

Contrail™

---

# Contrail Feature Guide

Published  
2025-08-18

RELEASE  
5.0.3

Juniper Networks, Inc.  
1133 Innovation Way  
Sunnyvale, California 94089  
USA  
408-745-2000  
[www.juniper.net](http://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*

5.0.3

Copyright © 2025 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

[About This Guide | xx](#)

1

## **Overview**

[Understanding Contrail Controller | 2](#)

[Contrail Overview | 2](#)

[Contrail Description | 3](#)

2

## **Installing and Upgrading Contrail**

[Supported Platforms and Server Requirements | 6](#)

[Server Requirements and Supported Platforms | 6](#)

[Installing Contrail and Provisioning Roles | 7](#)

[Introduction to Containerized Contrail Modules | 7](#)

[Introduction to Contrail Microservices Architecture | 11](#)

[Downloading Installation Software | 13](#)

[Overview of contrail-ansible-deployer used in Contrail Command for Installing Contrail with Microservices Architecture | 13](#)

[Installing Contrail with OpenStack and Kolla Ansible | 20](#)

[Configuring the Control Node with BGP | 34](#)

[Contrail Global Controller | 39](#)

[Role and Resource-Based Access Control | 41](#)

[Installation and Configuration Scenarios | 51](#)

[Setting Up and Using a Simple Virtual Gateway with Contrail 4.0 | 51](#)

[Introduction to the Simple Gateway | 52](#)

[How the Simple Gateway Works | 52](#)

[Setup Without Simple Gateway | 52](#)

[Setup With a Simple Gateway | 53](#)

[Simple Gateway Configuration Features | 54](#)

[Packet Flows with the Simple Gateway | 55](#)

Packet Flow Process From the Virtual Network to the Public Network	56
Packet Flow Process From the Public Network to the Virtual Network	56
Methods for Configuring the Simple Gateway	57
Using the vRouter Configuration File to Configure the Simple Gateway	57
Using Thrift Messages to Dynamically Configure the Simple Gateway	57
How to Dynamically Create a Virtual Gateway	58
How to Dynamically Delete a Virtual Gateway	59
Using Devstack to Configure the Simple Gateway	59
Common Issues with Simple Gateway Configuration	60
Configuring MD5 Authentication for BGP Sessions	61
Configuring the Data Plane Development Kit (DPDK) Integrated with Contrail vRouter	62
Configuring Contrail DPDK vRouter to Run in a Docker Container	65
Configuring Single Root I/O Virtualization (SR-IOV)	66
Configuring Virtual Networks for Hub-and-Spoke Topology	73
Route Targets for Virtual Networks in Hub-and-Spoke Topology	74
Example: Configuring Hub-and-Spoke Virtual Networks	74
Troubleshooting Hub-and-Spoke Topology	75
Configuring Transport Layer Security-Based XMPP in Contrail	80
Configuring Graceful Restart and Long-lived Graceful Restart	83
Remote Compute	88
Dynamic Kernel Module Support (DKMS) for vRouter	97
<b>Upgrading Contrail Software  </b>	<b>98</b>
Contrail In-Service Software Upgrade from Releases 3.2 and 4.1 to 5.0.x using Ansible Deployer	98
Contrail In-Service Software Upgrade from Releases 3.2 and 4.1 to 5.0.x using Helm Deployer	108
<b>Backup and Restore Contrail Software  </b>	<b>120</b>
Backing up Contrail Databases in JSON Format	120
<b>Multicloud Contrail  </b>	<b>128</b>
Contrail Deployment on Microsoft Azure	128
Deploying Contrail on Microsoft Azure	129



- Deployment of Contrail on Azure | 129
- Deleting Contrail Deployment from Azure | 134

## On-Premise and Azure Multicloud Deployment | 135

- Installing On-Premise Contrail | 135
- Extending On-Premise Contrail To Microsoft Azure | 138

## Modifying Multicloud Topology | 147

## Deploying Contrail Enterprise Multicloud using REST API | 148

## Using Contrail with Kubernetes | 166

### Contrail Integration with Kubernetes | 166

### Installing and Managing Contrail 5.0 Microservices Architecture Using Helm Charts | 173

### Provisioning of Kubernetes Clusters | 177

- Provisioning of a Standalone Kubernetes Cluster | 177
- Provisioning of Nested Contrail Kubernetes Clusters | 178
  - Configure network connectivity to Contrail configuration and data plane functions. | 179
  - Generate a single yaml file to create a Contrail-k8s cluster | 180
  - Instantiate the Contrail-k8s cluster | 181
- Provisioning of Non-Nested Contrail Kubernetes Clusters | 181

### Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability | 184

### Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata | 194

### Accessing a Contrail OpenStack Helm Cluster | 198

### Frequently Asked Questions About Contrail and Helm Charts | 201

### Contrail Deployment with Helm | 205

### Verifying Configuration for CNI for Kubernetes | 211

- View Pod Name and IP Address | 212
- Verify Reachability of Pods | 212
- Verify If Isolated Namespace-Pods Are Not Reachable | 212
- Verify If Non-Isolated Namespace-Pods Are Reachable | 213
- Verify If a Namespace is Isolated | 214

### Kubernetes Updates to IP Fabric | 214

### Implementation of Kubernetes Network Policy with Contrail Firewall Policy | 217

## **Using VMware vCenter with Containerized Contrail | 233**

vCenter Integration for Contrail Release 5.0 | 233

vCenter Integration for Contrail Release 5.0.1 | 235

- Set Up the Controller Server | 236

- Download the Ansible Deployer with Contrail Playbooks | 236

- Configure Contrail Parameters and Install Contrail | 236

vCenter Integration for Contrail Release 5.0.2 | 237

- Prerequisites | 238

- ESX Agent Manager | 238

- Set Up vCenter Server | 238

- Configure Contrail Parameters | 243

- Install Contrail | 243

- Monitor and Manage ContrailVM from ESX Agent Manager | 243

Underlay Network Configuration for ContrailVM | 246

- Standard Switch Setup | 246

- Distributed Switch Setup | 248

- PCI Pass-Through Setup | 250

- SR-IOV Setup | 253

Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Releases 5.0 and 5.0.1 | 258

Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Release 5.0.2 | 273

Integrating Contrail Release 5.0.X with VMware vRealize Orchestrator | 280

Installing and Provisioning Contrail VMware vRealize Orchestrator Plugin | 286

- Accessing vRO Control Center | 286

- Installing vRO Plugin | 289

- Accessing vRO Desktop Client | 291

- Connecting to vRO using the Desktop Client | 291

- Connecting to Contrail Controller | 292

## **Using Contrail with Red Hat | 296**

Deploying Contrail with Red Hat OpenStack Platform Director 13 | 296

Provisioning Red Hat OpenShift Container Platform Clusters Using Ansible Deployer | 351

Installing a Standalone OpenShift Cluster Using Ansible Deployer | 351

Provisioning of Nested OpenShift Clusters Using Ansible Deployer—Beta | 361

Configure network connectivity to Contrail configuration and data plane functions | 361

Installing Nested OpenShift Cluster using Ansible Deployer | 365

## **Contrail and AppFormix Kolla/Ocata OpenStack Deployment | 366**

Contrail and AppFormix Deployment Requirements | 366

Preparing for the Installation | 367

Run the Playbooks | 371

Accessing Contrail in AppFormix Management Infrastructure in UI | 372

Notes and Caveats | 373

Example Instances.yml for Contrail and AppFormix OpenStack Deployment | 373

Installing AppFormix for OpenStack | 377

Installing AppFormix for OpenStack in HA | 382

## **Using Contrail with Juju Charms | 388**

Deploying Contrail by Using Juju Charms | 388

Preparing to Deploy Contrail by Using Juju Charms | 389

Deploying Contrail Charms | 391

Deploying Contrail Charms in a Bundle | 391

Deploying Juju Charms Manually | 398

Options for Juju Charms | 403

## **Contrail Command | 411**

Configuring Contrail Command | 411

Requirements | 411

Overview | 412

Configuration | 412

Sample command\_servers.yml File | 414

Deploying Contrail Cluster using the Contrail Command UI | 418

Requirements | 419

Overview | 419

Configuration | 421

Deploying Contrail Cluster using Contrail-Command and instances.yml | 431

Importing Contrail Cluster Data using Contrail Command | 439

## **Extending Contrail to Physical Routers, Bare Metal Servers, Switches, and Interfaces | 444**

Understanding Bare Metal Server Management | 444

Configuring High Availability for the Contrail OVSDb ToR Agent | 446

Using Device Manager to Manage Physical Routers | 453

SR-IOV VF as the Physical Interface of vRouter | 485

Using Gateway Mode to Support Remote Instances | 487

REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces | 489

## **Contrail for Data Center Automation and Fabric Management | 496**

Understanding Underlay Management | 496

Support for Intent Driven Automation Functionality using Ansible | 497

Providing Intent Driven Automation Capabilities on Physical Network Elements | 498

Image Management | 498

Upload a New Device Image | 498

Upgrade an Existing Device Image | 499

Fabric Management | 500

Create a Fabric | 501

Delete a Fabric | 504

Discover a Device | 504

Assign Role to a Device | 505

Manage Device Configuration | 506

View Node Profile Information | 506

Configuring Data Center Gateway | 506

Configuring QFX Series Devices as Data Center Gateway | 507

Onboard Brownfield Devices | 507

Add Bare Metal Server | 508

Create Tenant Virtual Network | 509

Add CSN Nodes | 517

Create Logical Routers | 518

- Verification | 520

- Configuring MX Series Routers as Data Center Gateway | 520

- Onboard Brownfield Devices | 521

- Create Virtual Network | 521

## Configuring Contrail

### Configuring Virtual Networks | 524

- Creating Projects in OpenStack for Configuring Tenants in Contrail | 524

- Creating a Virtual Network with Juniper Networks Contrail | 526

- Creating a Virtual Network with OpenStack Contrail | 530

- Creating an Image for a Project in OpenStack Contrail | 532

- Creating a Floating IP Address Pool | 536

- Using Security Groups with Virtual Machines (Instances) | 538

- Security Groups Overview | 538

- Creating Security Groups and Adding Rules | 539

- Support for IPv6 Networks in Contrail | 543

- Configuring EVPN and VXLAN | 547

- Configuring the VXLAN Identifier Mode | 549

- Configuring Forwarding | 551

- Configuring the VXLAN Identifier | 552

- Configuring Encapsulation Methods | 553

- Support for EVPN Route Type 5 | 556

### Example of Deploying a Multi-Tier Web Application Using Contrail | 558

- Example: Deploying a Multi-Tier Web Application | 558

- Multi-Tier Web Application Overview | 558

- Example: Setting Up Virtual Networks for a Simple Tiered Web Application | 559

- Verifying the Multi-Tier Web Application | 562

- Sample Addressing Scheme for Simple Tiered Web Application | 562

- Sample Physical Topology for Simple Tiered Web Application | 563

- Sample Physical Topology Addressing | 564

- Sample Network Configuration for Devices for Simple Tiered Web Application | 566

## Configuring Services | 573

### Configuring DNS Servers | 573

- DNS Overview | 573
- Defining Multiple Virtual Domain Name Servers | 574
- IPAM and Virtual DNS | 575
- DNS Record Types | 575
- Configuring DNS Using the Interface | 576
- Configuring DNS Using Scripts | 584

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

### Support for Multicast | 596

- Subnet Broadcast | 597
- All-Broadcast/Limited-Broadcast and Link-Local Multicast | 597
- Host Broadcast | 598

### Using Static Routes with Services | 599

- Static Routes for Service Instances | 599
- Configuring Static Routes on a Service Instance | 600
- Configuring Static Routes on Service Instance Interfaces | 601
- 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 | 651

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

Bidirectional Forwarding and Detection Health Check for BGPaaS | 656

Health Check of Transparent Service Chain | 656

Service Instance Fate Sharing | 656

## **Examples: Configuring Service Chaining | 658**

Example: Creating an In-Network Service Chain by Using Contrail Command | 658

Hardware and Software Requirements | 658

Overview | 659

Configuration | 659

Example: Creating an In-Network-NAT Service Chain by Using Contrail Command | 666

Prerequisites | 667

Overview | 668

Configuration | 668

Example: Creating a Transparent Service Chain by Using Contrail Command | 675

Prerequisites | 675

Overview | 676

Configuration | 676

## **Adding Physical Network Functions in Service Chains | 684**

Using Physical Network Functions in Contrail Service Chains | 684

PNF Service Chaining Objects | 684

Prerequisites and Assumptions | 685

Example: Adding a Physical Network Function Device to a Service Chain | 686

## **QoS Support in Contrail | 694**

Quality of Service in Contrail | 694

Configuring Network QoS Parameters | 703

Overview | 703

QoS Configuration Examples | 703

Limitations | 705

## **BGP as a Service | 706**

BGP as a Service | 706

BGP as a Service in Contrail Release 3.1 | 710

## **Load Balancers | 712**

Using Load Balancers in Contrail | 712

Support for OpenStack LBaaS Version 2.0 APIs | 727

Configuring Load Balancing as a Service in Contrail | 729

Overview: Load Balancing as a Service | 730

Contrail LBaaS Implementation | 731

Configuring LBaaS Using CLI | 732

Configuring LBaaS using the Contrail Web UI | 734

## **Optimizing Contrail | 744**

Route Target Filtering | 744

Introduction | 744

Debugging and Troubleshooting Route Target Filtering | 745

RTF Limitations in Contrail 1.10 | 746

Source Network Address Translation (SNAT) | 747

Overview | 747

Neutron APIs for Routers | 748

Network Namespace | 748

Using the Web UI to Configure Routers with SNAT | 749

Using the Web UI to Configure Distributed SNAT | 750

Multiqueue Virtio Interfaces in Virtual Machines | 752

vRouter Command Line Utilities | 753

Overview | 754

vif Command | 755

flow Command | 759

vrfstats Command | 760

rt Command | 761

dropstats Command | 763

mpls Command | 767

mirror Command | 769

vxlan Command | 771

nh Command | 772



## 4

**Contrail Security****Contrail Security | 777**

Security Policy Enhancements | 777

## 5

**Monitoring and Troubleshooting Contrail****Configuring Traffic Mirroring to Monitor Network Traffic | 798**

Configuring Traffic Analyzers and Packet Capture for Mirroring | 798

Traffic Analyzer Images | 798

Configuring Traffic Analyzers | 799

Setting Up Traffic Mirroring Using Configure &gt; Networking &gt; Services | 799

Configuring Interface Monitoring and Mirroring | 806

Mirroring Enhancements | 807

Analyzer Service Virtual Machine | 809

Mapping VLAN Tags from a Physical NIC to a VMI (NIC-Assisted Mirroring) | 812

**Understanding Contrail Analytics | 814**

Understanding Contrail Analytics | 814

Contrail Alerts | 815

Underlay Overlay Mapping in Contrail | 819

Overview: Underlay Overlay Mapping using Contrail Analytics | 820

Underlay Overlay Analytics Available in Contrail | 820

Architecture and Data Collection | 821

New Processes/Services for Underlay Overlay Mapping | 821

External Interfaces Configuration for Underlay Overlay Mapping | 822

Physical Topology | 822

SNMP Configuration | 823

Link Layer Discovery Protocol (LLDP) Configuration | 823

IPFIX and sFlow Configuration | 823

Sending pRouter Information to the SNMP Collector in Contrail | 826

pRouter UVEs | 826

Contrail User Interface for Underlay Overlay Analytics | 828

Enabling Physical Topology on the Web UI | 829

Viewing Topology to the Virtual Machine Level | 829

- Viewing the Traffic of any Link | 829
- Trace Flows | 830
- Search Flows and Map Flows | 831
- Overlay to Underlay Flow Map Schemas | 832
- Module Operations for Overlay Underlay Mapping | 835
- SNMP Collector Operation | 835
- Topology Module Operation | 837
- IPFIX and sFlow Collector Operation | 838
- Troubleshooting Underlay Overlay Mapping | 839
- Script to add pRouter Objects | 839

## **Configuring Contrail Analytics | 842**

Analytics Scalability | 842

High Availability for Analytics | 844

System Log Receiver in Contrail Analytics | 844

- Overview | 845
- Redirecting System Logs to Contrail Collector | 845
- Exporting Logs from Contrail Analytics | 845

Sending Flow Messages to the Contrail System Log | 845

Ceilometer Support in a Contrail Cloud | 846

- Overview | 847
- Ceilometer Details | 847
- Verification of Ceilometer Operation | 848
- Contrail Ceilometer Plugin | 850
- Ceilometer Installation and Provisioning | 853

User Configuration for Analytics Alarms and Log Statistics | 854

- Configuring Alarms Based on User-Visible Entities Data | 854
- Examples: Detecting Anomalies | 856
- Configuring the User-Defined Log Statistic | 857
- Implementing the User-Defined Log Statistic | 860

Alarms History | 863

Node Memory and CPU Information | 865

Role- and Resource-Based Access Control for the Contrail Analytics API | 866

Configuring Analytics as a Standalone Solution | 867

Configuring Secure Sandesh and Introspect for Contrail Analytics | 870

## Using Contrail Analytics to Monitor and Troubleshoot the Network | 873

Monitoring the System | 873

Debugging Processes Using the Contrail Introspect Feature | 877

Monitor > Infrastructure > Dashboard | 882

- Monitor Dashboard | 883

- Monitor Individual Details from the Dashboard | 883

- Using Bubble Charts | 884

- Color-Coding of Bubble Charts | 885

Monitor > Infrastructure > Control Nodes | 886

- Monitor Control Nodes Summary | 886

- Monitor Individual Control Node Details | 887

- Monitor Individual Control Node Console | 889

- Monitor Individual Control Node Peers | 892

- Monitor Individual Control Node Routes | 894

Monitor > Infrastructure > Virtual Routers | 897

- Monitor vRouters Summary | 897

- Monitor Individual vRouters Tabs | 899

- Monitor Individual vRouter Details Tab | 899

- Monitor Individual vRouters Interfaces Tab | 901

- Monitor Individual vRouters Networks Tab | 903

- Monitor Individual vRouters ACL Tab | 904

- Monitor Individual vRouters Flows Tab | 906

- Monitor Individual vRouters Routes Tab | 907

- Monitor Individual vRouter Console Tab | 908

Monitor > Infrastructure > Analytics Nodes | 911

- Monitor Analytics Nodes | 911

- Monitor Analytics Individual Node Details Tab | 913

- Monitor Analytics Individual Node Generators Tab | 914

- Monitor Analytics Individual Node QE Queries Tab | 915

- Monitor Analytics Individual Node Console Tab | 916

## Monitor > Infrastructure > Config Nodes | 919

- Monitor Config Nodes | 919
- Monitor Individual Config Node Details | 920
- Monitor Individual Config Node Console | 921

## Monitor > Networking | 923

- Monitor > Networking Menu Options | 923
- Monitor -> Networking -> Dashboard | 924
- Monitor > Networking > Projects | 926
- Monitor Projects Detail | 927
- Monitor > Networking > Networks | 930

## Query > Flows | 935

- Query > Flows > Flow Series | 936
- Example: Query Flow Series | 939
- Query > Flow Records | 941
- Query > Flows > Query Queue | 944

## Query > Logs | 945

- Query > Logs Menu Options | 946
- Query > Logs > System Logs | 946
- Sample Query for System Logs | 948
- Query > Logs > Object Logs | 950

## Understanding Flow Sampling | 952

### Example: Debugging Connectivity Using Monitoring for Troubleshooting | 957

- Using Monitoring to Debug Connectivity | 957

## Common Support Answers | 964

### Debugging Ping Failures for Policy-Connected Networks | 964

### Debugging BGP Peering and Route Exchange in Contrail | 972

- Example Cluster | 972
- Verifying the BGP Routers | 972
- Verifying the Route Exchange | 976
- Debugging Route Exchange with Policies | 978
- Debugging Peering with an MX Series Router | 980
- Debugging a BGP Peer Down Error with Incorrect Family | 982

- Configuring MX Peering (iBGP) | 984
- Checking Route Exchange with an MX Series Peer | 986
- Checking the Route in the MX Series Router | 988

## Troubleshooting the Floating IP Address Pool in Contrail | 990

- Example Cluster | 991
- Example | 992
- Example: MX80 Configuration for the Gateway | 993
- Ping the Floating IP from the Public Network | 996
- Troubleshooting Details | 997
- Get the UUID of the Virtual Network | 997
- View the Floating IP Object in the API Server | 998
- View floating-ips in floating-ip-pools in the API Server | 1002
- Check Floating IP Objects in the Virtual Machine Interface | 1005
- View the BGP Peer Status on the Control Node | 1009
- Querying Routes in the Public Virtual Network | 1010
- Verification from the MX80 Gateway | 1012
- Viewing the Compute Node Vnsw Agent | 1014
- Advanced Troubleshooting | 1016

## Removing Stale Virtual Machines and Virtual Machine Interfaces | 1019

- Problem Example | 1019
- Show Virtual Machines | 1021
- Show Virtual Machines Using Python API | 1022
- Delete Methods | 1024

## Troubleshooting Link-Local Services in Contrail | 1024

- Overview of Link-Local Services | 1024
- Troubleshooting Procedure for Link-Local Services | 1025
- Metadata Service | 1026
- Troubleshooting Procedure for Link-Local Metadata Service | 1026

## Contrail Commands and APIs

### Contrail Commands | 1030

#### Getting Contrail Node Status | 1030

- Overview | 1030
- UVE for NodeStatus | 1031

- Node Status Features | **1031**
- Using Introspect to Get Process Status | **1038**
- contrail-status script | **1040**

contrail-logs (Accessing Log File Messages) | **1042**

contrail-status (Viewing Node Status) | **1045**

contrail-version (Viewing Version Information) | **1047**

service (Managing Services) | **1050**

Backing Up Contrail Databases Using JSON Format | **1052**

## **Contrail Application Programming Interfaces (APIs) | 1060**

Contrail Analytics Application Programming Interfaces (APIs) and User-Visible Entities (UVEs) | **1060**

- User-Visible Entities | **1061**
- Common UVEs in Contrail | **1062**
- Virtual Network UVE | **1062**
- Virtual Machine UVE | **1063**
- vRouter UVE | **1063**
- UVEs for Contrail Nodes | **1064**
- Wild Card Query of UVEs | **1064**
- Filtering UVE Information | **1064**

Log and Flow Information APIs | **1074**

- HTTP GET APIs | **1074**
- HTTP POST API | **1075**
- POST Data Format Example | **1075**
- Query Types | **1077**
- Examining Query Status | **1077**
- Examining Query Chunks | **1078**
- Example Queries for Log and Flow Data | **1078**

Working with Neutron | **1082**

- Data Structure | **1082**
- Network Sharing in Neutron | **1083**
- Commands for Neutron Network Sharing | **1084**
- Support for Neutron APIs | **1084**
- Contrail Neutron Plugin | **1085**

DHCP Options | **1085**

Incompatibilities | **1085**

Support for Amazon VPC APIs on Contrail OpenStack | **1086**

Overview of Amazon Virtual Private Cloud | **1086**

Mapping Amazon VPC Features to OpenStack Contrail Features | **1087**

VPC and Subnets Example | **1088**

Euca2ools CLI for VPC and Subnets | **1089**

Security in VPC: Network ACLs Example | **1089**

Euca2ools CLI for Network ACLs | **1091**

Security in VPC: Security Groups Example | **1091**

Euca2ools CLI for Security Groups | **1092**

Elastic IPs in VPC | **1093**

Euca2ools CLI for Elastic IPs | **1093**

Euca2ools CLI for Route Tables | **1094**

Supported Next Hops | **1094**

Internet Gateway Next Hop Euca2ools CLI | **1095**

NAT Instance Next Hop Euca2ools CLI | **1095**

Example: Creating a NAT Instance with Euca2ools CLI | **1095**

# About This Guide

Use this guide to understand Contrail and its ability to create and orchestrate highly secure virtual networks. This guide also provides information about Docker containers, microservices architecture, deployment of Web applications, service chaining, and configuration of Contrail Analytics for monitoring and troubleshooting the network. Information about commonly used Contrail CLI commands and APIs are also included.

## RELATED DOCUMENTATION

<a href="#">README Access to Contrail Registry 5.0</a>
<a href="#">Contrail Release 5.0</a>
<a href="#">Release Notes 5.0</a>
<a href="#">Day One: Understanding OpenContrail Architecture</a>
<a href="#">Juniper Contrail Configuration API Reference</a>
<a href="#">Juniper Networks TechWiki: Contrail</a>



# 1

PART

## Overview

---

- [Understanding Contrail Controller | 2](#)
-

# Understanding Contrail Controller

## IN THIS CHAPTER

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

## 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*

# 2

PART

## Installing and Upgrading Contrail

---

- Supported Platforms and Server Requirements | 6
  - Installing Contrail and Provisioning Roles | 7
  - Installation and Configuration Scenarios | 51
  - Upgrading Contrail Software | 98
  - Backup and Restore Contrail Software | 120
  - Multicloud Contrail | 128
  - Using Contrail with Kubernetes | 166
  - Using VMware vCenter with Containerized Contrail | 233
  - Using Contrail with Red Hat | 296
  - Contrail and AppFormix Kolla/Ocata OpenStack Deployment | 366
  - Using Contrail with Juju Charms | 388
  - Contrail Command | 411
  - Extending Contrail to Physical Routers, Bare Metal Servers, Switches, and Interfaces | 444
  - Contrail for Data Center Automation and Fabric Management | 496
-

# Supported Platforms and Server Requirements

## IN THIS CHAPTER

- [Server Requirements and Supported Platforms | 6](#)

## Server Requirements and Supported Platforms

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

For a list of supported platforms, see *Supported Platforms Contrail 5.0*.

## RELATED DOCUMENTATION

| [Downloading Installation Software](#)

# Installing Contrail and Provisioning Roles

## IN THIS CHAPTER

- Introduction to Containerized Contrail Modules | 7
- Introduction to Contrail Microservices Architecture | 11
- Downloading Installation Software | 13
- Overview of contrail-ansible-deployer used in Contrail Command for Installing Contrail with Microservices Architecture | 13
- Installing Contrail with OpenStack and Kolla Ansible | 20
- Configuring the Control Node with BGP | 34
- Contrail Global Controller | 39
- Role and Resource-Based Access Control | 41

## Introduction to Containerized Contrail Modules

### IN THIS SECTION

- Why Use Containers? | 8
- Overview of Contrail Containers | 8
- Contrail 4.0 Containers | 9
- DPDK vRouter | 11
- Summary of Container Design, Configuration Management, and Orchestration | 11

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.

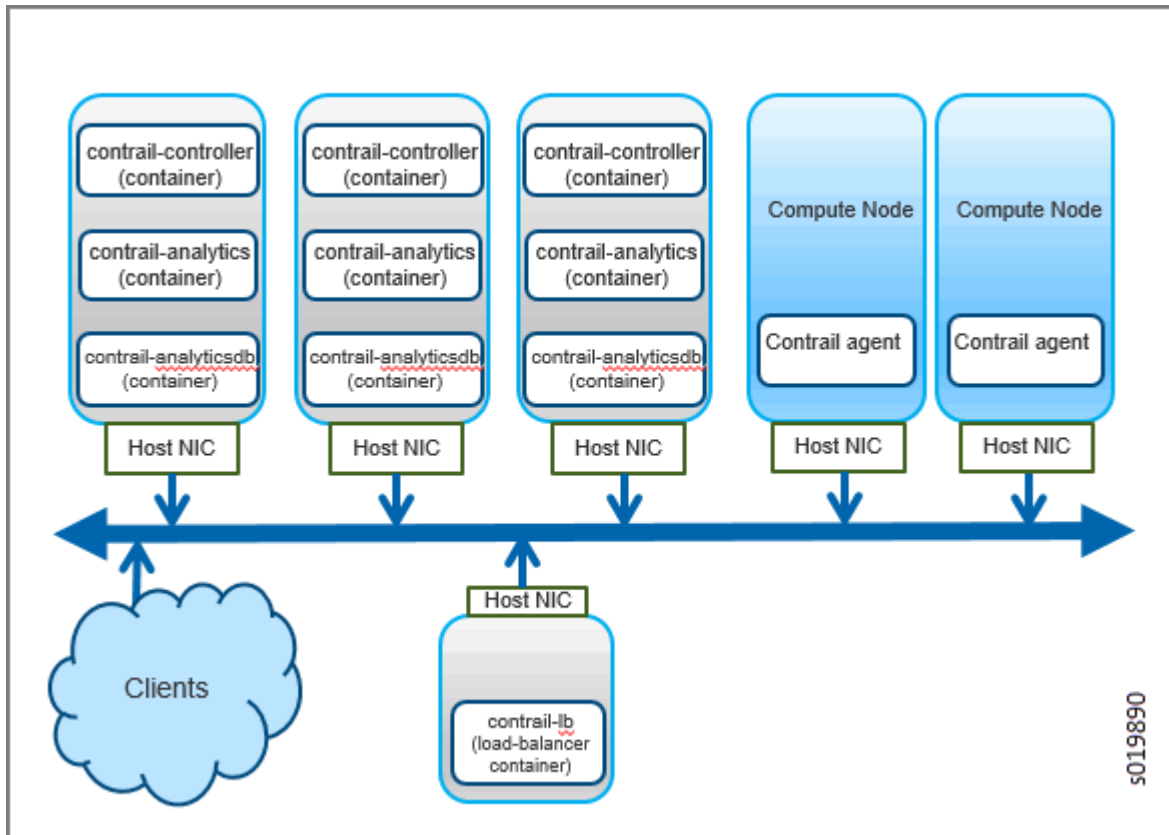
In Contrail 4.0.1, the containerized components include OpenStack Ocata services. Only OpenStack Ocata services are containerized. Mitaka and Newton SKUs of OpenStack are still provisioned as non-containerized host services.

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.

## DPDK vRouter

Starting with Contrail release 5.0, you can configure the Contrail DPDK vRouter to run in a Docker container. In earlier releases, DPDK vRouter runs on a compute host. The `contrail-vrouter-dpdk` binary file provides data plane functionality when Contrail vRouter is run in DPDK mode in a Contrail cluster.

## 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.
- 
- 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.

## Introduction to Contrail Microservices Architecture

### IN THIS SECTION

- [What is Contrail Microservices Architecture? | 12](#)
- [Installing Contrail with Microservices Architecture | 12](#)

With Contrail 4.0, Contrail started moving to an architecture of containers for major system components. Each container encapsulates the services needed for that container. The first phase of Contrail containers were characterized as fat containers, where multiple processes run within the container.

Starting with Contrail Release 5.0, more components are being containerized, and the fat containers are being decomposed into thin containers with microservices. The microservices are still encapsulated in

their respective containers, however, only the essential functions relative to each container's functions are present as microservices. This enables a more agile system, avoiding monolithic containers.

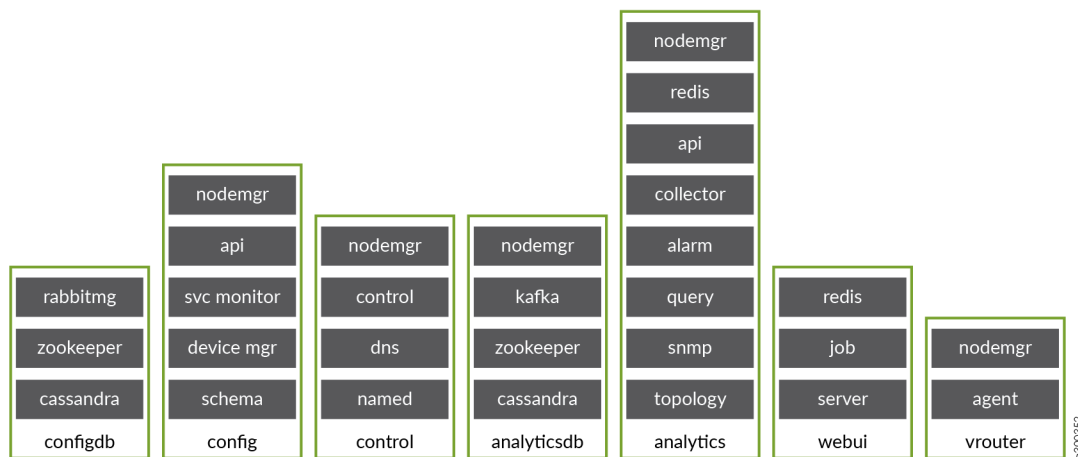
## What is Contrail Microservices Architecture?

Nothing is changing with regard to Contrail functionality, however, employing microservices provides a number of benefits, including the ability to deploy patches without updating the entire Contrail deployment, offering better ways to manage the lifecycles of containers, and improving user experiences with Contrail provisioning and upgrading. The microservices architecture enables provisioning with minimum information provided, and enables every feature to be configurable. Utilizing microservices also simplifies application complexity by implementing small, independent processes.

The containers and their processes are grouped as services and microservices, and are similar to pods in the Kubernetes open-source software used to manage containers on a server cluster.

Figure 2 on page 12 shows how the Contrail containers and microservices are grouped into a pod structure upon installation.

**Figure 2: Contrail Containers, Pods, and Microservices**



## Installing Contrail with Microservices Architecture

Procedures have been developed to simplify the installation and management of Contrail with microservices architecture. Refer to the following topics for installation for the operating system appropriate for your system:

- *Understanding contrail-ansible-deployer used in Contrail Command*
- [Installing and Managing Contrail Microservices Architecture Using Helm Charts](#)

## 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

## Overview of contrail-ansible-deployer used in Contrail Command for Installing Contrail with Microservices Architecture

### IN THIS SECTION

- [What is the contrail-ansible-deployer? | 14](#)
- [Preparing to Install with Contrail | 14](#)
- [Supported Providers | 15](#)
- [Configure a Yaml File for Your Environment | 15](#)
- [Installing a Contrail System | 20](#)

Contrail Release 5.0 introduces microservices architecture. This section provides an overview of using contrail-ansible-deployer used by The Contrail Command tool for installing Contrail Networking with a microservices architecture. For an introduction to Contrail microservices, refer to *Introduction to Contrail Microservices Architecture*. For step by step procedure on how to install contrail using contrail command deployer, refer to *Installing Contrail Cluster using Contrail Command and instances.yml*.

## What is the contrail-ansible-deployer?

The contrail-ansible-deployer is a set of Ansible playbooks designed to deploy Contrail 5.x with microservices architecture.

The contrail-ansible-deployer contains three plays:

### playbooks/provision\_instances.yml

This play provisions the operating system instances for hosting the containers, currently for these infrastructure providers:

- kvm
- gce
- aws

### playbooks/configure\_instances.yml

This play configures the provisioned instances. The playbook installs software and configures the operating system to meet prerequisite standards. This is applicable to all providers.

### playbooks/install\_contrail.yml

This play pulls, configures, and starts the Contrail containers.

## Preparing to Install with Contrail

This section helps you prepare your system before installing Contrail 5.0.x using contrail-command-deployer.

### Prerequisites

Make sure your system is operational with the following before installation with contrail-command-deployer.

- CentOS 7.4, kernel  $\geq$  3.10.0-693.17.1
- Ansible 2.4.2.0
- Name resolution is operational for long and short host names of the cluster nodes, through either DNS or the host file.
- Docker engine (tested version is 17.03.1-ce)
- The docker-compose installed (tested version is 1.17.0)

- The docker-compose Python library (tested version is 1.9.0)
- If using Kubernetes (k8s), the tested version is 1.9.2.0
- For high availability (HA), the time must be in sync between the cluster nodes.

## Supported Providers

The playbooks support installing Contrail on the following providers:

- bms—bare metal server
- kvm—kernel-based virtual machine (KVM)-hosted virtual machines
- gce—Google compute engine (GCE)-hosted virtual machines
- aws—Amazon Web Services (AWS)-hosted virtual machines

## Configure a Yaml File for Your Environment

The configuration for all three plays is contained in a single file:

`config/instances.yaml`

The configuration has multiple main sections, including:

The main sections of the `config/instances.yaml` file are described in this section. Using the sections that are appropriate for your system, configure each with parameters specific to your environment.

## Provider Configuration

The section `provider_config` configures provider-specific settings.

## KVM Provider Example

Use this example if you are in a kernel-based virtual machine-hosted environment.

```
provider_config:                                     # the provider section contains all provider
relevant configuration
  kvm:                                                # Mandatory.
    image: CentOS-7-x86_64-GenericCloud-1710.qcow2.xz # Mandatory for provision play. Image
to be deployed.
    image_url: https://cloud.centos.org/centos/7/images/ # Mandatory for provision play. Path/
url to image.
```

Use this example if you are in a bare metal server environment.

```
provider_config:
  bms:                                # Mandatory.
  ssh_pwd: contrail123                # Optional. Not needed if ssh keys are used.
  ssh_user: centos                    # Mandatory.
  ssh_public_key: /home/centos/.ssh/id_rsa.pub # Optional. Not needed if ssh password is used.
  ssh_private_key: /home/centos/.ssh/id_rsa    # Optional. Not needed if ssh password is used.
  ntpserver: ntp-server-ip-address           # Optional. Needed if ntp server
should be configured.
  domainsuffix: local                 # Optional. Needed if configuration play
should configure /etc/hosts
```

Use this example if you are in an Amazon Web Services environment.

```
provider_config:
  aws: # Mandatory.
```



```

ec2_access_key: THIS_IS_YOUR_ACCESS_KEY      # Mandatory.
ec2_secret_key: THIS_IS_YOUR_SECRET_KEY      # Mandatory.
ssh_public_key: /home/centos/.ssh/id_rsa.pub  # Optional.
ssh_private_key: /home/centos/.ssh/id_rsa     # Optional.
ssh_user: centos                             # Mandatory.
instance_type: t2.xlarge                     # Mandatory.
image: ami-337be65c                          # Mandatory.
region: eu-central-1                        # Mandatory.
security_group: SECURITY_GROUP_ID            # Mandatory.
vpc_subnet_id: VPC_SUBNET_ID                # Mandatory.
assign_public_ip: yes                        # Mandatory.
volume_size: 50                             # Mandatory.
key_pair: KEYPAIR_NAME                      # Mandatory.

```

## GCE Provider Example

Use this example if you are in a Google Cloud environment.

```

provider_config:
  gce:
    # Mandatory.
    service_account_email: # Mandatory. GCE service account email address.
    credentials_file:      # Mandatory. Path to GCE account json file.
    project_id:            # Mandatory. GCE project name.
    ssh_user:              # Mandatory. Ssh user for GCE instances.
    ssh_pwd:               # Optional. Ssh password used by ssh user, not needed when
public is used
    ssh_private_key:       # Optional. Path to private SSH key, used by by ssh user, not
needed when ssh-agent loaded private key
    machine_type: n1-standard-4 # Mandatory. Default is too small
    image: centos-7           # Mandatory. For provisioning and configuration only centos-7
is currently supported.
    network: microservice-vn  # Optional. Defaults to default
    subnetwork: microservice-sn # Optional. Defaults to default
    zone: us-west1-aA        # Optional. Defaults to ?
    disk_size: 50            # Mandatory. Default is too small

```

## Global Services Configuration

This section sets global service parameters. All parameters are optional.

```
global_configuration:
  CONTAINER_REGISTRY: hub.juniper.net/contrail
  REGISTRY_PRIVATE_INSECURE: True
  CONTAINER_REGISTRY_USERNAME: YourRegistryUser
  CONTAINER_REGISTRY_PASSWORD: YourRegistryPassword
```

## Contrail Services Configuration

This section sets global Contrail service parameters. All parameters are optional.

```
contrail_configuration:      # Contrail service configuration section
  CONTRAIL_VERSION: latest
  UPGRADE_KERNEL: true
```

For a complete list of parameters available for `contrail_configuration.md`, see [Contrail Configuration Parameters for Ansible Deployer](#).

## Kolla Services Configuration

If OpenStack Kolla is deployed, this section defines the parameters for Kolla.

```
kolla_config:
```

## Instances Configuration

Instances are the operating systems on which the containers will be launched. The instance configuration has a few provider-specific knobs. The instance configuration specifies which roles are installed on which instance. Additionally, instance-wide and role-specific Contrail and Kolla configurations can be specified, overwriting the parameters from the global Contrail and Kolla configuration settings.

## GCE Default All-in-One Instance

The following example is a very simple all-in-one GCE instance. It will install all Contrail roles and the Kubernetes master and node, using the default configuration.

```
instances:
  gce1:                                # Mandatory. Instance name
    provider: gce                      # Mandatory. Instance runs on GCE
```

## AWS Default All-in-One Instance

The following example uses three AWS EC2 instances to deploy, and an all-in-one high availability setup with all roles and default parameters.

```
instances:
  aws1:
    provider: aws
  aws2:
    provider: aws
  aws3:
    provider: aws
```

## KVM Contrail Plane Instance

The following example is a KVM-based instance only, installing Contrail control plane containers.

```
instances:
  kvm1:
    provider: kvm
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      kubemanager:
      k8s_master:
```

## More Examples

Refer to the following for more configuration examples for instances.

- [GCE Kubernetes \(k8s\) HA with separate control and data plane instances](#)
- [AWS Kolla HA with separate control and data plane instances](#)

## Installing a Contrail System

To perform a full installation of a Contrail system, refer to the installation instructions in: *Installing Contrail Cluster using Contrail Command and instances.yml*.

### RELATED DOCUMENTATION

| *Installing Contrail Cluster using Contrail Command and instances.yml*

## Installing Contrail with OpenStack and Kolla Ansible

### IN THIS SECTION

- [Set Up the Base Host | 21](#)
- [Run OpenStack Commands | 24](#)
- [Multiple Interface Configuration Sample for Multinode OpenStack HA and Contrail | 25](#)
- [Single Interface Configuration Sample for Multinode OpenStack HA and Contrail | 27](#)
- [Frequently Asked Questions | 30](#)

This topic provides the steps needed to install Contrail Release 5.0.X. with OpenStack, using Kolla Ansible playbook `contrail-kolla-ansible`. Kolla is an OpenStack project that provides Docker containers and Ansible playbooks to provide production-ready containers and deployment tools for operating OpenStack clouds.

The `contrail-kolla-ansible` playbook works in conjunction with `contrail-ansible-deployer` to install OpenStack and Contrail Release 5.0.x. containers.

To deploy a Contrail Cluster using Contrail Command, see *Installing Contrail Cluster using Contrail Command and instances.yml*.

Deployment of Kolla containers using `contrail-kolla-ansible` and Contrail containers using `contrail-ansible-deployer` is presented in this topic:

## Set Up the Base Host

This procedure assumes you are installing with CentOS 7.5 kernel 3.10.0-862.11.6.el7.x86\_64. The vRouter has a [dependency](#) with the host kernel. Install this kernel version on the target nodes before provisioning.

To set up the base host:

1. Download the appropriate installer package from the [Contrail Download](#) page.

2. Install Ansible.

```
yum -y install epel-release
```

```
yum -y install git ansible-2.4.2.0
```

3. Untar the `tgz` file.

```
- tar xvf contrail-ansible-deployer-5.0.1-0.214.tgz
```

The `instances.yml` is located at the `contrail-ansible-deployer/config/`

4. Configure Contrail and Kolla parameters in the file `instances.yml`, using the following guidelines:

- The provider configuration (`provider_config`) section refers to the cloud provider where the Contrail cluster will be hosted, and contains all parameters relevant to the provider. For bare metal servers, the provider is `bms`.
- The `kolla_globals` section refers to OpenStack services. For more information about all possible `kolla_globals`, see <https://github.com/Juniper/contrail-kolla-ansible/.../globals.yml>.
- Additional Kolla configurations (`contrail-kolla-ansible`) are possible as `contrail_additions`. For more information about all possible `contrail_additions` to Kolla, see <https://github.com/Juniper/contrail-kolla-ansible/.../all.yml>.
- The `contrail_configuration` section contains parameters for Contrail services.
  - `CONTAINER_REGISTRY` specifies the registry from which to pull Contrail containers. It can be set to your local Docker registry if you are building your own containers. If a registry is not specified, it will try to pull the containers from the Docker hub.

If a custom registry is specified, also specify the same registry under `kolla_globals` as `contrail_docker_registry`.

- `CONTRAIL_VERSION`, if not specified, will default to the "latest" tag. It is possible to specify a tag from nightly builds.
- For more information about all possible parameters for `contrail_configuration`, see <https://github.com/Juniper/contrail-container-builder/.../common.sh>.
- If "roles" is not specified, the following roles are assumed.

```
config_database:
  config:
  control:
  analytics_database:
  analytics:
  webui:
  vrouter:
  openstack:
  openstack_compute:
```

- If there are host-specific values per host, for example, if the names of the interfaces used for "network\_interface" are different on the servers in your cluster, use the example configuration at [Configuration Sample for Multi Node OpenStack HA and Contrail \(multi interface\)](#).
- Many of the parameters are automatically derived to sane defaults (how the first configuration works). You can explicitly specify variables to override the derived values if required. Review the code to see the derivation logic.
- `CONTROL_DATA_NET_LIST` can be a comma separated list of CIDR subnets that can be designated for CONTROL/DATA plane traffic. The 'kolla' parameters 'network\_interface' will be derived from this subnet as the interface that corresponds to an IP address in this subnet. `CONTROL_DATA_NET_LIST` can still be used in a single interface setup by specifying the management subnet as the value so that the interface names need not be specified.

### Example: instances.yaml

This example is a bare minimum configuration for a single node, single interface, all-in-one cluster.

```
provider_config:
  bms:
    ssh_pwd: <password>
    ssh_user: root
```

```

    ntpserver: <IP NTP server>
    domainsuffix: local
instances:
  bms1:
    provider: bms
    ip: <IP BMS>1
contrail_configuration:
  RABBITMQ_NODE_PORT: 5673
  AUTH_MODE: keystone
  KEYSTONE_AUTH_URL_VERSION: /v3
kolla_config:
  kolla_globals:
    enable_haproxy: no
  kolla_passwords:
    keystone_admin_password: <Keystone admin password>

```

### Example: instances.yaml

This example is a more elaborate configuration for a single node, single interface, all-in-one cluster.

```

provider_config:
  bms:
    ssh_pwd: <password>
    ssh_user: root
    ntpserver: <IP NTP server>
    domainsuffix: local
instances:
  bms1:
    provider: bms
    ip: <IP BMS>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      vrouter:
      openstack:
      openstack_compute:
global_configuration:
  CONTAINER_REGISTRY: <Registry FQDN/IP>:<Registry Port>

```

```

    REGISTRY_PRIVATE_INSECURE: True
contrail_configuration:
    CONTRAIL_VERSION: latest
    CLOUD_ORCHESTRATOR: openstack
    VROUTER_GATEWAY: <IP gateway>
    RABBITMQ_NODE_PORT: 5673
    PHYSICAL_INTERFACE: <interface name>
    AUTH_MODE: keystone
    CONTROL_DATA_NET_LIST: 198.168.10.0/24
    KEYSTONE_AUTH_URL_VERSION: /v3
kolla_config:
    kolla_globals:
        kolla_internal_vip_address: <Internal VIP>
        contrail_api_interface_address: <Contrail API Addr>
        enable_haproxy: no
    kolla_passwords:
        keystone_admin_password: <Keystone Admin Password>

```

##### 5. Run the following Commands:

- `ansible-playbook -e orchestrator=openstack -i inventory/ playbooks/configure_instances.yml`
- `ansible-playbook -i inventory/ playbooks/install_openstack.yml`
- `ansible-playbook -e orchestrator=openstack -i inventory/ playbooks/install_contrail.yml`

##### 6. Open web browser and type **https://contrail-server-ip:8143** to access Contrail WebUI.

The default login user name is **admin**. Use the same password which was entered in step "4" on page [21](#)

## Run OpenStack Commands

At this time, it is necessary to manually install the OpenStack client (python-openstackclient) using pip. You cannot install using Yum repos because some dependent Python libraries conflict with the installation of the python-openstackclient. You also cannot install using pip repos because Ansible libraries can be overwritten.

##### 1. Manually install the python-openstackclient.

```

yum install -y gcc python-devel

pip install python-openstackclient

```



```
pip install python-ironicclient
```

## 2. Test the setup with VM-to-VM ping.

```
source /etc/kolla/admin-openrc.sh
wget http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img
openstack image create cirros2 --disk-format qcow2 --public --container-format bare --file
cirros-0.4.0-x86_64-disk.img
openstack network create testvn
openstack subnet create --subnet-range 198.168.100.0/24 --network testvn subnet1
openstack flavor create --ram 512 --disk 1 --vcpus 1 m1.tiny
NET_ID=`openstack network list | grep testvn | awk -F '|' '{print $2}' | tr -d ' '`
openstack server create --flavor m1.tiny --image cirros2 --nic net-id=${NET_ID} test_vm1
openstack server create --flavor m1.tiny --image cirros2 --nic net-id=${NET_ID} test_vm2
```

## Multiple Interface Configuration Sample for Multinode OpenStack HA and Contrail

This is a configuration sample for a multiple interface, multiple node deployment of high availability OpenStack and Contrail Release 5.0.x. Use this sample to configure parameters specific to your system.

For more information or for recent updates, refer to the github topic [Configuration Sample for Multi Node OpenStack HA and Contrail \(multi interface\)](#).

### Configuration Sample—Multiple Interface



**NOTE:** This example shows host-specific parameters, where interface names are different on each host and are specified under each role. The most specific setting takes precedence. As an example, if there was no `network_interface` setting under the role `openstack` for `bms1`, then it would take the name value `eth2` from the global variable. However, because there is a setting under the `bms1 openstack` section, that `network_interface` name will be `eno1`.

```
provider_config:
  bms:
    ssh_pwd: <Pwd>
    ssh_user: root
    ntpserver: <NTP Server>
    domainsuffix: local
instances:
  bms1:
```

```

    provider: bms
    ip: <BMS1 IP>
    roles:
      openstack:
bms2:
    provider: bms
    ip: <BMS2 IP>
    roles:
      openstack:
bms3:
    provider: bms
    ip: <BMS3 IP>
    roles:
      openstack:
bms4:
    provider: bms
    ip: <BMS4 IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
bms5:
    provider: bms
    ip: <BMS5 IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
bms6:
    provider: bms
    ip: <BMS6 IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:

```

```

    webui:
bms7:
  provider: bms
  ip: <BMS7 IP>
  roles:
    vrouter:
      PHYSICAL_INTERFACE: <Interface name>
      VROUTER_GATEWAY: <Gateway IP>
    openstack_compute:
bms8:
  provider: bms
  ip: <BMS8 IP>
  roles:
    vrouter:
      # Add following line for TSN Compute Node
      TSN_EVPN_MODE: True
    openstack_compute:
contrail_configuration:
  CLOUD_ORCHESTRATOR: openstack
  CONTROL_DATA_NET_LIST: <Control Data Subnet CIDR>
  KEYSTONE_AUTH_URL_VERSION: /v3
  IPFABRIC_SERVICE_HOST: <Service Host IP>
  # Add following line for TSN Compute Node
  TSN_NODES: <TSN NODE IP List>
  # For EVPN VXLAN TSN
  ENCAP_PRIORITY: "VXLAN,MPLSoUDP,MPLSoGRE"
  PHYSICAL_INTERFACE: <Interface name>
kolla_config:
  kolla_globals:
    kolla_internal_vip_address: <Internal VIP>
    kolla_external_vip_address: <External VIP>
    contrail_api_interface_address: <Contrail API IP>
  kolla_passwords:
    keystone_admin_password: <Keystone Admin Password>

```

## Single Interface Configuration Sample for Multinode OpenStack HA and Contrail

This is a configuration sample for a multiple interface, single node deployment of high availability OpenStack and Contrail Release 5.0.x. Use this sample to configure parameters specific to your system.

For more information or for recent updates, refer to the github topic [Configuration Sample for Multi Node OpenStack HA and Contrail \(single interface\)](#).

## Configuration Sample—Single Interface

```

provider_config:
  bms:
    ssh_pwd: <password>
    ssh_user: root
    ntpserver: xx.xx.x.xx
    domainsuffix: local
instances:
  centos1:
    provider: bms
    ip: ip-address
    roles:
      openstack:
  centos2:
    provider: bms
    ip: ip-address
    roles:
      openstack:
  centos3:
    provider: bms
    ip: ip-address
    roles:
      openstack:
  centos4:
    provider: bms
    ip: ip-address
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
  centos5:
    provider: bms
    ip: ip-address
    roles:
      config_database:
      config:
      control:
      analytics_database:

```

```

    analytics:
    webui:
centos6:
  provider: bms
  ip: ip-address
  roles:
    config_database:
    config:
    control:
    analytics_database:
    analytics:
    webui:
centos7:
  provider: bms
  ip: ip-address
  roles:
    vrouter:
    openstack_compute:
centos8:
  provider: bms
  ip: ip-address
  roles:
    vrouter:
    openstack_compute:
contrail_configuration:
  CONTRAIL_VERSION: <contrail_version>
  CONTROLLER_NODES: ip-addresses separated by comma
  CLOUD_ORCHESTRATOR: openstack
  RABBITMQ_NODE_PORT: 5673
  VROUTER_GATEWAY: gateway-ip-address
  PHYSICAL_INTERFACE: eth1
  IPFABRIC_SERVICE_IP: ip-address
  KEYSTONE_AUTH_HOST: ip-address
  KEYSTONE_AUTH_URL_VERSION: /v3
kolla_config:
  kolla_globals:
    kolla_internal_vip_address: ip-address
    contrail_api_interface_address: ip-address
    network_interface: "eth1"
    enable_haproxy: "yes"
  kolla_passwords:
    keystone_admin_password: <password>

```



**NOTE:** Replace `<contrail_version>` with the correct `contrail_container_tag` value for your Contrail release. The respective `contrail_container_tag` values are listed in [README Access to Contrail Registry](#).

## Frequently Asked Questions

This section presents some common error situations and gives guidance on how to resolve the error condition.

### Using Host-Specific Parameters

You might have a situation where you need to specify host-specific parameters, for example, the interface names are different for the different servers in the cluster. In this case, you could specify the individual names under each role, and the more specific setting takes precedence.

For example, if there is no "network\_interface" setting under the role "openstack" for example "bms1", then it will take its setting from the global variable.

An extended example is available at: [Configuration Sample for Multi Node OpenStack HA and Contrail](#).

### Containers from Private Registry Not Accessible

1. You might have a situation in which containers that are pulled from a private registry named `CONTAINER_REGISTRY` are not accessible.
2. To resolve, check to ensure that `REGISTRY_PRIVATE_INSECURE` is set to **True**.

### Error: Failed to insert vrouter kernel module

1. You might have a situation in which the vrouter module is not getting installed on the compute nodes, with the vrouter container in an error state and errors are shown in the Docker logs.

```
[srvr5] ~ # docker logs vrouter_vrouter-kernel-init_1
/bin/cp: cannot create regular file '/host/bin/vif': No such file or directory
INFO: Load kernel module for kver=3.10.0
INFO: Modprobing vrouter /opt/contrail/vrouter-kernel-modules/3.10.0-862.11.6.el7.x86_64/
vrouter.ko
```

	total	used	free	shared	buff/cache	available
Mem:	62G	999M	55G	9.1M	5.9G	60G
Swap:	0B	0B	0B			

```

          total      used      free      shared  buff/cache   available
Mem:      62G        741M        61G         9.1M        923M         61G
Swap:      0B          0B          0B
insmod: ERROR: could not insert module /opt/contrail/vrouter-kernel-modules/
3.10.0-862.11.6.el7.x86_64/vrouter.ko: Unknown symbol in module
ERROR: Failed to insert vrouter kernel module

```

2. In this release, the vrouter module requires the host kernel version to be 3.10.0-862.11.6.el7.x86\_64. To get this kernel version, before running provision, install the kernel version on the target nodes.

```

yum -y install
kernel-3.10.0-862.11.6.el7.x86_64

yum update
reboot

```

## Fatal Error When Vrouter Doesn't Specify OpenStack

1. You might encounter a fatal error when vrouter needs to be provisioned without nova-compute.

```

2018-03-21 00:47:16,884 p=16999 u=root | TASK [iscsi : Ensuring config directories exist]
*****

2018-03-21 00:47:16,959 p=16999 u=root | fatal: [ip-address]: FAILED! => {"msg": "The
conditional check
'inventory_hostname in groups['compute'] or inventory_hostname in groups['storage']'
failed. The error was:
error while evaluating conditional (inventory_hostname in groups['compute'] or
inventory_hostname in
groups['storage']): Unable to look up a name or access an attribute in template string ({%
if
inventory_hostname in groups['compute'] or inventory_hostname in groups['storage'] %) True
{% else %} False
{% endif %}).\nMake sure your variable name does not contain invalid characters like '-':
argument of type
'StrictUndefined' is not iterable\n\nThe error appears to have been in '/root/contrail-
kolla-
ansible/ansible/roles/iscsi/tasks/config.yml': line 2, column 3, but may\nbe elsewhere in
the file depending
on the exact syntax problem.\n\nThe offending line appears to be:\n\n---\n- name: Ensuring

```

```

config
  directories exist\n  ^ here\n"}

2018-03-21 00:47:16,961 p=16999 u=root |          to retry, use: --limit @/root/contrail-
ansible-
  deployer/playbooks/install_contrail.retry

```

2. There is a use case in which vrouter needs to be provisioned without being accompanied by nova-compute. Consequently, the "openstack\_compute" is not automatically inferred when "vrouter" role is specified. To resolve this issue, the "openstack\_compute" role needs to be explicitly stated along with "vrouter".

For more information about this use case, refer to the bug [#1756133](#).

## Need for HAProxy and Virtual IP on a Single OpenStack Cluster

By default, all OpenStack services listen on the IP interface provided by the `kolla_internal_vip_address/network_interface` variables under the `kolla_globals` section in **config/instances.yaml**. In most cases this corresponds to the ctrl-data network, which means that even Horizon will now run only on the ctrl-data network. The only way Kolla provides access to Horizon on the management network is by using HAProxy and keepalived. Enabling keepalived requires a virtual IP for VRRP, and it cannot be the interface IP. There is no way to enable HAProxy without enabling keepalived when using Kolla configuration parameters. For this reason, you need to provide two virtual IP addresses: one on management (`kolla_external_vip_address`) and one on ctrl-data-network (`kolla_internal_vip_address`). With this configuration, Horizon will be accessible on the management network by means of the `kolla_external_vip_address`.

## Using the kolla\_toolbox Container to Run OpenStack Commands

The directory `/etc/kolla/kolla-toolbox` on the base host on which OpenStack containers are running is mounted and accessible as `/var/lib/kolla/config_files` from inside the `kolla_toolbox` container. If you need other files when executing OpenStack commands, for example the command `openstack image create` needs an image file, you can copy the relevant files into the `/etc/kolla/kolla-toolbox` directory of the base host and use them inside the container.

The following example shows how to run OpenStack commands in this way:

```

# ON BASE HOST OF OPENSTACK CONTROL NODE
cd /etc/kolla/kolla-toolbox
wget http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img

docker exec -it kolla_toolbox bash

```



```
# NOW YOU ARE INSIDE THE KOLLA_TOOLBOX CONTAINER
(kolla-toolbox)[ansible@server1 /]$ source /var/lib/kolla/config_files/admin-openrc.sh
(kolla-toolbox)[ansible@server1 /]$ cd /var/lib/kolla/config_files
(kolla-toolbox)[ansible@server1 /var/lib/kolla/config_files]$ openstack image create cirros2
--disk-format qcow2 --public --container-format bare --file cirros-0.4.0-x86_64-disk.img
```

Field	Value
checksum	443b7623e27ecf03dc9e01ee93f67afe
container_format	bare
created_at	2018-03-29T21:37:48Z
disk_format	qcow2
file	/v2/images/e672b536-0796-47b3-83a6-df48a5d074be/file
id	e672b536-0796-47b3-83a6-df48a5d074be
min_disk	0
min_ram	0
name	cirros2
owner	371bdb766278484bbabf868cf7325d4c
protected	False
schema	/v2/schemas/image
size	12716032
status	active
tags	
updated_at	2018-03-29T21:37:50Z
virtual_size	None
visibility	public

```
(kolla-toolbox)[ansible@server1 /var/lib/kolla/config_files]$ openstack image list
```

ID	Name	Status
e672b536-0796-47b3-83a6-df48a5d074be	cirros2	active
57e6620e-796a-40ee-ae6e-ea1daa253b6c	cirros2	active

## RELATED DOCUMENTATION

*Installing Contrail Cluster using Contrail Command and instances.yml*

## 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.

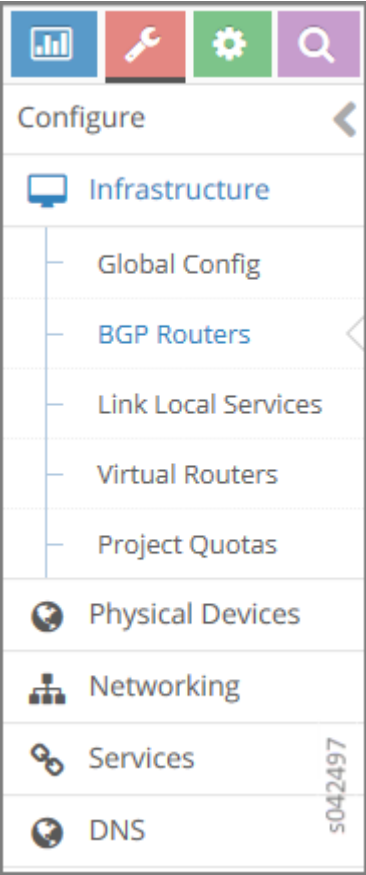
Before you begin, ensure that the following tasks are completed:

- 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:8143>, 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:8143>), select **Configure > Infrastructure > BGP Routers**; see [Figure 3 on page 35](#).

Figure 3: Configure> Infrastructure > BGP Routers



A summary screen of the control nodes and BGP routers is displayed; see [Figure 4 on page 35](#).

Figure 4: 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				


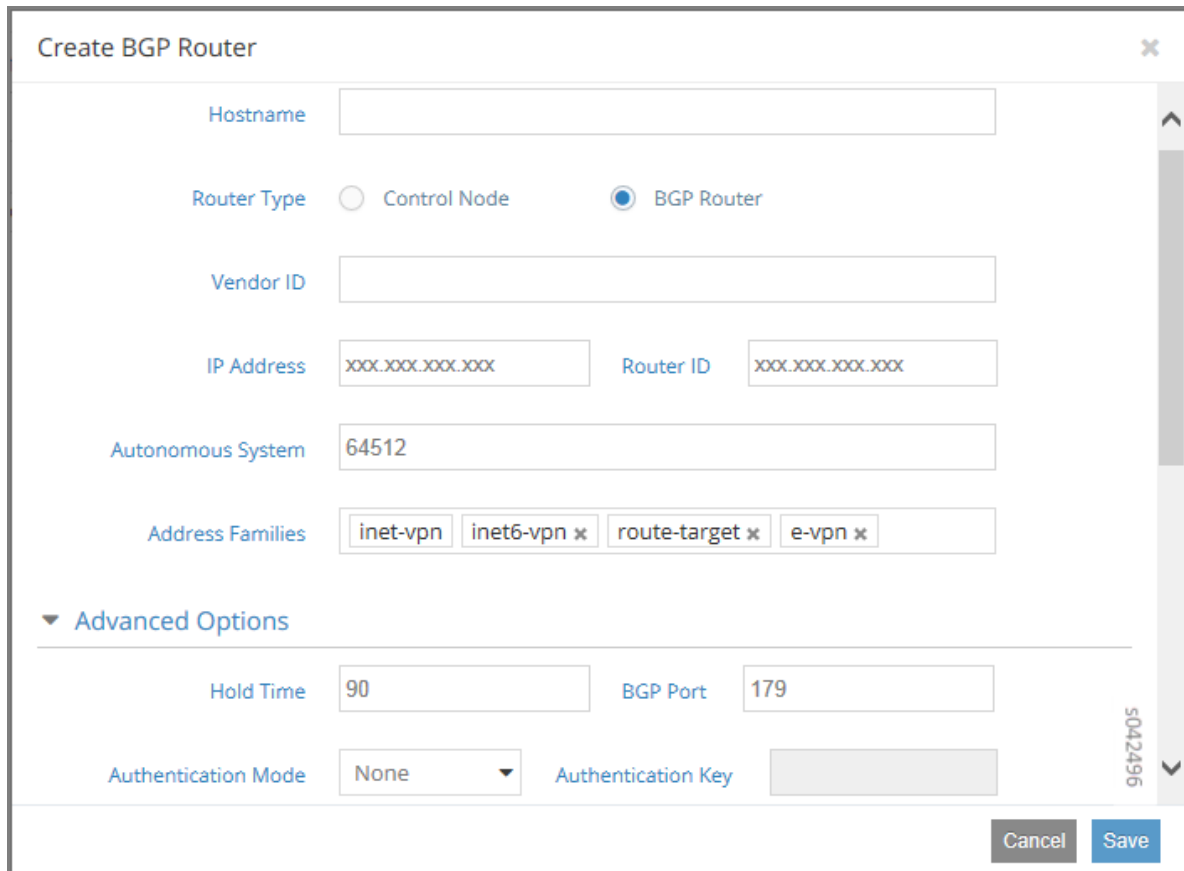
2. (Optional) The global AS number is 64512 by default. To change the AS number, on the **BGP Router** summary screen click the gear wheel and select **Edit**. In the Edit BGP Router window enter the new number.
3. To create control nodes and BGP routers, on the **BGP Routers** summary screen, click the  icon. The **Create BGP Router** window is displayed; see [Figure 5 on page 36](#).

Figure 5: 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 1 on page 37](#).

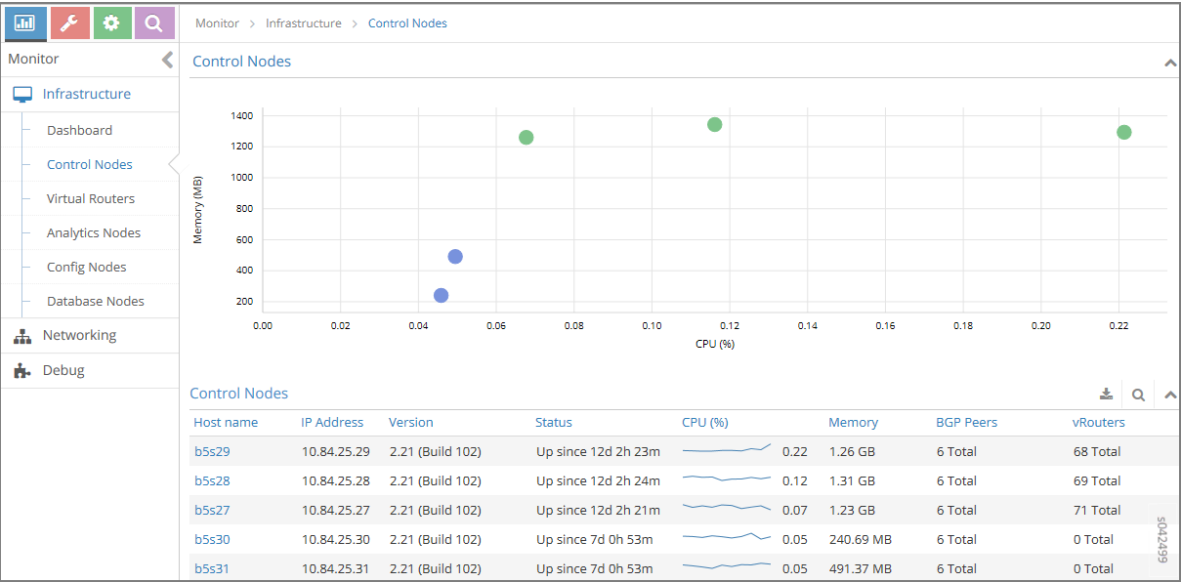
Table 1: Create BGP Router Fields

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.

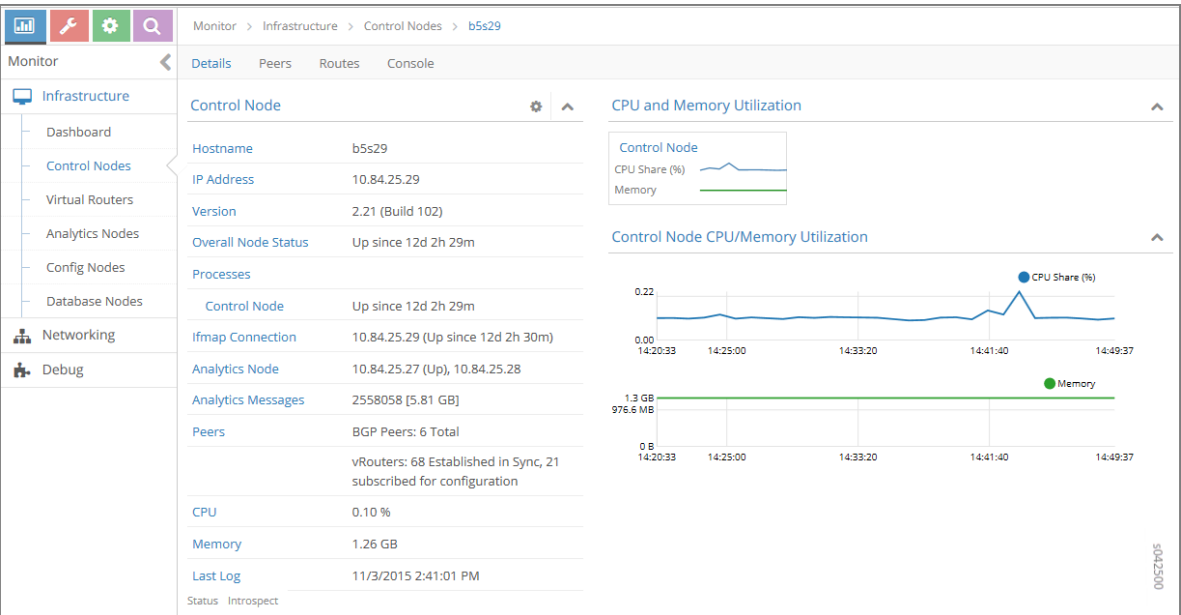
7. You can check for peers by selecting **Monitor > Infrastructure > Control Nodes**; see [Figure 6 on page 38](#).

Figure 6: Control Nodes



In the **Control Nodes** window, click any hostname in the memory map to view its details; see [Figure 7 on page 38](#).

Figure 7: Control Node Details



8. Click the **Peers** tab to view the peers of a control node; see [Figure 8 on page 39](#).

Figure 8: Control Node Peers Tab

Monitor > Infrastructure > Control Nodes > b5s29

DetailsPeersRoutesConsole

Peers

DownloadSearchExpand

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",

```

5042501

RELATED DOCUMENTATION

- Creating a Virtual Network with Juniper Networks Contrail
- Creating a Virtual Network with OpenStack Contrail

Contrail Global Controller

IN THIS SECTION

- Resource Identifier Management | 40
- Multiple Location Resource Provisioning | 40

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 later

## 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 | 41](#)
- [API-Level Access Control | 42](#)
- [Object Level Access Control | 43](#)
- [Configuration | 43](#)
- [Upgrading from Previous Releases | 46](#)
- [Configuring RBAC Using the Contrail User Interface | 46](#)
- [RBAC Resources | 49](#)

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

Contrail Release 3.0 and later supports 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
```

## 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.

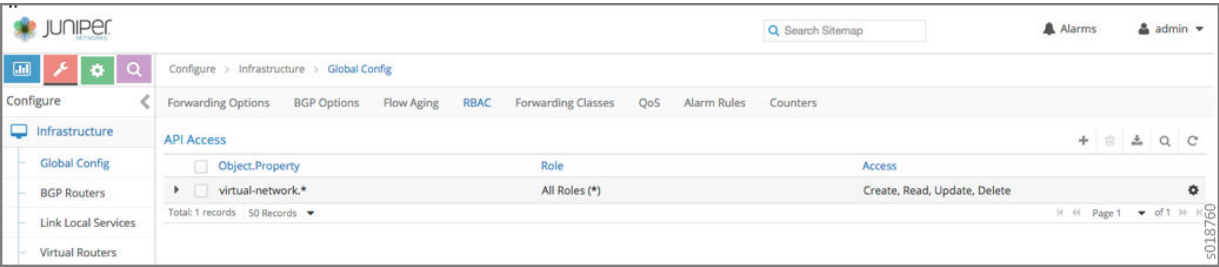
```
service contrail-analytics-api restart
```

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 9 on page 47](#).

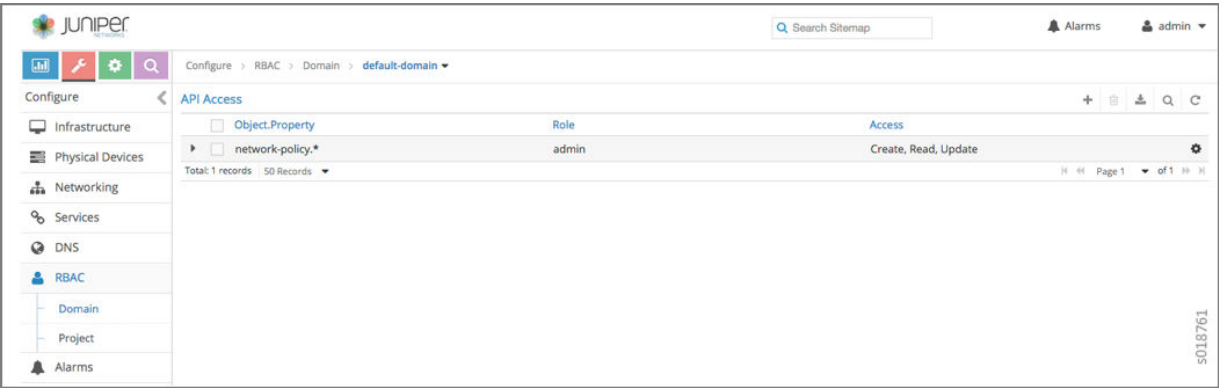
Figure 9: RBAC Global Level



Configuring RBAC at the Domain Level

To configure RBAC at the domain level, navigate to **Configure > RBAC > Domain**, see [Figure 10 on page 47](#).

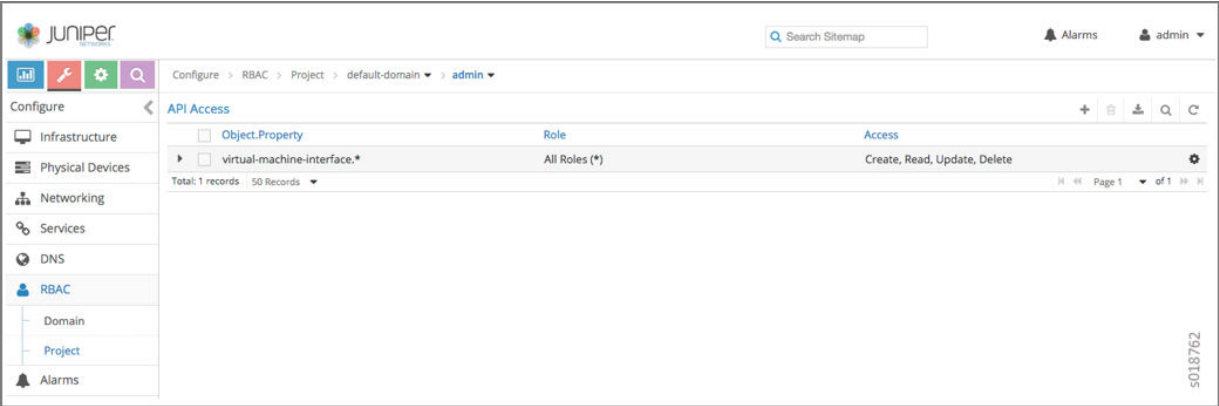
Figure 10: RBAC Domain Level



Configuring RBAC at the Project Level

To configure RBAC at the project level, navigate to **Configure > RBAC > Project**, see [Figure 11 on page 48](#).

Figure 11: 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 12 on page 48](#).

Figure 12: RBAC Details API Access

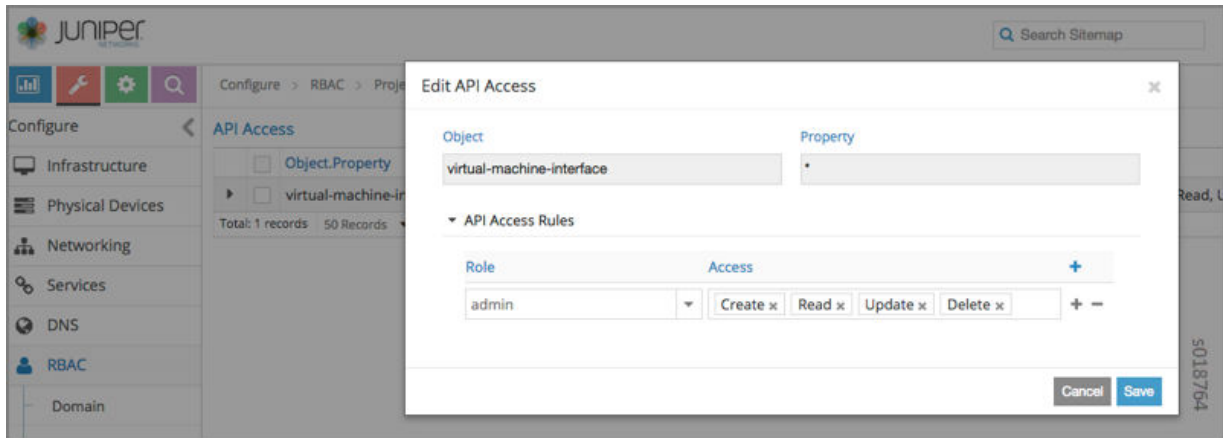


Creating or Editing API Level Access

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 13 on page 49](#).



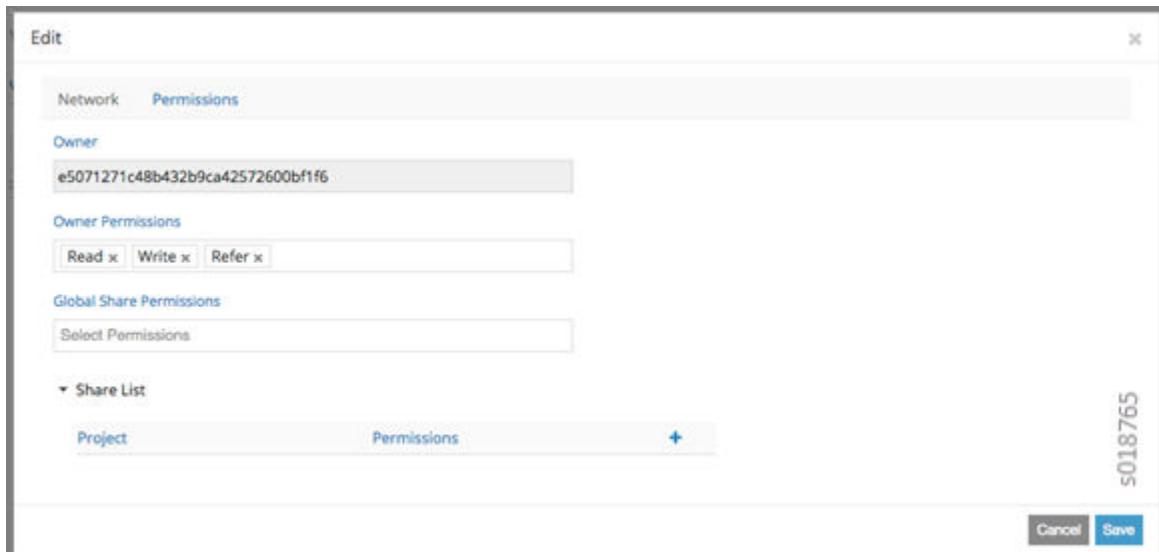
Figure 13: Edit API Access



## 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 14 on page 49](#).

Figure 14: 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 | 51](#)
- [Configuring MD5 Authentication for BGP Sessions | 61](#)
- [Configuring the Data Plane Development Kit \(DPDK\) Integrated with Contrail vRouter | 62](#)
- [Configuring Contrail DPDK vRouter to Run in a Docker Container | 65](#)
- [Configuring Single Root I/O Virtualization \(SR-IOV\) | 66](#)
- [Configuring Virtual Networks for Hub-and-Spoke Topology | 73](#)
- [Configuring Transport Layer Security-Based XMPP in Contrail | 80](#)
- [Configuring Graceful Restart and Long-lived Graceful Restart | 83](#)
- [Remote Compute | 88](#)
- [Dynamic Kernel Module Support \(DKMS\) for vRouter | 97](#)

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

## IN THIS SECTION

- [Introduction to the Simple Gateway | 52](#)
- [How the Simple Gateway Works | 52](#)
- [Setup Without Simple Gateway | 52](#)
- [Setup With a Simple Gateway | 53](#)
- [Simple Gateway Configuration Features | 54](#)
- [Packet Flows with the Simple Gateway | 55](#)
- [Packet Flow Process From the Virtual Network to the Public Network | 56](#)
- [Packet Flow Process From the Public Network to the Virtual Network | 56](#)
- [Methods for Configuring the Simple Gateway | 57](#)

- [Using the vRouter Configuration File to Configure the Simple Gateway | 57](#)
- [Using Thrift Messages to Dynamically Configure the Simple Gateway | 57](#)
- [Common Issues with Simple Gateway Configuration | 60](#)

## 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

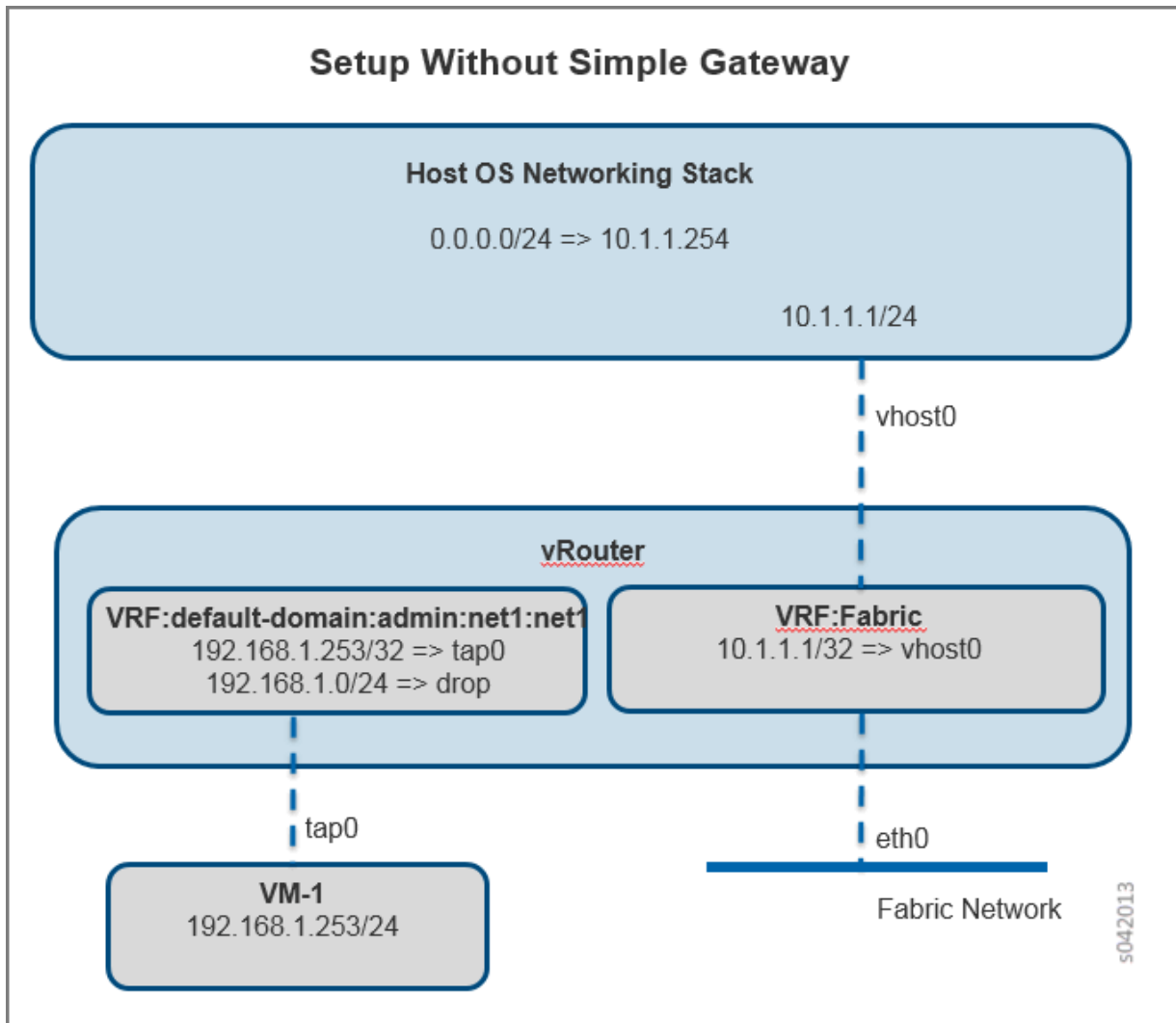
The following sections illustrate how the simple gateway works, first, by showing a virtual network setup with no simple gateway, then illustrating the same setup with a simple gateway configured.

### 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

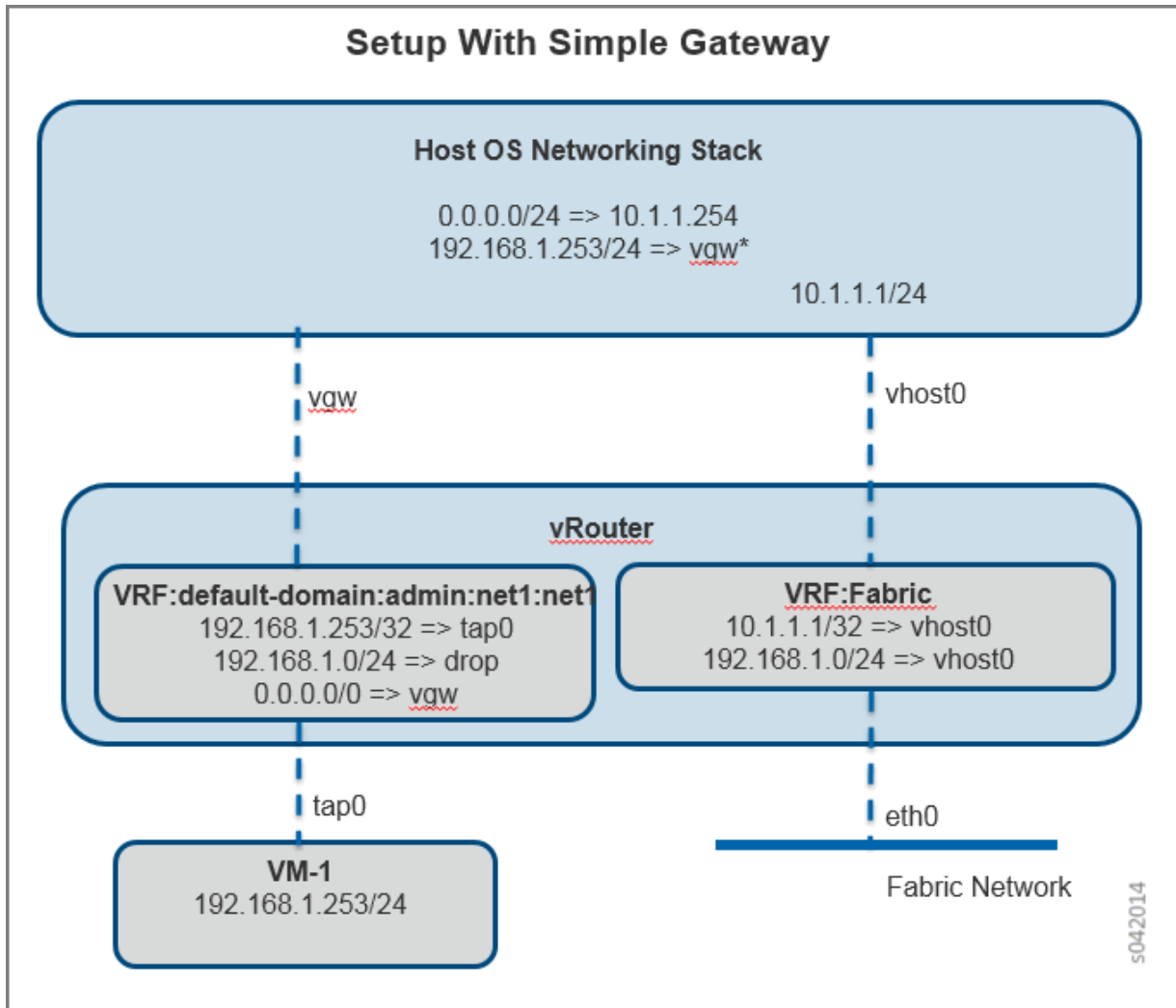
Figure 15 on page 54 shows a virtual network setup with the simple gateway configured for the `default-domain:admin:net1` virtual network.

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 15 on page 54 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 15: 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 | 58](#)
- [How to Dynamically Delete a Virtual Gateway | 59](#)
- [Using Devstack to Configure the Simple Gateway | 59](#)

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`

## 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 16 on page 62](#), a list of BGP peers is displayed.

Figure 16: 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 | 63](#)
- [Preparing the Environment File for Provisioning a Cluster Node with DPDK | 63](#)

## 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 Release 5.0, the DPDK configuration should be defined in `instances.yaml` for ansible based provision, or in `host.yaml` for helm-based provision. The `AGENT_MODE` configuration specifies whether the hypervisor is provisioned in the DPDK mode of operation.

When a Contrail compute node is provisioned with DPDK, the corresponding yaml file specifies the number of CPU cores to use for forwarding packets, the number of huge pages to allocate for DPDK, and the uio driver to use 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.

- **CPU\_CORE\_MASK**—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”.

The supported format is hexadecimal (for example, 0xf).

- **DPDK\_UIO\_DRIVER**—Specify the UIO driver to be used with DPDK.

The supported values include:

- `vfio-pci`—Specify that the `vfio` module in the Linux kernel should be used instead of `uio`. The `vfio` module protects memory access 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 built into the Linux kernel should be used. This option does not support the use of SR-IOV VFs. This is the default option if `DPDK_UIO_DRIVER` is not specified.
- `mlx` – For Mellanox ConnectX-5 NICs. This is available starting from Contrail release 5.0.1.



**NOTE:** For RHEL and Intel x710 Fortville-based NIC, use `vfio-pci` instead of the default `uio_pci_generic`.

Use the standard Ansible or helm-based provision procedure. Upon completion, your cluster, with specified nodes using the DPDK vRouter implementation, is ready to use.

#### Sample configuration in `instances.yml` for ansible-based provision

```
Bms1:
  provider: bms
  ip: ip-address
  roles:
    vrouter:
      AGENT_MODE: dpdk
      CPU_CORE_MASK: "0xff"
      DPDK_UIO_DRIVER: uio_pci_generic
      HUGE_PAGES: 32000
```

#### Sample configuration in `host.yml` for helm-based provision

```
...
AGENT_MODE: dpdk
CPU_CORE_MASK: "0xff"
DPDK_UIO_DRIVER: uio_pci_generic
HUGE_PAGES: 32000
```



## Creating a Flavor for DPDK

To launch a VM in a DPDK-enabled vRouter hypervisor, the flavor for the VM should be set to 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.

```
# openstack flavor set m1.large --property 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.



**NOTE:** Note: By default, 2MB huge pages are provisioned. If 1GB huge page is required, it must be done by the Administrator.

## RELATED DOCUMENTATION

[Configuring Single Root I/O Virtualization \(SR-IOV\) | 66](#)

<http://www.dpdk.org>

## Configuring Contrail DPDK vRouter to Run in a Docker Container

Starting with Contrail Release 5.0, you can configure the Contrail DPDK vRouter to run in a Docker container. In earlier releases, DPDK vRouter runs on a compute host. The `contrail-vrouter-dpdk` binary file provides data plane functionality when Contrail vRouter runs in DPDK mode in a Contrail cluster.

Complete these steps to configure Contrail DPDK vRouter to run in a Docker container.

1. Run the following commands on the compute host:

```
edit /etc/sysctl.conf and add below parameters a.vm.nr_hugepages = 48341 b.vm.max_map_count = 96682 c.kernel.core_pattern = /var/crashes/core.%e.%p.%h.%t
```

```
mkdir -p /hugepages
echo "hugetlbfs /hugepages hugetlbfs defaults 0 0 " >> /etc/fstab
4.sudo mount -t hugetlbfs
hugetlbfs /hugepages
```

2. Perform the following steps on the Docker container:

- a. Install pciutils.
- b. Install the following packages in the container:
  - contrail-vrouter-dpdk-init
  - contrail-vrouter-dpdk
  - contrail-vrouter-utils
  - python-opencontrail-vrouter-netns
  - python-contrail-vrouter-api

3. Run the following commands:

```
/opt/contrail/bin/dpdk_nic_bind.py -b ixgbe 0000:02:00.0
/opt/contrail/bin/dpdk_nic_bind.py -s
taskset 0xf /usr/bin/contrail-vrouter-dpdk --no-daemon
```

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

### IN THIS SECTION

- [Overview: Configuring SR-IOV | 67](#)
- [Enabling ASPM in BIOS | 67](#)
- [Configuring SR-IOV Using the Ansible Deployer | 67](#)
- [Configuring SR-IOV using Helm | 69](#)
- [Launching SR-IOV Virtual Machines | 71](#)

## Overview: Configuring SR-IOV

Contrail Release 3.0 through Release 4.0 supports single root I/O virtualization (SR-IOV) on Ubuntu systems only. With Release 4.1, Contrail supports SR-IOV on Red Hat Enterprise Linux (RHEL) operating systems as well.

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.

## 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 Using the Ansible Deployer

You must perform the following tasks to enable SR-IOV on a system.

1. Enable the Intel Input/Output Memory Management Unit (IOMMU) on Linux.
2. Enable the required number of Virtual Functions (VFs) on the selected NIC.
3. Configure the names of the physical networks whose VMs can interface with the VFs.
4. Reboot Nova compute.

```
service nova-compute restart
```

5. 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
```

The above tasks are handled by the Ansible Deployer playbook. The cluster members and its configuration parameters are specified in the **instances.yaml** file located in the config directory within the ansible-deployer repository.

The compute instances that are going to be in SR-IOV mode should have an SR-IOV configuration. The **instance.yaml** snippet below shows a sample instance definition.

```

instances:
  bms1:
    provider: bms
    ip: ip-address
    roles:
      openstack:
  bms2:
    provider: bms
    ip: ip-address
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
  bms3:
    provider: bms
    ip: ip-address
    roles:
      openstack_compute:
      vrouter:
        SRIOV: true
        SRIOV_VF: 3

```

```
SRIOV_PHYSICAL_INTERFACE: eno1
SRIOV_PHYS_NET: physnet1
```

## Configuring SR-IOV using Helm

You must perform the following tasks to enable SR-IOV on a system.

1. Enable the Intel Input/Output Memory Management Unit (IOMMU) on Linux.
2. Enable the required number of Virtual Functions (VFs) on the selected NIC.
3. Configure the names of the physical networks whose VMs can interface with the VFs.
4. Reboot Nova compute.

```
service nova-compute restart
```

5. 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
```

The above tasks are handled by the Helm charts. The cluster members and its configuration parameters are specified in the **multinode-inventory** file located in the config directory within the openstack-helm-infra repository.

For Helm, the configuration and SR-IOV environment-specific parameters must be updated in three different places:

- The compute instance must be set as contrail-vrouter-sriov.

For example, the following is a snippet from the **tools/gate/devel/multinode-inventory.yaml** file in the openstack-helm-infra repository.

```
all:
  children:
    primary:
      hosts:
        node1:
          ansible_port: 22
          ansible_host: host-ip-address
          ansible_user: ubuntu
          ansible_ssh_private_key_file: /home/ubuntu/.ssh/insecure.pem
          ansible_ssh_extra_args: -o StrictHostKeyChecking=no
  nodes:
    children:
      openstack-compute:
        children:
          contrail-vrouter-sriov: #compute instance set to contrail-vrouter-sriov
            hosts:
              node7:
                ansible_port: 22
                ansible_host: host-ip-address
                ansible_user: ubuntu
                ansible_ssh_private_key_file: /home/ubuntu/.ssh/insecure.pem
                ansible_ssh_extra_args: -o StrictHostKeyChecking=no
```

- Contrail-vrouter-sriov must be labeled appropriately.

For example, the following is a snippet from the **tools/gate/devel/multinode-vars.yaml** in the openstack-helm-infra repository.

```
nodes:
  labels:
    primary:
      - name: openstack-helm-node-class
        value: primary

    all:
      - name: openstack-helm-node-class
        value: general
  contrail-controller:
```

```

- name: opencontrail.org/controller
  value: enabled
openstack-compute:
- name: openstack-compute-node
  value: enabled
contrail-vrouter-dpdk:
- name: opencontrail.org/vrouter-dpdk
  value: enabled
contrail-vrouter-sriov: # label as contrail-vrouter-sriov
- name: vrouter-sriov
  value: enabled

```

- SR-IOV config parameters must be updated in the **contrail-vrouter/values.yaml** file.

For example, the following is a snippet from the **contrail-vrouter/values.yaml** file in the contrail-helm-deployer repository.

```

contrail_env_vrouter_kernel:
  AGENT_MODE: kernel

contrail_env_vrouter_sriov:
  SRIOV: true
  per_compute_info:
    node_name: k8snode1
    SRIOV_VF: 10
    SRIOV_PHYSICAL_INTERFACE: enp129s0f1
    SRIOV_PHYS_NET: physnet1

```

## 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 17 on page 72](#).

Figure 17: 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 18 on page 72](#).

Figure 18: Create Port

Create

×

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

S018551

Cancel Save

- 3. 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:

1. 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
```

2. Create a subnet in `vn1`.

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

3. 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
```

4. 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>
```

5. 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](#) | 62

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

### IN THIS SECTION

- [Route Targets for Virtual Networks in Hub-and-Spoke Topology](#) | 74
- [Example: Configuring Hub-and-Spoke Virtual Networks](#) | 74

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/utlils# 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"
        }
      }
    ]
  }
}

```

```

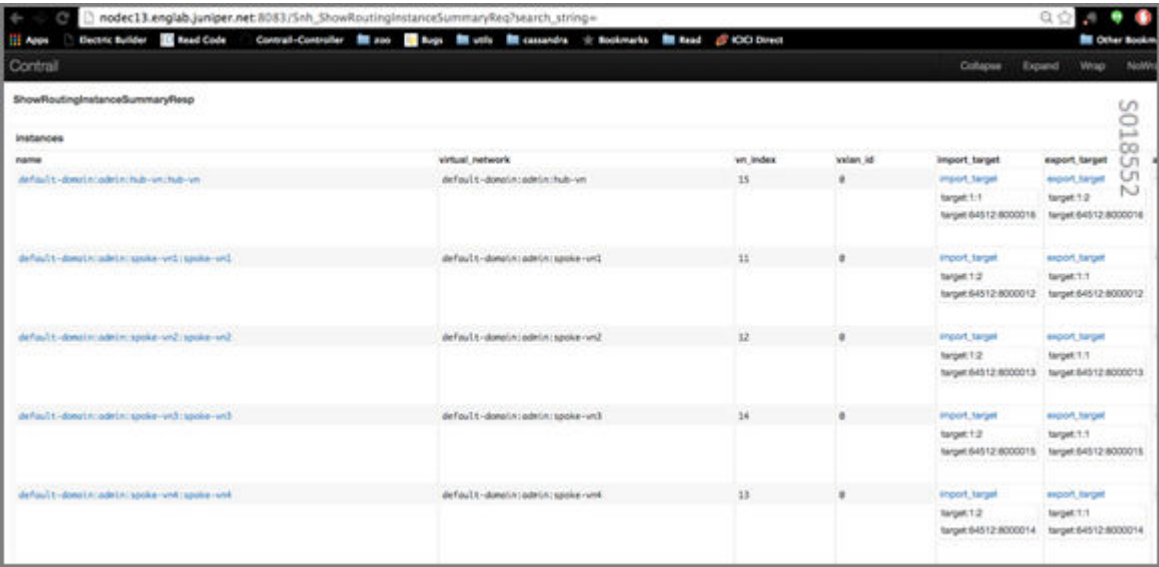
        },
        "href": "http://<host ip>:8095/route-target/446a3bbe-f263-4b58-
a537-8333878dd7c3",
        "to": [
            "target:1:1"
        ],
        "uuid": "446a3bbe-f263-4b58-a537-8333878dd7c3"
    },
    {
        "attr": {
            "import_export": "export"
        },
        "href": "http://<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 19 on page 80 shows the import and export targets for hub-vn and spoke-vns, by invoking `contrail-control-introspect`.

Figure 19: Contrail Introspect



The screenshot shows the Contrail Introspect web interface. The browser address bar displays 'nodec13.englab.juniper.net:8083/5nh\_ShowRoutingInstanceSummaryReq?search\_string='. The page title is 'Contrail'. Below the title, there's a 'ShowRoutingInstanceSummaryResp' section. A table lists routing instances with columns: name, virtual\_network, vn\_index, vxlan\_id, import\_target, and export\_target. The table contains five rows of data. On the right side of the table, there is a vertical text '5018552'.

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

## Configuring Transport Layer Security-Based XMPP in Contrail

### IN THIS SECTION

- Overview: TLS-Based XMPP | 80
- Configuring XMPP Client and Server in Contrail | 81

### 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.  <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	/etc/contrail/ssl/certs/server.pem
xmpp_server_key	Path to server's/node's private key	/etc/contrail/ssl/certs/server-privkey.pem
xmpp_ca_cert	Path to CA certificate	/etc/contrail/ssl/certs/ca-cert.pem
xmpp_dns_auth_enable=true	Enables SSL based XMPP	Default is set to false, XMPP is disabled.  <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 xmpp_dns_auth_enable=true	Enables SSL based XMPP	Default is set to false, XMPP is disabled.  <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 | 84](#)
- [BGP Graceful Restart Helper Mode | 84](#)
- [Feature Highlights | 84](#)
- [XMPP Helper Mode | 85](#)
- [Configuration Parameters | 85](#)
- [Cautions for Graceful Restart | 87](#)
- [Configuring Graceful Restart with the Contrail User Interface | 87](#)

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.

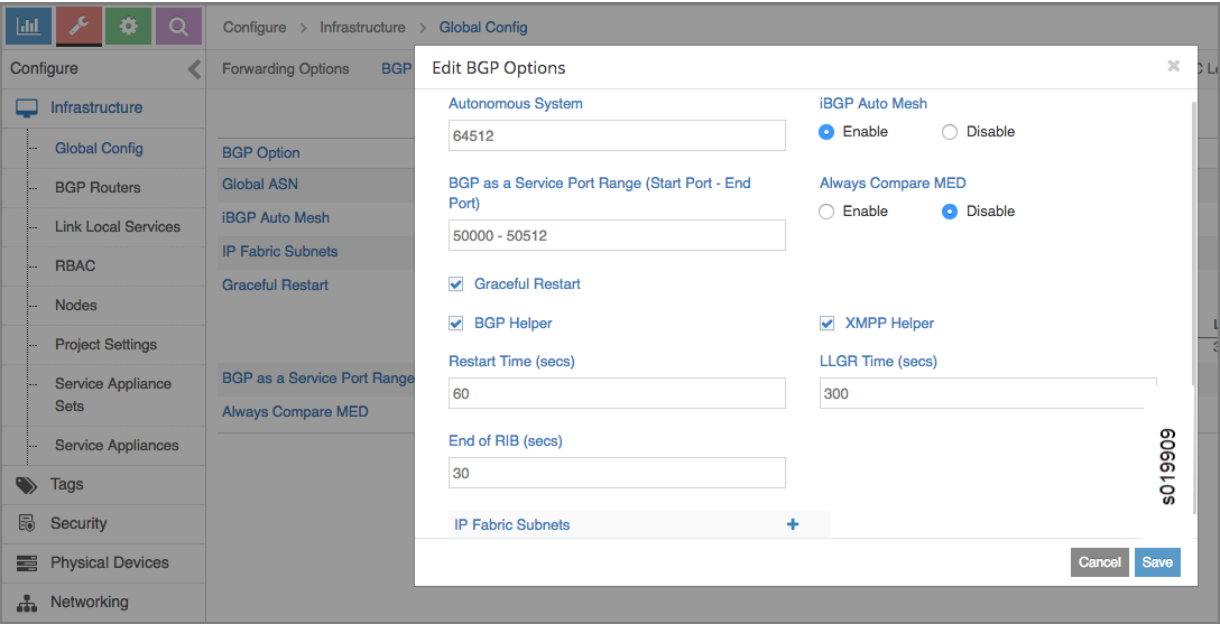
- Using the graceful restart/long-lived graceful restart feature with a peer is effective either to all negotiated address families or to none. If a peer signals support for graceful restart/long-lived graceful restart for only a subset of the negotiated address families, the graceful restart helper 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 20 on page 88](#).

Figure 20: Configuring Graceful Restart



## Remote Compute

### IN THIS SECTION

- [Remote Compute Overview | 89](#)
- [Remote Compute Features | 89](#)
- [Provisioning a Remote Compute Cluster | 90](#)

Contrail Release 5.0.1 introduces remote compute, a method of managing a Contrail deployment across many small distributed data centers efficiently and cost effectively.



## Remote Compute Overview

The purpose of remote compute is to enable the deployment of Contrail in many small distributed data centers, up to hundreds or even thousands, for telecommunications point-of-presence (PoPs) or central offices (COs). Each small data center has only a small number of computes, typically 5-20 in a rack, running a few applications such as video caching, traffic optimization, and virtual Broadband Network Gateway (vBNG). It is not cost effective to deploy a full Contrail controller cluster of nodes of control, configuration, analytics, database, and the like, in each distributed PoP on dedicated servers. Additionally, manually managing hundreds or thousands of clusters is not feasible operationally.

## Remote Compute Features

Remote compute is implemented by means of a subcluster that manages compute nodes at remote sites to receive configurations and exchange routes.

The key concepts of Contrail remote compute include:

- Remote compute employs a subcluster to manage remote compute nodes away from the primary data center.
- The Contrail control cluster is deployed in large centralized data centers, where it can remotely manage compute nodes in small distributed small data centers.
- A lightweight version of the controller is created, limited to the control node, and the config node, analytics, and analytics database are shared across several control nodes.
- Many lightweight controllers are co-located on a small number of servers to optimize efficiency and cost.
- The control nodes peer with the remote compute nodes by means of XMPP and peer with local gateways by means of MP-eBGP.

## Remote Compute Operations

A subcluster object is created for each remote site, with a list of links to local compute nodes that are represented as vrouter objects, and a list of links to local control nodes that are represented as BGP router objects, with an ASN as property.

The subclusters are identified in the provision script. The vrouter and bgp-router provision scripts take each subcluster as an optional argument to link or delink with the subcluster object.

It is recommended to spawn the control nodes of the remote cluster in the primary cluster, and they are IGBP-meshed among themselves within that subcluster. The control nodes BGP-peer with their respective SDN gateway, over which route exchange occurs with the primary control nodes.

Compute nodes in the remote site are provisioned to connect to their respective control nodes to receive configuration and exchange routes. Data communication among workloads between these clusters occurs through the provider backbone and their respective SDN gateways. The compute nodes and the control nodes push analytics data to analytics nodes hosted on the primary cluster.

## Subcluster Properties

The Contrail UI shows a list of subcluster objects, each with a list of associated vrouters and BGP routers that are local in that remote site and the ASN property.

General properties of subclusters include:

- A subcluster control node never directly peers with another subcluster control node or with primary control nodes.
- A subcluster control node has to be created, and is referred to, in virtual-router and bgp-router objects.
- A subcluster object and the control nodes under it should have the same ASN.
- The ASN cannot be modified in a subcluster object.

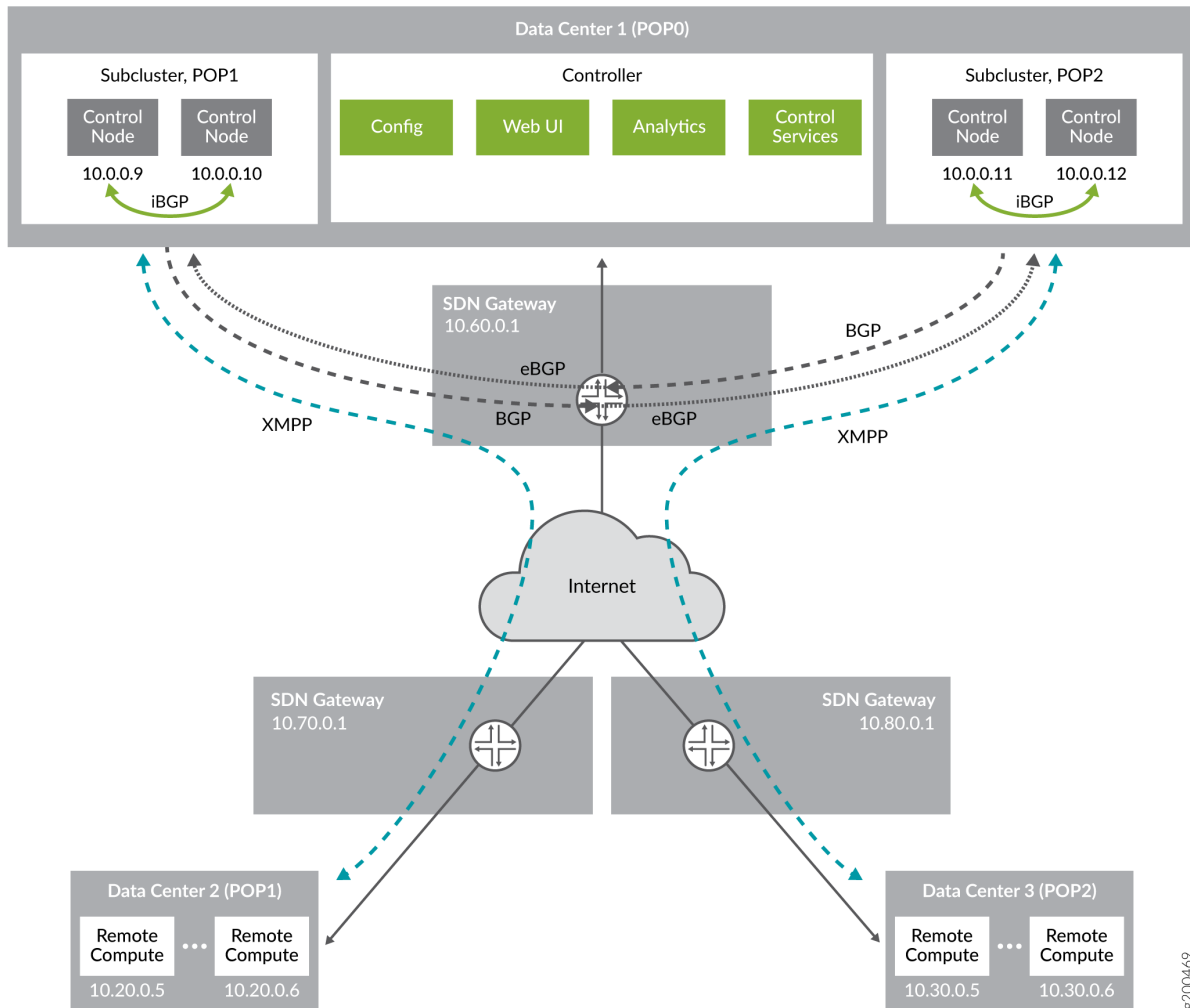


**NOTE:** Multinode service chaining across subclusters is not supported.

## Provisioning a Remote Compute Cluster

With Contrail Release 5.0.1, you provision for remote compute using an `instances.yaml` file. *Deploying Contrail Cluster using Contrail-Command and instances.yml* shows a bare minimum configuration. The YAML file described in this section builds upon that minimum configuration and uses [Figure 21 on page 91](#) as an example data center network topology.

### Figure 21: Example Multi-Cluster Topology



In this topology, there is one main data center (**pop0**) and two remote data centers (**pop1** and **pop2**.) **pop0** contains two subclusters: one for **pop1**, and the other for **pop2**. Each subcluster has two control nodes. The control nodes within a subcluster, for example 10.0.0.9 and 10.0.0.10, communicate with each other through iBGP.

Communication between the control nodes within a subcluster and the remote data center is through the SDN Gateway; there is no direct connection. For example, the remote compute in pop1 (IP address 10.20.0.5) communicates with the control nodes (IP addresses 10.0.0.9 and 10.0.0.10) in subcluster 1 through the SDN Gateway.

To configure remote compute in the YAML file:

1. First, create the remote locations or subclusters. In this example, we create data centers 2 and 3 (with the names **pop1** and **pop2**, respectively), and define unique ASN numbers for each. Subcluster names must also be unique.

```
remote_locations:
  pop1:
    BGP_ASN: 12345
    SUBCLUSTER: pop1
  pop2:
    BGP_ASN: 12346
    SUBCLUSTER: pop2
```

2. Create the control nodes for pop1 and pop2 and assign an IP address and role. These IP addresses are the local IP address. In this example, there are two control nodes for each sub-cluster.

```
control_1_only_pop1:      # Mandatory. Instance name
  provider: bms           # Mandatory. Instance runs on BMS
  ip: 10.0.0.9
  roles:
    control:
      location: pop1
control_2_only_pop1:      # Mandatory. Instance name
  provider: bms           # Mandatory. Instance runs on BMS
  ip: 10.0.0.10
  roles:
    control:
      location: pop1
control_1_only_pop2:      # Mandatory. Instance name
  provider: bms           # Mandatory. Instance runs on BMS
  ip: 10.0.0.11
  roles:                  # Optional.
    control:
      location: pop2
control_2_only_pop2:      # Mandatory. Instance name
  provider: bms           # Mandatory. Instance runs on BMS
  ip: 10.0.0.12
  roles:                  # Optional.
    control:
      location: pop2
```

3. Now, create the remote compute nodes for **pop1** and **pop2** and assign an IP address and role. In this example, there are two remote compute nodes for each data center. The 10.60.0.x addresses are the management IP addresses for the control service.

```
compute_1_pop1:          # Mandatory. Instance name
  provider: bms           # Mandatory. Instance runs on BMS
  ip: 10.20.0.5
  roles:
    openstack_compute:    # Optional.
  vrouters:
    CONTROL_NODES: 10.60.0.9,10.60.0.10
    VROUTER_GATEWAY: 10.70.0.1
  location: pop1
compute_2_pop1:          # Mandatory. Instance name
  provider: bms           # Mandatory. Instance runs on BMS
  ip: 10.20.0.6
  roles:
    openstack_compute:    # Optional.
  vrouters:
    CONTROL_NODES: 10.60.0.9,10.60.0.10
    VROUTER_GATEWAY: 10.70.0.1
  location: pop1
compute_1_pop2:          # Mandatory. Instance name
  provider: bms           # Mandatory. Instance runs on BMS
  ip: 10.30.0.5
  roles:
    openstack_compute:    # Optional.
  vrouters:
    CONTROL_NODES: 10.60.0.11,10.60.0.12
    VROUTER_GATEWAY: 10.80.0.1
  location: pop2
compute_2_pop2:          # Mandatory. Instance name
  provider: bms           # Mandatory. Instance runs on BMS
  ip: 10.30.0.6
  roles:
    openstack_compute:    # Optional.
  vrouters:
    CONTROL_NODES: 10.60.0.11,10.60.0.12
    VROUTER_GATEWAY: 10.80.0.1
  location: pop2
```

The entire YAML file is contained below.

### Example instance.yaml with sub-cluster configuration

```

provider_config:
  bms:
    ssh_pwd: <password>
    ssh_user: <root_user>
    ntpserver: 10.84.5.100
    domainsuffix: local
instances:
  openstack_node:          # Mandatory. Instance name
    provider: bms          # Mandatory. Instance runs on BMS
    ip: 10.0.0.4
    roles:                  # Optional.
      openstack:
  all_contrail_roles_default_pop: # Mandatory. Instance name
    provider: bms          # Mandatory. Instance runs on BMS
    ip: 10.0.0.5
    roles:                  # Optional.
      config_database:     # Optional.
      config:              # Optional.
      control:             # Optional.
      analytics_database:   # Optional.
      analytics:           # Optional.
      webui:               # Optional.
  compute_3_default_pop:    # Mandatory. Instance name
    provider: bms          # Mandatory. Instance runs on BMS
    ip: 10.0.0.6
    roles:
      openstack_compute:
      vrouter:
        VROUTER_GATEWAY: 10.60.0.1
  compute_1_default_pop:    # Mandatory. Instance name
    provider: bms          # Mandatory. Instance runs on BMS
    ip: 10.0.0.7
    roles:
      openstack_compute:
      vrouter:
        VROUTER_GATEWAY: 10.60.0.1
  compute_2_default_pop:    # Mandatory. Instance name
    provider: bms          # Mandatory. Instance runs on BMS
    ip: 10.0.0.8
    roles:

```

```

    openstack_compute:
    vrouter:
        VRROUTER_GATEWAY: 10.60.0.1
control_1_only_pop1:      # Mandatory. Instance name
    provider: bms          # Mandatory. Instance runs on BMS
    ip: 10.0.0.9
    roles:
        control:
            location: pop1
control_2_only_pop1:      # Mandatory. Instance name
    provider: bms          # Mandatory. Instance runs on BMS
    ip: 10.0.0.10
    roles:
        control:
            location: pop1
control_1_only_pop2:      # Mandatory. Instance name
    provider: bms          # Mandatory. Instance runs on BMS
    ip: 10.0.0.11
    roles:                  # Optional.
        control:
            location: pop2
control_2_only_pop2:      # Mandatory. Instance name
    provider: bms          # Mandatory. Instance runs on BMS
    ip: 10.0.0.12
    roles:                  # Optional.
        control:
            location: pop2
compute_1_pop1:           # Mandatory. Instance name
    provider: bms          # Mandatory. Instance runs on BMS
    ip: 10.20.0.5
    roles:
        openstack_compute: # Optional.
        vrouter:
            CONTROL_NODES: 10.60.0.9,10.60.0.10
            VRROUTER_GATEWAY: 10.70.0.1
            location: pop1
compute_2_pop1:           # Mandatory. Instance name
    provider: bms          # Mandatory. Instance runs on BMS
    ip: 10.20.0.6
    roles:
        openstack_compute: # Optional.
        vrouter:
            CONTROL_NODES: 10.60.0.9,10.60.0.10

```

```

    VROUTER_GATEWAY: 10.70.0.1
    location: pop1
compute_1_pop2:          # Mandatory. Instance name
  provider: bms          # Mandatory. Instance runs on BMS
  ip: 10.30.0.5
  roles:
    openstack_compute:   # Optional.
  vrouter:
    CONTROL_NODES: 10.60.0.11,10.60.0.12
    VROUTER_GATEWAY: 10.80.0.1
    location: pop2
compute_2_pop2:          # Mandatory. Instance name
  provider: bms          # Mandatory. Instance runs on BMS
  ip: 10.30.0.6
  roles:
    openstack_compute:   # Optional.
  vrouter:
    CONTROL_NODES: 10.60.0.11,10.60.0.12
    VROUTER_GATEWAY: 10.80.0.1
    location: pop2
global_configuration:
  CONTAINER_REGISTRY: 10.xx.x.81:5000
  REGISTRY_PRIVATE_INSECURE: True

contrail_configuration:    # Contrail service configuration section
  CONTRAIL_VERSION: <contrail_version>
  CONTROLLER_NODES: 10.60.0.5
  CLOUD_ORCHESTRATOR: openstack
  KEYSTONE_AUTH_HOST: 10.60.0.100
  KEYSTONE_AUTH_URL_VERSION: /v3
  RABBITMQ_NODE_PORT: 5673
  PHYSICAL_INTERFACE: eth1
  CONTROL_DATA_NET_LIST: 10.60.0.0/24,10.70.0.0/24,10.80.0.0/24

kolla_config:
  kolla_globals:
    network_interface: "eth1"
    enable_haproxy: "yes"
    contrail_api_interface_address: 10.60.0.5
    kolla_internal_vip_address: 10.60.0.100
    kolla_external_vip_address: 10.0.0.100
    kolla_external_vip_interface: "eth0"
  kolla_passwords:

```



```

keystone_admin_password: <password>

remote_locations:
  pop1:
    BGP_ASN: 12345
    SUBCLUSTER: pop1
  pop2:
    BGP_ASN: 12346
    SUBCLUSTER: pop2

```



**NOTE:** Replace *<contrail\_version>* with the correct `contrail_container_tag` value for your Contrail release. The respective `contrail_container_tag` values are listed in [README Access to Contrail Registry](#).

## 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>

# Upgrading Contrail Software

## IN THIS CHAPTER

- [Contrail In-Service Software Upgrade from Releases 3.2 and 4.1 to 5.0.x using Ansible Deployer | 98](#)
- [Contrail In-Service Software Upgrade from Releases 3.2 and 4.1 to 5.0.x using Helm Deployer | 108](#)

## Contrail In-Service Software Upgrade from Releases 3.2 and 4.1 to 5.0.x using Ansible Deployer

## IN THIS SECTION

- [Contrail In-Service Software Upgrade \(ISSU\) Overview | 98](#)
- [Prerequisites | 99](#)
- [Preparing the Contrail System for the Ansible Deployer ISSU Procedure | 99](#)
- [Provisioning Control Nodes and Performing Synchronization Steps | 101](#)
- [Transferring the Compute Nodes into the New Cluster | 104](#)
- [Finalizing the Contrail Ansible Deployer ISSU Process | 107](#)

## Contrail In-Service Software Upgrade (ISSU) Overview

If your installed version is Contrail Release 3.2 or higher, you can perform an in-service software upgrade (ISSU) to upgrade to Contrail Release 5.0.x using the Ansible deployer. In performing the ISSU, the Contrail controller cluster is upgraded side-by-side with a parallel setup, and the compute nodes are upgraded in place.



**NOTE:** We recommend that you take snapshots of your current system before you proceed with the upgrade process.

The procedure for performing the ISSU using the Contrail Ansible deployer is similar to previous ISSU upgrade procedures.



**NOTE:** This Contrail ansible deployer ISSU procedure does not include steps for upgrading OpenStack. If an OpenStack version upgrade is required, it should be performed using applicable OpenStack procedures.

In summary, the ISSU process consists of the following parts, in sequence:

1. Deploy the new cluster.
2. Synchronize the new and old clusters.
3. Upgrade the compute nodes.
4. Finalize the synchronization and complete the upgrades.

## Prerequisites

The following prerequisites are required to use the Contrail ansible deployer ISSU procedure:

- A previous version of Contrail installed, no earlier than Release 3.2.
- There are OpenStack controller and compute nodes, and Contrail nodes.
- OpenStack needs to have been installed from packages.
- Contrail and OpenStack should be installed on different nodes.



**NOTE:** Upgrade for compute nodes with Ubuntu 14.04 is not supported. Compute nodes need to be upgraded to Ubuntu 16.04 first.

## Preparing the Contrail System for the Ansible Deployer ISSU Procedure

In summary, these are the general steps for the system preparation phase of the Contrail ansible deployer ISSU procedure:

1. Deploy the 5.0.x version of Contrail using the Contrail ansible deployer, but make sure to include only the following Contrail controller services:

- Config
- Control
- Analytics
- Databases
- Any additional support services like rmq, kafka, and zookeeper. (The vrouter service will be deployed later on the old compute nodes.)



**NOTE:** You must provide keystone authorization information for setup.

2. After deployment is finished, you can log into the Contrail web interface to verify that it works.

The detailed steps for deploying the new cloud using the ansible deployer are as follows:

1. To deploy the new cloud, download **contrail-ansible-deployer-*release-tag*.tgz** onto your provisioning host from Juniper Networks.
2. The new cloud file **config/instances.yaml** appears as follows, with actual values in place of the variables as shown in the example:

```
provider_config:
  bms:
    domainsuffix: local
    ssh_user: user
    ssh_pwd: password
  instances:
    server1:
      ip: controller 1 ip
      provider: bms
      roles:
        analytics: null
        analytics_database: null
        config: null
        config_database: null
        control: null
        webui: null
  contrail_configuration:
    CONTROLLER_NODES: controller ip-s from api/mgmt network
    CONTROL_NODES: controller ip-s from ctrl/data network
    AUTH_MODE: keystone
```

```

KEYSTONE_AUTH_ADMIN_TENANT: old cloud's admin's tenant
KEYSTONE_AUTH_ADMIN_USER: old cloud's admin's user name
KEYSTONE_AUTH_ADMIN_PASSWORD: password for admin user
KEYSTONE_AUTH_HOST: keystone host/ip of old cloud
KEYSTONE_AUTH_URL_VERSION: "/v3"
KEYSTONE_AUTH_USER_DOMAIN_NAME: user's domain in case of keystone v3
KEYSTONE_AUTH_PROJECT_DOMAIN_NAME: project's domain in case of keystone v3
RABBITMQ_NODE_PORT: 5673
IPFABRIC_SERVICE_HOST: metadata service host/ip of old cloud
AAA_MODE: cloud-admin
METADATA_PROXY_SECRET: secret phrase that is used in old cloud
kolla_config:
  kolla_globals:
    kolla_internal_vip_address: keystone host/ip of old cloud
    kolla_external_vip_address: keystone host/ip of old cloud

```

3. Finally, run the ansible playbooks to deploy the new cloud.

```

ansible-playbook -v -e orchestrator=none -i inventory/ playbooks/configure_instances.yml
ansible-playbook -v -e orchestrator=openstack -i inventory/ playbooks/install_contrail.yml

```

After successful completion of these commands, the new cloud should be up and alive.

## Provisioning Control Nodes and Performing Synchronization Steps

In summary, these are the general steps for the node provisioning and synchronization phase of the Contrail ansible deployer ISSU procedure:

1. Provision new control nodes in the old cluster and old control nodes in the new cluster.
2. Stop the following containers in the new cluster on all nodes:
  - contrail-device-manager
  - contrail-schema-transformer
  - contrail-svcmonitor
3. Switch the new cloud into maintenance mode to prevent provisioning computes in the new cluster.
4. Prepare the config file for the ISSU.
5. Run the pre-sync script from the ISSU package.

6. Run the run-sync script from the ISSU package in background mode.

The detailed steps to provision the control nodes and perform the synchronization are as follows:

1. Pair the old control nodes in the new cluster. It is recommended to run it from any config-api container.

```
config_api_image='docker ps | awk '/config-api/{print $1}' | head'
```

2. Run the following command for each old control node, substituting actual values where indicated:

```
docker exec -it $config_api_image /bin/bash -c "LOG_LEVEL=SYS_NOTICE source /common.sh ;
python /opt/contrail/utils/provision_control.py --host_name hostname of old control node --
host_ip IP of old control node --api_server_ip $(hostname -i) --api_server_port 8082 --oper
add --router_asn 64512 --ibgp_auto_mesh \"$AUTH_PARAMS"
```

3. Pair the new control nodes in the old cluster with similar commands (the specific syntax depends on the deployment method of the old cluster), again substituting actual values where indicated.

```
python /opt/contrail/utils/provision_control.py --host_name new controller hostname --host_ip
new controller IP --api_server_ip old api-server IP/VIP --api_server_port 8082 --oper add --
admin_user admin --admin_password password --admin_tenant_name admin --router_asn 64512 --
ibgp_auto_mesh
```

4. Stop all the containers for contrail-device-manager, contrail-schema-transformer, and contrail-svcmonitor in the new cluster on all controller nodes.

```
docker stop config_devicemgr_1
docker stop config_schema_1
docker stop config_svcmonitor_1
```

These next steps should be performed from any new controller. Then the configuration prepared for ISSU runs. (For now, only manual preparation is available.)



**NOTE:** In various deployments, old cassandra may use port 9160 or 9161. You can learn the configuration details for the old services on any old controller node, in the file `/etc/contrail-contrail-api.conf`.

The configuration appears as follows and can be stored locally:

```
[DEFAULTS]
# details about oldrabbit
old_rabbit_user = contrail
old_rabbit_password = ab86245f4f3640a29b700def9e194f72
old_rabbit_q_name = vnc-config.issu-queue
old_rabbit_vhost = contrail
old_rabbit_port = 5672
old_rabbit_address_list = ip-addresses
# details about new rabbit
# new_rabbit_user = rabbitmq
# new_rabbit_password = password
# new_rabbit_ha_mode =
new_rabbit_q_name = vnc-config.issu-queue
new_rabbit_vhost = /
new_rabbit_port = 5673
new_rabbit_address_list = ip-addresses
# details about other old/new services
old_cassandra_user = controller
old_cassandra_password = 04dc0540b796492fad6f7cbdcfb18762
old_cassandra_address_list = ip-address:9161
old_zookeeper_address_list = ip-address:2181
new_cassandra_address_list = ip-address:9161 ip-address:9161 ip-address:9161
new_zookeeper_address_list = ip-address:2181
# details about new controller nodes
new_api_info = {"ip-address": [{"root"}, {"password"}], "ip-address": [{"root"}, {"password"}],
"ip-address": [{"root"}, {"password"}]}
```

## 1. Detect the config-api image ID.

```
image_id=`docker images | awk '/config-api/{print $3}' | head -1`
```

## 2. Run the pre-synchronization.

```
docker run --rm -it --network host -v $(pwd)/contrail-issu.conf:/etc/contrail/contrail-issu.conf --entrypoint /bin/bash -v /root/.ssh:/root/.ssh $image_id -c "/usr/bin/contrail-issu-pre-sync -c /etc/contrail/contrail-issu.conf"
```

### 3. Run the run-synchronization.

```
docker run --rm --detach -it --network host -v $(pwd)/contrail-issu.conf:/etc/contrail/
contrail-issu.conf --entrypoint /bin/bash -v /root/.ssh:/root/.ssh --name issu-run-sync
$image_id -c "/usr/bin/contrail-issu-run-sync -c /etc/contrail/contrail-issu.conf"
```

### 4. Check the logs of the run-sync process. To do this, open the run-sync container.

```
docker exec -it issu-run-sync /bin/bash
cat /var/log/contrail/issu_contrail_run_sync.log
```

### 5. Stop and remove the run-sync process after all compute nodes are upgraded.

```
docker rm -f issu-run-sync
```

## Transferring the Compute Nodes into the New Cluster

In summary, these are the general steps for the node transfer phase of the Contrail ansible deployer ISSU procedure:

1. Select the compute node(s) for transferring into the new cluster.
2. Move all workloads from the node(s) to other compute nodes. You also have the option to terminate workloads as appropriate.
3. For Contrail Release 3.x, remove Contrail from the node(s) as follows:
  - Stop the vrouter-agent service.
  - Remove the vhost0 interface.
  - Switch the physical interface down, then up.
  - Remove the vrouter.ko module from the kernel.
4. For Contrail Release 4.x, remove Contrail from the node(s) as follows:
  - Stop the agent container.
  - Restore the physical interface.
5. Add the required node(s) to **instances.yml** with the roles **vrouter** and **openstack\_legacy\_compute**.



6. Run the Contrail ansible deployer to deploy the new vrouter and to configure the old compute service.
7. All new compute nodes will have:
  - The collector setting pointed to the new Contrail cluster
  - The Control/DNS nodes pointed to the new Contrail cluster
  - The config-api setting in **vnc\_api\_lib.ini** pointed to the new Contrail cluster
8. (Optional) Run a test workload on transferred nodes to ensure the new vrouter-agent works correctly.

Follow these steps to rollback a compute node, if needed:

1. Move the workload from the compute node.
2. Stop the Contrail Release 5.0.x containers.
3. Ensure the network configuration has been successfully reverted.
4. Deploy the previous version of Contrail using the deployment method for that version.

The detailed steps for transferring compute nodes into the new cluster are as follows:



**NOTE:** After moving workload from the chosen compute nodes, you should remove the previous version of contrail-agent. For example, for Ubuntu 16.04 and vrouter-agent installed directly on the host, these would be the steps to remove the previous contrail-agent:

```
# stop services
systemctl stop contrail-vrouter-nodemgr
systemctl stop contrail-vrouter-agent
# remove packages
apt-get purge -y contrail*
# restore original interfaces definition
cd /etc/network/interfaces.d/
cp 50-cloud-init.cfg.save 50-cloud-init.cfg
rm vrouter.cfg
# restart networking
systemctl restart networking.service
# remove old kernel module
rmmod vrouter
```

```
# maybe you need to restore default route
ip route add 0.0.0.0/0 via 10.0.10.1 dev ens3
```

1. The new instance should be added to **instances.yaml** with two roles: vrouter and openstack\_compute\_legacy. To avoid reprovisioning the compute node, set the maintenance mode to TRUE. For example:

```
instances:
  server10:
    ip: compute 10 ip
    provider: bms
    roles:
      vrouter:
        MAINTENANCE_MODE: TRUE
        VROUTER_ENCRYPTION: FALSE
      openstack_compute_legacy: null
```

2. Run the ansible playbooks.

```
ansible-playbook -v -e orchestrator=none -e config_file=/root/contrail-ansible-deployer/
instances.yaml playbooks/configure_instances.yml
ansible-playbook -v -e orchestrator=openstack -e config_file=/root/contrail-ansible-deployer/
instances.yaml playbooks/install_contrail.yml
```

3. The contrail-status for the compute node appears as follows:

```
vrouter kernel module is PRESENT
== Contrail vrouter ==
nodemgr: active
agent: initializing (No Configuration for self)
```

4. Restart contrail-control on all new controller nodes after the upgrade is complete:

```
docker restart control_control_1
```

5. Check status of new compute nodes by running `contrail-status` on them. All components should be active now. You can also check the status of the new instance by creating AZ/aggregates with the new compute nodes and run some test workloads to ensure it operates correctly.

## Finalizing the Contrail Ansible Deployer ISSU Process

Finalize the Contrail ansible deployer ISSU as follows:

1. Stop the `issu-run-sync` container.

```
docker rm -f issu-run-sync
```

2. Run the post synchronization commands.

```
docker run --rm -it --network host -v $(pwd)/contrail-issu.conf:/etc/contrail/contrail-issu.conf --entrypoint /bin/bash -v /root/.ssh:/root/.ssh --name issu-run-sync $image_id -c "/usr/bin/contrail-issu-post-sync -c /etc/contrail/contrail-issu.conf"
docker run --rm -it --network host -v $(pwd)/contrail-issu.conf:/etc/contrail/contrail-issu.conf --entrypoint /bin/bash -v /root/.ssh:/root/.ssh --name issu-run-sync $image_id -c "/usr/bin/contrail-issu-zk-sync -c /etc/contrail/contrail-issu.conf"
```

3. Disengage maintenance mode and start all previously stopped containers. To do this, set the entry `MAINTENANCE_MODE` in **instances.yaml** to `FALSE`, then run the following command from the deployment node:

```
ansible-playbook -v -e orchestrator=openstack -i inventory/ playbooks/install_contrail.yml
```

4. Clean up and remove the old Contrail controllers. Use the **provision-issu.py** script called from the `config-api` container with the config **issu.conf**. Replace the credential variables and API server IP with appropriate values as indicated.

```
[DEFAULTS]
db_host_info={"ip-address": "node-ip-address", "ip-address": "node-ip-address", "ip-address": "node-ip-address"}
config_host_info={"ip-address": "node-ip-address", "ip-address": "node-ip-address", "ip-address": "node-ip-address"}
analytics_host_info={"ip-address": "node-ip-address", "ip-address": "node-ip-address", "ip-address": "node-ip-address"}
control_host_info={"ip-address": "node-ip-address", "ip-address": "node-ip-address", "ip-
```

```
address": "node-ip-address"}
admin_password = admin password
admin_tenant_name = admin tenant
admin_user = admin username
api_server_ip= any IP of new config-api controller
api_server_port=8082
```

5. Run the following commands from any controller node.



**NOTE:** All *\*host\_info* parameters should contain the list of new hosts.

```
docker cp issu.conf config_api_1:issu.conf
docker exec -it config_api_1 python /opt/contrail/utils/provision_issu.py -c issu.conf
```

6. Servers can be cleaned up if there are no other services present.
7. All configurations for the neutron-api must be edited to have the parameter `api_server_ip` point to the list of new config-api IP addresses. Locate **ContrailPlugin.ini** (or other file that contains this parameter) and change the IP addresses to the list of new config-api IP addresses.
8. The heat configuration needs the same changes. Locate the parameter `[clients_contrail]/api_server` and change it to point to the list of the new config-api IP addresses.

## Contrail In-Service Software Upgrade from Releases 3.2 and 4.1 to 5.0.x using Helm Deployer

### IN THIS SECTION

- [Contrail In-Service Software Upgrade \(ISSU\) Overview | 109](#)
- [Prerequisites | 109](#)
- [Preparing the Contrail System for the Helm Deployer ISSU Procedure | 110](#)
- [Provisioning Control Nodes and Performing Synchronization Steps | 110](#)
- [Transferring the Compute Nodes into the New Cluster | 113](#)
- [Finalizing the Contrail Helm Deployer ISSU Process | 117](#)

## Contrail In-Service Software Upgrade (ISSU) Overview

If your installed version is Contrail Release 3.2 or higher, you can perform an in-service software upgrade (ISSU) to upgrade to Contrail Release 5.0.x using the Helm deployer. In performing the ISSU, the Contrail controller cluster is upgraded side-by-side with a parallel setup, and the compute nodes are upgraded in place.



**NOTE:** We recommend that you take snapshots of your current system before you proceed with the upgrade process.

The procedure for performing the ISSU using the Contrail Helm deployer is similar to previous ISSU upgrade procedures.



**NOTE:** This Contrail Helm deployer ISSU procedure does not include steps for upgrading OpenStack. If an OpenStack version upgrade is required, it should be performed using applicable OpenStack procedures.

In summary, the ISSU process consists of the following parts, in sequence:

1. Deploy the new cluster.
2. Synchronize the new and old clusters.
3. Upgrade the compute nodes.
4. Finalize the synchronization and complete the upgrades.

## Prerequisites

The following prerequisites are required to use the Contrail Helm deployer ISSU procedure:

- A previous version of Contrail installed, not earlier than Release 3.2.
- There are OpenStack controller and compute nodes, and Contrail nodes.
- OpenStack needs to have been installed from packages.
- Contrail and OpenStack should be installed on different nodes.



**NOTE:** Upgrade for compute nodes with Ubuntu 14.04 is not supported. Compute nodes need to be upgraded to Ubuntu 16.04 first.

## Preparing the Contrail System for the Helm Deployer ISSU Procedure

In summary, these are the general steps for the system preparation phase of the Contrail Helm deployer ISSU procedure:

1. Deploy the 5.0.x version of Contrail using the Contrail Helm deployer, but make sure to include only the following Contrail controller services:
  - Config
  - Control
  - Analytics
  - Databases
  - Any additional support services like rmq, kafka, and zookeeper. (The vrouter service will be deployed later on the old compute nodes.)



**NOTE:** You must provide keystone authorization information for setup.

2. After deployment is finished, you can log into the Contrail web interface to verify that it works.

Detailed instructions for deploying the new cloud using Helm are provided in [Installing Contrail Networking for Kubernetes using Helm](#).

## Provisioning Control Nodes and Performing Synchronization Steps

In summary, these are the general steps for the node provisioning and synchronization phase of the Contrail Helm deployer ISSU procedure:

1. Provision new control nodes in the old cluster and old control nodes in the new cluster.
2. Stop the following containers in the new cluster on all nodes:
  - contrail-device-manager
  - contrail-schema-transformer
  - contrail-svcmonitor
3. Switch the new cloud into maintenance mode to prevent provisioning computes in the new cluster.
4. Prepare the config file for the ISSU.
5. Run the pre-sync script from the ISSU package.

6. Run the run-sync script from the ISSU package in background mode.

The detailed steps to provision the control nodes and perform the synchronization are as follows:

1. Pair the old control nodes in the new cluster. It is recommended to run it from any config-api container:

```
config_api_cid=`docker ps | awk '/config-api/{print $1}' | head`
```

2. Run this command for each old control node, substituting actual values where indicated:

```
docker exec -it $config_api_cid /bin/bash -c "LOG_LEVEL=SYS_NOTICE source /common.sh ;
python /opt/contrail/utils/provision_control.py --host_name hostname of old control node --
host_ip IP of old control node --api_server_ip $(hostname -i) --api_server_port 8082 --oper
add --router_asn 64512 --ibgp_auto_mesh \"$AUTH_PARAMS"
```

3. Pair the new control nodes in the old cluster with similar commands (the specific syntax depends on the deployment method of the old cluster), again substituting actual values where indicated.

```
python /opt/contrail/utils/provision_control.py --host_name new controller hostname --host_ip
new controller IP --api_server_ip old api-server IP/VIP --api_server_port 8082 --oper add --
admin_user admin --admin_password password --admin_tenant_name admin --router_asn 64512 --
ibgp_auto_mesh
```

4. Stop all the containers for contrail-device-manager, contrail-schema-transformer, and contrail-svcmonitor in the new cluster on all controller nodes.

```
docker ps | grep config-devicemgr | awk '{print $1}' | xargs docker pause
docker ps | grep config-schema | awk '{print $1}' | xargs docker pause
docker ps | grep config-svcmonitor | awk '{print $1}' | xargs docker pause
```

These next steps should be performed from any new Contrail controller. Then the configuration prepared for ISSU runs. (For now, only manual preparation is available.)



**NOTE:** In various deployments, old cassandra may use port 9160 or 9161. You can learn the configuration details for the old services on any old controller node, in the file `/etc/contrail-contrail-api.conf`.

The configuration appears as follows and can be stored locally:

```
[DEFAULTS]
# details about oldrabbit
old_rabbit_user = contrail
old_rabbit_password = ab86245f4f3640a29b700def9e194f72
old_rabbit_q_name = vnc-config.issu-queue
old_rabbit_vhost = contrail
old_rabbit_port = 5672
old_rabbit_address_list = ip-address
# details about new rabbit
# new_rabbit_user = rabbitmq
# new_rabbit_password = password
# new_rabbit_ha_mode =
new_rabbit_q_name = vnc-config.issu-queue
new_rabbit_vhost = /
new_rabbit_port = 5673
new_rabbit_address_list = rabbitmq.contrail
# details about other old/new services
old_cassandra_user = controller
old_cassandra_password = 04dc0540b796492fad6f7cbdcfb18762
old_cassandra_address_list = ip-address:9161
old_zookeeper_address_list = ip-address:2181
new_cassandra_address_list = ip-address:9161 ip-address:9161 ip-address:9161
new_zookeeper_address_list = ip-address:2181
# details about new controller nodes
new_api_info = {"ip-address": [{"root"}, {"password"}], "ip-address": [{"root"}, {"password"}],
"ip-address": [{"root"}, {"password"}]}
```

### 1. Detect the config-api image ID:

```
image_id=`docker images | awk '/config-api/{print $3}' | head -1`
```

### 2. Run the pre-synchronization.

```
docker run --rm -it --network host -v $(pwd)/contrail-issu.conf:/etc/contrail/contrail-issu.conf --entrypoint /bin/bash -v /root/.ssh:/root/.ssh $image_id -c "/usr/bin/contrail-issu-pre-sync -c /etc/contrail/contrail-issu.conf"
```



### 3. Run the run-synchronization.

```
docker run --rm --detach -it --network host -v $(pwd)/contrail-issu.conf:/etc/contrail/
contrail-issu.conf --entrypoint /bin/bash -v /root/.ssh:/root/.ssh --name issu-run-sync
$image_id -c "/usr/bin/contrail-issu-run-sync -c /etc/contrail/contrail-issu.conf"
```

### 4. Check the logs of the run-sync process. To do this, open the run-sync container.

```
docker exec -it issu-run-sync /bin/bash
cat /var/log/contrail/issu_contrail_run_sync.log
```

### 5. Stop and remove the run-sync process after all compute nodes are upgraded.

```
docker rm -f issu-run-sync
```

## Transferring the Compute Nodes into the New Cluster

In summary, these are the general steps for the node transfer phase of the Contrail Helm deployer ISSU procedure:

1. Select the compute node(s) for transferring into the new cluster.
2. Move all workloads from the node(s) to other compute nodes. You also have the option to terminate workloads as appropriate.
3. For Contrail Release 3.x, remove Contrail from the node(s) as follows:
  - Stop the vrouter-agent service.
  - Remove the vhost0 interface.
  - Switch the physical interface down, then up.
  - Remove the vrouter.ko module from the kernel.
4. For Contrail Release 4.x, remove Contrail from the node(s) as follows:
  - Stop the agent container.
  - Restore the physical interface.
5. Add the required node(s) to **instances.yml** with the roles **vrouter** and **openstack\_legacy\_compute**.

6. Run the Contrail Helm deployer to deploy the new vrouter and to configure the old compute service.
7. All new compute nodes will have:
  - The collector setting pointed to the new Contrail cluster
  - The Control/DNS nodes pointed to the new Contrail cluster
  - The config-api setting in **vnc\_api\_lib.ini** pointed to the new Contrail cluster
8. (Optional) Run a test workload on transferred nodes to ensure the new vrouter-agent works correctly.

Follow these steps to rollback a compute node, if needed:

1. Move the workload from the compute node.
2. Stop the Contrail Release 5.0.x containers.
3. Ensure the network configuration has been successfully reverted.
4. Deploy the previous version of Contrail using the deployment method for that version.

The detailed steps for transferring compute nodes into the new cluster are as follows:



**NOTE:** After moving workload from the chosen compute nodes, you should remove the previous version of contrail-agent. For example, for Ubuntu 16.04 and vrouter-agent installed directly on the host, these would be the steps to remove the previous contrail-agent:

```
# stop services
systemctl stop contrail-vrouter-nodemgr
systemctl stop contrail-vrouter-agent
# remove packages
apt-get purge -y contrail*
# restore original interfaces definition
cd /etc/network/interfaces.d/
cp 50-cloud-init.cfg.save 50-cloud-init.cfg
rm vrouter.cfg
# restart networking
systemctl restart networking.service
# remove old kernel module
rmmod vrouter
# maybe you need to restore default route
ip route add 0.0.0.0/0 via 10.0.10.1 dev ens3
```

The new instance requires two Helm repositories which can be downloaded from Juniper Networks.

1. Download the file **contrail-helm-deployer-*release-tag*.tgz** onto your provisioning host
2. Run the command `scp contrail-helm-deployer-release-tag.tgz` for all nodes in the cluster
3. Untar **contrail-helm-deployer-*release-tag*.tgz** on all nodes:

```
tar -zxf contrail-helm-deployer-release-tag.tgz -C /opt/
```

The next set of steps sets up the new compute nodes for Contrail deployment.



**NOTE:** You should run the steps in the following procedure from the same node where Contrail was deployed.

1. Add the new instance to **/opt/openstack-helm-infra/tools/gate/devel/multinode-inventory.yaml**, in the nodes section.
2. Prepare the new compute nodes for Contrail deployment:

```
export BASE_DIR=/opt
export OSH_INFRA_PATH=${BASE_DIR}/openstack-helm-infra
export CHD_PATH=${BASE_DIR}/contrail-helm-deployer
cd ${OSH_INFRA_PATH}
make dev-deploy setup-host multinode
make dev-deploy k8s multinode
```

3. Verify the new node names by using the command `kubectl get nodes`.
4. Label the new nodes as follows:

```
kubectl label node name --overwrite openstack-control-plane=disable
kubectl label node name opencontrail.org/vrouter-kernel=enabled
```

5. To avoid reprovisioning compute nodes when adding them, set the maintenance mode to TRUE in **values.yaml**. For example:

```
global:
  contrail_env_vrouter_kernel:
    MAINTENANCE_MODE: TRUE
```

6. If adding vrouter with the DPDK or SRIOV role, switch the kernel to dpdk or sriov mode as appropriate.



**NOTE:** You need only to deploy the vrouter Helm chart just once for the first compute node or nodes. Upon subsequent deployments, k8s will automatically deploy vrouter on the new nodes.

7. Add vrouter as follows:

```
helm install --name contrail-vrouter ${CHD_PATH}/contrail-vrouter --namespace=contrail --
values=/tmp/values.yaml
```

8. After labeling and installing the new nodes, get the pods to verify they are operational.

```
kubectl get pods -n contrail
```



**NOTE:** If the new nodes are not deployed correctly, check for the presence of a default route. If a default route is not present, restore it.

9. At this point, contrail-status for compute nodes should have output as follows:

```
vrouter kernel module is PRESENT
== Contrail vrouter ==
nodemgr: active
agent: initializing (No Configuration for self)
```

10. Restart contrail-control on all the new controller nodes after upgrading the compute nodes.

```
docker ps | grep control-control | awk '{print $1}' | xargs docker
```

11. Transfer the new code into the compute node as follows:

```
pythonpath=`python -c "import sys; paths = [path for path in sys.path if 'packages' in path] ; print(paths[-1])"`
init_image_id=`docker images | awk '/contrail-vrouter-agent/{print $1":"$2}' | head -1 | sed 's/contrail-vrouter-agent/contrail-openstack-compute-init/'`
docker run --rm -it --network host -v /usr/bin:/opt/plugin/bin -v $pythonpath:/opt/plugin/site-packages $init_image_id
```

12. Check status of new compute nodes by running contrail-status on them. All components should be active now. You can also check the status of the new instance by creating AZ/aggregates with the new compute nodes and run some test workloads to ensure it operates correctly.

## Finalizing the Contrail Helm Deployer ISSU Process

Finalize the Contrail Helm deployer ISSU as follows:

1. Stop the issu-run-sync container.

```
docker rm -f issu-run-sync
```

2. Run the post synchronization commands.

```
docker run --rm -it --network host -v $(pwd)/contrail-issu.conf:/etc/contrail/contrail-issu.conf --entrypoint /bin/bash -v /root/.ssh:/root/.ssh --name issu-run-sync $image_id -c "/usr/bin/contrail-issu-post-sync -c /etc/contrail/contrail-issu.conf"
docker run --rm -it --network host -v $(pwd)/contrail-issu.conf:/etc/contrail/contrail-issu.conf --entrypoint /bin/bash -v /root/.ssh:/root/.ssh --name issu-run-sync $image_id -c "/usr/bin/contrail-issu-zk-sync -c /etc/contrail/contrail-issu.conf"
```

3. Start all previously stopped containers.

```
docker ps | grep config-devicemgr | awk '{print $1}' | xargs docker unpause | xargs docker restart
```

```
docker ps | grep config-schema | awk '{print $1}' | xargs docker unpause | xargs docker
restart
docker ps | grep config-svcmonitor | awk '{print $1}' | xargs docker unpause | xargs docker
restart
```

4. Disengage maintenance mode. To do this, set the entry `MAINTENANCE_MODE` in `values.yaml` to `FALSE`, then run the following command from the deployment node:

```
helm upgrade -f /tmp/values.yaml contrail-vrouter /opt/contrail-helm-deployer/contrail-vrouter
```

5. Clean up and remove the old Contrail controllers. Use the `provision-issu.py` script called from the `config-api` container, with the config `issu.conf`. Replace the credential variables and API server IP with appropriate values as indicated.

```
[DEFAULTS]
db_host_info={"ip-address": "node-ip-address", "ip-address": "node-ip-address", "ip-address":
"node-ip-address"}
config_host_info={"ip-address": "node-ip-address", "ip-address": "node-ip-address", "ip-
address": "node-ip-address"}
analytics_host_info={"ip-address": "node-ip-address", "ip-address": "node-ip-address", "ip-
address": "node-ip-address"}
control_host_info={"ip-address": "node-ip-address", "ip-address": "node-ip-address", "ip-
address": "node-ip-address"}
admin_password = admin password
admin_tenant_name = admin tenant
admin_user = admin username
api_server_ip= any IP of new config-api controller
api_server_port=8082
```

6. Run the following commands from any controller node:



**NOTE:** All *\*host\_info* parameters should contain the list of new hosts.

```
config_api_cid=`docker ps | awk '/config-api/{print $1}' | head`
docker cp issu.conf $config_api_cid:issu.conf
docker exec -it $config_api_cid python /opt/contrail/utils/provision-issu.py -c issu.conf
```

7. Servers can be cleaned up if there are no other services present.

8. All configurations for the neutron-api must be edited to have the parameter `api_server_ip` point to the list of new config-api IP addresses. Locate **ContrailPlugin.ini** (or other file that contains this parameter) and change the IP addresses to the list of new config-api IP addresses.
9. The heat configuration needs the same changes. Locate the parameter `[clients_contrail]/api_server` and change it to point to the list of the new config-api IP addresses.

# Backup and Restore Contrail Software

## IN THIS CHAPTER

- [Backing up Contrail Databases in JSON Format | 120](#)

## Backing up Contrail Databases in JSON Format

### IN THIS SECTION

- [Preliminary Caution | 120](#)
- [Simple Database Backup in JSON Format | 121](#)
- [Restore Database from the Backup | 121](#)
- [Example Backup and Restore in JSON | 123](#)

This document shows how to take backup of Contrail databases (Cassandra and Zookeeper) in JSON format.

### Preliminary Caution



**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 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 Database Backup in JSON Format

Perform a simple backup (database dump). Use `db_json_exim.py`, located at `/usr/lib/python2.7/site-packages/cfgm_common` on controller node.



**NOTE:** The controller node for non-containerized Contrail is a virtual machine (VM).  
The controller node for containerized Contrail is a controller container.

1. Run the following command on any one of the controller nodes to go to `config_api_1` container.

```
docker exec -it config_api_1 bash
```

2. Backup data with `db_json_exim` in JSON format.

```
cd /usr/lib/python2.7/site-packages/cfgm_common
```

```
python db_json_exim.py --export-to db-dump.json
```

3. See a cleaner version of the dump.

```
cat db-dump.json | python -m json.tool | less
```

4. Omit keyspace in the dump, for example, to share with Juniper Networks.

```
python db_json_exim.py --export-to db-dump.json --omit-keyspace dm_keyspace
```

## Restore Database from the Backup

Use the following steps to restore a system from a simple backup.

1. Copy `db-dump.json` and `contrail-api.conf` to the host.

```
mkdir /tmp/db-dump docker cp config_api_1:/etc/contrail/contrail-api.conf /tmp/db-dump/ docker cp  
config_api_1:/usr/lib/python2.7/site-packages/cfgm_common/db-dump.json /tmp/db-dump/
```

2. Stop config services on all the controllers.

```
docker stop config_svcmonitor_1
```

```
docker stop config_devicemgr_1
```

```
docker stop config_schema_1
```

```
docker stop config_api_1
```

```
docker stop config_nodemgr_1
```

```
docker stop config_database_nodemgr_1
```

3. Stop Cassandra on all the config-db controllers or verify it is already stopped.

```
docker stop config_database_cassandra_1
```

4. Stop Zookeeper on all the controllers or verify it is already stopped.

```
docker stop config_database_zookeeper_1
```

5. Stop Kafka on all controllers. Check analytics controllers.

```
docker stop analytics_database_kafka_1
```

6. Backup the Zookeeper data directory on all the controllers.

```
cd /var/lib/docker/volumes/config_database_config_zookeeper
```

```
cp -R _data/version-2/ version-2-save
```

7. Wipe out the Zookeeper data directory contents on all the controllers.

```
rm -rf _data/version-2/*
```

8. Backup the Cassandra data directory on all the controllers.

```
cd /var/lib/docker/volumes/config_database_config_cassandra
```

```
cp -R _data/ Cassandra_data-save
```

9. Wipe out the Cassandra data directory contents on all controllers.

```
rm -rf _data/*
```

10. Start Zookeeper on all the controllers.

```
docker start config_database_zookeeper_1
```

11. Start Cassandra on all the controllers.

```
docker start config_database_cassandra_1
```

12. List docker image to the name/ID of config-api image.

```
docker image ls | grep config-api
```

13. Run a new docker using the name or ID of the config-api image.

```
docker run --rm -it -v /tmp/db-dump:/tmp/ --network host --entrypoint=/bin/bash ci-<repository>:5000/
contrail-controller-config-api:5.0-latest
```

14. Restore the data in new running docker.

```
cd /usr/lib/python2.7/site-packages/cfgm_common python db_json_exim.py --import-from /tmp/db-dump.json --api-
conf /tmp/contrail-api.conf
```

**15. Start Kafka on all controllers. Check analytics controllers.**

```
docker start analytics_database_kafka_1
```

**16. Start config services on all the controllers.**

```
docker start config_svcmonitor_1
```

```
docker start config_devicemgr_1
```

```
docker start config_schema_1
```

```
docker start config_api_1
```

```
docker start config_nodemgr
```

```
docker start config_database_nodemgr
```

## Example Backup and Restore in 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:

- nodec53
- nodec54
- nodec55

### Example: Perform Simple Database Backup in JSON Format

```
[root@nodec54 ~]# docker exec -it config_api_1 bash
(config-api)[root@nodec54 /root]$ cd /usr/lib/python2.7/site-packages/cfgm_common/
(config-api)[root@nodec54 /usr/lib/python2.7/site-packages/cfgm_common]$ python db_json_exim.py
--export-to db-dump.json
(config-api)[root@nodec54 /usr/lib/python2.7/site-packages/cfgm_common]$ cat db-dump.json |
python -m json.tool |less
{
  "cassandra": {
    "config_db_uuid": {
      "obj_fq_name_table": {
```

```
"access_control_list": {
<snip>
```

## Example: Restore Database from the Backup

1. Copy db-dump.json and contrail-api.conf to the host.

```
root@nodec54 ~]# mkdir /tmp/db-dump
root@nodec54 ~]# docker cp config_api_1:/etc/contrail/contrail-api.conf /tmp/db-dump/
root@nodec54 ~]# docker cp config_api_1:/usr/lib/python2.7/site-packages/cfgm_common/db-
dump.json /tmp/db-dump/
```

2. Stop config services on all the controllers.

```
[root@nodec53 ~]# docker stop config_schema_1
[root@nodec53 ~]# docker stop config_svcmonitor_1
[root@nodec53 ~]# docker stop config_devicemgr_1
[root@nodec53 ~]# docker stop config_nodemgr
[root@nodec53 ~]# docker stop config_database_nodemgr

root@nodec54~]# docker stop config_schema_1
[root@nodec54 ~]# docker stop config_svcmonitor_1
[root@nodec54 ~]# docker stop config_devicemgr_1
[root@nodec54 ~]# docker stop config_nodemgr
[root@nodec54 ~]# docker stop config_database_nodemgr

root@nodec55~]# docker stop config_schema_1
[root@nodec55 ~]# docker stop config_svcmonitor_1
[root@nodec55 ~]# docker stop config_devicemgr_1
[root@nodec55 ~]# docker stop config_nodemgr
[root@nodec55 ~]# docker stop config_database_nodemgr
```

3. Stop Cassandra on all the config-db controllers or verify it is already stopped.

```
[root@nodec53 ~]# docker stop config_database_cassandra_1

[root@nodec54 ~]# docker stop config_database_cassandra_1
```

```
[root@nodec55 ~]# docker stop config_database_cassandra_1
```

4. Stop Zookeeper on all the controllers or verify it is already stopped.

```
[root@nodec53 ~]# docker stop config_database_zookeeper_1
```

```
[root@nodec54 ~]# docker stop config_database_zookeeper_1
```

```
[root@nodec55 ~]# docker stop config_database_zookeeper_1
```

5. Stop Kafka on all the controllers. Check analytics controllers.

```
[root@nodec53 ~]# docker stop analytics_database_kafka_1
```

```
[root@nodec54 ~]# docker stop analytics_database_kafka_1
```

```
[root@nodec55 ~]# docker stop analytics_database_kafka_1
```

6. Stop Kafka on all the controllers. Check analytics controllers.

```
[root@nodec53 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra
```

```
[root@nodec53 config_database_config_cassandra]# rm -rf _data/*
```

```
[root@nodec54 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra
```

```
[root@nodec54 config_database_config_cassandra]# rm -rf _data/*
```

```
[root@nodec55 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra
```

```
[root@nodec55 config_database_config_cassandra]# rm -rf _data/*
```

7. Delete config Cassandra.

```
[root@nodec53 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra
```

```
[root@nodec53 config_database_config_cassandra]# rm -rf _data/*
```

```
[root@nodec54 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra
```

```
[root@nodec54 config_database_config_cassandra]# rm -rf _data/*
```

```
[root@nodec55 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra
```

```
[root@nodec55 config_database_config_cassandra]# rm -rf _data/*
```

## 8. Delete config Zookeeper.

```
[root@nodec53 _data]# cd /var/lib/docker/volumes/config_database_config_zookeeper
[root@nodec53 config_database_config_zookeeper]# rm -rf _data/version-2/*

[root@nodec54 _data]# cd /var/lib/docker/volumes/config_database_config_zookeeper
[root@nodec54 config_database_config_zookeeper]# rm -rf _data/version-2/*

[root@nodec55 _data]# cd /var/lib/docker/volumes/config_database_config_zookeeper
[root@nodec55 config_database_config_zookeeper]# rm -rf _data/version-2/*
```

## 9. Start config Cassandra and Zookeeper on all the controllers.

```
[root@nodec53 ~]# docker start config_database_zookeeper_1
[root@nodec53 ~]# docker start config_database_cassandra_1

[root@nodec54 ~]# docker start config_database_zookeeper_1
[root@nodec54 ~]# docker start config_database_cassandra_1

[root@nodec55 ~]# docker start config_database_zookeeper_1
[root@nodec55 ~]# docker start config_database_cassandra_1
```

## 10. Run db\_json\_exim.py to restore the data from json dump.

```
root@nodec54 ~]# docker image ls | grep config-api
root@nodec54 ~]# docker run --rm -it -v /tmp/db-dump:/tmp/ --network host --
entrypoint=/bin/bash ci-<repository>:5000/contrail-controller-config-api:5.0-latest
(config-api)[root@nodec54 /root]$ cd /usr/lib/python2.7/site-packages/cfgm_common/
(config-api)[root@nodec54 /usr/lib/python2.7/site-packages/cfgm_common]$ python
db_json_exim.py --import-from /tmp/db-dump.json --api-conf /tmp/contrail-api.conf
```

## 11. Start Kafka on all the controllers. Check analytics controllers.

```
[root@nodec53 ~]# docker start analytics_database_kafka_1
[root@nodec54 ~]# docker start analytics_database_kafka_1
[root@nodec55 ~]# docker start analytics_database_kafka_1
```

## 12. Start config services on all the controllers.

```
[root@nodec53 ~]# docker start config_schema_1
[root@nodec53 ~]# docker start config_svcmonitor_1
[root@nodec53 ~]# docker start config_devicemgr_1
[root@nodec53 ~]# docker start config_nodemgr
[root@nodec53 ~]# docker start config_database_nodemgr
[root@nodec53 ~]# docker start config_api _1
```

```
[root@nodec54~]# docker start config_schema_1
[root@nodec54 ~]# docker start config_svcmonitor_1
[root@nodec54 ~]# docker start config_devicemgr_1
[root@nodec54 ~]# docker start config_nodemgr
[root@nodec54 ~]# docker start config_database_nodemgr
[root@nodec54 ~]# docker start config_api _1
```

```
[root@nodec55 ~]# docker start config_schema_1
[root@nodec55 ~]# docker start config_svcmonitor_1
[root@nodec55 ~]# docker start config_devicemgr_1
[root@nodec55 ~]# docker start config_nodemgr
[root@nodec55 ~]# docker start config_database_nodemgr
[root@nodec55 ~]# docker start config_api _1
```

# Multicloud Contrail

## IN THIS CHAPTER

- [Contrail Deployment on Microsoft Azure | 128](#)
- [Deploying Contrail on Microsoft Azure | 129](#)
- [On-Premise and Azure Multicloud Deployment | 135](#)
- [Modifying Multicloud Topology | 147](#)
- [Deploying Contrail Enterprise Multicloud using REST API | 148](#)

## Contrail Deployment on Microsoft Azure

Contrail Release 5.0.2 supports extending of on-premise Contrail capability on to Microsoft Azure public cloud. The multicloud gateway feature enables leveraging Contrail services to the public cloud seamlessly.

Ansible is used to deploy Contrail on the public cloud. Terraform is used to build the resources on the public cloud creates a template of all Azure objects. These templates are autogenerated in Contrail Release 5.0.2. These templates take care of all Contrail requirements including creating VMs in Azure, connecting the network, providing IP addresses for the VMs and so on. Secure connectivity to the network is provided through the Contrail multicloud gateway. The core stack of the multicloud gateway comprises Contrail vRouter, BGP, and IPsec over SSL. IPsec provides the VPN capabilities. After creating VMs and providing secure connectivity to the network through the multicloud gateway, you can deploy Contrail on the secure fabric and all on-premise Contrail features and services are available on the cloud.

Consider that you have an on-premise environment with multiple applications or workloads running on it. The workloads include front-end, middle tier, and back-end applications or a database. To virtualize the workloads on Azure, Contrail creates a multicloud gateway on the on-premise site as well as on Azure. The multicloud gateway provides seamless and secure connectivity between the on-premise system and Azure. Once secure connectivity is established, the on-premise workloads can be deployed on Azure. You can choose to deploy all the workloads, or some workloads, or also have a hybrid environment where some workloads are running in the on-premise system and some on the public cloud.



This workflow of spinning up Contrail SDN in a multicloud environment for Azure is automated. See *Deploying Contrail on Microsoft Azure* for information on deploying Contrail on Azure.

## RELATED DOCUMENTATION

*Deploying Contrail on Microsoft Azure*

*On-Premise and Azure Public Cloud Deployment*

*Adding a Compute Host to Multicloud*

## Deploying Contrail on Microsoft Azure

### IN THIS SECTION

- [Deployment of Contrail on Azure | 129](#)
- [Deleting Contrail Deployment from Azure | 134](#)

Starting from Contrail Release 5.0.2, you can deploy Contrail on Microsoft Azure public cloud. This topic describes Contrail deployment procedures on Azure and also the procedure to delete the deployment.

### Deployment of Contrail on Azure

Ensure that you have a valid subscription to an Azure account for virtual networks and virtual machines (VMs). Create the `contrail-multicloud` resource group on the Azure portal. Ensure that you have installed the Docker on the local deployer host.

Perform the following detailed steps for deploying Contrail on Azure.

Perform the following steps to create a topology with two virtual networks, two gateways, two compute hosts and one controller in Azure.

1. To download the Multicloud Deployer package file, follow these steps:
  - a. Select Contrail version 5.0.x from the **Version** list in the Juniper Networks [Software Downloads](#) page.
  - b. In the **Application Tools** section, click the Multicloud Deployer **tgz** file.

You are now redirected to the **Software Download** page.

- c. Log in to the download page.

A **End User License Agreement** is displayed. Select **I Agree** and click on **Proceed**.

- d. Download the file on your localhost or on your device.
- e. Follow the **Usage Instructions** on the download page to install the file.

2. Extract the contents of the .tgz file.

```
# tar -xzf contrail-multicloud-deployer-5.0.2-0.XXX.tgz
```

3. Create the **secret.yml** file. The **secrets.yml** file contains required credentials for multicloud deployment. For Azure you need to add only the public\_key.

```
# vi secrets.yml
public_key: "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQ..."
```

4. Define the topology. The **topology.yml** file comprises the resource group, virtual networks or clouds, and instances. Instances can have roles such as the gateway role for the multicloud gateway, controller and k8s\_master roles for the controller nodes, and compute role for the compute nodes. The instance type as defined in standard Azure documentation and you must be aware of what is available in Azure for populating the topology.

```
# vi topology.yml
```

Here is an example of a **topology.yml** file.

```
- provider: azure
  organization: Juniper
  project: contrail-multicloud
  regions:
    - name: WestUS2
      resource_group: contrail-multicloud-training
      vnet:
        - name: contrail-az-1
          cidr_block: 192.168.0.0/16
          subnets:
            - name: subnet_contrail_az_1
              cidr_block: 192.168.100.0/24
              security_group: allow_all_protocols
          security_groups:
```

```

- name: allow_all_protocols-contrail-az-1
  rules:
    - name: all_in-contrail-az-1
      direction: inbound
    - name: all_out-contrail-az-1
      direction: outbound
instances:
- name: az-contrail-gw-1
  roles:
    - gateway
  provision: true
  username: ubuntu
  os: ubuntu16
  instance_type: Standard_F16s_v2
  subnets: subnet_contrail_az_1
  interface: eth1

- name: controller-contrail-az-1
  provision: true
  username: ubuntu
  roles:
    - controller
    - k8s_master
  os: ubuntu16
  instance_type: Standard_F32s_v2
  subnets: subnet_contrail_az_1
  interface: eth0

- name: compute-contrail-az-1
  provision: true
  username: ubuntu
  roles:
    - compute_node
  os: ubuntu16
  instance_type: Standard_F16s_v2
  subnets: subnet_contrail_az_1
  interface: eth0

- name: contrail-az-2
  cidr_block: 10.0.0.0/16
  subnets:
    - name: subnet_contrail_az_2
      cidr_block: 10.0.100.0/24

```

```

    security_group: allow_all_protocols-contrail-az-2
security_groups:
  - name: allow_all_protocols-contrail-az-2
    rules:
      - name: all_in-contrail-az-2
        direction: inbound
      - name: all_out-contrail-az-2
        direction: outbound
instances:
  - name: az-contrail-gw-2
    roles:
      - gateway
    provision: true
    username: ubuntu
    os: ubuntu16
    instance_type: Standard_F16s_v2
    subnets: subnet_contrail_az_2
    interface: eth1

  - name: compute-contrail-az-2
    provision: true
    username: ubuntu
    roles:
      - compute_node
    os: ubuntu16
    instance_type: Standard_F16s_v2
    subnets: subnet_contrail_az_2
    interface: eth0

```

5. (Optional) On Linux-based systems, when the ssh-agent is running, the deployer.sh can add the keys to ssh-agent. Use the following command to start ssh-agent.

```
eval `ssh-agent -s`
```

On Linux-based systems, if the added keys are removed during cluster provisioning, add the keys to the ssh-agent by using the following command.

```
ssh-add <path-to-keyfile>
```

For example:

```
ssh-add contrail-multi-cloud/keys/contrail-multicloud-key-7755
```

6. Set up the deployer.

```
# ./deployer.sh [-r registry -v <local|docker> -a access_key -s secret_key -k private_key ]
```

For example:

```
# ./deployer.sh -r <username> -t 5.0.1 -v $PWD:/root/multicloud -k
```

Use the password for the user on the local system. The contrail-multicloud-deployer deployer Docker container is created.

7. Log in to the deployer Docker container. Password for the root user is multicloud.

```
# ssh -o PreferredAuthentications=password -o PubkeyAuthentication=no -A root@127.0.0.1 -p 2222
```

8. Navigate to the multicloud directory.

```
# cd multicloud
```

9. Log in to Azure and authenticate your session.

- Register your device and log in to Azure. Using the `az login` command displays a secure link to the Azure portal and a code for device authentication.

```
# az login
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and
enter the code xxxxxxxx to authenticate.
```

- Use a Web browser to open the displayed URL <https://microsoft.com/devicelogin>.
- Enter the displayed code in the portal.
- Enter your Azure account login credentials.

Upon successful sign-in, your device and session is authenticated and you are logged into Azure.

10. (Optional) View your subscription details.

```
# az account list
```

11. Navigate to the **one-click-deployer** directory.

```
# cd one-click-deployer
```

12. Run the **deploy.sh** script to generate the topology and deploy Contrail. The **deploy.sh** script is available in the <https://github.com/Juniper/contrail-multi-cloud> repository.

```
# ./deploy.sh
```

13. (Optional) After Contrail deployment, if the kube-dns pod is stuck in CreatingContainer or ErrorCreating, ensure that the kube-dns pod is recreated. This might occur required if the container fails during provisioning.

Check for the kube-dns pod name using the following command.

```
kubectl get pods --all-namespaces | grep kube-dns | awk '{print $2}'
```

Delete kube-dns pod using the following command.

```
kubectl delete pod <kube-dns-xxxxx> -n kube-system
```

## Deleting Contrail Deployment from Azure

To delete Contrail from Azure, perform the following steps.

1. Navigate to the **one-click-deployer** directory.

```
# cd multicloud/one-click-deployer
```

2. Tear down the objects using the **teardown.sh** script.

```
./teardown.sh
```

3. Delete the deployer Docker, keys, and generated files.

```
cd contrail-multi-cloud
./cleanup.sh
```

## RELATED DOCUMENTATION

*Contrail Deployment on Microsoft Azure*

*On-Premise and Azure Public Cloud Deployment*

*Adding a Compute Host to Multicloud*

## On-Premise and Azure Multicloud Deployment

### IN THIS SECTION

- [Installing On-Premise Contrail | 135](#)
- [Extending On-Premise Contrail To Microsoft Azure | 138](#)

This topic describes the steps involved in deploying an on premise setup and extending it to Microsoft Azure cloud in two different availability zones. This topic also displays examples of the files involved in configuring the setup.

### Installing On-Premise Contrail

#### Before you begin

You must configure VLANs referring the irb interface units so that nodes are reachable to each other. Also, you must set static routes to the public cloud private network (here 17x.xx.1.0/24 and 17x.xx.3.0/24) on the switch to go through the gateway node (19x.xxx.2.1). Also, routes for the 17x.xx.0.0 network must be set on the controllers and computes to go out through the interface connected to the switch to the irb interface IP.

In real-time deployment scenarios, the controller nodes, compute nodes, and the gateway nodes are in different subnets in a data center. This example on-premise setup topology consists of one controller node, two compute nodes, and one multicloud gateway node.

## Configuration

The following steps describe how to bring up on-premise Contrail setup using contrail-ansible-deployer.

1. Ensure that the source files are available and that the ubuntu-16.04.3 installation is operational.

```
>> cat /etc/apt/sources.list
# # ##### Ubuntu Main Repos
deb http://us.archive.ubuntu.com/ubuntu/ xenial main restricted universe
# # ##### Ubuntu Update Repos
deb http://us.archive.ubuntu.com/ubuntu/ xenial-updates main restricted universe
>> cat /etc/apt/sources.list.d/ansible-ubuntu-ansible-2.4-xenial.list
deb http://ppa.launchpad.net/ansible/ansible-2.4/ubuntu xenial main
```

2. Run an update after changing the sources files.

```
>> apt-get update
```

3. Install the dependent packages needed for the contrail-ansible-deployer to bring up the on-premise setup.

```
>> apt-get install git ansible vim screen net-tools python-pip
```

4. To download the Ansible Deployer package file, follow these steps:

- a. Select Contrail version from the **Version** list in the Juniper Networks [Software Downloads](#) page.

- b. In the **Application Tools** section, click the Ansible Deployer **tgz** file.

You are now redirected to the **Software Download** page.

- c. Log in to the download page.

A **End User License Agreement** is displayed. Select **I Agree** and click on **Proceed**.

- d. Download the file on your localhost or on your device.

- e. Follow the **Usage Instructions** on the download page to install the file.

5. Create the **instances.yaml** file for Contrail deployment with OpenStack as orchestrator for on-premise deployments. Create **instances.yaml** under the **contrail-ansible-deployer/config/** directory as shown in the example below:

```
provider_config:
  bms:
    ssh_pwd: c0ntrail123
    ssh_user: root
    ssh_public_key:
    ssh_private_key:
```



```

ntpserver: 10.204.217.158
domainsuffix: device.example.net

instances:
  bms1:
    provider: bms
    ip: 19x.xxx.1.1
    private_ip: 19x.xxx.1.1
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      openstack:

  bms2:
    provider: bms
    ip: 19x.xxx.1.2
    private_ip: 19x.xxx.1.2
    roles:
      openstack_compute:
      vrouter:

  bms3:
    provider: bms
    ip: 19x.xxx.1.3
    private_ip: 19x.xxx.1.3
    roles:
      openstack_compute:
      vrouter:

global_configuration:
  CONTAINER_REGISTRY: "hub.juniper.net/contrail"
  CONTAINER_REGISTRY_USERNAME: < username >
  CONTAINER_REGISTRY_PASSWORD: f*****7

contrail_configuration:
  CLOUD_ORCHESTRATOR: openstack
  OPENSTACK_VERSION: queens
  CONTRAIL_CONTAINER_TAG: 5.0.2-0.360-queens
  CONTROLLER_NODES: 19x.xxx.1.1

```

```

CONTROL_NODES: 19x.xxx.1.1
VROUTER_GATEWAY: 19x.xxx.1.254
RABBITMQ_NODE_PORT: 5673
KEYSTONE_AUTH_HOST: 19x.xxx.1.1
KEYSTONE_AUTH_ADMIN_PASSWORD: c0ntrail123
KEYSTONE_AUTH_URL_VERSION: /v3

kolla_config:
  kolla_globals:
    contrail_api_interface_address: 19x.xxx.1.1
    enable_haproxy: "no"
    enable_ironic: "no"
    enable_swift: "no"

  kolla_passwords:
    metadata_secret: c0ntrail123
    keystone_admin_password: c0ntrail123

```

6. Navigate to the **contrail-ansible-deployer** directory to run the required Ansible playbooks for OpenStack and Contrail installation.

```
>> cd contrail-ansible-deployer/
```

7. Set up basic packages and check out Kolla Ansible for OpenStack installation.

```
>> ansible-playbook -i inventory/ -e orchestrator=openstack playbooks/configure_instances.yml
```

8. Provision Openstack.

```
>> ansible-playbook -i inventory/ playbooks/install_openstack.yml
```

9. Provision Contrail.

```
>> ansible-playbook -i inventory/ -e orchestrator=openstack playbooks/install_contrail.yml
```

## Extending On-Premise Contrail To Microsoft Azure

### Before you begin

You need a deployer node from where to orchestrate the bringing up of the cloud setup on Azure and on-premise gateways. Ensure that you have Git access from the deployer node and also that you have Docker containers on the node.

The Azure cloud setup described in this example consists of two availability zones. Each availability zone has two multicloud gateway nodes in HA and two compute nodes. To bring up the Azure setup you need an Azure account with login credentials.

### Configuration

The following steps describe how to bring up of the cloud setup on Azure and on-premise gateways using contrail-multi-cloud deployer.

1. Use one of the following methods to download the deployer package.

- Untar the deployer package.

```
>> tar -xvzf contrail-multicloud-deployer-5.0.2-0.XXX.tgz
```

- Git clone the **contrail-multi-cloud.git** repository.

```
>> git clone -b R5.0 https://github.com/Juniper/contrail-multi-cloud.git
```

2. Navigate to the deployer directory.

```
>> cd contrail-multi-cloud/
```

3. Start the ssh agent if it is not already running.

```
>> ssh-add -l
```

The agent has no identities.

```
>> eval $(ssh-agent -s)
Agent pid 18333
```

4. Edit the **topology.yml** topology file under the **contrail-multi-cloud/** directory.

```
- provider: OnPrem
  organization: Juniper
  project: multicloud
  instances:
    - name: nodec33
      roles:
        - gateway
      provision: true
      username: root
      password: c0ntrail123
      public_ip: 1x.xxx.217.168
      private_ip: 19x.xxx.2.1
      private_subnet:
        - 19x.xxx.2.0/24
        - 19x.xxx.1.0/24
      protocols_mode:
        - ssl_client
      interface: enp1s0f1
      gateway: 19x.xxx.2.254
```

```

- name: nodec28
  roles:
    - controller: false
    - k8s_master
  provision: true
  username: root
  password: c0ntrail123
  public_ip: 1x.xxx.217.13
  private_ip: 19x.xxx.1.1
  private_subnet: 19x.xxx.1.0/24
  interface: enp1s0f1
- name: nodec10
  roles:
    - compute_node: false
    - k8s_node
  provision: true
  username: root
  password: c0ntrail123
  public_ip: 1x.xxx.217.176
  private_ip: 19x.xxx.1.2
  private_subnet: 19x.xxx.1.0/24
  interface: enp1s0f1
  gateway: 19x.xxx.1.254
- name: nodec50
  roles:
    - compute_node: false
    - k8s_node
  provision: true
  username: root
  password: c0ntrail123
  public_ip: 1x.xxx.217.153
  private_ip: 19x.xxx.1.3
  private_subnet: 19x.xxx.1.0/24
  interface: enp1s0f1
  gateway: 19x.xxx.1.254
- provider: azure
  organization: Juniper
  project: multicloud
  regions:
    - name: WestUS2
      resource_group: contrail-test-west-us-2
      vnet:
        - name: rg-vpc-1

```

```

cidr_block: 17x.xx.1.0/24
subnets:
  - name: rg-subnet-1
    cidr_block: 17x.xx.1.0/25
    security_group: rg-sg-1
security_groups:
  - name: rg-sg-1
    rules:
      - name: rg-all_in_1
        direction: inbound
      - name: rg-all_out_1
        direction: outbound
instances:
- name: rg-gw-1
  availability_zone: 1
  provision: true
  username: ubuntu
  os: ubuntu16
  os_version: 16.04.201705080
  instance_type: Standard_F2
  subnets: rg-subnet-1
  interface: eth1
  roles:
    - gateway
  protocols_mode:
    - ssl_server
    - ipsec_server
    - ipsec_client
- name: rg-gw-2
  availability_zone: 1
  provision: true
  username: ubuntu
  os: ubuntu16
  os_version: 16.04.201705080
  instance_type: Standard_F2
  subnets: rg-subnet-1
  interface: eth1
  roles:
    - gateway
  protocols_mode:
    - ssl_server
    - ipsec_server
    - ipsec_client

```

```

- name: rg-compute-1
  availability_zone: 1
  provision: true
  username: ubuntu
  os: ubuntu16
  os_version: 16.04.201705080
  instance_type: Standard_F2
  subnets: rg-subnet-1
  interface: eth0
  roles:
    - compute_node
- name: rg-compute-2
  availability_zone: 1
  provision: true
  username: ubuntu
  os: ubuntu16
  os_version: 16.04.201705080
  instance_type: Standard_F2
  subnets: rg-subnet-1
  interface: eth0
  roles:
    - compute_node
- name: rg-vpc-2
  cidr_block: 17x.xx.3.0/24
  subnets:
    - name: rg-subnet-2
      cidr_block: 17x.xx.3.0/25
      security_group: rg-sg-2
  security_groups:
    - name: rg-sg-2
      rules:
        - name: rg-all_in_2
          direction: inbound
        - name: rg-all_out_2
          direction: outbound
  instances:
    - name: rg-gw-21
      availability_zone: 2
      provision: true
      username: ubuntu
      os: ubuntu16
      os_version: 16.04.201705080
      instance_type: Standard_F2

```

```

subnets: rg-subnet-2
interface: eth1
roles:
  - gateway
protocols_mode:
  - ssl_server
  - ipsec_server
  - ipsec_client
- name: rg-gw-22
  availability_zone: 2
  provision: true
  username: ubuntu
  os: ubuntu16
  os_version: 16.04.201705080
  instance_type: Standard_F2
  subnets: rg-subnet-2
  interface: eth1
  roles:
    - gateway
  protocols_mode:
    - ssl_server
    - ipsec_server
    - ipsec_client
- name: rg-compute-21
  availability_zone: 2
  provision: true
  username: ubuntu
  os: ubuntu16
  os_version: 16.04.201705080
  instance_type: Standard_F2
  subnets: rg-subnet-2
  interface: eth0
  roles:
    - compute_node
- name: rg-compute-22
  availability_zone: 2
  provision: true
  username: ubuntu
  os: ubuntu16
  os_version: 16.04.201705080
  instance_type: Standard_F2
  subnets: rg-subnet-2
  interface: eth0

```

```
roles:
  - compute_node
```

5. Edit the following **common.yml** Ansible files.

- contrail common yaml file

```
# vi contrail-multi-cloud/ansible/contrail/common.yml
ANSIBLE_DEPLOYER_BRANCH: R5.0
global_configuration:
  #REGISTRY_PRIVATE_INSECURE: TRUE
  CONTAINER_REGISTRY: "hub.juniper.net/contrail"
  CONTAINER_REGISTRY_USERNAME: XXXX
  CONTAINER_REGISTRY_PASSWORD: XXXX
contrail_user: admin
contrail_password: c0ntrail123
contrail_tenant: admin
contrail_port: 8082
contrail_tenant: default-project
provider_config:
  bms:
    ssh_pwd: c0ntrail123
    ssh_user: root
contrail_configuration:
  CONTRAIL_VERSION: 5.0.2-0.360
  ENCAP_PRIORITY: "MPLSoUDP,MPLSoGRE,VXLAN"
  CLOUD_ORCHESTRATOR: kubernetes
# VROUTER_ENCRYPTION: True
```

- gateway common yaml file

```
# vi contrail-multi-cloud/ansible/gateway/common.yml
PATH_CONFIG: "/etc/multicloud"
PATH_SSL_CONFIG_LOCAL: "~/.multicloud/ssl"
PATH_SSL_CONFIG: "{{ PATH_CONFIG }}/ssl"
PATH_OPENVPN_CONFIG: "{{ PATH_CONFIG }}/openvpn"
PATH_BIRD_CONFIG: "{{ PATH_CONFIG }}/bird"
PATH_STRONGSWAN_CONFIG: "{{ PATH_CONFIG }}/strongswan"
PATH_VRRP_CONFIG: "{{ PATH_CONFIG }}/vrrp"
PATH_AWS_CONFIG: "{{ PATH_CONFIG }}/aws"
PATH_INTERFACE_CONFIG: "/etc/network/interfaces.d"
PATH_FW_CONFIG: "{{ PATH_CONFIG }}/firewall"
```



```

PATH_GCP_CONFIG: "{{ PATH_CONFIG }}/gcp"
PATH_SECRET_CONFIG: "{{ PATH_CONFIG }}/secret"
CONTAINER_REGISTRY: "hub.juniper.net/contrail"
CONTRAIL_MULTICLOUD_VERSION: 5.0.2-0.360
CONTAINER_REGISTRY_USERNAME: XXXX
CONTAINER_REGISTRY_PASSWORD: XXXX
UPGRADE_KERNEL: False
AS: 65000
vpn_lo_network: 1xx.x5.0.0/16
vpn_network: 1xx.x4.0.0/16
required_bgp_rrs: 1
openvpn_port: 443
local_interface: eth1
bfd_interval: 200ms
bfd_multiplier: 5
bfd_interval_multihop: 500ms
bfd_multiplier_multihop: 5
core_bgp_secret: bgp_

```

6. Log in to the Docker repository to check out the **contrail-multi-cloud** deployer container.  
 >> docker login hub.juniper.net/contrail-nightly -u XXX -p xxx
7. Start the deployer container and mount the deployer files to the container.  
 >> ./deployer.sh -r hub.juniper.net/contrail-nightly -t 5.0.2-0.349 -v \$PWD:/root/multicloud -k
8. Check if the generated key file is added to the ssh agent. If the key file has not been added, add it manually.  
 >> ssh-add -l  
 >> ssh-add keys/<keyfile>
9. Log in to the container with default multicloud password.  
 >> ssh -o PreferredAuthentications=password -o PubkeyAuthentication=no -A root@127.0.0.1 -p 2222
10. Ensure that the ssh agent is running and contains the correct key file. Else, start the agent and add the keyfile.  
 >> ssh-add -l
11. Check if the ssh-keys are added to all the on-premise nodes for Ansible playbooks to succeed.  
 >> sshpass -p \"c0ntrail123\" ssh-copy-id -o StrictHostKeyChecking=no -i /root/multicloud/keys/contrail-multicloud-key-11513.pub root@<OnPrem nodes mgmt IP>
12. (Optional) If step 11 fails, use the following command to add keys to all four on-premise nodes.  
 >> ssh-copy-id root@<OnPrem nodes mgmt IP>
13. Register your device and log in to Azure.

Using the `az login` command displays a secure link to the Azure portal and a code for device authentication.

- `# az login`  
To sign in, use a web browser to open the page <https://microsoft.com/devicelogin> and enter the code <XXXXXX> to authenticate.
- Use a Web browser to open the displayed URL <https://microsoft.com/devicelogin>.
- Enter the displayed code in the portal.
- Enter your Azure account login credentials.

Upon successful sign-in, your device and session is authenticated and you are logged into Azure.

```
# az login
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter
the code <XXXXXX> to authenticate.
[
{
  "cloudName": "AzureCloud",
  "id": "0e42336f-d930-41f9-9661-582c24337897",
  "isDefault": true,
  "name": "Pay-As-You-Go",
  "state": "Enabled",
  "tenantId": "bea78b3c-4cdb-4130-854a-1d193232e5f4",
  "user": {
    "name": "username@juniper.net",
    "type": "user"
  }
}
]
```

14. Navigate to the **one-click-deployer** directory in the container and use the deploy script to bring up the setup on Azure.

```
>> cd multicloud/one-click-deployer/
```

```
>> ./deploy.sh
```

You can now log in to the Azure portal and view your resources. You can also use the following `ssh` key file and log in to the Azure VMs from the on-premise machines or to the public IP addresses on Azure from any server.

- `ssh -i contrail-multicloud-key-11513 ubuntu@17x.xx.1.5`

- `ssh -i contrail-multicloud-key-11513 ubuntu@<public-ip-of-Azure-GW-VM>`

## RELATED DOCUMENTATION

*Contrail Deployment on Microsoft Azure*

*Deploying Contrail on Microsoft Azure*

*Adding a Compute Host to Multicloud*

## Modifying Multicloud Topology

You can modify the multicloud topology if the existing deployment capabilities are low. Perform the following steps to add a new compute host to the VPC as well as a new VPC altogether.

1. Edit the **topology.yml** to reflect your new topology. Ensure that the new VPC uses a different IP address pool.

```
# vi topology.yml
```

2. Navigate to the **one-click-deployer** directory.

```
# cd multicloud/one-click-deployer
```

3. Run the **modify.sh** script to generate the topology and deploy Contrail. The **modify.sh** script is available in <https://github.com/Juniper/contrail-multi-cloud/tree/master/one-click-deployer>.

```
# ./modify.sh
```



**NOTE:** If you are not able to access the <https://github.com/Juniper/contrail-multi-cloud/tree/master/one-click-deployer> page, it may be because of a permission issue. Request the owner of the page for necessary permissions.

## RELATED DOCUMENTATION

*Contrail Deployment on Microsoft Azure*

## Deploying Contrail Enterprise Multicloud using REST API

### IN THIS SECTION

- Prerequisites and Assumptions | 148
- Objective and Workflow | 149
- Deploying the Public Cloud Infrastructure | 149
- Creating Contrail Roles Specific to Public Cloud Instances | 155
- Creating On-Premise Cloud Objects | 156
- Extending On-Premise Contrail Cluster to Public Cloud | 164

This section explains how to deploy Contrail Enterprise Multicloud (CEM) using REST API.

### Prerequisites and Assumptions

The following are the assumptions for Contrail Enterprise Multicloud deployment:

- Contrail Controller cluster is already deployed on the on-premise side. On-premise implies that the software is installed locally on the organization or data center servers.
- All nodes on the on-premise side have a management IP (declared in node object as `ip_address` field) control/data IP as a child port object `ip_address` field.
- Ensure that static routes are added in the on-premise multicloud gateway, Contrail controller, and top-of-rack (TOR) node.
- Ensure that the on-premise Contrail cluster and multicloud gateway are in two different subnets.
- `contrail-api` is listening on the control-data IP.
- Understand the `contrail-go-api` server tools and concepts.
  - <https://github.com/Juniper/contrail/blob/master/doc/index.md>
  - <https://github.com/Juniper/contrail/blob/master/doc/cli.md>

- [https://github.com/Juniper/contrail/blob/master/doc/rest\\_api.md](https://github.com/Juniper/contrail/blob/master/doc/rest_api.md)
- [https://github.com/Juniper/contrail/blob/master/doc/rest\\_api.md#sync-api](https://github.com/Juniper/contrail/blob/master/doc/rest_api.md#sync-api)

## Objective and Workflow

The deployment consists of the following steps:

1. Create an entire public cloud infrastructure that includes Virtual Private Cloud (VPC)/virtual network, virtual machines, routes, and so on.
2. Deploy multicloud gateway roles for both on-premise site and public cloud sites.
3. Deploy Contrail and Kubernetes components needed on the public cloud site.
4. Establish connectivity between on-premise site and public cloud.

## Deploying the Public Cloud Infrastructure

When deploying the following example `deploy_public_cloud_infra.yml` file, multiple resources for Amazon Web Services (AWS) infrastructure are created. Summarized are important resources created using this yaml file.

- One VPC (192.168.100.0/24)
- One private subnet (192.168.100.128/25).
- Two security group rules.
- Two Elastic Compute Cloud (EC2) instances (gateway and compute)

Verify that the correct access key and secret key are entered in the `cloud_user` object.

Example: `deploy_public_cloud_infra.yml`

```
---
resources:
- data:
    name: public_cloud_tag
    uuid: efd769a8-2e6c-11e9-b210-d663bd873d93
    fq_name:
    - public_cloud_tag
    tag_type_name: label
    tag_value: public_cloud_provider_aws
```

```

kind: tag
operation: CREATE
- data:
  name: public_cloud_key
  uuid: 4e77005b-b7ba-489b-9891-8472cee8ghts
  parent_type: global-system-config
  fq_name:
  - default-global-system-config
  - public_cloud_key
kind: keypair
operation: CREATE
- data:
  name: public_cloud_credential
  uuid: 9d0fffff-3fd8-439c-bdb2-ff5800497579
  parent_type: global-system-config
  fq_name:
  - default-global-system-config
  - public_cloud_credential
  ssh_user: ec2-user
  keypair_refs:
  - uuid: 4e77005b-b7ba-489b-9891-8472cee8ghts
kind: credential
operation: CREATE
- data:
  uuid: 4e77005b-b7ba-489b-9891-8472cee9eadf
  name: public_cloud_user
  fq_name:
  - public_cloud_user
  perms2:
    owner: admin
  aws_credential:
    access_key: xxxxxxxxx
    secret_key: YYYYYYYYYYYYYYYYYYYY
  credential_refs:
  - uuid: 9d0fffff-3fd8-439c-bdb2-ff5800497579
kind: cloud_user
operation: CREATE
- data:
  provisioning_state: CREATED
  uuid: dfb40e0d-c9f4-47cd-bd5c-1efdd28fd4fc
  name: public_cloud
  fq_name:
  - public_cloud

```

```

    perms2:
      owner: admin
    organization: test
    project: 5.0.3
    cloud_user_refs:
      - uuid: 4e77005b-b7ba-489b-9891-8472cee9eadf
  kind: cloud
  operation: CREATE
- data:
  name: public_cloud_provider
  parent_type: cloud
  fq_name:
    - public_cloud
    - public_cloud_provider
  perms2:
    owner: admin
  type: aws
  kind: cloud_provider
  operation: CREATE
- data:
  name: us-west-1
  parent_type: cloud-provider
  fq_name:
    - public_cloud
    - public_cloud_provider
    - public_cloud_region
  display_name: public_cloud_region
  perms2:
    owner: admin
  kind: cloud_region
  operation: CREATE
- data:
  name: publc_virtual_cloud
  parent_type: cloud-region
  fq_name:
    - public_cloud
    - public_cloud_provider
    - public_cloud_region
    - publc_virtual_cloud
  perms2:
    owner: admin
  cidr_block: 192.168.100.0/24
  tag_refs:

```

```

    - uuid: efd769a8-2e6c-11e9-b210-d663bd873d93
kind: virtual_cloud
operation: CREATE
- data:
  uuid: 4bd887b1-3f65-59c1-bc2f-dfbaeac5798d
  name: public_cloud_private_subnet
  parent_type: virtual-cloud
  fq_name:
    - public_cloud
    - public_cloud_provider
    - public_cloud_region
    - public_virtual_cloud
    - public_cloud_private_subnet
  perms2:
    owner: admin
  cidr_block: 192.168.100.128/25
  availability_zone: a
kind: cloud_private_subnet
operation: CREATE
- data:
  uuid: 4bd887b1-3f65-59c1-bc2f-dfbaeac57123
  name: public_cloud_security_group
  parent_type: virtual-cloud
  fq_name:
    - public_cloud
    - public_cloud_provider
    - public_cloud_region
    - public_virtual_cloud
    - public_cloud_security_group
  perms2:
    owner: admin
kind: cloud_security_group
operation: CREATE
- data:
  name: public_cloud_security_group_rule_ingress
  parent_type: cloud-security-group
  fq_name:
    - public_cloud
    - public_cloud_provider
    - public_cloud_region
    - public_virtual_cloud
    - public_cloud_security_group
    - public_cloud_security_group_rule_ingress

```



```

perms2:
  owner: admin
  direction: ingress
  protocol: "-1"
  from_port: 0
  to_port: 0
  cidr_block: 0.0.0.0/0
kind: cloud_security_group_rule
operation: CREATE
- data:
  name: public_cloud_security_group_rule_egress
  parent_type: cloud-security-group
  fq_name:
    - public_cloud
    - public_cloud_provider
    - public_cloud_region
    - public_virtual_cloud
    - public_cloud_security_group
    - public_cloud_security_group_rule_egress
  perms2:
    owner: admin
    direction: egress
    protocol: "-1"
    from_port: 0
    to_port: 0
    cidr_block: 0.0.0.0/0
  kind: cloud_security_group_rule
  operation: CREATE
- data:
  uuid: 4bd887b1-3f65-59c1-bc2f-dfbaenew43526
  name: public_gateway_node
  parent_type: global-system-config
  fq_name:
    - default-global-system-config
    - public_gateway_node
  perms2:
    owner: admin
  hostname: gateway
  interface_name: eth1
  type: private
  cloud_info:
    availability_zone: a
    machine_id: ami-18726478

```

```

    instance_type: t2.xlarge
    roles:
    - gateway
  cloud_private_subnet_refs:
  - uuid: 4bd887b1-3f65-59c1-bc2f-dfbaeac5798d
  credential_refs:
  - uuid: 9d0fffff-3fd8-439c-bdb2-ff5800497579
  cloud_security_group_refs:
  - uuid: 4bd887b1-3f65-59c1-bc2f-dfbaeac57123
  tag_refs:
  - uuid: efd769a8-2e6c-11e9-b210-d663bd873d93
kind: node
operation: CREATE
- data:
  uuid: 4bd887b1-3f65-59c1-bc2f-dfbaenew43634
  name: public_compute_node
  parent_type: global-system-config
  fq_name:
  - default-global-system-config
  - public_compute_node
  perms2:
    owner: admin
  hostname: compute
  interface_name: eth0
  type: private
  cloud_info:
    availability_zone: a
    machine_id: ami-18726478
    instance_type: t2.xlarge
    volume_size: 24
    roles:
    - compute
  cloud_private_subnet_refs:
  - uuid: 4bd887b1-3f65-59c1-bc2f-dfbaeac5798d
  credential_refs:
  - uuid: 9d0fffff-3fd8-439c-bdb2-ff5800497579
  cloud_security_group_refs:
  - uuid: 4bd887b1-3f65-59c1-bc2f-dfbaeac57123
  tag_refs:
  - uuid: efd769a8-2e6c-11e9-b210-d663bd873d93
kind: node
operation: CREATE

```

## Creating Contrail Roles Specific to Public Cloud Instances

Use these guidelines in the `create_contrail_roles_for_cloud_objects.yml` file in this procedure:

- `a5063dde-2681-11e9-8021-0050568a3bf0` is the `contrail_cluster` UUID. Also `a50635c8-2681-11e9-8021-0050568a3bf0` is the `kubernetes_cluster` UUID. It is assumed that both UUIDs are already created.
- `node_refs` is the UUID of the nodes that were created in the previous topic “Deploy Public Cloud Infrastructure.”
- `parent_uuid` is the UUID of the `kubernetes_cluster` for the `kubernetes_node` role object.
- For `contrail_multicloud_gw_node` and `contrail_vrouter_node` role, object `parent_uuid` is the `contrail_cluster` objects UUID.

To create Contrail roles specific to public cloud instances, perform the following steps:

1. Enter these requests to locate the Contrail and Kubernetes cluster UUIDs.

```
contrailcli list contrail_cluster | grep uuid
contrailcli list kubernetes_cluster | grep uuid
```

2. Use the following request payload to create the contrail roles for cloud objects.

Example: `create_contrail_roles_for_cloud_objects.yml`

```
---
resources:
- data:
    name: public_contrail_multicloud_gw_node
    node_refs:
    - uuid: 4bd887b1-3f65-59c1-bc2f-dfbaenew43526
    protocols_mode:
    - ssl_server
    - ipsec_server
    - ipsec_client
    parent_type: contrail-cluster
    parent_uuid: a5063dde-2681-11e9-8021-0050568a3bf0
    kind: contrail_multicloud_gw_node
    operation: CREATE
```

```

- data:
  name: public_kubernetes_node
  node_refs:
  - uuid: 4bd887b1-3f65-59c1-bc2f-dfbaenew43634
  parent_type: kubernetes-cluster
  parent_uuid: a50635c8-2681-11e9-8021-0050568a3bf0
  kind: kubernetes_node
  operation: CREATE

- data:
  name: public_contrail_vrouter_node
  node_refs:
  - uuid: 4bd887b1-3f65-59c1-bc2f-dfbaenew43634
  parent_type: contrail-cluster
  parent_uuid: a5063dde-2681-11e9-8021-0050568a3bf0
  kind: contrail_vrouter_node
  operation: CREATE

```

3. Update the provisioning\_state of cloud object to NOSTATE to trigger the deployment of the public cloud.

```

- data:
  provisioning_state: NOSTATE
  uuid: dfb40e0d-c9f4-47cd-bd5c-1efdd28fd4fc
  kind: cloud
  Operation: UPDATE

```

Wait for the cloud deployment logs in `/var/log/contrail/cloud.log` to complete before proceeding to the next steps. When completed, the provisioning\_state of the cloud resource change from NOSTATE to either UPDATED or UPDATE\_FAILED.

## Creating On-Premise Cloud Objects

In the following `create_onprem_pvt_port.yml` file, node objects were already created. You are updating the `cloud_private_subnet` and `tag_refs`. Per the requirement, you need to have multicloud gateway and other roles on the on-premise cluster (Contrail controller, Kubernetes nodes, OpenStack nodes) on two different networks connected through a TOR. Hence, in the following yaml file there are two private subnets created. Be careful when adding the `cloud_private_subnet_refs` to the nodes.

To create on-premise cloud objects:

1. Create the private port.

If the private interface is not already created for the on-premise cluster nodes by using the UI, create them here. In the following example file, you are using the UUID of each on-premise cluster node resource.

Example: create\_onprem\_pvt\_port.yml

```
---
resources:
# Create private interface for onprem compute node
- data:
    parent_type: node
    parent_uuid: 4bd887b1-3f65-59c1-bc2f-dfbaenew43634
    name: bond0
    ip_address: 192.168.1.2
    pxe_enabled: false
    kind: port

# Create private interface for onprem controller
- data:
    parent_type: node
    parent_uuid: c8d9d4ec-2f4a-11e9-bfac-0050568a3bf0
    name: bond0
    ip_address: 192.168.1.1
    pxe_enabled: false
    kind: port
```

2. Create onprem tag. This is used later to link on-premise nodes to the on-premise virtual\_cloud.

Example: create\_onprem\_tag.yml

```
---
resources:
- data:
    name: onprem_cloud_tag
    uuid: 2eefeb06-2e7c-11e9-b210-d663bd873d93
    fq_name:
      - onprem_cloud_tag
    tag_type_name: label
    tag_value: onprem_cloud_provider
```

```
kind: tag
Operation: CREATE
```

3. Update the on-premise credential with the public cloud keypair reference.

- a. Use the UUID of the already created credential resource. List the credentials using the following requests to obtain the UUID.

```
contrailcli list contrail_control_node -d | grep -A 1 node_refs
contrailcli show node <uuidOfNodeRefsFromPreviousCommand> | grep -A 1 credential_refs
```

- b. In `keypair_refs` use the UUID of the keypair created as part of deploying public cloud in the previous topic “Deploy Public Cloud Infrastructure.”

Example: `update_onprem_keypair.yml`

```
---
resources:
- data:
    uuid: c8d9bf8e-2f4a-11e9-bfac-0050568a3bf0
    keypair_refs:
    - uuid: 4e77005b-b7ba-489b-9891-8472cee8ghts
    kind: credential
    operation: UPDATE
```

4. Create the on-premise `cloud_user` with the `credential_refs` pointing to the on-premise credential UUID, that was updated in Step “3” on page 158.

Example: `create_onprem_clouduser.yml`

```
---
resources:
- data:
    uuid: 4e77005b-b7ba-489b-9891-aGFjawo9eadf
    name: onprem_cloud_user
    fq_name:
    - onprem_user
```

```

credential_refs:
- uuid: c8d9bf8e-2f4a-11e9-bfac-0050568a3bf0
perms2:
  owner: admin
kind: cloud_user
operation: CREATE

```

## 5. Create the on-premise cloud objects.

Cloud object refers to `cloud_user` created in Step "4" on page 158 and the `virtual_cloud` reference tag created in Step "2" on page 157.

Example: `create_onprem_cloud_objects.yml`

```

---
resources:
- data:
    provisioning_state: CREATED
    uuid: dfb40e0d-c9f4-47cd-bd5c-MWVmZGQyOGZkNGZjCg
    name: onprem_cloud
    fq_name:
    - onprem_cloud
    perms2:
      owner: admin
    organization: juniper
    project: juniper-private
    cloud_user_refs:
    - uuid: 4e77005b-b7ba-489b-9891-aGFjawo9eadf
    kind: cloud
    operation: CREATE

- data:
    name: onprem_cloud_provider
    parent_type: cloud
    fq_name:
    - onprem_cloud
    - onprem_cloud_provider
    perms2:
      owner: admin
    type: private
    kind: cloud_provider

```

```
operation: CREATE
```

```
- data:
```

```
  name: onprem_cloud_region
  parent_type: cloud-provider
  fq_name:
    - onprem_cloud
    - onprem_cloud_provider
    - onprem_cloud_region
  perms2:
```

```
    owner: admin
```

```
kind: cloud_region
```

```
operation: CREATE
```

```
- data:
```

```
  name: onprem_virtual_cloud
  parent_type: cloud-region
  fq_name:
    - onprem_cloud
    - onprem_cloud_provider
    - onprem_cloud_region
    - onprem_virtual_cloud
  perms2:
```

```
    owner: admin
```

```
  tag_refs:
```

```
    - uuid: 2eefeb06-2e7c-11e9-b210-d663bd873d93
```

```
kind: virtual_cloud
```

```
operation: CREATE
```

```
- data:
```

```
  name: onprem_cloud_private_subnet
  uuid: 5ecfeb06-2e7c-11e9-b210-d663bd873d93
  parent_type: virtual-cloud
  fq_name:
    - onprem_cloud
    - onprem_cloud_provider
    - onprem_cloud_region
    - onprem_virtual_cloud
    - onprem_cloud_private_subnet
  perms2:
```

```
    owner: admin
```

```
  cidr_block: 192.168.1.0/24
```

```
kind: cloud_private_subnet
```



```

operation: CREATE

- data:
  name: onprem_cloud_private_subnet_gw
  uuid: 3defeb06-2e7c-11e9-b210-d663bd873d93
  parent_type: virtual-cloud
  fq_name:
    - onprem_cloud
    - onprem_cloud_provider
    - onprem_cloud_region
    - onprem_virtual_cloud
    - onprem_cloud_private_subnet_gw
  perms2:
    owner: admin
  cidr_block: 192.168.2.0/24
  kind: cloud_private_subnet
  operation: CREATE

```

## 6. Create the on-premise gateway node.

- tag\_refs, credential\_refs, and cloud\_private\_subnet\_refs are the UUID of the respective resources created or updated in [Step "2" on page 157](#), [Step "3" on page 158](#), and [Step "5" on page 159](#) respectively.
- cloud\_private\_subnet\_refs is specifically from the cloud\_private\_subnet created for the on-premise gateway.

Example: create\_onprem\_mcgw\_node.yml

```

---
resources:
- data:
  uuid: 41f99f2d-a5a4-4e2c-b598-c173cf748953
  name: onprem_gateway
  type: private
  hostname: onprem_gateway
  ip_address: 10.87.74.132
  interface_name: eno1
  fq_name:
    - default-global-system-config
    - onpre_virtual_cloud
  parent_type: global-system-config

```

```

tag_refs:
- uuid: 2eefeb06-2e7c-11e9-b210-d663bd873d93
credential_refs:
- uuid: c8d9bf8e-2f4a-11e9-bfac-0050568a3bf0
cloud_private_subnet_refs:
- uuid: 3defeb06-2e7c-11e9-b210-d663bd873d93
kind: node

# Create private interface for onprem gateway
- data:
  parent_type: node
  parent_uuid: 41f99f2d-a5a4-4e2c-b598-c173cf748953
  name: bond0
  ip_address: 192.168.2.1
  pxe_enabled: false
  kind: port

```

7. Create the on-premise `contrail_multicloud_gateway_node` role and update `parent_uuid` with `contrail_cluster` UUID.

Use the following request to get the `contrail_cluster` UUID:

```
contrailcli list contrail_cluster | grep uuid
```

Update `node_refs` UUID with the gateway node created earlier in this step.

Example: `create_onprem_mcgw_node_role.yml`

```

---
resources:
- data:
  name: onprem_contrail_multicloud_gw_node
  node_refs:
  - uuid: 41f99f2d-a5a4-4e2c-b598-c173cf748953
  protocols_mode:
  - ssl_client
  default_gateway: 192.168.2.254
  parent_type: contrail-cluster
  parent_uuid: a5063dde-2681-11e9-8021-0050568a3bf0
  kind: contrail_multicloud_gw_node

```

## 8. Update the on-premise compute and controller node.

Link the on-premise cluster nodes (compute/controller) to the `virtual_cloud` created for the on-premise cluster using tag. Use the UUID of the node object created using the UI as part of the Contrail cluster deployment.

Use the following request to get the node UUID:

```
contrailcli list contrail_control_node | grep uuid
contrailcli list contrail_vrouter_node | grep uuid
```

`tag_refs`, and `cloud_private_subnet_refs` are the UUID of the respective resources created or updated in [Step "2" on page 157](#) and [Step "5" on page 159](#).

Example: `update_onprem_nodes.yml`

```
---
resources:
#Link onprem cluster nodes to the virtual_cloud created for onprem cluster
- data:
  uuid: c8d9d4ec-2f4a-11e9-bfac-0050568a3bf0
  cloud_private_subnet_refs:
  - uuid: 5ecfeb06-2e7c-11e9-b210-d663bd873d93
  tag_refs:
  - uuid: 2eefeb06-2e7c-11e9-b210-d663bd873d93
  kind: node
  operation: UPDATE

- data:
  uuid: c8d9c1b4-2f4a-11e9-bfac-0050568a3bf0
  cloud_private_subnet_refs:
  - uuid: 5ecfeb06-2e7c-11e9-b210-d663bd873d93
  tag_refs:
  - uuid: 2eefeb06-2e7c-11e9-b210-d663bd873d93
  kind: node
  operation: UPDATE
```

9. Update the on-premise cloud state with NOSTATE to trigger deployment of the on-premise cloud. Use the `onprem_cloud` objects UUID created in Step "5" on page 159.

```
---
resources:
- data:
    provisioning_state: NOSTATE
    uuid: dfb40e0d-c9f4-47cd-bd5c-MWVmZGQyOGZkNGZjCg
    kind: cloud
    operation: UPDATE
```

Wait for the cloud deployment logs in `/var/log/contrail/cloud.log` to complete before proceeding to the next steps. When completed, the `provisioning_state` of the cloud resource changes from NOSTATE to either UPDATED or UPDATE\_FAILED.

## Extending On-Premise Contrail Cluster to Public Cloud

To extend the on-premise Contrail cluster to the public cloud:

1. Use the following request to get the cloud UUIDs.

```
contrailcli list cloud | grep uuid
```

2. Use the following request to get the UUID of the `contrail_cluster`.

```
contrailcli list contrail_cluster | grep uuid
```

3. Run the following request payload to extend the on-premise Contrail cluster to the public cloud.

Example: `extend_onprem_to_coud.yml`

```
---
resources:
- data:
    uuid: a5063dde-2681-11e9-8021-0050568a3bf0
    provisioning_state: NOSTATE
    provisioning_action: ADD_CLOUD
    cloud_refs:
```

```

- uuid: dfb40e0d-c9f4-47cd-bd5c-MWVmZGQyOGZkNGZjCg
- uuid: dfb40e0d-c9f4-47cd-bd5c-1efdd28fd4fc
mc_gw_info:
  AS: 65000
  openvpn_port: 443
  vpn_lo_network: 100.65.0.0/16
  vpn_network: 100.64.0.0/16
  bfd_interval: 200ms
  bfd_multiplier: 5
  bfd_interval_multihop: 500ms
  bfd_multiplier_multihop: 5
kind: contrail_cluster
operation: UPDATE

```

With this request, you trigger the Contrail multicloud Ansible playbooks to start deploying Contrail roles on the public cloud, which includes the Contrail multicloud gateway role.

Wait for the cloud deployment logs in **/var/log/contrail/cloud.log** to complete before proceeding to the next steps. When completed, the `provisioning_state` of the cloud resource changes from `NOSTATE` to either `UPDATED` or `UPDATE_FAILED`.

## RELATED DOCUMENTATION

[Deleting the Contrail MultiCloud Cluster](#)

[Updating the Contrail Multicloud Cluster](#)

## CHAPTER 8

# Using Contrail with Kubernetes

**IN THIS CHAPTER**

- [Contrail Integration with Kubernetes | 166](#)
- [Installing and Managing Contrail 5.0 Microservices Architecture Using Helm Charts | 173](#)
- [Provisioning of Kubernetes Clusters | 177](#)
- [Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability | 184](#)
- [Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata | 194](#)
- [Accessing a Contrail OpenStack Helm Cluster | 198](#)
- [Frequently Asked Questions About Contrail and Helm Charts | 201](#)
- [Contrail Deployment with Helm | 205](#)
- [Verifying Configuration for CNI for Kubernetes | 211](#)
- [Kubernetes Updates to IP Fabric | 214](#)
- [Implementation of Kubernetes Network Policy with Contrail Firewall Policy | 217](#)

## Contrail Integration with Kubernetes

**IN THIS SECTION**

- [What is Kubernetes? | 167](#)
- [Configuration Modes for Contrail Integrated with Kubernetes | 168](#)
- [Kubernetes Services | 170](#)
- [Ingress | 171](#)
- [Contrail Kubernetes Solution | 171](#)

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 2 on page 167](#) lists the mapping between Kubernetes concepts and OpenContrail resources.

**Table 2: 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 712](#) 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.

## Kubernetes Support for Security Policy

Network policies created in a Kubernetes environment are implemented by using Contrail Security Policy framework. Labels from the Kubernetes environment are exposed as tags in Contrail. Starting in Contrail Release 5.0, you can define tags for a Kubernetes environment. Contrail security policy uses these tags to implement specified Kubernetes policies. You can define tags in the UI or upload configurations in JSON format. The newly-defined tags can be used to create and enforce policies in Contrail Security.

## 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](#)

[Verifying Configuration for CNI for Kubernetes](#) | 211

## Installing and Managing Contrail 5.0 Microservices Architecture Using Helm Charts

### IN THIS SECTION

- [Understanding Helm Charts](#) | 173
- [Contrail Helm Deployer Charts](#) | 174
- [Contrail Kubernetes Resource implementation](#) | 175
- [Example: Contrail Pods Deployment Options](#) | 175
- [Installing Contrail Using Helm Charts](#) | 176

Contrail Release 5.0 introduces microservices architecture. This section provides an overview of using Helm charts when installing Contrail with a microservices architecture. Contrail Helm charts work together with OpenStack Helm for an OpenStack Contrail deployment. For an introduction to Contrail microservices, refer to *Introduction to Contrail Microservices Architecture*.

### Understanding Helm Charts

Helm is the package manager for Kubernetes, an open source software for managing containerized systems. The packaging format used by Helm is a chart, a collection of files that describe a related set of Kubernetes resources. Helm charts enable you to define, install, and configure your Kubernetes application. A chart can be used to deploy something simple, like a memcached pod, or something complex, like a full web application stack complete with HTTP servers, databases, and the like.

Starting with Contrail Release 5.0, Contrail Helm charts give you complete life cycle management of installation, update, and delete of Contrail Docker-based containers in a microservices architecture.

The Contrail Helm deployer supports deploying Contrail for OpenStack.

## Contrail Helm Deployer Charts

The Contrail Helm deployer uses the following charts.

- **helm-toolkit chart**

Contains common templates and functions that are used by every other Contrail Helm chart.

- **contrail-thirdparty chart**

Defines and deploys third party containers as Kubernetes resources for Contrail, including:

- RabbitMQ
- ZooKeeper
- Cassandra
- Kafka
- Redis

- **contrail-controller chart**

Deploys and manages Contrail components as Kubernetes resources, including:

- control
- config
- webui

- **contrail-analytics chart**

Deploys and manages Contrail analytics components as Kubernetes resources.

- **contrail-vrouter chart**

Deploys and manages Contrail vrouter components as Kubernetes resources.

- **contrail-superset chart**

A superset of all other Contrail Helm charts, can be used to install all Kubernetes resources defined in other Contrail charts.

## Contrail Kubernetes Resource implementation

All Contrail Helm charts follow a similar approach to implementing Kubernetes resources. For each of the Contrail Release 5.0 containers, configuration input is given as an environment variable in the file `values.yaml`. Use the variable `.Values.contrail_env` to define environment variables for the containers.

```
contrail_env:
  CONTROLLER_NODES: <Controller-Nodes-IP-Address>
  LOG_LEVEL: SYS_NOTICE
  CLOUD_ORCHESTRATOR: openstack
  AAA_MODE: cloud-admin
```

All of the environment variables are stored in Kubernetes resources called configmaps. The configmaps are loaded into specific containers as environment variables.

Because Contrail is an infrastructure-level application, every pod of Contrail is hosted on the host network namespace. Consequently, the daemonset controller is used to define all Contrail pods, so that each of the Contrail pods are brought up on different nodes to avoid port conflicts.

### Example: Contrail Pods Deployment Options



**NOTE:** By default, the `contrail-thirdparty` Helm chart creates a separate pod for each of the third party services.

```
pods:
  - contrail-control
    containers:
      - contrail-control
      - contrail-dns
      - contrail-named
      - control-nodemgr
  - contrail-config
    containers:
      - config-api
      - schema-transformer
      - svc-monitor
      - device-manager
      - config-nodemgr
  - contrail-webui
    containers:
```

```

- contrail-webui
- contrail-middleware
- contrail-analytics
containers:
- analytics-api
- analytics-collector
- snmp-collector
- query-engine
- alarm-gen
- contrail-topology
- contrail-vrouter
containers:
- vrouter-kernel/vrouter-dpdk/vrouter-sriov
- vrouter-agent
- vrouter-nodemgr

```

## Installing Contrail Using Helm Charts

Use one of the following procedures to install Contrail with OpenStack Ocata using Helm charts:

- ["Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability " on page 184](#)
- ["Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata " on page 194](#)

## RELATED DOCUMENTATION

---

[Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability | 184](#)

---

[Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata | 194](#)

---

[Accessing a Contrail OpenStack Helm Cluster | 198](#)

---

[Frequently Asked Questions About Contrail and Helm Charts | 201](#)



## Provisioning of Kubernetes Clusters

### IN THIS SECTION

- [Provisioning of a Standalone Kubernetes Cluster | 177](#)
- [Provisioning of Nested Contrail Kubernetes Clusters | 178](#)
- [Provisioning of Non-Nested Contrail Kubernetes Clusters | 181](#)

Contrail Release 5.0.1 supports the following ways of provisioning Kubernetes clusters

### Provisioning of a Standalone Kubernetes Cluster

You can provision a standalone Kubernetes cluster using contrail-ansible-deployer.

Perform the following steps to install one Kubernetes cluster and one Contrail cluster and integrate them together.

1. Re-image the node to CentOS 7.4 using Linux kernel version 3.10.0-862.3.2.
2. Install the necessary tools.
 

```
yum -y install epel-release git ansible net-tools
```
3. Download the contrail-ansible-deployer-5.0.1-0.214.tgz Ansible Deployer application tool package onto your provisioning host from [Juniper Networks](#) and extract the package.
 

```
- tar xvf contrail-ansible-deployer-5.0.1-0.214.tgz
```
4. Navigate to the **contrail-ansible-deployer** directory.
 

```
cd contrail-ansible-deployer
```
5. Edit the **config/instances.yaml** and enter the necessary values. See *Overview of contrail-ansible-deployer used in Contrail Command for Installing Contrail with Microservices Architecture* for a sample **config/instances.yaml** file.
6. Turn off the swap functionality on all nodes.
 

```
swapoff -a
```
7. Configure the nodes.
 

```
ansible-playbook -e orchestrator=kubernetes -i inventory/ playbooks/configure_instances.yml
```
8. Install Kubernetes and Contrail.
 

```
ansible-playbook -e orchestrator=kubernetes -i inventory/ playbooks/install_k8s.yml
```

```
ansible-playbook -e orchestrator=kubernetes -i inventory/ playbooks/install_contrail.yml
```

## Provisioning of Nested Contrail Kubernetes Clusters

### IN THIS SECTION

- [Configure network connectivity to Contrail configuration and data plane functions. | 179](#)
- [Generate a single yaml file to create a Contrail-k8s cluster | 180](#)
- [Instantiate the Contrail-k8s cluster | 181](#)

When Contrail provides networking for a Kubernetes cluster that is provisioned on the workloads of a Contrail-OpenStack cluster, it is called a nested Kubernetes cluster. Contrail components are shared between the two clusters.

### Prerequisites

Ensure that the following prerequisites are met before provisioning a nested Kubernetes cluster:

1. Ensure that you have an operational Contrail-OpenStack cluster based on Contrail Release 5.0.1.
2. Ensure that you have an operational Kubernetes v1.9.2 cluster on virtual machines created on an Contrail-OpenStack cluster.
3. Update the `/etc/hosts` file on the Kubernetes master node with entries for each node of the cluster.

For example, if the Kubernetes cluster is made up of three nodes such as master1 (IP: x.x.x.x), minion1 (IP: y.y.y.y), and minion2 (IP: z.z.z.z). The `/etc/hosts` on the Kubernetes master node must have the following entries:

```
x.x.x.x master1
y.y.y.y minion1
z.z.z.z minion2
```

4. If Contrail container images are stored in **private/secure docker registry**, a Kubernetes secret must be created and referenced during "[Generate a single yaml file to create a Contrail-k8s cluster](#)" on [page 180](#), with credentials of the private docker registry.

```
kubect1 create secret docker-registry name --docker-server=registry --docker-username=username --docker-password=password --docker-email=email -n namespace
```

Command options:

- *name*—Name of the secret
- *registry*—Name of the registry. Example: hub.juniper.net/contrail
- *username*—Username to log in to the registry
- *password*—Password to log in to the registry
- *email*—Registered email of the registry account
- *namespace*—Kubernetes namespace where the secret must be created. This should be the namespace where you intend to create the Contrail pods.

The following steps describe how to provision a nested Contrail Kubernetes cluster.

### **Configure network connectivity to Contrail configuration and data plane functions.**

A nested Kubernetes cluster is managed by the same Contrail control processes that manage the underlying OpenStack cluster.

The kube-manager is essentially a part of the Contrail Config function. In a nested deployment, one kube-manager instance will be provisioned in each overlay cluster. This necessitates the need for the kube-manager running in the overlay to have network reachability to Contrail config functions of the underlay OpenStack cluster.

Network connectivity for the following Contrail config functions are required:

- Contrail Config
- Contrail Analytics
- Contrail Msg Queue
- Contrail VNC DB
- Keystone

In addition to config connectivity, the CNI for the Kubernetes cluster needs network reachability to the vRouter on its Compute node. Network connectivity for the vRouter data plane function is also required.

You can use the link local service feature or a combination of link local service with fabric Source Network Address Translation (SNAT) feature of Contrail to provide IP reachability to and from the overlay Kubernetes cluster config and data components to corresponding config and data components of the underlay OpenStack cluster.

To provide IP reachability to and from the Kubernetes cluster using the fabric SNAT with link local service, perform the following steps.

1. Enable fabric SNAT on the virtual network of the VMs.

The fabric SNAT feature must be enabled on the virtual network of the virtual machines on which the Kubernetes master and minions are running.

2. Create a link local service for the Container Network Interface (CNI) to communicate with its vRouter Agent. This link local service should be configured using the Contrail GUI, in the following example:

Contrail Process	Service IP	Service Port	Fabric IP	Fabric Port
vRouter	<i>Service-IP for the active node</i>	9091	127.0.0.1	9091



**NOTE:** Fabric IP address is 127.0.0.1 since you must make the CNI communicate with the vRouter on its underlay node.

For example, the following link local services must be created:

Link Local Service Name	Service IP	Service Port	Fabric IP	Fabric Port
K8s-cni-to-agent	10.10.10.5	9091	127.0.0.1	9091



**NOTE:** Here 10.10.10.5 is the Service IP address that you chose. This can be any unused IP in the cluster. This IP address is primarily used to identify link local traffic and has no other significance.

### Generate a single yaml file to create a Contrail-k8s cluster

Contrail components are installed on the Kubernetes cluster as pods. The configuration to create these pods in Kubernetes is encoded in a yaml file.

This file can be generated as follows:

1. Download the `contrail-ansible-deployer-5.0.1-0.214.tgz` Ansible Deployer application tool package onto your provisioning host from [Juniper Networks](#) and extract the package.
 

```
- tar xvf contrail-ansible-deployer-5.0.1-0.214.tgz
```
2. Navigate to the **contrail-container-builder** directory.
 

```
cd contrail-container-builder
```
3. Populate the **common.env** file located in the top directory of the cloned `contrail-container-builder` repo with information corresponding to your cluster and environment.

For your reference, see a sample **common.env** file with required bare minimum configurations here [https://github.com/Juniper/contrail-container-builder/blob/master/kubernetes/sample\\_config\\_files/common.env.sample.nested\\_mode](https://github.com/Juniper/contrail-container-builder/blob/master/kubernetes/sample_config_files/common.env.sample.nested_mode).



**NOTE:** If Contrail container images are stored in **private/secure docker registry**, a Kubernetes secret must be created and referenced as documented in "4" on page 178 of Prerequisites. Populate the variable **KUBERNETES\_SECRET\_CONTRAIL\_REPO=<secret-name>** with the name of the generated Kubernetes secret, in the **common.env** file.

4. Generate the yaml file as following in your shell:

```
cd contrail-container-build-repo/kubernetes/manifests

./resolve-manifest.sh contrail-kubernetes-nested.yaml > nested-contrail.yaml
```

5. Copy the output (or file) generated from 4 to the master node in your Kubernetes cluster.

### Instantiate the Contrail-k8s cluster

Create contrail components as pods on the Kubernetes cluster.

```
root@k8s:~# kubectl get pods -n kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
contrail-kube-manager-lcjbc	1/1	Running	0	3d
contrail-kubernetes-cni-agent-w8shc	1/1	Running	0	3d

You will see the following pods running in the kube-system namespace:

contrail-kube-manager-xxxxxx—This is the manager that acts as conduit between Kubernetes and OpenStack clusters

contrail-kubernetes-cni-agent-xxxxx—This installs and configures Contrail CNI on Kubernetes nodes

## Provisioning of Non-Nested Contrail Kubernetes Clusters

### Prerequisites

Ensure that the following prerequisites are met before provisioning a non-nested Kubernetes cluster:

1. You must have an installed and operational Contrail OpenStack cluster based on the Contrail Release 5.0.1 release.

2. You must have an installed and operational Kubernetes cluster on the server where you want to install the non-nested Contrail Kubernetes cluster.
3. Label the Kubernetes master node with the Contrail controller label:

```
kubect1 label node node node-role.opencontrail.org/controller=true
```

4. Ensure that the Kubelet running on the Kubernetes master node is not run with network plugin options. If kubelet is running with network plugin option, then disable or comment out the KUBELET\_NETWORK\_ARGS option in the `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` configuration file.



**NOTE:** It is recommended that the Kubernetes master should not be configured with a network plugin, so as to not install vRouter kernel module on the control node. However, this is optional.

5. Restart the kubelet service:

```
systemctl daemon-reload;
systemctl restart kubelet.service
```

In non-nested mode, a Kubernetes cluster is provisioned side by side with an OpenStack cluster with networking provided by the same Contrail components of the OpenStack cluster.

### Provisioning a Contrail Kubernetes Cluster

Follow these steps to provision Contrail Kubernetes cluster.

1. Download the `contrail-ansible-deployer-5.0.1-0.214.tgz` Ansible Deployer application tool package onto your provisioning host from [Juniper Networks](#) and extract the package.

```
- tar xvf contrail-ansible-deployer-5.0.1-0.214.tgz
```

2. Navigate to the **contrail-container-builder** directory.

```
cd contrail-container-builder
```

3. Populate the **common.env** file located in the top directory of the cloned `contrail-container-builder` repo with information corresponding to your cluster and environment.

For a sample **common.env** file with required bare minimum configurations see [https://github.com/Juniper/contrail-container-builder/blob/master/kubernetes/sample\\_config\\_files/common.env.sample.non\\_nested\\_mode](https://github.com/Juniper/contrail-container-builder/blob/master/kubernetes/sample_config_files/common.env.sample.non_nested_mode).



**NOTE:** If Config API is not secured by keystone, ensure that *AUTH\_MODE* and *KEYSTONE\_\** variables are not configured or present while populating the **common.env** file.

4. Generate the yaml file as shown below:

```
cd contrail-container-build-repo/kubernetes/manifests

./resolve-manifest.sh contrail-kubernetes-nested.yaml > non-nested-contrail.yaml
```

5. Copy the file generated from 4 to the master node in your Kubernetes cluster.
6. Create contrail components as pods on the Kubernetes cluster as follows:

```
kubectl apply -f non-nested-contrail.yaml
```

7. Create the following Contrail pods on the Kubernetes cluster. Ensure that contrail-agent pod is created only on the worker node.

```
[root@b4s403 manifests]# kubectl get pods --all-namespaces -o wide
```

	NAMESPACE	NAME	READY	STATUS	RESTARTS
AGE	IP	NODE			
	kube-system	contrail-agent-mxkcq	2/2	Running	0
1m	<x.x.x.x>	b4s402			
	kube-system	contrail-kube-manager-gl5m	1/1	Running	0
1m	<x.x.x.x>	b4s403			

## RELATED DOCUMENTATION

[Contrail Integration with Kubernetes](#) | 166

## Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability

### IN THIS SECTION

- [System Specifications | 184](#)
- [Preparing to Install | 184](#)
- [Installation of OpenStack Helm Charts | 188](#)
- [Installation of Contrail Helm Charts | 189](#)
- [Basic Testing OpenStack Helm Contrail Cluster | 193](#)
- [Accessing the Contrail OpenStack Helm Cluster | 194](#)

This is the installation procedure for using Helm charts to provision a multinode Contrail system with OpenStack Ocata and high availability.

### System Specifications

This procedure uses Juniper OpenStack Helm infrastructure and the OpenStack Helm repository to provision an OpenStack Ocata Contrail multinode deployment.

This procedure is tested with:

- Operating system: Ubuntu 16.04.3 LTS
- Kernel: 4.4.0-87-generic
- Docker: 1.13.1-cs9
- Helm: v2.7.2
- Kubernetes: v1.8.3
- OpenStack: Ocata

### Preparing to Install

This section is the prerequisites needed to prepare your system before provisioning multinode Contrail with OpenStack Ocata and high availability.



1. Generate SSH key on primary node and copy to all nodes, in below example three nodes with IP addresses 10.13.82.43, 10.13.82.44, and 10.13.82.45 are used.

```
(k8s-master)> ssh-keygen

(k8s-master)> ssh-copy-id -i ~/.ssh/id_rsa.pub 10.13.82.43
(k8s-master)> ssh-copy-id -i ~/.ssh/id_rsa.pub 10.13.82.44
(k8s-master)> ssh-copy-id -i ~/.ssh/id_rsa.pub 10.13.82.45
```

2. Make sure NTP is configured in all nodes and each node is synched to the time-server in your environment. In below example the NTP server IP is "10.84.5.100".

```
(k8s-all-nodes)> ntpq -p
      remote           refid      st t when poll reach   delay   offset  jitter
=====
*10.84.5.100      66.129.255.62      2 u  15   64  377   72.421  -22.686   2.628
```

3. Get the contrail-helm-deployer.

From [Juniper Networks](#), download contrail-helm-deployer-5.0.0-0.40.tgz onto your provisioning host.

- scp contrail-helm-deployer-5.0.0-0.40.tgz to all nodes on your cluster.
- Untar contrail-helm-deployer-5.0.0-0.40.tgz on all nodes.

```
tar -zxf contrail-helm-deployer-5.0.0-0.40.tgz -C /opt/
```

4. Export required variables.

```
(k8s-master)> cd /opt
(k8s-master)> export BASE_DIR=$(pwd)
(k8s-master)> export OSH_PATH=${BASE_DIR}/openstack-helm
(k8s-master)> export OSH_INFRA_PATH=${BASE_DIR}/openstack-helm-infra
(k8s-master)> export CHD_PATH=${BASE_DIR}/contrail-helm-deployer
```

5. Install necessary packages and deploy Kubernetes.



**NOTE:** If you want to install a different version of Kubernetes, CNI, or Calico, edit `{OSH_INFRA_PATH}/tools/gate/devel/local-vars.yaml` to override the default values in `{OSH_INFRA_PATH}/tools/gate/playbooks/vars.yaml`.

```
(k8s-master)> cd ${OSH_PATH}
(k8s-master)> ./tools/deployment/developer/common/001-install-packages-opencontrail.sh
```

6. Create an inventory file on the primary node for Ansible base provisioning. In the following output, 10.13.82.43/.44/.45 are the IP addresses of the nodes, and will use the SSK-key generated in Step 1.

```
#!/bin/bash
(k8s-master)> set -xe
(k8s-master)> cat > /opt/openstack-helm-infra/tools/gate/devel/multinode-inventory.yaml <<EOF
all:
  children:
    primary:
      hosts:
        node_one:
          ansible_port: 22
          ansible_host: 10.13.82.43
          ansible_user: root
          ansible_ssh_private_key_file: /root/.ssh/id_rsa
          ansible_ssh_extra_args: -o StrictHostKeyChecking=no
        nodes:
          hosts:
            node_two:
              ansible_port: 22
              ansible_host: 10.13.82.44
              ansible_user: root
              ansible_ssh_private_key_file: /root/.ssh/id_rsa
              ansible_ssh_extra_args: -o StrictHostKeyChecking=no
            node_three:
              ansible_port: 22
              ansible_host: 10.13.82.45
              ansible_user: root
              ansible_ssh_private_key_file: /root/.ssh/id_rsa
              ansible_ssh_extra_args: -o StrictHostKeyChecking=no
EOF
```

## 7. Create an environment file on the primary node for the cluster.



**NOTE:** By default, Kubernetes v1.9.3, Helm v2.7.2, and CNI v0.6.0 are installed. If you want to install a different version, edit the `${OSH_INFRA_PATH}/tools/gate/devel/multinode-vars.yaml` file to override the values given in `${OSH_INFRA_PATH}/playbooks/vars.yaml`.

Sample `multinode-vars.yaml` :

```
(k8s-master)> cat > /opt/openstack-helm-infra/tools/gate/devel/multinode-vars.yaml <<EOF
# version fields
version:
  kubernetes: v1.9.3
  helm: v2.7.2
  cni: v0.6.0

kubernetes:
  network:
    # enp0s8 is your control/data interface, to which kubernetes will bind to
    default_device: enp0s8
  cluster:
    cni: calico
    pod_subnet: 192.168.0.0/16
    domain: cluster.local
  docker:
    # list of insecure_registries, from where you will be pulling container images
    insecure_registries:
      - "10.87.65.243:5000"
    # list of private secure docker registry auth info, from where you will be pulling container
    images
    #private_registries:
    # - name: <docker-registry-name>
    #   username: username@abc.xyz
    #   email: username@abc.xyz
    #   password: password
    #   secret_name: contrail-image-secret
    #   namespace: openstack
EOF
```

8. Run playbooks on the primary node.

```
(k8s-master)> set -xe
(k8s-master)> cd ${OSH_INFRA_PATH}
(k8s-master)> make dev-deploy setup-host multinode
(k8s-master)> make dev-deploy k8s multinode
```

9. Verify the kube-dns connection from all nodes. Use `nslookup` to verify that you are able to resolve Kubernetes cluster-specific names.

```
(k8s-all-nodes)> nslookup
> kubernetes.default.svc.cluster.local
Server:          10.96.0.10
Address:         10.96.0.10#53

Non-authoritative answer:
Name:   kubernetes.default.svc.cluster.local
Address: 10.96.0.1
```

## Installation of OpenStack Helm Charts

Use this procedure to install the OpenStack Helm charts.

1. Before installing the OpenStack Helm charts, review the default labels for the nodes.

The default nodes have the labels `openstack-control-plane` and `openstack-compute-node`. The default configuration creates OpenStack Helm (OSH) pods on all the nodes. Use the following commands to check the default OpenStack labels.

```
(k8s-master)> kubectl get nodes -o wide -l openstack-control-plane=enabled
(k8s-master)> kubectl get nodes -o wide -l openstack-compute-node=enabled
```

If you need to restrict the creation of OSH pods on specific nodes, disable the OpenStack labels. The following example shows how to disable the `openstack-compute-node` label on the `ubuntu-contrail-9` node.

```
(k8s-master)> kubectl label node ubuntu-contrail-9 --overwrite openstack-compute-node=disabled
```

## 2. Deploy OpenStack Helm charts.

```
(k8s-master)> set -xe
(k8s-master)> cd ${OSH_PATH}

(k8s-master)> ./tools/deployment/multinode/010-setup-client.sh
(k8s-master)> ./tools/deployment/multinode/021-ingress-opencontrail.sh
(k8s-master)> ./tools/deployment/multinode/030-ceph.sh
(k8s-master)> ./tools/deployment/multinode/040-ceph-ns-activate.sh
(k8s-master)> ./tools/deployment/multinode/050-mariadb.sh
(k8s-master)> ./tools/deployment/multinode/060-rabbitmq.sh
(k8s-master)> ./tools/deployment/multinode/070-memcached.sh
(k8s-master)> ./tools/deployment/multinode/080-keystone.sh
(k8s-master)> ./tools/deployment/multinode/090-ceph-radosgateway.sh
(k8s-master)> ./tools/deployment/multinode/100-glance.sh
(k8s-master)> ./tools/deployment/multinode/110-cinder.sh
(k8s-master)> ./tools/deployment/multinode/131-libvirt-opencontrail.sh
# Edit ${OSH_PATH}/tools/overrides/backends/opencontrail/nova.yaml and
# ${OSH_PATH}/tools/overrides/backends/opencontrail/neutron.yaml
# to make sure that you are pulling init container image from correct registry and tag
(k8s-master)> ./tools/deployment/multinode/141-compute-kit-opencontrail.sh
(k8s-master)> ./tools/deployment/developer/ceph/100-horizon.sh
```

## Installation of Contrail Helm Charts

Use this procedure to install the Contrail Helm charts.

1. Label the Contrail pods. All Contrail pods are to be deployed in the namespace `contrail`, using the following labels:

- Controller components—`config`, `control`, `analytics`
- vRouter kernel—`opencontrail.org/vrouter-kernel`
- vRouter DPDK—`opencontrail.org/vrouter-dpdk`

The following example shows how to label `ubuntu-contrail-11` as DPDK and label `ubuntu-contrail-10` as kernel vrouter.

```
(k8s-master)> kubectl label node ubuntu-contrail-11 opencontrail.org/vrouter-dpdk=enabled
(k8s-master)> kubectl label node ubuntu-contrail-10 opencontrail.org/vrouter-kernel=enabled
```

```
(k8s-master)> kubectl label nodes ubuntu-contrail-9 ubuntu-contrail-10 ubuntu-contrail-11
opencontrail.org/controller=enabled
```

## 2. Create Kubernetes ClusterRoleBinding for Contrail.

```
(k8s-master)> cd $CHD_PATH
(k8s-master)> kubectl replace -f ${CHD_PATH}/rbac/cluster-admin.yaml
```

## 3. Set up the Contrail Helm charts and set the configuration settings specific to your system in the values.yaml file for each of the charts.

```
(k8s-master)> cd $CHD_PATH
(k8s-master)> make

# Please note in below example, 192.168.1.0/24 is "Control/Data" network
# Export variables
(k8s-master)> export CONTROLLER_NODES="192.168.1.43,192.168.1.44,192.168.1.45"
(k8s-master)> export VROUTER_GATEWAY="192.168.1.1"
(k8s-master)> export CONTROL_DATA_NET_LIST="192.168.1.0/24"
(k8s-master)> export BGP_PORT="1179"

# [Optional] By default, it will pull latest image from opencontrailnightly

(k8s-master)> export CONTRAIL_REGISTRY="opencontrailnightly"
(k8s-master)> export CONTRAIL_TAG="latest"

# [Optional] only if you are pulling images from a private docker registry
export CONTRAIL_REG_USERNAME="abc@abc.com"
export CONTRAIL_REG_PASSWORD="password"

tee /tmp/contrail-env-images.yaml << EOF
global:
  contrail_env:
    CONTROLLER_NODES: ${CONTROLLER_NODES}
    CONTROL_NODES: ${CONTROL_NODES:-CONTROLLER_NODES}
    LOG_LEVEL: SYS_NOTICE
    CLOUD_ORCHESTRATOR: openstack
    AAA_MODE: cloud-admin
    VROUTER_GATEWAY: ${VROUTER_GATEWAY}
    BGP_PORT: ${BGP_PORT}
  contrail_env_vrouter_kernel:
```

```

CONTROL_DATA_NET_LIST: ${CONTROL_DATA_NET_LIST}
AGENT_MODE: nic
contrail_env_vrouter_dpdn:
  AGENT_MODE: dpdk
images:
  tags:
    kafka: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-external-kafka:${CONTRAIL_TAG:-latest}"
    cassandra: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-external-cassandra:${CONTRAIL_TAG:-latest}"
    redis: "redis:4.0.2"
    zookeeper: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-external-zookeeper:${CONTRAIL_TAG:-latest}"
    contrail_control: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-control-control:${CONTRAIL_TAG:-latest}"
    control_dns: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-control-dns:${CONTRAIL_TAG:-latest}"
    control_named: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-control-named:${CONTRAIL_TAG:-latest}"
    config_api: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-config-api:${CONTRAIL_TAG:-latest}"
    config_devicemgr: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-config-devicemgr:${CONTRAIL_TAG:-latest}"
    config_schema_transformer: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-config-schema:${CONTRAIL_TAG:-latest}"
    config_svcmonitor: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-config-svcmonitor:${CONTRAIL_TAG:-latest}"
    webui_middleware: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-webui-job:${CONTRAIL_TAG:-latest}"
    webui: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-webui-web:${CONTRAIL_TAG:-latest}"
    analytics_api: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-api:${CONTRAIL_TAG:-latest}"
    contrail_collector: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-collector:${CONTRAIL_TAG:-latest}"
    analytics_alarm_gen: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-alarm-gen:${CONTRAIL_TAG:-latest}"
    analytics_query_engine: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-query-engine:${CONTRAIL_TAG:-latest}"
    analytics_snmp_collector: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-snmp-collector:${CONTRAIL_TAG:-latest}"
    contrail_topology: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-topology:${CONTRAIL_TAG:-latest}"

```

```

    build_driver_init: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-kernel-
build-init:${CONTRAIL_TAG:-latest}"
    vrouter_agent: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-agent:${
CONTRAIL_TAG:-latest}"
    vrouter_init_kernel: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-
kernel-init:${CONTRAIL_TAG:-latest}"
    vrouter_dpdk: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-agent-dpdk:${
CONTRAIL_TAG:-latest}"
    vrouter_init_dpdk: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-kernel-
init-dpdk:${CONTRAIL_TAG:-latest}"
    nodemgr: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-nodemgr:${CONTRAIL_TAG:-
latest}"
    contrail_status: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-status:${
CONTRAIL_TAG:-latest}"
    node_init: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-node-init:${
CONTRAIL_TAG:-latest}"
    dep_check: quay.io/stackanetes/kubernetes-entrpoint:v0.2.1
EOF

```



**NOTE:** If any other environment variables need to be added, add them in the values.yaml file of the respective charts.

```

# [Optional] only if you are pulling contrail images from a private registry
tee /tmp/contrail-registry-auth.yaml << EOF
global:
  images:
    imageCredentials:
      registry: ${CONTRAIL_REGISTRY:-opencontrailnightly}
      username: ${CONTRAIL_REG_USERNAME}
      password: ${CONTRAIL_REG_PASSWORD}
EOF

# [Optional] only if you are pulling images from a private registry
export CONTRAIL_REGISTRY_ARG="--values=/tmp/contrail-registry-auth.yaml "

```

#### 4. Use Helm install commands to deploy each of the Contrail Helm charts.

```

(k8s-master)> helm install --name contrail-thirdparty ${CHD_PATH}/contrail-thirdparty \
--namespace=contrail \

```



```
--values=/tmp/contrail-env-images.yaml \
${CONTRAIL_REGISTRY_ARG}

(k8s-master)> helm install --name contrail-controller ${CHD_PATH}/contrail-controller \
--namespace=contrail \
--values=/tmp/contrail-env-images.yaml \
${CONTRAIL_REGISTRY_ARG}

(k8s-master)> helm install --name contrail-analytics ${CHD_PATH}/contrail-analytics \
--namespace=contrail \
--values=/tmp/contrail-env-images.yaml \
${CONTRAIL_REGISTRY_ARG}

# Edit contrail-vrouter/values.yaml and make sure that
global.images.tags.vrouter_init_kernel is right. Image tag name will be different depending
upon your linux. Also set the global.node.host_os to ubuntu or centos depending on your system

(k8s-master)> helm install --name contrail-vrouter ${CHD_PATH}/contrail-vrouter \
--namespace=contrail \
--values=/tmp/contrail-env-images.yaml \
${CONTRAIL_REGISTRY_ARG}
```

5. When the Contrail pods are up and running, deploy the OpenStack Heat chart.

```
# Edit ${OSH_PATH}/tools/overrides/backends/opencontrail/nova.yaml and
# ${OSH_PATH}/tools/overrides/backends/opencontrail/heat.yaml
# to make sure that you are pulling the right opencontrail init container image
(k8s-master)> ./tools/deployment/multinode/151-heat-opencontrail.sh
```

6. When finished, run the compute kit test.

```
(k8s-master)> ./tools/deployment/multinode/143-compute-kit-opencontrail-test.sh
```

## Basic Testing OpenStack Helm Contrail Cluster

Use the following commands to perform basic testing on the virtual network and the virtual machines in your OpenStack Helm Contrail cluster.

```
(k8s-master)> export OS_CLOUD=openstack_helm

(k8s-master)> openstack network create MGMT-VN
```

```
(k8s-master)> openstack subnet create --subnet-range 172.16.1.0/24 --network MGMT-VN MGMT-VN-subnet
```

```
(k8s-master)> openstack server create --flavor m1.tiny --image 'Cirros 0.3.5 64-bit' \
--nic net-id=MGMT-VN \
Test-01
```

```
(k8s-master)> openstack server create --flavor m1.tiny --image 'Cirros 0.3.5 64-bit' \
--nic net-id=MGMT-VN \
Test-02
```

## Accessing the Contrail OpenStack Helm Cluster

Use the following topic to access the OpenStack and Contrail Web UI and prepare the OpenStack client for command-line interface (CLI):

["Accessing a Contrail OpenStack Helm Cluster" on page 198](#)

### RELATED DOCUMENTATION

[Installing and Managing Contrail 5.0 Microservices Architecture Using Helm Charts | 173](#)

[Frequently Asked Questions About Contrail and Helm Charts | 201](#)

[Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata | 194](#)

[Accessing a Contrail OpenStack Helm Cluster | 198](#)

## Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata

### IN THIS SECTION

- [System Specifications | 195](#)
- [Installation Steps | 195](#)
- [Accessing the Contrail OpenStack Helm Cluster | 198](#)

This is the installation procedure for using Helm charts to provision an all-in-one Contrail system with OpenStack Ocata. This is not a high availability configuration.



**NOTE:** All-in-one systems are only used for testing or for demonstration purposes.

## System Specifications

This procedure uses Helm to provision an OpenStack Ocata Contrail all-in-one cluster without high availability.

This procedure is tested with:

- Operating system: Ubuntu 16.04.3 LTS
- Kernel: 4.4.0-87-generic
- Docker: 1.13.1-cs9
- Helm: v2.7.2
- Kubernetes: v1.8.3
- OpenStack: Ocata

This setup was tested on a system with the following specifications:

- CPU: 8
- RAM: 32 GB
- HDD: 120 GB

## Installation Steps

1. Get the contrail-helm-deployer.

From [Juniper Networks](#), download `contrail-helm-deployer-5.0.0-0.40.tgz` onto your provisioning host.

- Untar `contrail-helm-deployer-5.0.0-0.40.tgz`.

```
tar -zxf contrail-helm-deployer-5.0.0-0.40.tgz -C /opt/
```

## 2. Export required variables.

```
export BASE_DIR=$(pwd)
export OSH_PATH=${BASE_DIR}/openstack-helm
export OSH_INFRA_PATH=${BASE_DIR}/openstack-helm-infra
export CHD_PATH=${BASE_DIR}/contrail-helm-deployer
Export variables
```

## 3. Install necessary packages and deploy Kubernetes.



**NOTE:** If you want to install a different version of Kubernetes, CNI, or Calico, edit \$ {OSH\_INFRA\_PATH}/tools/gate/devel/local-vars.yaml to override the default values in \$ {OSH\_INFRA\_PATH}/tools/gate/playbooks/vars.yaml.

```
cd ${OSH_PATH}
./tools/deployment/developer/common/001-install-packages-opencontrail.sh
./tools/deployment/developer/common/010-deploy-k8s.sh
```

## 4. Install OpenStack and the Heat client.

```
./tools/deployment/developer/common/020-setup-client.sh
```

## 5. Deploy OpenStack Helm-related charts.

```
./tools/deployment/developer/nfs/031-ingress-opencontrail.sh
./tools/deployment/developer/nfs/040-nfs-provisioner.sh
./tools/deployment/developer/nfs/050-mariadb.sh
./tools/deployment/developer/nfs/060-rabbitmq.sh
./tools/deployment/developer/nfs/070-memcached.sh
./tools/deployment/developer/nfs/080-keystone.sh
./tools/deployment/developer/nfs/100-horizon.sh
./tools/deployment/developer/nfs/120-glance.sh
./tools/deployment/developer/nfs/151-libvirt-opencontrail.sh
./tools/deployment/developer/nfs/161-compute-kit-opencontrail.sh
```

## 6. Deploy Contrail Helm charts.

```
cd $CHD_PATH

make

# Set the IP of your CONTROL_NODES (specify your control data ip, if you have one)
export CONTROL_NODES=10.87.65.245
# set the control data network cidr list separated by comma and set the respective gateway
export CONTROL_DATA_NET_LIST=10.87.65.128/25
export VROUTER_GATEWAY=10.87.65.129

kubectl label node opencontrail.org/controller=enabled --all
kubectl label node opencontrail.org/vrouter-kernel=enabled --all

kubectl replace -f ${CHD_PATH}/rbac/cluster-admin.yaml

tee /tmp/contrail.yaml << EOF
global:
  contrail_env:
    CONTROLLER_NODES: 172.17.0.1
    CONTROL_NODES: ${CONTROL_NODES}
    LOG_LEVEL: SYS_NOTICE
    CLOUD_ORCHESTRATOR: openstack
    AAA_MODE: cloud-admin
    CONTROL_DATA_NET_LIST: ${CONTROL_DATA_NET_LIST}
    VROUTER_GATEWAY: ${VROUTER_GATEWAY}
EOF

helm install --name contrail ${CHD_PATH}/contrail \
--namespace=contrail --values=/tmp/contrail.yaml
```

## 7. Deploy Heat charts.

```
cd ${OSH_PATH}
./tools/deployment/developer/nfs/091-heat-opencontrail.sh
```

## Accessing the Contrail OpenStack Helm Cluster

Use the following topic to access the OpenStack and Contrail Web UI and prepare the OpenStack client for command-line interface (CLI):

["Accessing a Contrail OpenStack Helm Cluster" on page 198](#)

### RELATED DOCUMENTATION

- [Installing and Managing Contrail 5.0 Microservices Architecture Using Helm Charts | 173](#)
- [Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability | 184](#)
- [Accessing a Contrail OpenStack Helm Cluster | 198](#)
- [Frequently Asked Questions About Contrail and Helm Charts | 201](#)

## Accessing a Contrail OpenStack Helm Cluster

### IN THIS SECTION

- [Overview | 198](#)
- [Installing the OpenStack Client | 199](#)
- [Create openstackrc File and Test OpenStack Client | 199](#)
- [Accessing the Contrail Web UI | 200](#)
- [Accessing OpenStack Horizon | 200](#)
- [Accessing the Virtual Machine Console from Horizon | 201](#)
- [OpenStack References | 201](#)

When the provisioning of Contrail with Helm charts is completed, use this topic to access the OpenStack and Contrail Web UI and prepare the OpenStack client for command-line interface (CLI).

### Overview

This topic assumes you have already installed Contrail and OpenStack using Helm charts, typically by using these procedures:

- ["Installing and Managing Contrail 5.0 Microservices Architecture Using Helm Charts" on page 173](#)
- ["Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability " on page 184](#)
- ["Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata " on page 194](#)
- ["Frequently Asked Questions About Contrail and Helm Charts" on page 201](#)

## Installing the OpenStack Client

Use this procedure to install the OpenStack CLI tool.

1. Install the OpenStack client CLI tool on the primary Ubuntu host.

```
apt install python-dev python-pip -y
pip install --upgrade pip
pip install python-openstackclient OR
apt-get install python-openstackclient
```

2. If you have problems installing the python-dev package, add another repository.

```
Add following repo to source "/etc/apt/sources.list"
deb http://archive.ubuntu.com/ubuntu/ xenial-updates main universe multiverse
apt-get update
apt-get install python-dev
```

## Create openstackrc File and Test OpenStack Client

1. Create an openstackrc file.

```
cat > /root/openstackrc << EOF
export OS_USERNAME=admin
export OS_PASSWORD=password
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://keystone-api.openstack:35357/v3
# The following lines can be omitted
#export OS_TENANT_ID=tenantIDString
#export OS_REGION_NAME=regionName
export OS_IDENTITY_API_VERSION=3
```

```
export OS_USER_DOMAIN_NAME=${OS_USER_DOMAIN_NAME:-"Default"}
export OS_PROJECT_DOMAIN_NAME=${OS_PROJECT_DOMAIN_NAME:-"Default"}
EOF
```

## 2. Test the OpenStack client.

```
source openstackrc
openstack server list
openstack stack list
openstack --help
```

## Accessing the Contrail Web UI

1. Access the Contrail Web UI using port 8143. Use the IP address of the host where the contrail-webui pod is running, with the port 8143.

```
https://<IP address host with contrail-webui>:8143
```

2. At the Contrail login screen, enter the default username and password: admin, password.

## Accessing OpenStack Horizon

The OpenStack Web UI (GUI) service is exposed by the Kubernetes service, using the IP address of the node port and the default port 31000.

1. Check the NodePort used for the OpenStack Web UI pod.

```
kubectl get svc -n openstack | grep horizon-int
```

horizon-int	NodePort	10.99.150.28	<none>	80:31000/TCP	4d
-------------	----------	--------------	--------	--------------	----

2. Access the OpenStack Web UI and log in with the default username and password: admin, password.

```
http://<IP address NodePort>:31000/auth/login/?next=/
```



## Accessing the Virtual Machine Console from Horizon

To access the virtual machine (VM) console, add the nova novncproxy fully-qualified domain name (FQDN) in the `/etc/hosts` file, using the host-ip where the osh-ingress pod is running.

The following example for MAC-OS shows the ingress pod running on the host with IP address 10.13.82.233.

```
/private/etc/
hosts

127.0.0.1    localhost
255.255.255.255 broadcasthost
::1         localhost
10.13.82.233 nova-novncproxy.openstack.svc.cluster.local
```



**NOTE:** If you don't want to make changes in `/etc/hosts`, you can replace the `nova-novncproxy.openstack.svc.cluster.local` portion in the URL with the IP address where the OSH ingress pod is running.

## OpenStack References

For more information about accessing and using OpenStack, see the following OpenStack resources:

- [Create OpenStack client environment scripts](#)
- [Install the OpenStack command-line clients](#)
- [External DNS to FQDN/Ingress](#)

## Frequently Asked Questions About Contrail and Helm Charts

### IN THIS SECTION

- [How do I set up the vhost0 interface for the vrouter on the non-management interface of the compute node? | 202](#)
- [How do I configure the Contrail control BGP server to listen on a different port? | 203](#)

- How can I pass additional parameters to services in Contrail by using the configuration file in INI format? | 203
- How do I configure services for the vrouter agent? | 203
- What are the Contrail services that can be configured? | 204
- How can I pass additional parameters to the Contrail Web UI services with configuration file in JS format? | 205
- How can I verify all pods of Contrail are up and running? | 205
- How can I see the logs of each of the containers? | 205
- How can I enter into a pod? | 205

This topic presents frequently asked questions and answers about Contrail and Helm Charts.

## How do I set up the vhost0 interface for the vrouter on the non-management interface of the compute node?



**NOTE:** Some Contrail versions assume a single name for all of the non-management interfaces in your cluster.

If your non-management interface is eth1, in the `contrail-vrouter/values.yaml` set the `contrail_env.PHYSICAL_INTERFACE` to `eth1` and set the `contrail_env.VROUTER_GATEWAY` to the IP address of the non-management gateway.

```
# Sample config
contrail_env:
  CONTROLLER_NODES: 1.1.1.10
  LOG_LEVEL: SYS_NOTICE
  CLOUD_ORCHESTRATOR: openstack
  AAA_MODE: cloud-admin
  PHYSICAL_INTERFACE: eth1
  VROUTER_GATEWAY: 1.1.1.1
```

## How do I configure the Contrail control BGP server to listen on a different port?

To configure a non-default BGP port, in the `contrail-controller/values.yaml` set the `contrail_env.BGP` to the desired port.

```
# Sample config
contrail_env:
  CONTROLLER_NODES: 1.1.1.10
  LOG_LEVEL: SYS_NOTICE
  CLOUD_ORCHESTRATOR: openstack
  AAA_MODE: cloud-admin
  BGP_PORT: 1179
```

## How can I pass additional parameters to services in Contrail by using the configuration file in INI format?

The following example configures the `minimum_diskGB` parameter for the node manager of the analytics database.

```
# Sample config
contrail_env:
  DATABASE_NODEMGR__DEFAULTS__minimum_diskGB: "2"
```

## How do I configure services for the vrouter agent?

The following is an example configuration for the vrouter agent.

```
# Sample config
contrail_env:
  VROUTER_AGENT__FLOWS__thread_count: "2"
  VROUTER_AGENT__METADATA__metadata_use_ssl = True
  VROUTER_AGENT__METADATA__metadata_client_cert = /usr/share/ca-certificates/contrail/
client_cert.pem
  VROUTER_AGENT__METADATA__metadata_client_key = /usr/share/ca-certificates/contrail/
client_key.pem
  VROUTER_AGENT__METADATA__metadata_ca_cert = /usr/share/ca-certificates/contrail/cacert.pem
```

## What are the Contrail services that can be configured?

Configurable services at this time include the following:

- Configurable services for config node:
  - SVC\_MONITOR
  - API
  - DEVICE\_MANAGER
  - SCHEMA
  - CONFIG\_NODEMGR
- Configurable services for control:
  - CONTROL
  - DNS
  - CONTROL\_NODEMGR
- Configurable services for analytics:
  - ALARM\_GEN
  - TOPOLOGY
  - ANALYTICS\_API
  - COLLECTOR
  - SNMP\_COLLECTOR
  - QUERY\_ENGINE
  - ANALYTICS\_NODEMGR
- Configurable services for database:
  - DATABASE\_NODEMGR
- Configurable services for vrouter:
  - VROUTER\_AGENT
  - VROUTER\_AGENT\_NODEMGR

## How can I pass additional parameters to the Contrail Web UI services a with configuration file in JS format?

Define the exact variable in the environment. Available configuration settings can be found in the source code, see <https://github.com/Juniper/contrail-container-builder/blob/master/containers/controller/webui/base/entrypoint.sh#L31-L199> .

```
# Sample config
contrail_env:
  WEBUI_SSL_CIPHERS: "ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384"
```

## How can I verify all pods of Contrail are up and running?

Use the following command to list all pods of Contrail.

```
kubectl get pods -n openstack -o wide | grep contrail-
```

## How can I see the logs of each of the containers?

Contrail logs are stored under `/var/log/contrail/` on each node. To check for the standard output (stdout) log for each container:

```
kubectl logs -f <contrail-pod-name> -n openstack
```

## How can I enter into a pod?

Use the `kubectl` command.

```
kubectl exec -it <contrail-pod> -n openstack -- bash
```

## Contrail Deployment with Helm

This procedure describes how to deploy Contrail with Helm charts, but without OpenStack.



**NOTE:** Nodes should be configured so the primary can ssh into Minion. If ssh keys are needed, these should be specified in the inventory file.

Follow these steps to deploy Contrail with Helm:

1. Download the file **contrail-helm-deployer-release-tag.tgz** onto your provisioning host. It contains the required two required Helm repositories: **/opt/openstack-helm-infra** (which contains code to deploy k8s) and **/opt/contrail-helm-deployer**.
2. Run the command `scp contrail-helm-deployer-release-tag.tgz` for all nodes in the cluster.
3. Untar **contrail-helm-deployer-release-tag.tgz** on all nodes:

```
tar -xzf contrail-helm-deployer-release-tag.tgz -C /opt/
```

4. Using any node in the cluster, export the following variables:

```
export BASE_DIR=/opt
export OSH_INFRA_PATH=${BASE_DIR}/openstack-helm-infra
export CHD_PATH=${BASE_DIR}/contrail-helm-deployer
```

5. In this step, all the required packages are installed and Kubernetes is deployed. If you want to install a different version of Kubernetes or CNI, edit the file **\${OSH\_INFRA\_PATH}/tools/gate/devel/multinode-vars.yaml**. Doing this overrides the default values in **\${OSH\_INFRA\_PATH}/playbooks/vars.yaml**. Following is an example **multinode-vars.yaml** file, with sample values indicated for the **private\_registries** section:

```
version:
  kubernetes: v1.9.3
  helm: v2.7.2
  cni: v0.6.0
docker:
  # list of insecure_registries, from where you will be pulling container images
  insecure_registries:
    - "10.87.65.243:5000"
  # list of private secure docker registry auth info, from where you will be pulling
  container images
  #private_registries:
  # - name: docker-registry-name
```

```
#   username: username@abc.xyz
#   email: username@abc.xyz
#   password: password
#   secret_name: contrail-image-secret
#   namespace: openstack
kubernetes:
  network:
    default_device: ens3
  cluster:
    cni: calico
    pod_subnet: 192.168.0.0/16
    domain: cluster.local
```

6. Install the dependent packages using `sudo apt-get`.

```
sudo apt-get update
sudo apt-get install --no-install-recommends -y ca-certificates make jq nmap curl uuid-
runtime ipcalc linux-headers-$(uname -r)
```

7. Prepare the nodes definition in `$OSH_INFRA_PATH/tools/gate/devel/multinode-inventory.yaml`, similar to this example:

```
all:
  children:
    primary:
      hosts:
        controller1:
          ansible_port: 22
          ansible_host: 10.10.0.1
          ansible_user: root
          ansible_ssh_extra_args: -o StrictHostKeyChecking=no
          ansible_ssh_private_key_file: /path/to/ssh/key/file
      nodes:
        hosts:
          controller2:
            ansible_port: 22
            ansible_host: 10.10.0.2
            ansible_user: root
            ansible_ssh_extra_args: -o StrictHostKeyChecking=no
            ansible_ssh_private_key_file: /path/to/ssh/key/file
```

8. Deploy k8s to the nodes and use the `kubectl get nodes` command to verify the deployment is successful.

```
cd ${OSH_INFRA_PATH}
make dev-deploy setup-host multinode
make dev-deploy k8s multinode

nslookup kubernetes.default.svc.cluster.local || /bin/true
kubectl get nodes -o wide
```

9. Set the correct labels for the nodes.

```
kubectl label node controller1.localdomain --overwrite openstack-compute-node=disable
kubectl label node controller1.localdomain opencontrail.org/controller=enabled
kubectl label node controller2.localdomain --overwrite openstack-compute-node=disable
kubectl label node controller2.localdomain opencontrail.org/controller=enabled
```

10. Deploy the OpenContrail charts.

```
cd $CHD_PATH
make
# Change k8s rbac settings
kubectl replace -f ${CHD_PATH}/rbac/cluster-admin.yaml
```

11. Prepare the values for Contrail in `/tmp/contrail.yml`, similar to the following example.



**NOTE:** This example uses bash variables you should replace with exact values using any preferred means (sed, eval, cat, and so on). Similarly, replace the other variables with actual values where indicated, including `IPDATA_SERVICE_HOST`, `METADATA_PROXY_SECRET`, and keystone IP/VIP details.

```
global:
  images:
    tags:
      kafka: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-external-kafka:${CONTRAIL_TAG:-latest}"
      cassandra: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-external-cassandra:${CONTRAIL_TAG:-latest}"
```



```

redis: "redis:4.0.2"
zookeeper: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-external-zookeeper:${CONTRAIL_TAG:-latest}"
contrail_control: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-control-control:${CONTRAIL_TAG:-latest}"
control_dns: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-control-dns:${CONTRAIL_TAG:-latest}"
control_named: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-control-named:${CONTRAIL_TAG:-latest}"
config_api: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-config-api:${CONTRAIL_TAG:-latest}"
config_devicemgr: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-config-devicemgr:${CONTRAIL_TAG:-latest}"
config_schema_transformer: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-config-schema:${CONTRAIL_TAG:-latest}"
config_svcmonitor: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-config-svcmonitor:${CONTRAIL_TAG:-latest}"
webui_middleware: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-webui-job:${CONTRAIL_TAG:-latest}"
webui: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-webui-web:${CONTRAIL_TAG:-latest}"
analytics_api: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-api:${CONTRAIL_TAG:-latest}"
contrail_collector: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-collector:${CONTRAIL_TAG:-latest}"
analytics_alarm_gen: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-alarm-gen:${CONTRAIL_TAG:-latest}"
analytics_query_engine: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-query-engine:${CONTRAIL_TAG:-latest}"
analytics_snmp_collector: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-snmp-collector:${CONTRAIL_TAG:-latest}"
contrail_topology: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-topology:${CONTRAIL_TAG:-latest}"
build_driver_init: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-kernel-build-init:${CONTRAIL_TAG:-latest}"
vrouter_agent: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-agent:${CONTRAIL_TAG:-latest}"
vrouter_init_kernel: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-kernel-init:${CONTRAIL_TAG:-latest}"
vrouter_dpdk: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-agent-dpdk:${CONTRAIL_TAG:-latest}"
vrouter_init_dpdk: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-kernel-init-dpdk:${CONTRAIL_TAG:-latest}"

```

```

    nodemgr: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-nodemgr:${CONTRAIL_TAG:-latest}"
    contrail_status: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-status:${CONTRAIL_TAG:-latest}"
    node_init: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-node-init:${CONTRAIL_TAG:-latest}"
    dep_check: quay.io/stackanetes/kubernetes-entrypoint:v0.2.1

contrail_env:
    CONTROLLER_NODES: 10.10.0.1,10.10.0.2
    LOG_LEVEL: SYS_DEBUG
    CLOUD_ORCHESTRATOR: openstack
    JVM_EXTRA_OPTS: "-Xms1g -Xmx2g"
    BGP_PORT: "1179"
    CONFIG_DATABASE_NODEMGR__DEFAULTS__minimum_diskGB: "2"
    DATABASE_NODEMGR__DEFAULTS__minimum_diskGB: "2"
    IPFABRIC_SERVICE_HOST: metadata IP of old OpenStack setup
    METADATA_PROXY_SECRET: metadata proxy secret of old OpenStack setup
endpoints:
    keystone:
        auth:
            username: admin
            password: password
            project_name: admin
            user_domain_name: admin_domain
            project_domain_name: admin_domain
            region_name: RegionOne
        hosts:
            default: keystone IP/VIP
        path:
            default: /v3
        port:
            admin:
                default: 35357
            api:
                default: 5000
        scheme:
            default: http
        host_fqdn_override:
            default: keystone IP/VIP
        namespace: null

```

12. If you are using a private registry, add the username and password under the `imageCredentials` section as follows:

```
global:
  images:
    imageCredentials:
      registry: ${CONTRAIL_REGISTRY:-opencontrailnightly}
      username: ${CONTRAIL_REG_USERNAME}
      password: ${CONTRAIL_REG_PASSWORD}
```

13. Finally, deploy the Contrail charts:

```
helm install --name contrail-thirdparty ${CHD_PATH}/contrail-thirdparty --
namespace=contrail --values=/tmp/contrail.yaml
helm install --name contrail-analytics ${CHD_PATH}/contrail-analytics --namespace=contrail
--values=/tmp/contrail.yaml
helm install --name contrail-controller ${CHD_PATH}/contrail-controller --
namespace=contrail --values=/tmp/contrail.yaml
```

After all containers are deployed, you can check cluster status using the `contrail-status` command. You can also use the Contrail web browser interface to view and verify the cluster status.

## Verifying Configuration for CNI for Kubernetes

### IN THIS SECTION

- [View Pod Name and IP Address | 212](#)
- [Verify Reachability of Pods | 212](#)
- [Verify If Isolated Namespace-Pods Are Not Reachable | 212](#)
- [Verify If Non-Isolated Namespace-Pods Are Reachable | 213](#)
- [Verify If a Namespace is Isolated | 214](#)

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 | 166](#)

[Installing and Provisioning Containerized Contrail Controller for Kubernetes](#)

## Kubernetes Updates to IP Fabric

### IN THIS SECTION

- [Reachability to Kubernetes Pods Using the IP Fabric Forwarding Feature | 215](#)
- [Service Isolation Through Virtual Networks | 215](#)

- [Contrail ip-fabric-snat Feature | 216](#)
- [Third-Party Ingress Controllers | 216](#)
- [Custom Network Support for Ingress Resources | 216](#)
- [Kubernetes Probes and Kubernetes Service Node-Port | 216](#)
- [Kubernetes 1.9 Network-Policy Support | 217](#)

This topic describes the updates to Kubernetes and supported features in Contrail Release 5.0:

## Reachability to Kubernetes Pods Using the IP Fabric Forwarding Feature

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. Since pods are in the overlay network, they cannot be reached directly from the underlay without a gateway or vRouter. In Contrail Release 5.0, the IP fabric forwarding (`ip-fabric-forwarding`) feature enables virtual networks to be created as part of the underlay network and eliminates the need for encapsulation and decapsulation of data. The `ip-fabric-forwarding` feature is only applicable for pod networks. If `ip-fabric-forwarding` is enabled, pod-networks are associated to `ip-fabric-ipam` instead of `pod-ipam` which is also a flat subnet.

The `ip-fabric-forwarding` feature is enabled and disabled in the global and namespace levels. By default, `ip-fabric-forwarding` is disabled in the global level. To enable it in global level, you must set “`ip_fabric_forwarding`” to “`true`” in the “[KUBERNETES]” section of the `/etc/contrail/contrail-kubernetes.conf` file. To enable or disable the feature in namespace level, you must set “`ip_fabric_forwarding`” to “`true`” or “`false`” respectively in namespace annotation. For example, “`opencontrail.org/ip_fabric_forwarding`”: “`true`”. Once the feature is enabled, it cannot be disabled.

For more information, see [Gateway-less Forwarding](#).

## Service Isolation Through Virtual Networks

In namespace isolation mode, services in one namespace are not accessible from other namespaces, unless security groups or network policies are explicitly defined to allow access. If any Kubernetes service is implemented by pods in an isolated namespace, those services are reachable only to pods in the same namespace through the Kubernetes `service-ip`.

The Kubernetes `service-ip` is allocated from the cluster network despite being in an isolated namespace. So, by default, service from one namespace can reach services from another namespace. However, security groups in isolated namespaces prevent reachability from external namespace and also prevent reachability from outside of the cluster. In order to enable access by external namespaces, the security group must be edited to allow access to all namespaces which defeats the purpose of isolation.

Contrail Release 5.0 enables service or ingress reachability from external clusters in isolated namespaces. Two virtual networks are created in isolated namespaces. One network is dedicated to pods and one is dedicated to services. Contrail network-policy is created between the pod network and the service network for reachability between pods and services. Service uses the same service-ipam which is a flat-subnet like pod-ipam. It is applicable for default namespace as well.

## Contrail ip-fabric-snat Feature

With the Contrail ip-fabric-snat feature, pods that are in the overlay can reach the Internet without floating IPs or a logical-router. The ip-fabric-snat feature uses compute node IP for creating a source NAT to reach the required services and is applicable only to pod networks. The kube-manager reserves ports 56000 through 57023 for TCP and 57024 through 58047 for UDP to create a source NAT in global-config during the initialization.

The ip-fabric-snat feature can be enabled or disabled in the global or namespace levels. By default, the feature is disabled in the global level. To enable the ip-fabric-snat feature in the global level, you must set “ip-fabric-snat” to “true” in the “[KUBERNETES]” section in the `/etc/contrail/contrail-kubernetes.conf` file. To enable or disable it in the namespace level, you must set “ip\_fabric\_snat” to “true” or “false” respectively in namespace annotation. For example, “opencontrail.org/ip\_fabric\_snat”: “true”. The ip\_fabric\_snat feature can be at enabled and disabled any time. To enable or disable the ip\_fabric\_snat feature in the default-pod-network, default namespace must be used. If the ip\_fabric\_forwarding is enabled, ip\_fabric\_snat is ignored.

For more information, see [Distributed SNAT](#).

## Third-Party Ingress Controllers

Multiple ingress controllers can co-exist in Contrail. If “kubernetes.io/ingress.class” is absent or is “opencontrail” in the annotations of the Kubernetes ingress resource, the kube-manager creates a HAProxy loadbalancer. Otherwise it is ignored and the respective ingress controller handles the ingress resource. Since Contrail ensures the reachability between pods and services, any ingress controller can reach the endpoints or pods directly or through services.

## Custom Network Support for Ingress Resources

Contrail supports custom networks in namespace level for pods. Starting with Contrail Release 5.0, custom networks are supported for ingress resources as well.

## Kubernetes Probes and Kubernetes Service Node-Port

The Kubelet needs reachability to pods for liveness and readiness probes. Contrail network policy is created between the IP fabric network and pod network to provide reachability between node and pods.



Whenever the pod network is created, the network policy is attached to the pod network to provide reachability between node and pods. So, any process in the node can reach the pods.

Kubernetes Service Node-Port is based on node reachability to pods. Since Contrail provides connectivity between node and pods through Contrail the network policy, Node Port is supported.

## Kubernetes 1.9 Network-Policy Support

Contrail Release 5.0 supports implementing Kubernetes network policy in Contrail using the Contrail firewall security policy framework. While Kubernetes network policy can be implemented using other security objects in Contrail like security groups and Contrail network policies, the support of tags by Contrail firewall security policy aids in the simplification and abstraction of workloads.

For more information, see ["Implementation of Kubernetes Network Policy with Contrail Firewall Policy" on page 217](#).

### RELATED DOCUMENTATION

[Implementation of Kubernetes Network Policy with Contrail Firewall Policy | 217](#)

[Contrail Integration with Kubernetes | 166](#)

## Implementation of Kubernetes Network Policy with Contrail Firewall Policy

### IN THIS SECTION

- [Kubernetes Network Policy Characteristics | 218](#)
- [Representing Kubernetes Network Policy as Contrail Firewall Security Policy | 219](#)
- [Contrail Firewall Policy Naming Convention | 220](#)
- [Implementation of Kubernetes Network Policy | 221](#)
- [Example Network Policy Configurations | 222](#)
- [Cluster-wide Policy Action Enforcement | 230](#)

Contrail Release 5.0 supports implementing Kubernetes network policy in Contrail using the Contrail firewall security policy framework. While Kubernetes network policy can be implemented using other

security objects in Contrail like security groups and Contrail network policies, the support of tags by Contrail firewall security policy aids in the simplification and abstraction of Kubernetes workloads.

Contrail firewall security policy allows decoupling of routing from security policies and provides multi-dimension segmentation and policy portability while significantly enhancing user visibility and analytics functions. Contrail firewall security policy uses tags to achieve multi-dimension traffic segmentation among various entities, and with security features. Tags are key-value pairs associated with different entities in the deployment. Tags can be pre-defined or custom defined. Kubernetes network policy is a specification of how groups of Kubernetes workloads, which are hereafter referred to as pods, are allowed to communicate with each other and other network endpoints. Network policy resources use labels to select pods and define rules which specify what traffic is allowed to the selected pods.

## Kubernetes Network Policy Characteristics

Kubernetes network policies have the following characteristics:

- A network policy is pod specific and applies to a pod or a group of pods. If a specified network policy applies to a pod, the traffic to the pod is dictated by rules of the network policy.
- If a network policy is not applied to a pod then the pod accepts traffic from all sources.
- A network policy can define traffic rules for a pod at the ingress, egress, or both directions. By default, a network policy is applied to the ingress direction, if no direction is explicitly specified.
- When a network policy is applied to a pod, the policy must have explicit rules to specify a whitelist of permitted traffic in the ingress and egress directions. All traffic that does not match the whitelist rules are denied and dropped.
- Multiple network policies can be applied on any pod. Traffic matching any one of the network policies must be permitted.
- A network policy acts on connections rather than individual packets. For example, if traffic from pod A to pod B is allowed by the configured policy, then the return packets for that connection from pod B to pod A are also allowed, even if the policy in place does not allow pod B to initiate a connection to pod A.
- **Ingress Policy:** An ingress rule consists of the identity of the source and the protocol:port type of traffic from the source that is allowed to be forwarded to a pod.

The identity of the source can be of the following types:

- Classless Interdomain Routing (CIDR) block—If the source IP address is from the CIDR block and the traffic matches the protocol:port, then traffic is forwarded to the pod.
- Kubernetes namespaces—Namespace selectors identify namespaces, whose pods can send the defined protocol:port traffic to the ingress pod.

- Pods—Pod selectors identify the pods in the namespace corresponding to the network policy, that can send matching protocol:port traffic to the ingress pods.
- **Egress Policy:** This specifies a whitelist CIDR to which a particular protocol:port type of traffic is permitted from the pods targeted by the network policy

The identity of the destination can be of the following types:

- CIDR block—If the destination IP address is from the CIDR block and the traffic matches the protocol:port, then traffic is forwarded to the destination.
- Kubernetes namespaces—Namespace selectors identify namespaces, whose pods can send the defined protocol:port traffic to the egress pod.
- Pods—Pod selectors identify the pods in the namespace corresponding to the network policy, that can receive matching protocol:port traffic from the egress pods.

### Representing Kubernetes Network Policy as Contrail Firewall Security Policy

Kubernetes and Contrail firewall policy are different in terms of the semantics in which network policy is specified in each. The key to efficient implementation of a Kubernetes network policy through Contrail firewall policy is in mapping the corresponding configuration constructs between these two entities.

The constructs are mapped as displayed in [Table 3 on page 219](#):

**Table 3: Kubernetes Network Policy and Contrail Firewall Policy Mapping**

Kubernetes Network Policy Constructs	Contrail Firewall Policy Constructs
Label	Custom Tag (one for each label)
Namespace	Custom Tag (one for each namespace)
Network Policy	Firewall Policy (one firewall policy per Network Policy)
Rule	Firewall Rule (one firewall rule per network policy rule)
CIDR Rules	Address Group
Cluster	Default Application Policy Set



**NOTE:** The project in which Contrail firewall policy constructs are created is the one that houses the Kubernetes cluster. For example, the Contrail firewall policy constructs are created in the global scope, if the Kubernetes cluster is a standalone cluster and the Contrail firewall policy constructs are created in the project scope, if the Kubernetes cluster is a nested cluster.

## Resolving Kubernetes Network Policy Labels

The representation of pods in Contrail firewall policy is exactly the same as in the corresponding Kubernetes network policy. Contrail firewall policy deals with labels or tags in Contrail terminology. Contrail does not expand labels to IP addresses.

For example, in the default namespace, if network policy-podSelector specifies: `role=db`, then the corresponding firewall rule specifies the pods as `(role=db && namespace=default)`. No other translations to pod IP address or otherwise are done.

If the same network-policy also has namespaceSelector as `namespace=myproject`, then the corresponding firewall rule represents that namespace as `(namespace=myproject)`. No other translations or rules representing pods in "myproject" namespace is done.

Similarly, each CIDR is represented by one rule. In essence, the Kubernetes network policy is translated 1:1 to Contrail firewall policy. There is only one additional firewall rule created for each Kubernetes network policy. The purpose of that rule is to implement the implicit deny requirements of the network policy and no other rule is created.

## Contrail Firewall Policy Naming Convention

Contrail firewall security policies and rules are named as follows:

- A Contrail firewall security policy created for a Kubernetes network policy is named in the following format:

```
< Namespace-name >-< Network Policy Name >
```

For example, a network policy "world" in namespace "Hello" is named:

```
Hello-world
```

- Contrail firewall rules created for a Kubernetes network policy are named in the following format:

```
< Namespace-name >-<PolicyType>-< Network Policy Name >-<Index of from/to blocks>-<selector
type>-<rule-index>-<svc/port index>
```

For example:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: world
  namespace: hello
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: frontend
```

A rule corresponding to this policy is named:

```
hello-ingress-world-0-podSelector-0-0
```

## Implementation of Kubernetes Network Policy

The contrail-kube-manager daemon binds Kubernetes and Contrail together. This daemon connects to the API server of Kubernetes clusters and converts Kubernetes events, including network policy events, into appropriate Contrail objects. With respect to a Kubernetes network policy, contrail-kube-manager performs the following actions:

- Creates a Contrail tag for each Kubernetes label
- Creates a firewall policy for each Kubernetes network policy
- Creates an Application Policy Set (APS) to represent the cluster. All firewall policies created in that cluster are attached to this application policy set.

- Modifications to existing Kubernetes network policies result in the corresponding firewall policies being updated.

## Example Network Policy Configurations

The following examples illustrate various sample network policies and the corresponding firewall security policies created.

### Example 1 - Conditional egress and ingress traffic

The following policy specifies a sample network policy with specific conditions for ingress and egress traffic to and from all pods in a namespace:

#### Sample Kubernetes network policy

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
    - Ingress
    - Egress
  ingress:
    - from:
        - ipBlock:
            cidr: 172.17.0.0/16
            except:
              - 172.17.1.0/24
        - namespaceSelector:
            matchLabels:
              project: myproject
        - podSelector:
            matchLabels:
              role: frontend
  ports:
    - protocol: TCP
      port: 6379
```

```

egress:
- to:
  - ipBlock:
      cidr: 10.0.0.0/24
  ports:
  - protocol: TCP
    port: 5978

```

### Sample Contrail firewall security policy

The test-network-policy defined in Kubernetes results in the following objects being created in Contrail.

*Tags*—The following tags are created, if they do not exist. In a regular workflow, these tags must have been created by the time the namespace and pods were created.

Key	Value
role	db
namespace	default

### Address Groups

The following address groups are created:

Name	Prefix
172.17.1.0/24	172.17.1.0/24
172.17.0.0/16	172.17.0.0/16
10.0.0.0/24	10.0.0.0/24

### Firewall Rules

The following firewall rules are created:

Rule Name	Action	Services	Endpoint1	Dir	Endpoint2	Match Tags
default-ingress-test-network-policy-0-ipBlock-0-172.17.1.0/24-0	deny	tcp:6379	Address Group: 172.17.1.0/24	>	role=db && namespace=default	
default-ingress-test-network-policy-0-ipBlock-0-cidr-172.17.0.0/16-0	pass	tcp:6379	Address Group: 172.17.0.0/16	>	role=db && namespace=default	
default-ingress-test-network-policy-0-namespaceSelector-1-0	pass	tcp:6379	project=myproject	>	role=db && namespace=default	
default-ingress-test-network-policy-0-podSelector-2-0	pass	tcp:6379	namespace=default && role=frontend	>	role=db && namespace=default	
default-egress-test-network-policy-ipBlock-0-cidr-10.0.0.0/24-0	pass	tcp:5978	role=db && namespace=default	>	Address Group: 10.0.0.0/24	

### Firewall Policy

The following firewall security policy is created with the following rules.

Name	Rules
default-test-network-policy	<ul style="list-style-type: none"> <li>• default-ingress-test-network-policy-0-ipBlock-0-172.17.1.0/24-0</li> <li>• default-ingress-test-network-policy-0-ipBlock-0-cidr-172.17.0.0/16-0</li> <li>• default-ingress-test-network-policy-0-namespaceSelector-1-0</li> <li>• default-ingress-test-network-policy-0-podSelector-2-0</li> <li>• default-egress-test-network-policy-ipBlock-0-cidr-10.0.0.0/24-0</li> </ul>



Example 2 - Allow all Ingress Traffic

The following policy explicitly allows all traffic for all pods in a namespace:

Sample Kubernetes network policy

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-ingress
spec:
  podSelector:
    ingress:
      - {}
```

Sample Contrail firewall security policy

*Tags*—The following tags are created, if they do not exist. In a regular workflow, these tags are created before the namespace and pods are created.

Key	Value
namespace	default

*Address Groups* - None

*Firewall Rules*

The following firewall rule is created:

Rule Name	Action	Services	Endpoint1	Dir	Endpoint2	Match Tags
default-ingress-allow-all-ingress-0-allow-all-0	pass	any	any	>	namespace=default	

*Firewall Policy*

The following firewall policy are created:

Name	Rules
default-allow-all-ingress	default-ingress-allow-all-ingress-0-allow-all-0

### Example 3 - Deny all ingress traffic

The following policy explicitly denies all ingress traffic to all pods in a namespace:

#### Sample Kubernetes network policy

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-ingress
spec:
  podSelector:
  policyTypes:
    - Ingress
```

#### Sample Contrail firewall security policy

Tags—The following tags are created, if they do not exist. In a regular workflow, these tags are created before the namespace and pods are created.

Key	Value
namespace	default

*Address Groups - None*

*Firewall Rules - None*



**NOTE:** The implicit behavior of any network policy is to deny traffic not matching explicit allow flows. However in this policy, there are no explicit allow rules. Hence, no firewall rules are created for this policy.

#### Firewall Policy

The following firewall policy is created:

Name	Rules
default-deny-ingress	

### Example 4 - Allow all egress traffic

The following policy explicitly allows all egress traffic from all pods in a namespace:

#### Sample Kubernetes network policy

```

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-egress
spec:
  podSelector:
    egress:
      - {}

```

#### Sample Contrail firewall security policy

Tags—The following tag is created, if they do not exist. In a regular workflow, these tags are created before the namespace and pods are created.

Key	Value
namespace	default

*Address Groups - None*

*Firewall Rules*

The following firewall rule is created:

Rule Name	Action	Services	Endpoint1	Dir	Endpoint2	Match Tags
default-egress-allow-all-egress-allow-all-0	pass	any	namespace=default	>	any	

### *Firewall Policy*

The following firewall policy is created:

Name	Rules
default-allow-all-egress	default-egress-allow-all-egress-allow-all-0

### **Example 5 - Default deny all egress traffic**

The following policy explicitly denies all egress traffic from all pods in a namespace:

#### **Sample Kubernetes network policy**

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all-egress
spec:
  podSelector: {}
  policyTypes:
    - Egress
```

#### **Sample Contrail firewall security policy**

Tags—The following tag is created, if they do not exist. In a regular workflow, these tags are created before the namespace and pods are created.

Key	Value
namespace	default

*Address Groups - None*

*Firewall Rules - None*



**NOTE:** The implicit behavior of any network policy with egress policy type is to deny egress traffic not matching explicit egress allow flows. In this policy, there are no explicit egress allow rules. Hence, no firewall rules are created for this policy.

*Firewall Policy*

The following firewall policy is created:

Name	Rules
default-deny-all-egress	

**Example 6 - Default deny all ingress and egress traffic**

The following policy explicitly denies all ingress and egress traffic to and from all pods in that namespace:

**Sample Kubernetes network policy**

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all-ingress-egress
spec:
  podSelector:
  policyTypes:
    - Ingress
    - Egress
```

**Sample Contrail firewall security policy**

Tags—The following tags is created, if they do not exist. In a regular workflow, these tags are created before the namespace and pods are created.

Key	Value
namespace	default

*Address Groups - None*

*Firewall Rules - None*



**NOTE:** The implicit behavior of any network policy with ingress/egress policy type is to deny corresponding traffic not matching explicit allow flows. In this policy, there are no explicit allow rules. Hence, no firewall rules are created for this policy.

*Firewall Policy*

The following firewall policy is created:

Name	Rules
default-deny-all-ingress-egress	

## Cluster-wide Policy Action Enforcement

The specification and the syntax of network policies allow for maximum flexibility and varied combinations. However, you must exercise caution while configuring the network policies.

Consider a case where two network policies are created:

Policy 1: Pod A can send to Pod B.

Policy 2: Pod B can only receive from Pod C.

From a networking flow perspective, there is an inherent contradiction between the above policies. Policy 1 states that a flow from Pod A to Pod B is allowed. Policy 2 implies that flow from Pod A to Pod B is not allowed. From a networking perspective, Contrail prioritizes flow behavior as more critical. In the event of inherent contradiction in network policies, Contrail will honor the flow perspective. One of the core aspects of this notion is that if a policy matches a flow, the action is honored cluster-wide.

For instance, if a flow matches a policy at the source, the flow will match the same policy in the destination as well.

Hence, the flow behavior in a Contrail-managed Kubernetes cluster is as follows:

1. Flow from Pod A to Pod B is allowed (due to Policy 1)
2. Flow from Pod C to Pod B is allowed (due to Policy 2)
3. Any other flow to Pod B is disallowed (due to Policy 2)

## Example Network Policy Action Enforcement Scenarios

Consider the following examples of network policy action enforcement:

- Allow all egress traffic and deny all ingress traffic

Setup: Namespace NS1 has two pods, Pod A and Pod B.

Policy: A network policy applied on namespace NS1 states:

- Rule 1. Allow all egress traffic from all pods in NS1.
- Rule 2. Deny all ingress traffic to all pods in NS1.

Behavior:

- Pod A can send traffic to Pod B (due to rule 1)
- Pod B can send traffic to Pod A (due to rule 1)
- PodX from a different namespace cannot send traffic to Pod A or Pod B (due to rule 2)
- Allow all ingress traffic and deny all egress traffic

Setup: Namespace NS1 has two pods, Pod A and Pod B.

Policy: A network policy applied on namespace NS1 states:

- Rule 1. Allow all ingress traffic to all pods in NS1
- Rule 2. Deny all egress traffic from all pods in NS1.

Behavior:

- Pod A can send traffic to Pod B (due to rule 1)
- Pod B can send traffic to Pod A (due to rule 1)
- Pod A and Pod B cannot send traffic to pods in any other namespace.
- Egress CIDR rule

Setup: Namespace NS1 has two pods, Pod A and Pod B.

Policy: A network policy applied on namespace NS1 states:

- Policy 1: Allow Pod A to send traffic to CIDR of Pod B.
- Policy 2: Deny all ingress traffic to all pods in NS1.
- Behavior:
  - Pod A can send traffic to Pod B (due to Policy 1)
  - All other traffic to Pod A and Pod B is dropped (due to policy 2)

## RELATED DOCUMENTATION

[Implementing Kubernetes Network Policy Using Contrail Firewall Policy](#)

[Kubernetes Updates to IP Fabric | 214](#)



# Using VMware vCenter with Containerized Contrail

## IN THIS CHAPTER

- [vCenter Integration for Contrail Release 5.0 | 233](#)
- [vCenter Integration for Contrail Release 5.0.1 | 235](#)
- [vCenter Integration for Contrail Release 5.0.2 | 237](#)
- [Underlay Network Configuration for ContrailVM | 246](#)
- [Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Releases 5.0 and 5.0.1 | 258](#)
- [Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Release 5.0.2 | 273](#)
- [Integrating Contrail Release 5.0.X with VMware vRealize Orchestrator | 280](#)
- [Installing and Provisioning Contrail VMware vRealize Orchestrator Plugin | 286](#)

## vCenter Integration for Contrail Release 5.0

### IN THIS SECTION

- [Prerequisites | 234](#)
- [Set Up vCenter Server | 234](#)
- [Configure Contrail Parameters and Install | 235](#)

This topic provides the steps for integrating Contrail Release 5.0 and microservices with VMware vCenter.

## Prerequisites

Before starting the integration, prepare your system by performing the following steps.

```
yum update -y

yum install -y yum-plugin-priorities https://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm

yum install -y python-pip git gcc python-devel sshpass

yum install -y git

yum install ansible-2.3.2.0

pip install pyvmomi
```

## Set Up vCenter Server

Set up your vCenter server with parameters defining such things as the data center, cluster, DVSwitches, ContrailVM, and the like.

Use the following to get the Ansible Deployer with Contrail playbooks.

1. From [Juniper Networks](#), download Ansible Deployer (contrail-ansible-deployer-5.0.0-0.40.tgz) onto your provisioning host.
2. Untar the tgz.

```
- tar xvf contrail-ansible-deployer-5.0.0-0.40.tgz
```

3. Copy the playbooks.

```
cp playbooks/roles/vcenter/vars/vcenter_vars.yml.sample playbooks/roles/vcenter/vars/vcenter_vars.yml
```

4. Prepare a file `vcenter_vars.yml` populated with vCenter server and ESXi hosts parameters. You can download the CentOS 7 + ESXi VM Host from [Juniper Networks](#).

For an example file, see [https://github.com/Juniper/contrail-ansible-deployer/blob/master/playbooks/roles/vcenter/vars/vcenter\\_vars.yml.sample](https://github.com/Juniper/contrail-ansible-deployer/blob/master/playbooks/roles/vcenter/vars/vcenter_vars.yml.sample).

5. Run the Contrail vCenter playbook.

```
ansible-playbook playbooks/vcenter.yml
```

## Configure Contrail Parameters and Install

1. Populate the file `config/instances.yaml` with Contrail roles.

For an example file, see [https://github.com/Juniper/contrail-ansible-deployer/blob/master/config/instances.yaml.vcenter\\_example](https://github.com/Juniper/contrail-ansible-deployer/blob/master/config/instances.yaml.vcenter_example).

2. Install Contrail by running the Contrail playbooks.

```
ansible-playbook -i inventory/ -e orchestrator=vcenter playbooks/configure_instances.yml
```

```
ansible-playbook -i inventory/ -e orchestrator=vcenter playbooks/install_contrail.yml
```

### RELATED DOCUMENTATION

[Installing and Provisioning VMware vCenter with Contrail](#)

[vCenter Integration for Contrail Release 5.0.1 | 235](#)

[vCenter Integration for Contrail Release 5.0.2 | 237](#)

[Underlay Network Configuration for ContrailVM | 246](#)

[Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Releases 5.0 and 5.0.1 | 258](#)

[Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Release 5.0.2 | 273](#)

## vCenter Integration for Contrail Release 5.0.1

### IN THIS SECTION

- [Set Up the Controller Server | 236](#)
- [Download the Ansible Deployer with Contrail Playbooks | 236](#)
- [Configure Contrail Parameters and Install Contrail | 236](#)

Starting in Contrail Release 5.0.1, the only supported mode of vCenter integration with Contrail is the vCenter-as-orchestrator mode. From Release 5.0.1, Contrail no longer supports the vCenter-as-compute mode, where the orchestrator is Openstack, and the vCenter cluster acts as a nova-compute node to the Openstack controller.

These topics provide instructions for integrating Contrail Release 5.0.1 and microservices with VMware vCenter.

## Set Up the Controller Server

The controller server or host server is installed with CentOS 7.5.

Before starting the integration, prepare your system by performing the following steps:

```
yum update -y
yum install -y yum-plugin-priorities
https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
yum install -y python-pip git gcc python-devel sshpass
pip install "ansible==2.5" pyvmomi
```

## Download the Ansible Deployer with Contrail Playbooks

1. Download the Contrail Ansible Deployer (**contrail-ansible-deployer-5.0.1-0.214.tgz**) onto your provisioning host. You can download the deployer from <https://www.juniper.net/support/downloads/?p=contrail#sw>.
2. Untar the **tgz**.

```
- tar xvf contrail-ansible-deployer-5.0.1-0.214.tgz
```

3. Prepare a **vcenter\_vars.yml** file populated with vCenter server and ESXi hosts parameters. You can download the CentOS 7.5 and ESXi VM Host from <https://www.juniper.net/support/downloads/?p=contrail#sw>.

For an example file, see [https://github.com/Juniper/contrail-ansible-deployer/blob/master /playbooks/roles/vcenter/vars/vcenter\\_vars.yml.sample](https://github.com/Juniper/contrail-ansible-deployer/blob/master/playbooks/roles/vcenter/vars/vcenter_vars.yml.sample).

4. Run the Contrail vCenter playbook.

```
ansible-playbook playbooks/vcenter.yml
```

## Configure Contrail Parameters and Install Contrail

1. Populate the **config/instances.yml** file with Contrail roles.

For an example file, see [https://github.com/Juniper/contrail-ansible-deployer/blob/master /config/instances.yml.vcenter\\_example](https://github.com/Juniper/contrail-ansible-deployer/blob/master/config/instances.yml.vcenter_example).

## 2. Install Contrail by running the Contrail playbooks.

```
ansible-playbook -i inventory/ -e orchestrator=vcenter playbooks/configure_instances.yml
ansible-playbook -i inventory/ -e orchestrator=vcenter playbooks/install_contrail.yml
```

### RELATED DOCUMENTATION

[Installing and Provisioning VMware vCenter with Contrail](#)

[vCenter Integration for Contrail Release 5.0 | 233](#)

[vCenter Integration for Contrail Release 5.0.2 | 237](#)

[Underlay Network Configuration for ContrailVM | 246](#)

[Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Releases 5.0 and 5.0.1 | 258](#)

[Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Release 5.0.2 | 273](#)

## vCenter Integration for Contrail Release 5.0.2

### IN THIS SECTION

- [Prerequisites | 238](#)
- [ESX Agent Manager | 238](#)
- [Set Up vCenter Server | 238](#)
- [Configure Contrail Parameters | 243](#)
- [Install Contrail | 243](#)
- [Monitor and Manage ContrailVM from ESX Agent Manager | 243](#)

These topics provide instructions for integrating Contrail Release 5.0.2 and microservices with VMware vCenter.

## Prerequisites

Before you start the integration, ensure that the contrail controller meets the prerequisites given in *Server Requirements and Supported Platforms*.

Follow these steps to prepare Contrail controller(s):

```
yum update -y

yum install -y yum-plugin-priorities https://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm

yum install -y python-pip git gcc python-devel sshpass

yum install -y git

pip install "ansible==2.5.0" pyvmomi
```

## ESX Agent Manager

VMware provides a standard vCenter solution called vSphere ESX Agent Manager (EAM), that allows you to deploy, monitor, and manage ContrailVMs on ESXi hosts.

Starting in Contrail Release 5.0.2, the ContrailVM is deployed as an Agent VM that is monitored by EAM. With this integration, ContrailVMs are marked as more critical and privileged than other tenant VMs on the host.

The following are the benefits of running ContrailVM as an AgentVM from EAM:

- Auto-deploy ContrailVMs on ESXi hosts in scope (clusters).
- Manage and Monitor ContrailVMs through EAM in the vSphere web client.
- Integrate with other vCenter features like AddHos, Maintenance Mode, vSphere DRS, vSphere DPM, and VMWare HA.

These topics provide instructions for integrating Contrail Release 5.0.2 and microservices with VMware vCenter.

## Set Up vCenter Server

Follow these steps to set up the vCenter server.

1. Download the Contrail Ansible Deployer (**contrail-ansible-deployer-*< >*.tgz**) onto your provisioning host. You can download the deployer from <https://www.juniper.net/support/downloads/?p=contrail#sw>.

## 2. Untar the **tgz**.

```
- tar xvf contrail-ansible-deployer-< >.tgz
```

3. Prepare a **vcenter\_vars.yml** file populated with vCenter server and ESXi hosts parameters. You can download the CentOS 7.5 and ESXi VM Host from <https://www.juniper.net/support/downloads/?p=contrail#sw>.



**NOTE:** You can see a sample of the **vcenter\_vars.yml** file in the **contrail-ansible-deployer/playbooks /roles/vcenter/vars/vcenter\_vars.yml** after you extract the image files.



**NOTE:** The ContrailVM's Open Virtualization Format (OVF) image must be hosted on an http or https server which runs on and is reachable from the vCenter server. The location of the OVF is provided as a URL path for **vmrk**: as shown in the example given below.

### Example: Enabling HA and DRS in the cluster

**vcenter\_servers:**

```
- SRV1:
  hostname:
  username:
  password:
  # Optional: defaults to False
  #validate_certs: False
  datacentername:
  clusternames:
  #path to the ovf, is needed for ESX Agent Manager to deploy ContrailVMs
  vmrk: http://<ip-address>/centos-7.5/LATEST/ContrailVM.ovf
  # Optional: If not specified HA and DRS are turned off on the clusters.
  enable_ha: yes
  enable_drs: yes
```

For definition examples, refer **contrail-ansible-deployer/playbooks/roles/vcenter/vars/vcenter\_vars.yml.sample**.

To enable HA and DRS in the cluster, set **enable\_ha** and **enable\_drs** to **yes** in the **vcenter\_vars.yml** file. If these flags are not enabled, HA and DRS is turned off by default for newly created and existing clusters.

## Example instances.yaml File

```

provider_config:
  bms:
    ssh_pwd: password
    ssh_user: root
    ntpserver: 8.8.8.8
    domainsuffix: blah.net

instances:
  bms1:
    provider: bms
    ip: <ip-address>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      vcenter_plugin:
  bms2:
    provider: bms
    esxi_host: <ip-address>
    ip: <ip-address>
    roles:
      vrouter:
      vcenter_manager:
        ESXI_USERNAME: root
        ESXI_PASSWORD: password
  bms3:
    provider: bms
    esxi_host: <ip-address>
    ip: <ip-address>
    roles:
      vrouter:
      vcenter_manager:
        ESXI_USERNAME: root
        ESXI_PASSWORD: password
  bms4:
    provider: bms
    esxi_host: <ip-address>

```



```

ip: <ip-address>
roles:
  vrouter:
    vcenter_manager:
      ESXI_USERNAME: root
      ESXI_PASSWORD: password

global_configuration:
  CONTAINER_REGISTRY: hub.juniper.net/contrail
  CONTAINER_REGISTRY_USERNAME: username
  CONTAINER_REGISTRY_PASSWORD: password
  REGISTRY_PRIVATE_INSECURE: False

contrail_configuration:
  CLOUD_ORCHESTRATOR: vcenter
  CONTROLLER_NODES: <ip-address>
  CONTRAIL_VERSION: 5.0.2-0.360
  RABBITMQ_NODE_PORT: 5673
  VCENTER_SERVER: <ip-address>
  VCENTER_USERNAME: administrator@vsphere.net
  VCENTER_PASSWORD: password
  VCENTER_DATACENTER: <DC name here>
  VCENTER_DVSWITCH: overlay
  VCENTER_WSDL_PATH: /usr/src/contrail/contrail-web-core/webroot/js/vim.wsd
  VCENTER_AUTH_PROTOCOL: https

```



**NOTE:** The default login credentials for Contrail OVF is

Username: *root*

Password: *cOntrail123*

### Example vcenter\_vars.yml File

```

---
vcenter_servers:
  - SRV1:
      hostname: <host-ip-address>
      username: administrator@vsphere.net
      password: password
      # Optional: defaults to False

```

```

#validate_certs: False
datacentername: "<your DC name here>"
clusternames:
  - "<your cluster name here>"
vmdk: http://<ip-address>/contrail/images/ContrailVM.ovf
dv_switch:
  dv_switch_name: overlay
dv_port_group:
  dv_portgroup_name: VM_pg
  number_of_ports: 1800

esxihosts:
  - name: <ip-address>
    username: root
    password: password
    datastore: <your local datastore here>
    datacenter: "<your DC name here>"
    cluster: "<your cluster name here>"
    contrail_vm:
      networks:
        - mac: 00:77:56:aa:bb:01
    vcenter_server: SRV1 #leave this
  - name: <ip-address>
    username: root
    password: password
    datastore: <your local datastore here>
    datacenter: "<your DC name here>"
    cluster: "<your cluster name here>"
    contrail_vm:
      networks:
        - mac: 00:77:56:aa:bb:02
    vcenter_server: SRV1 #leave this
  - name: <ip-address>
    username: root
    password: password
    datastore: <your local datastore here>
    datacenter: "<your DC name here>"
    cluster: "<your cluster name here>"
    contrail_vm:
      networks:
        - mac: 00:77:56:aa:bb:77
    vcenter_server: SRV1 #leave this

```

#### 4. Run the Contrail vCenter playbook.

```
ansible-playbook playbooks/vcenter.yml
```



**NOTE:** Verify that the hostnames for the contrail controller(s) and the ContrailVMs (vRouters) are unique in `/etc/hostname` file.

You can verify hostname from either the DHCP options (if the management network uses DHCP) or manually (if the management network uses static IP allocation).

### Configure Contrail Parameters

Populate the file `config/instances.yaml` with Contrail roles.

For an example file, see `contrail-ansible-deployer/config/instances.yaml.vcenter_example`.

### Install Contrail

Install Contrail by running the following Contrail playbooks:

```
ansible-playbook -i inventory/ -e orchestrator=vcenter playbooks/configure_instances.yml
ansible-playbook -i inventory/ -e orchestrator=vcenter playbooks/install_contrail.yml
```

### Monitor and Manage ContrailVM from ESX Agent Manager

ContrailVMs can be monitored from EAM by using ContrailVM-Agency.

Follow these steps to monitor and manage Contrail VM from EAM:

#### 1. Resolve issues from the ContrailVM-Agency.

The ContrailVM-Agency is in an alert state when the ContrailVM in any host is powered off or is deleted.

Click **Resolve All Issues** from the ContrailVM-Agency to correct the issue. The ContrailVM-Agency will attempt to correct the issue by bringing the ContrailVM back online or by spawning a ContrailVM from the OVF on the ESXi host.

Figure 22: vCenter Server Extensions

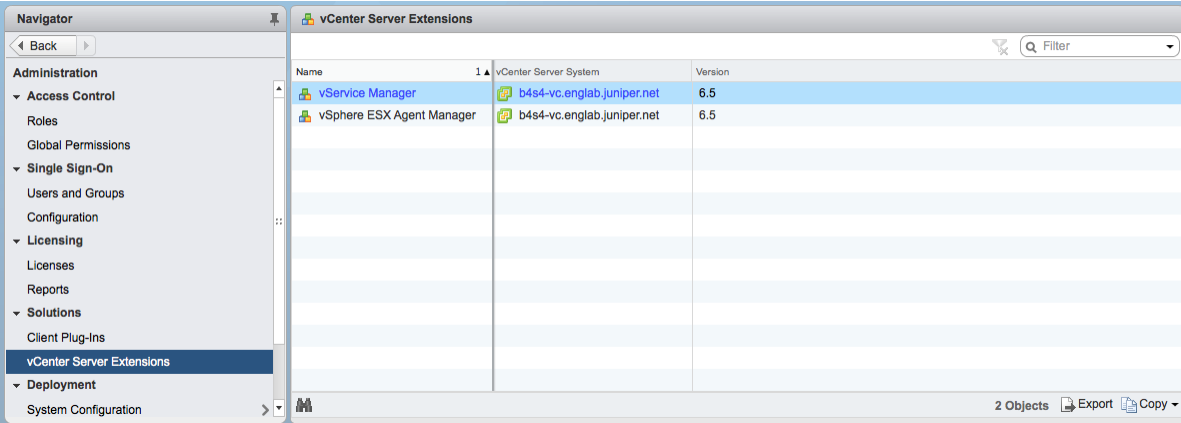
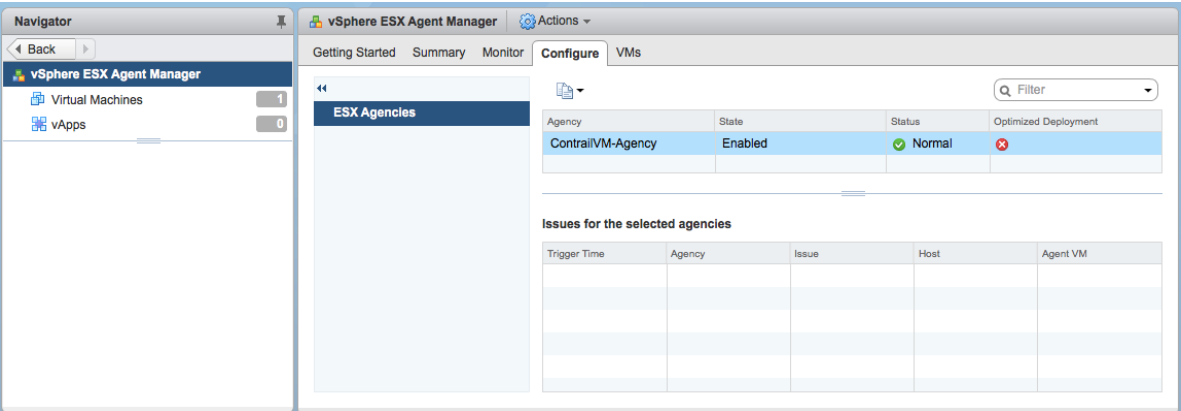
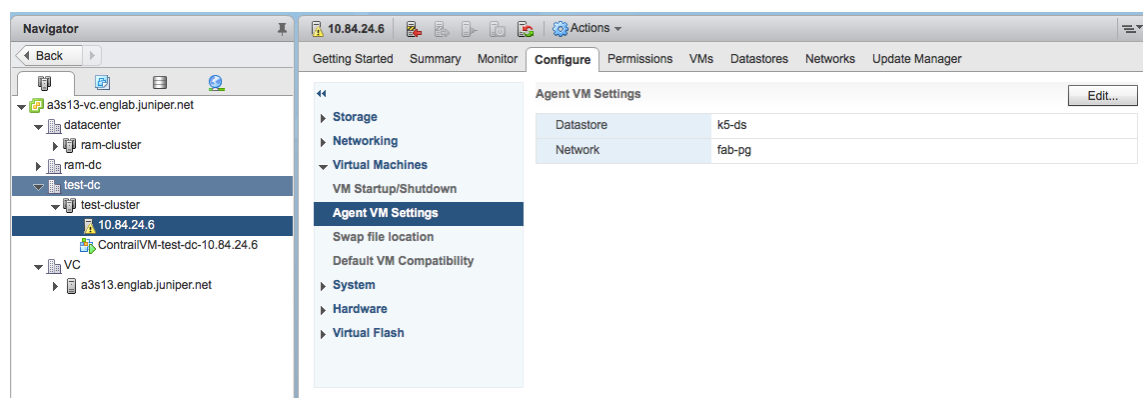


Figure 23: ESX Agencies



- 2. Add host.
  - a. Add ESXi host to the cluster.
  - b. Configure **Agent VM Settings** for the ESXI host.

Figure 24: Configure Agent VM Settings



For more information on configuring Agent VM, network, and datastore settings, see [Configure Agent VM Settings](#).

EAM deploys a ContrailVM (from the base OVF) on the ESXi host.

- c. Add ESXi host details to **vcenter\_vars.yml** and repeat step 4 to add appropriate interfaces to the ContrailVM and to configure necessary settings in the vCenter server.
  - d. Add ContrailVM details to **instances.yaml** and provision Contrail on the newly added ContrailVm (router). For more information on provisioning Contrail, see ["Install Contrail" on page 243](#).
3. Clean up the ContrailVM-Agency.
- Delete **ContrailVM-Agency** from the EAM user interface to delete ContrailVM and the agency.

## RELATED DOCUMENTATION

[Installing and Provisioning VMware vCenter with Contrail](#)

[vCenter Integration for Contrail Release 5.0 | 233](#)

[vCenter Integration for Contrail Release 5.0.1 | 235](#)

[Underlay Network Configuration for ContrailVM | 246](#)

[Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Releases 5.0 and 5.0.1 | 258](#)

[Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Release 5.0.2 | 273](#)

## Underlay Network Configuration for ContrailVM

### IN THIS SECTION

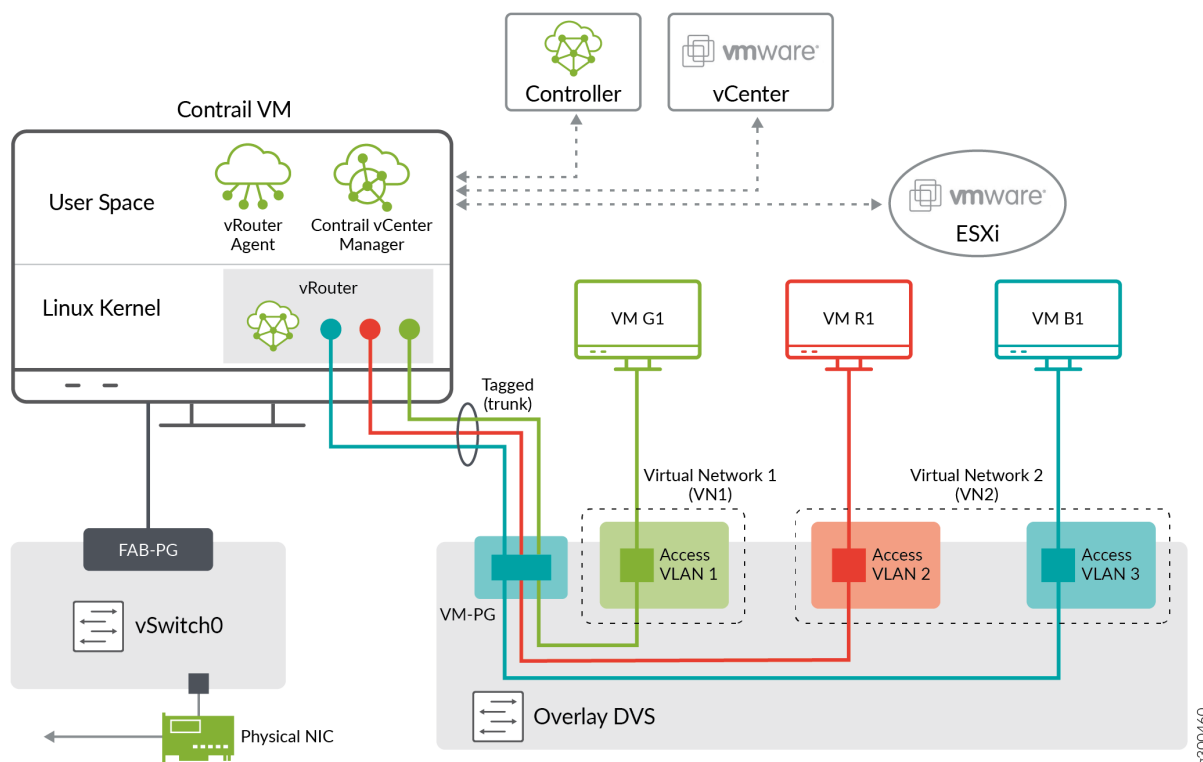
- [Standard Switch Setup | 246](#)
- [Distributed Switch Setup | 248](#)
- [PCI Pass-Through Setup | 250](#)
- [SR-IOV Setup | 253](#)

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 25 on page 247](#).

Figure 25: Standard Switch Setup



To set up the ContrailVM in this mode, the standard switch and port group must be configured in `vcenter_vars.yml`.

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 IP address of the `control_data` interface must be configured by the user and ensure connectivity. Additional configuration such as static routes and bond interface must be configured by the user.

The following is an example of configuration with standard switch.

```
- name: <esxi_host>
  username: <username>
  password: <password>
  datastore: <datastore>
  vcenter_server: <server>
  datacenter: <datacenter>
```

```
cluster: <cluster>
std_switch_list:
  - pg_name: mgmt-pg
    switch_name: vSwitch0
contrail_vm:
  networks:
    - mac: 00:77:56:aa:bb:03
      sw_type: standard
      switch_name: vSwitch0
      pg: mgmt-pg
```

## Distributed Switch Setup

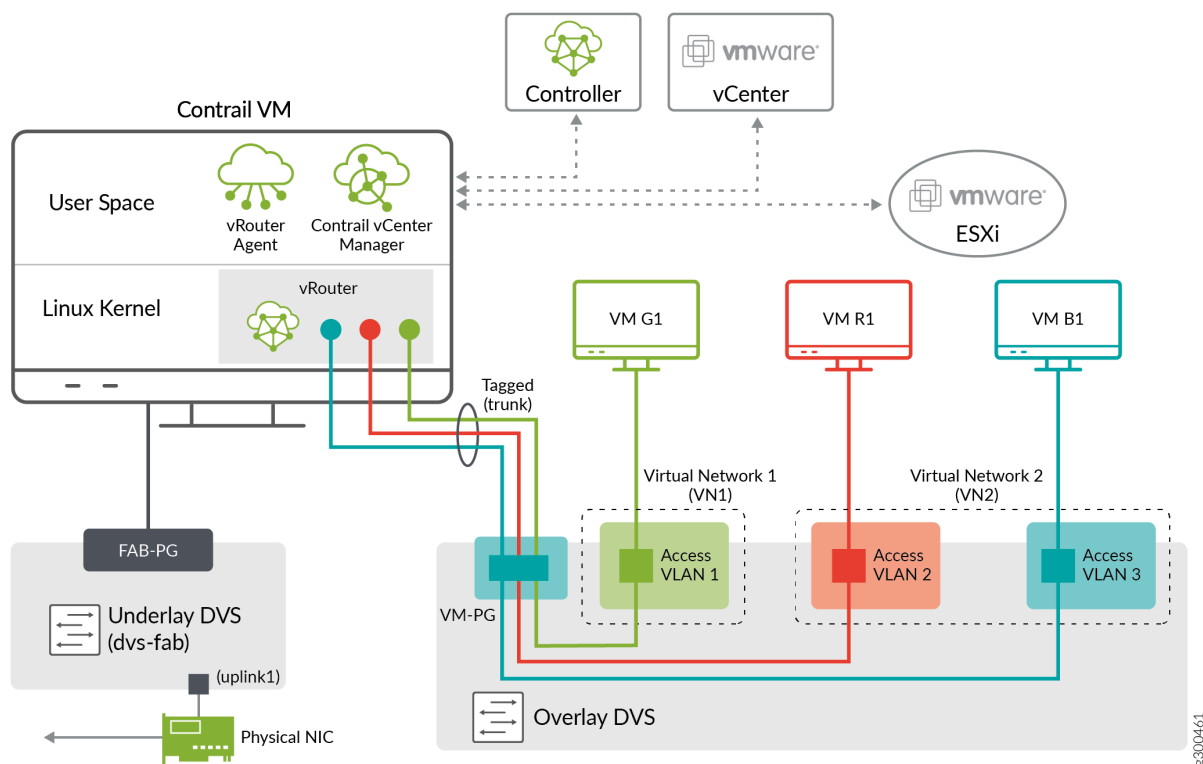
A distributed switch functions as a single virtual switch across associated hosts.

In the distributed switch setup, the ContrailVM is provided an interface through the distributed switch port group that is used for management and control data, see [Figure 26 on page 249](#).

The ContrailVM can be configured to use the management and control\_data NICs from DVS. When the DVS configuration is specified, the standard switch configuration is ignored.



Figure 26: Distributed Switch Setup



To set up the ContrailVM in this mode, configure the distributed switch, port group, number of ports in the port group, and the uplink in the `vcenter_servers` section in `vcenter_servers.yml`.



**NOTE:** The uplink can be a link aggregation group (LAG). If you use LAG, then DVS and LAG should be preconfigured.

The following is an example distributed switch configuration in `vcenter_vars.yml`.

```
vcenter_servers:
- SRV1:
  hostname: <server>
  username: <username>
  password: <password>
  datacentername: <datacenter>
  clusternames:
    - <cluster>
```

```

dv_switch:
  dv_switch_name: <dvs_name>
dv_port_group:
  dv_portgroup_name: <pg_name>
  number_of_ports: <num_of_ports>
dv_switch_control_data:
  dv_switch_name: <ctrl_dvs_name>
dv_port_group_control_data:
  dv_portgroup_name: <ctrl_pg_name>
  number_of_ports: <num_of_ports>
uplink:
  - 'vmnic3'

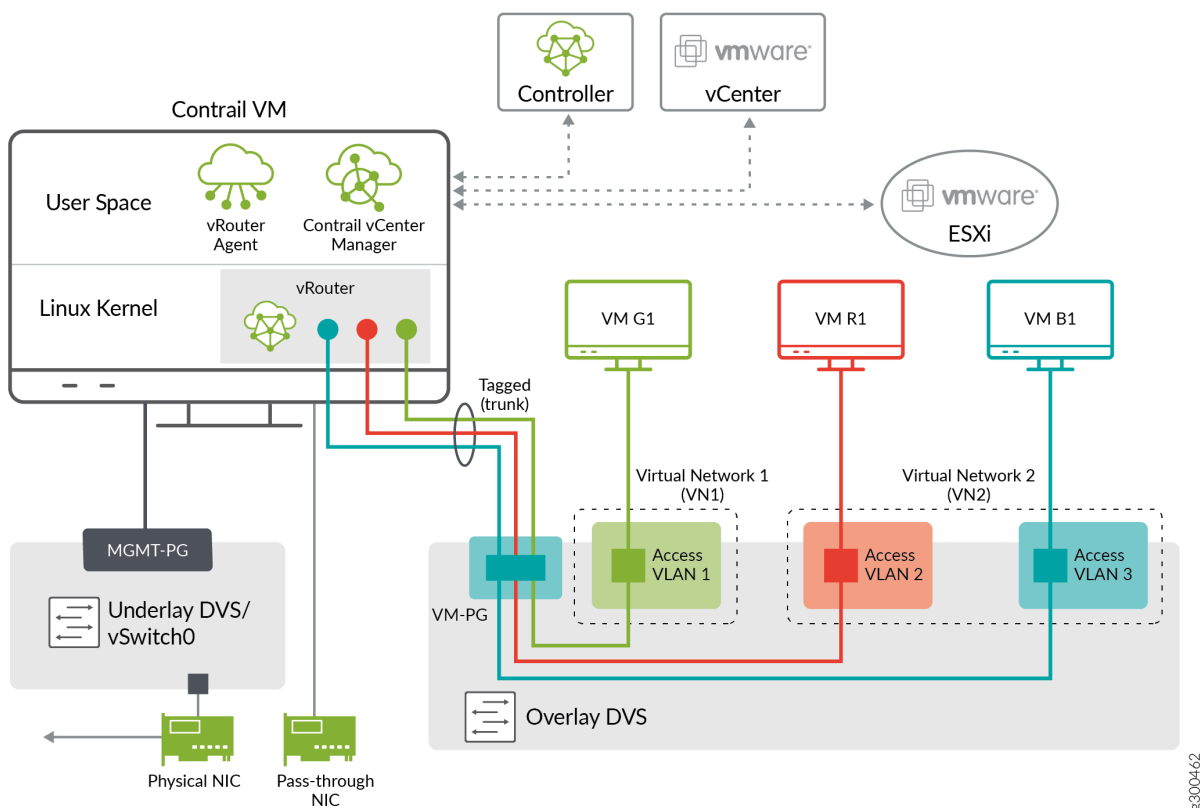
```

## 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 27 on page 251](#) shows a PCI pass-through setup with a single control\_data interface.

Figure 27: 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.

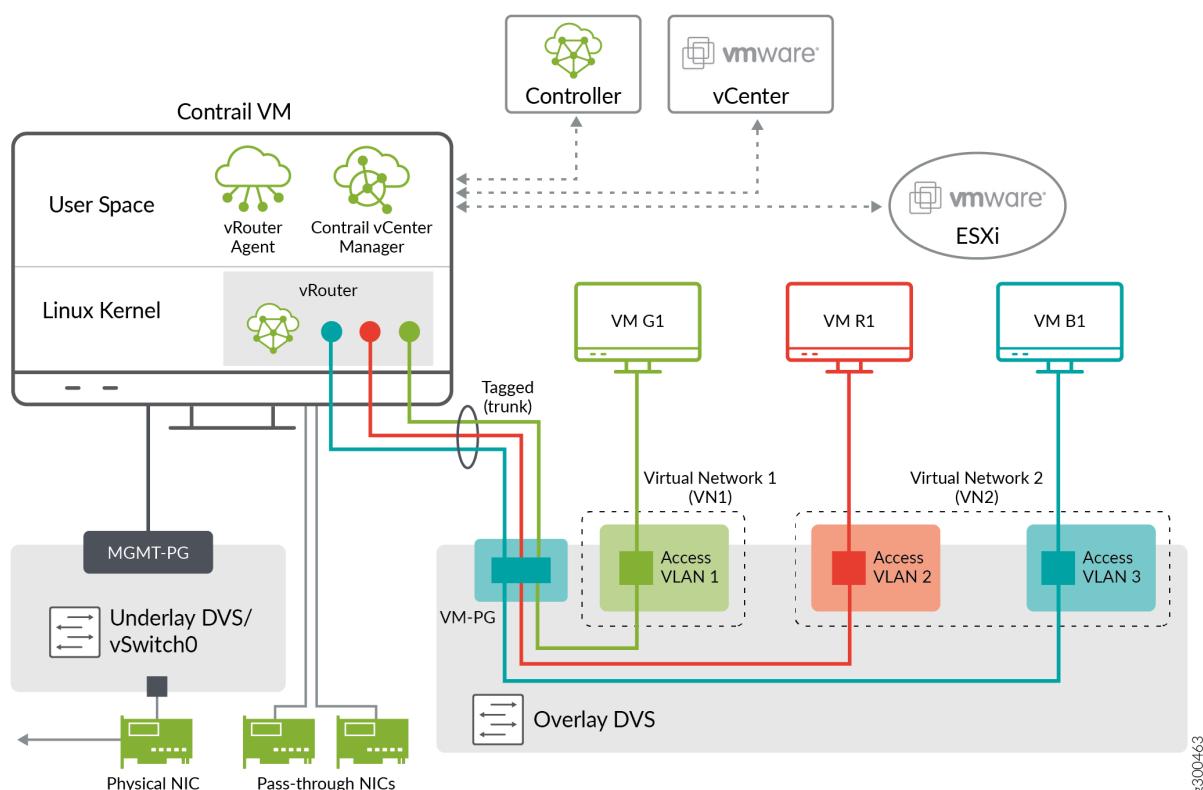
The following is an example PCI pass-through configuration with a single `control_data` interface:

```
esxihosts:
  - name: <esxi_host>
    username: <username>
    password: <password>
    datastore: <datastore>
    vcenter_server: <server>
    datacenter: <datacenter>
    cluster: <cluster>
    contrail_vm:
      networks:
        - mac: <mac_addr>
```

```
pci_devices:
  - '0000:04:00.0'
```

Figure 28 on page 252 shows a PCI pass-through setup with a bond\_control data interface, which has multiple pass-through NICs.

Figure 28: PCI Pass-Through Setup with Bond Control Interface



Update the ContrailVM section in `vcenter_vars.yml` with `pci_devices` as shown in the following example:

```
esxihosts:
  - name: <esxi_host>
    username: <username>
    password: <password>
    datastore: <datastore>
    vcenter_server: <server>
    datacenter: <datacenter>
    cluster: <cluster>
    contrail_vm:
```

```

networks:
  - mac: <mac_addr>
pci_devices:
  - '0000:04:00.0'
  - '0000:04:00.1'

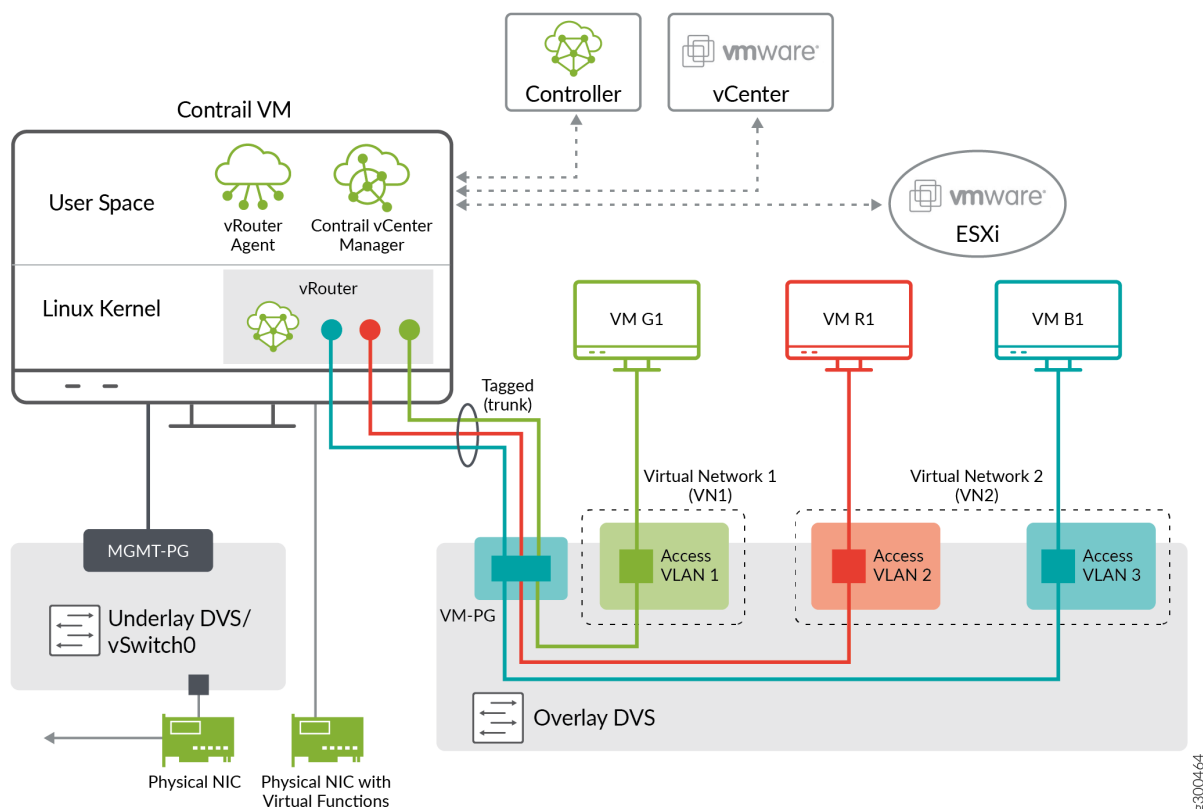
```

## 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 29 on page 253](#).

Figure 29: 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](#).

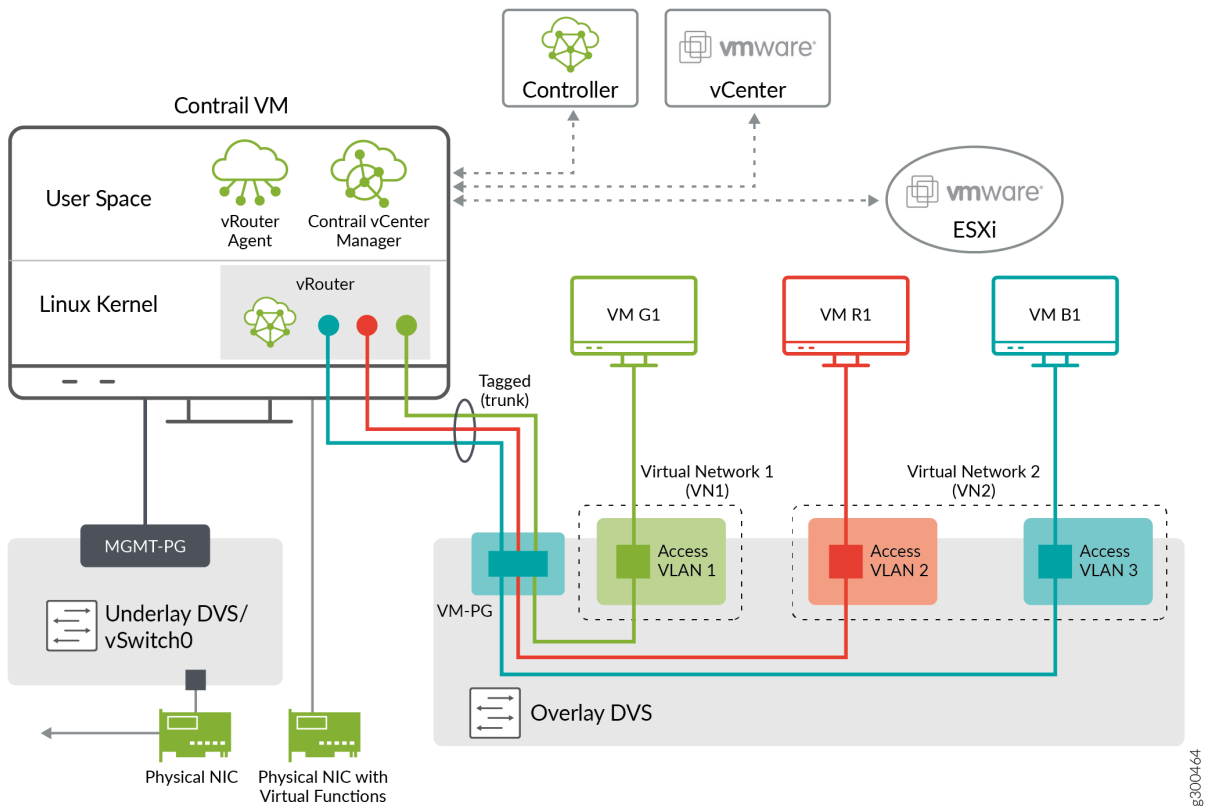
To set up the ContrailVM with SR-IOV interfaces, all configurations used for the standard switch setup are also used for the pass-through setup, providing management connectivity to the ContrailVM.

To provide the `control_data` interfaces, configure the SR-IOV-enabled physical interfaces in the `contrail_vm` section, and configure the `control_data` in the global section of `vcenter_vars.yml`.

Upon provisioning ESXi hosts in the installation process, the SR-IOV interfaces are exposed as Ethernet interfaces in the ContrailVM.

Figure 30 on page 254 shows a SR-IOV setup with a single `control_data` interface.

**Figure 30: SR-IOV With Single Control Data Interface**



The following is an example SR-IOV configuration for the cluster and server configuration.

The cluster configuration:

```
vcenter_servers:
  - SRV1:
    hostname: <server>
```

```

username: <username>
password: <password>
datacentername: <datacenter>
clusternames:
  - <cluster>

dv_switch:
  dv_switch_name: <dvs_name>
dv_port_group:
  dv_portgroup_name: <pg_name>
  number_of_ports: <num_of_ports>
dv_switch_sr_iov:
  dv_switch_name: <sriov_dvs_name>
dv_port_group_sr_iov:
  dv_portgroup_name: <sriov_pg_name>
  number_of_ports:

```

The server configuration:

```

esxihosts:
  - name: <esxi_host>
    username: <username>
    password: <password>
    datastore: <datastore>
    vcenter_server: <server>
    datacenter: <datacenter>
    cluster: <cluster>
    contrail_vm:

    networks:
      - mac: <mac_addr>
    sr_iov_nics:
      - 'vmnic0'

```

[Figure 31 on page 256](#) shows an SR-IOV configuration with a bond control\_data interface, which has multiple SR-IOV NICs.





```

    dv_portgroup_name: <pg_name>
    number_of_ports: <num_of_ports>
  dv_switch_sr_iov:
    dv_switch_name: <sriov_dvs_name>
  dv_port_group_sriov:
    dv_portgroup_name: <sriov_pg_name>
    number_of_ports:

```

The server configuration:

```

esxihosts:
  - name: <esxi_host>
    username: <username>
    password: <password>
    datastore: <datastore>
    vcenter_server: <server>
    datacenter: <datacenter>
    cluster: <cluster>
    contrail_vm:

    networks:
      - mac: <mac_addr>
    sr_iov_nics:
      - 'vmnic0'
      - 'vmnic1'

```

## RELATED DOCUMENTATION

[Installing and Provisioning VMware vCenter with Contrail](#)

[vCenter Integration for Contrail Release 5.0 | 233](#)

[vCenter Integration for Contrail Release 5.0.1 | 235](#)

[vCenter Integration for Contrail Release 5.0.2 | 237](#)

[Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Releases 5.0 and 5.0.1 | 258](#)

[Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Release 5.0.2 | 273](#)

## Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Releases 5.0 and 5.0.1

### IN THIS SECTION

- [Overview: User Interfaces for Contrail Integration with VMware vCenter | 258](#)
- [Feature Configuration for Contrail vCenter | 259](#)
- [Creating a Virtual Machine | 265](#)
- [Configuring the vCenter Network in Contrail UI | 272](#)

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

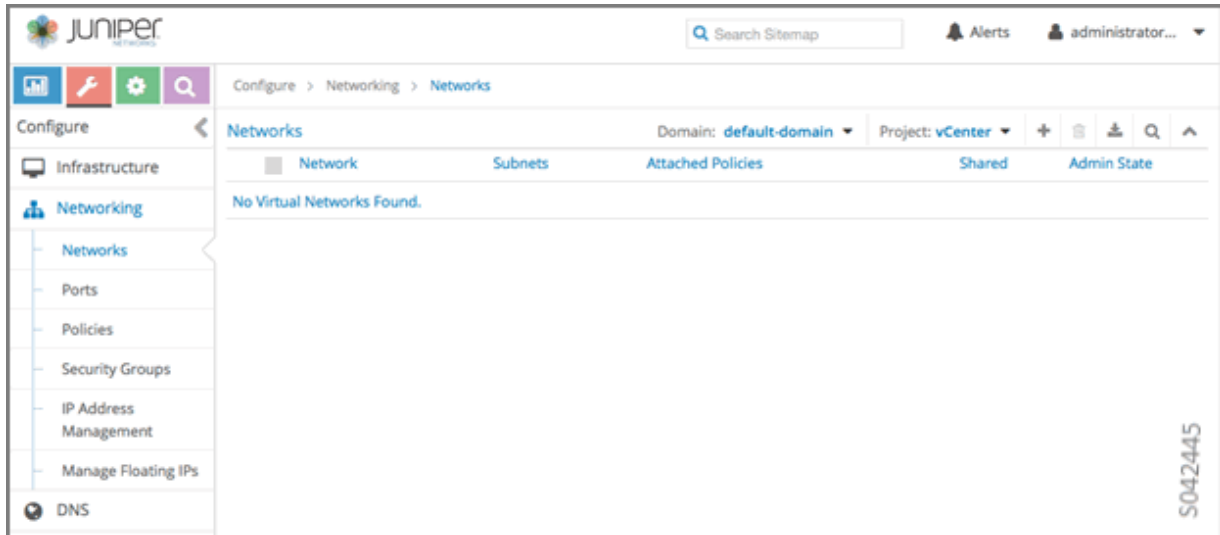
This section shows how to use the Contrail UI and the Contrail vCenter UI to configure features for the Contrail vCenter integrated system.

### 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.

Click **Save** to create the virtual network.

The virtual network you just created (Green-VN) is displayed in the Networks page.

## 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.

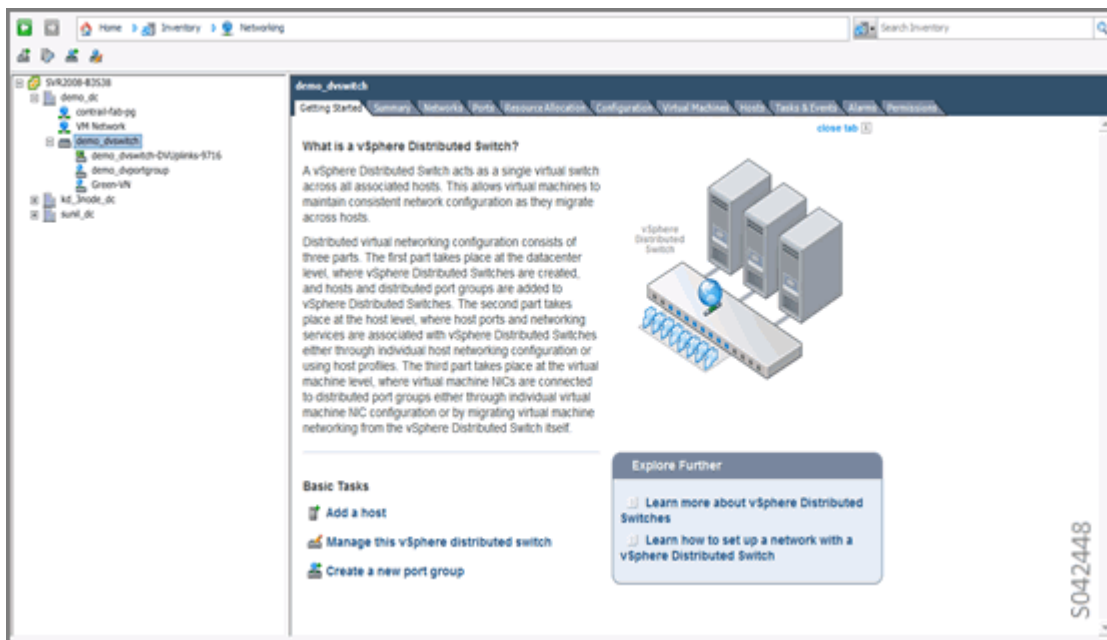
In vCenter, a virtual network is called a port group, which is part of a distributed switch.

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**.

**Create Distributed Port Group**

**Properties**  
How do you want to identify this network?

**Properties**  
Ready to Complete

Properties

Name: Red-VN

Number of Ports: 128

VLAN type: Private VLAN

Private VLAN Entry: Isolated (102, 103)

Help < Back Next > Cancel

S042449

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.

**Create Distributed Port Group**

**Ready to Complete**  
Verify the settings for new port group.

**Properties**  
Ready to Complete

The following port group will be created:

Name: Red-VN

Number of Ports: 128

VLAN type: Private VLAN

Private VLAN: Primary private VLAN ID: 102, Secondary private VLAN ID: 103, Type: Isolated

Help < Back Finish Cancel

S042450

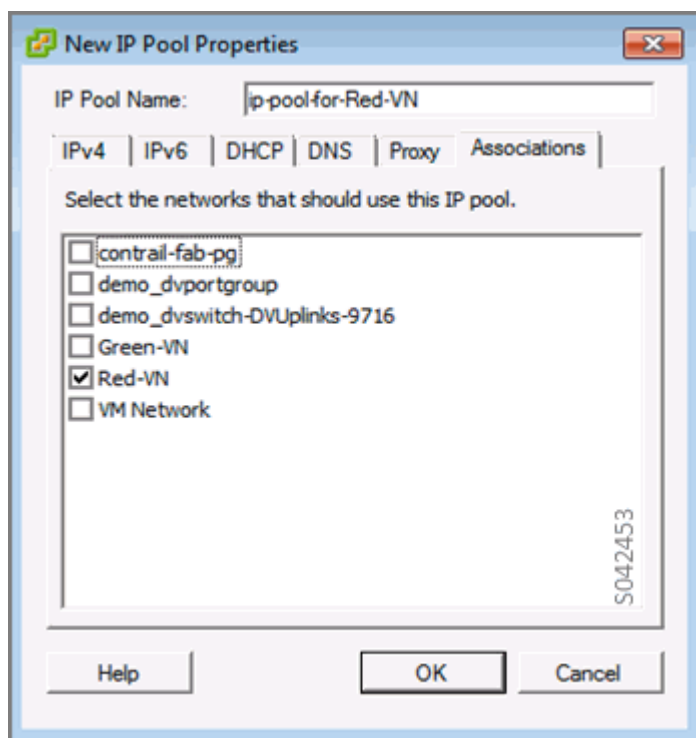
Next, create IP pools for the virtual network port group. Select the data center name in the left side panel, then click the **IP Pools** tab.

Near the top of the **IP Tools** window, click **Add** to open the **New IP Pool Properties** window. The **New 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 in the **IP Pool Name** field. 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.

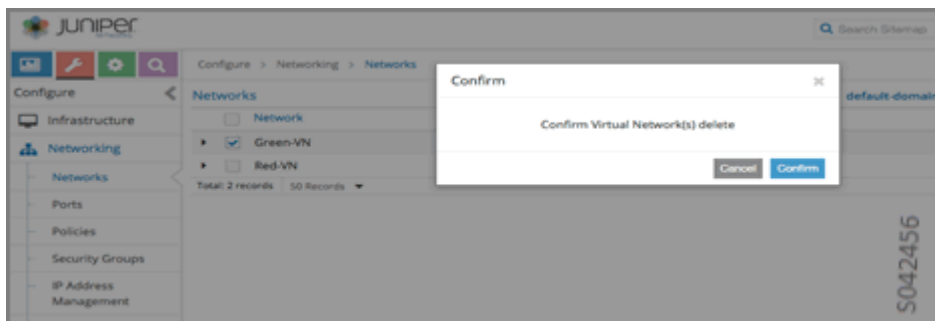
## Delete Virtual Network – Contrail UI

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, and click the delete 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.



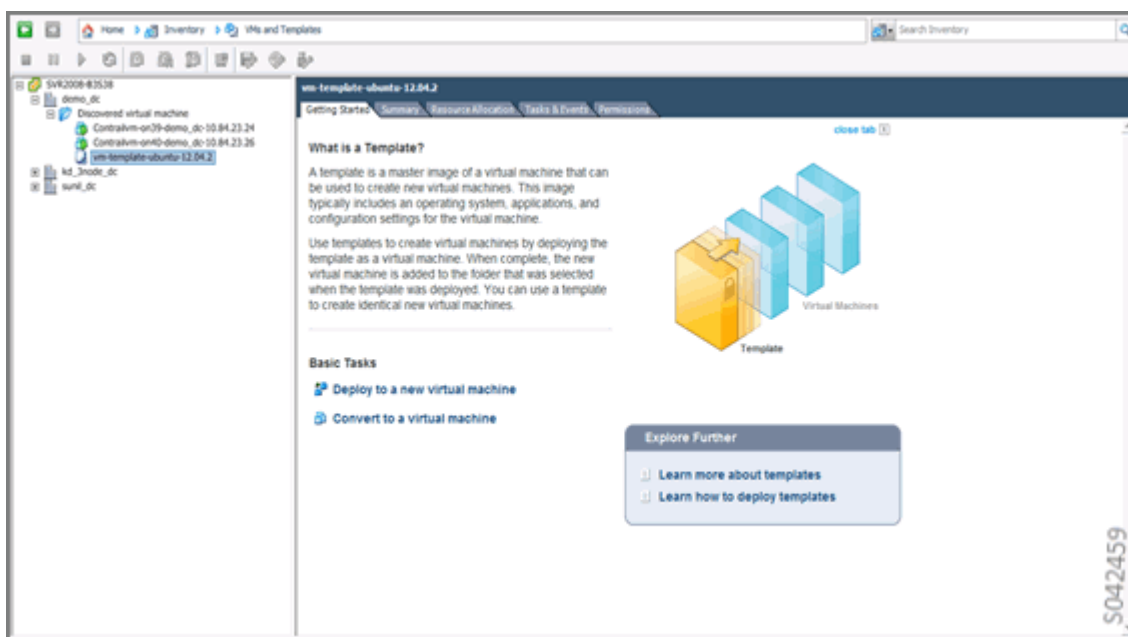
## Creating a Virtual Machine

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

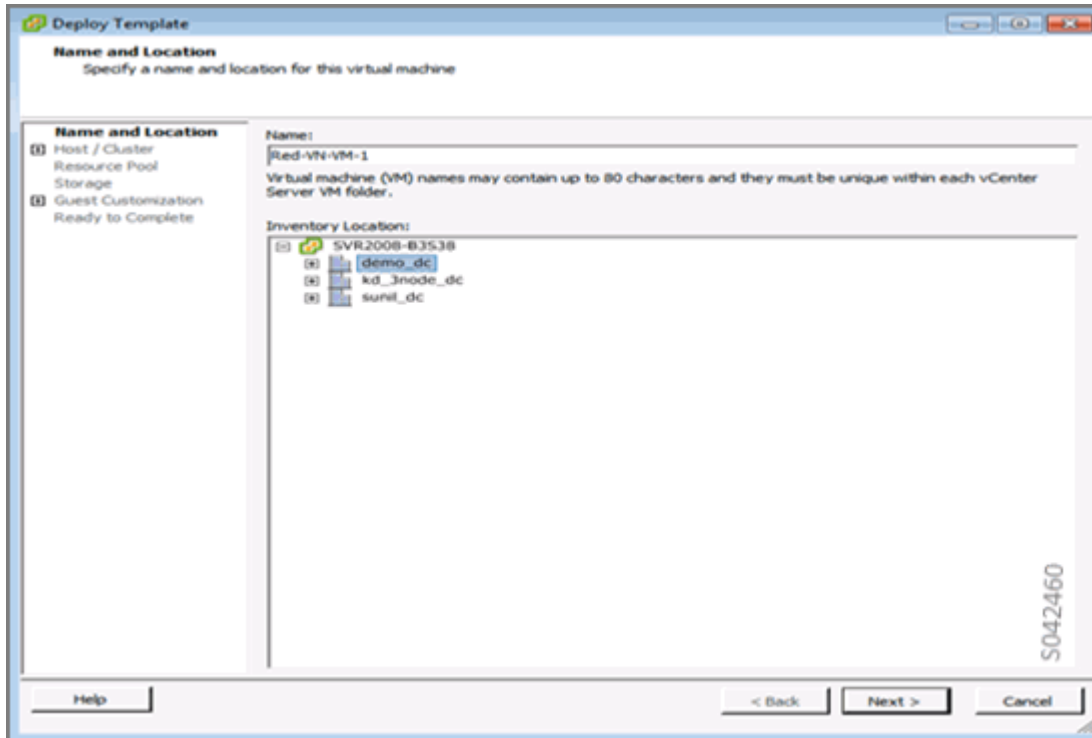
From the vCenter UI, select the virtual machine template from the left side panel. At the bottom of the right side pane, click **Deploy** to deploy a new virtual machine.

The following figure shows the **vm-template-ubuntu-12.04.2** virtual machine selected.



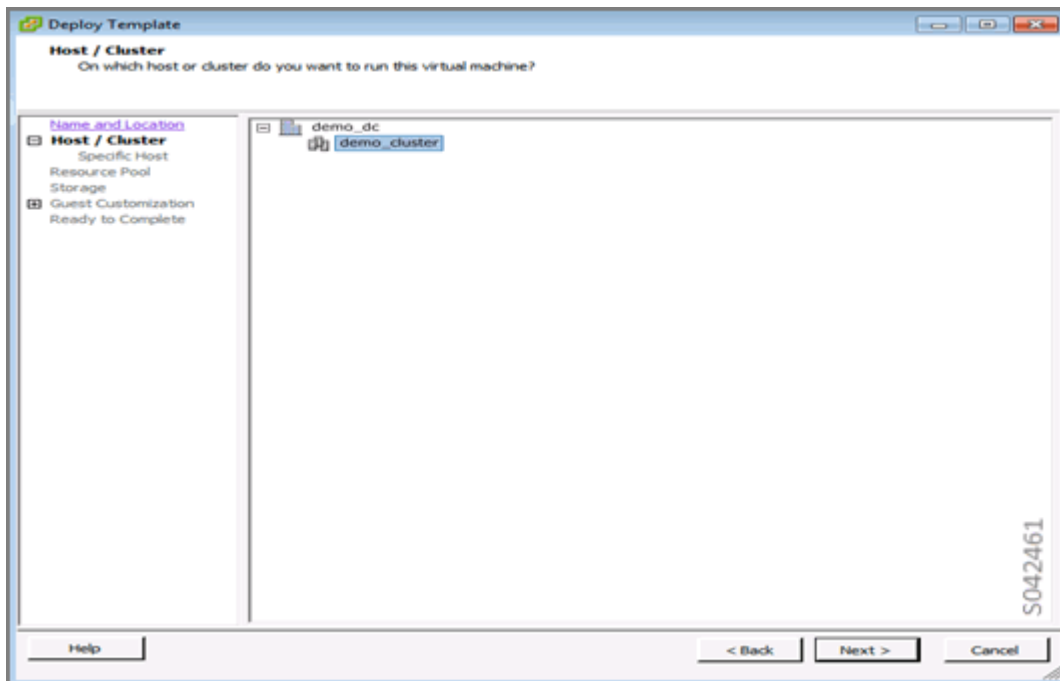
The **Deploy Template** Name and Location window is displayed, as in the following. Specify a name for the virtual machine and select the data center on which the virtual machine is to be spawned.

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.

Select the ESXi host on which to spawn the virtual machine and click **Next**.

From the **Storage** window, select the destination storage location for the virtual machine and click **Next**.

**Deploy Template**

**Storage**  
Select a destination storage for the virtual machine files

**Name and Location**  
Host / Cluster

**Storage**  
Guest Customization  
Ready to Complete

Select a virtual disk format:  
Same format as source

Select a destination storage for the virtual machine files:  
VM Storage Profile: [Warning Icon]

Name	Drive Type	Capacity	Provisioned	Free	Type	Thin Provisioned
b3s39-ds1	Non-SSD	465.50 GB	156.32 GB	309.18 GB	VMF55	Supported
b3s39-sys	SSD	104.25 GB	972.00 MB	103.30 GB	VMF55	Supported
cs-shared	Unknown	4.75 TB	2.13 TB	2.64 TB	NFS	Supported

☐ Disable Storage DRS for this virtual machine

Select a datastore:

Name	Drive Type	Capacity	Provisioned	Free	Type	Thin Provisioned
------	------------	----------	-------------	------	------	------------------

Advanced >>

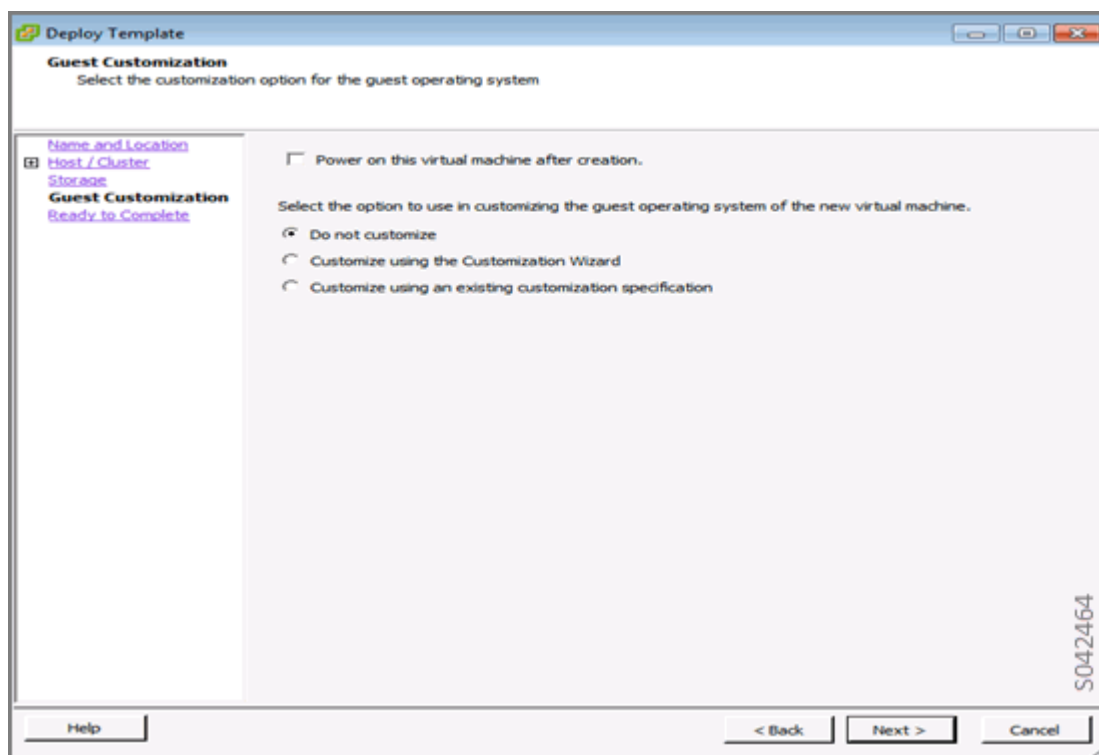
Compatibility:  
Validation succeeded

Help < Back Next > Cancel

S042463

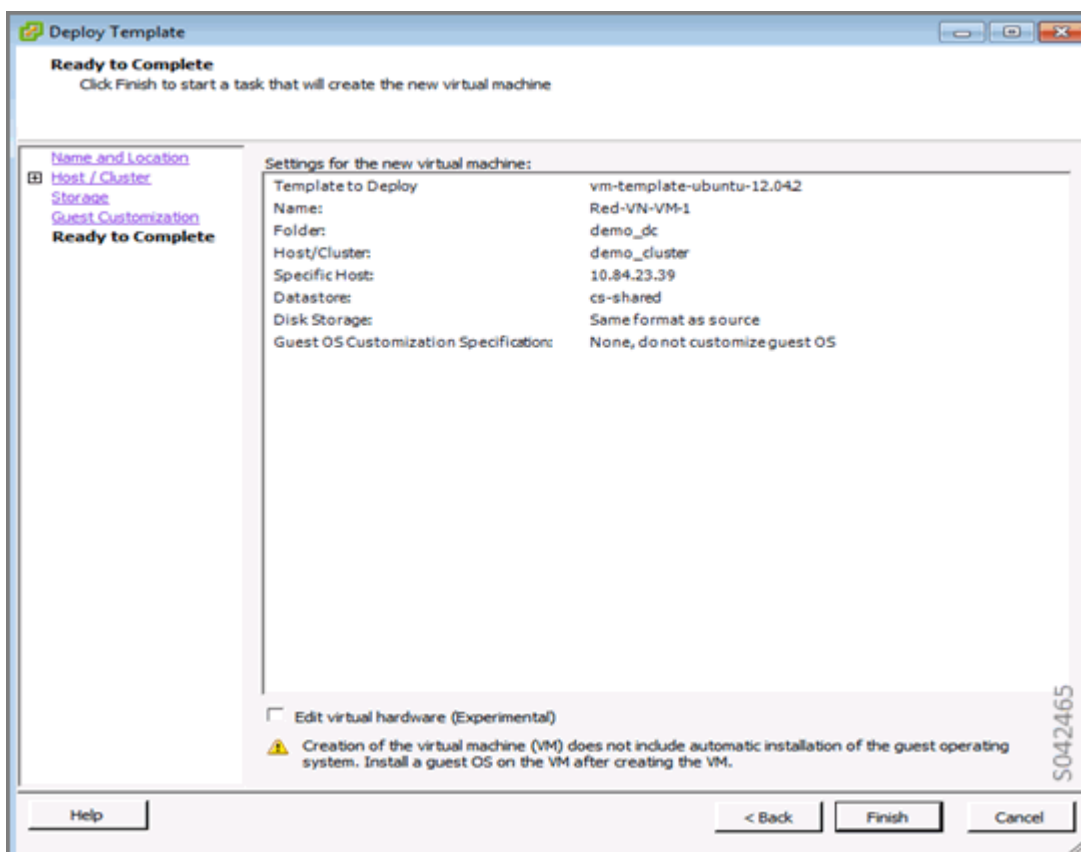
On the **Guest Customization** window, select **Do not customize**.

When you are finished, click **Next**.



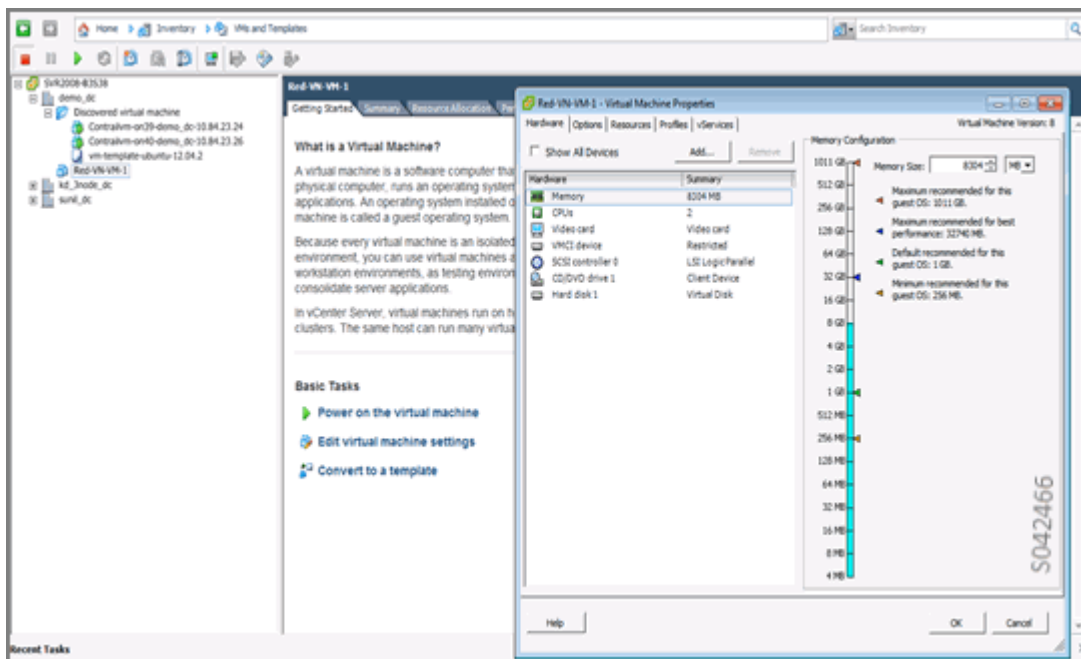
On the **Ready to Complete** window, review all of the virtual machine definitions that you have selected for the template.

If all the selections are correct, click **Finish**. This spawns the virtual machine.



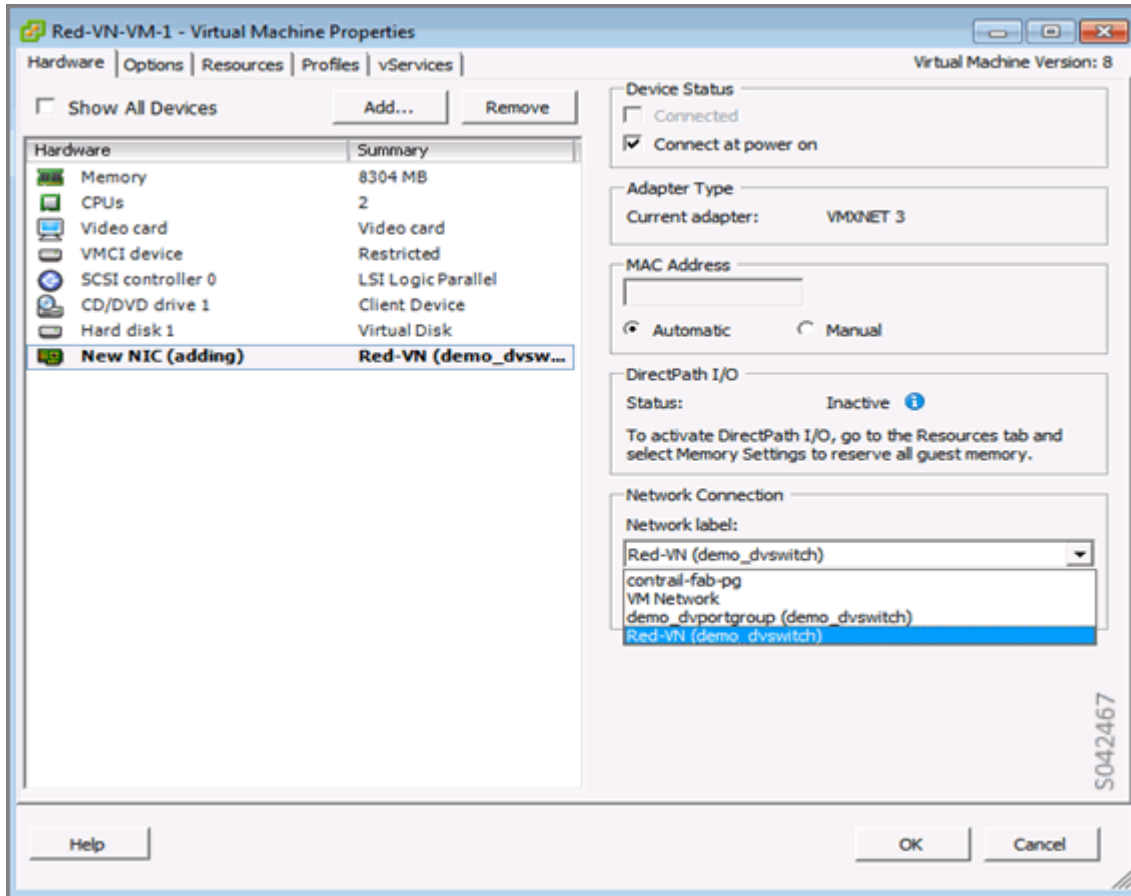
To complete the settings for the virtual machine, select the virtual machine to be edited in the left column of the main window of the vCenter UI. Then click **Edit virtual machine settings**.

The **Virtual Machine Properties** window is displayed, as in the following. From here you can update the virtual machine properties.

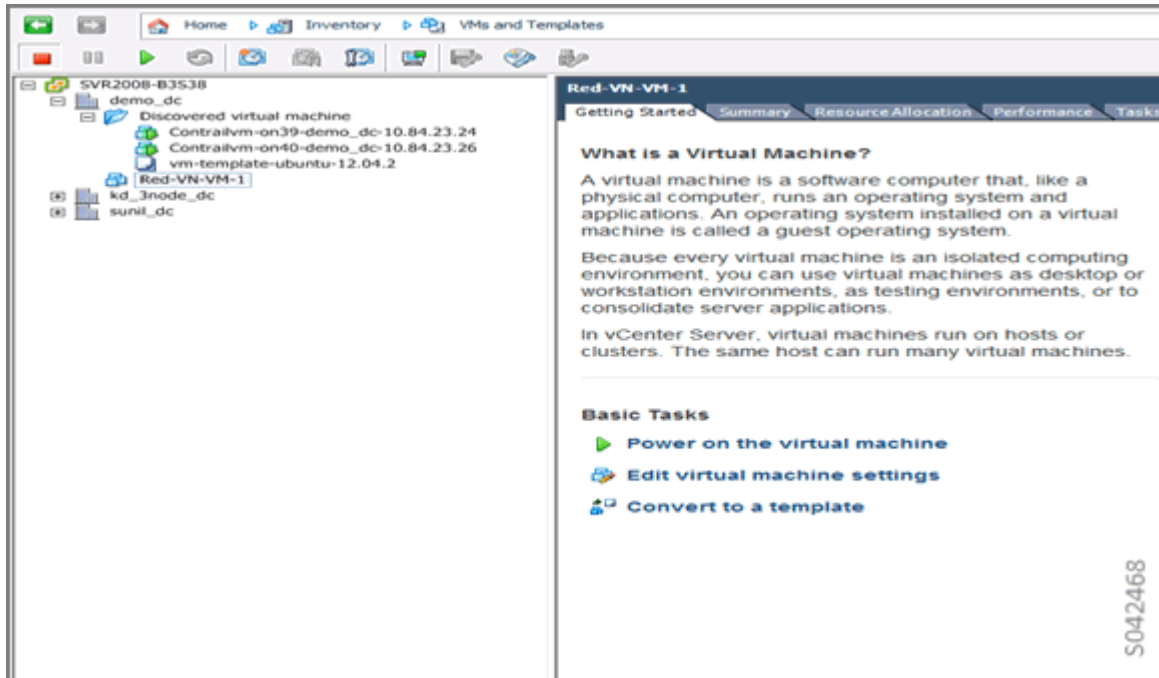


Click the **Hardware** tab in the **Virtual Machine Properties** window. Next, click **Add** to add a NIC and select the appropriate network. Select **Connect at power on**, as in the following.

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.

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.

## 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

[Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Release 5.0.2 | 273](#)

[Installing and Provisioning VMware vCenter with Contrail](#)

[vCenter Integration for Contrail Release 5.0 | 233](#)

[vCenter Integration for Contrail Release 5.0.1 | 235](#)

[vCenter Integration for Contrail Release 5.0.2 | 237](#)

[Underlay Network Configuration for ContrailVM | 246](#)

## Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Release 5.0.2

### IN THIS SECTION

- [Overview: User Interfaces for Contrail Integration with VMware vCenter | 273](#)
- [Feature Configuration for Contrail vCenter | 274](#)
- [Creating a Virtual Machine | 275](#)

You can install Contrail to work with the VMware vCenter Server version 6.5 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

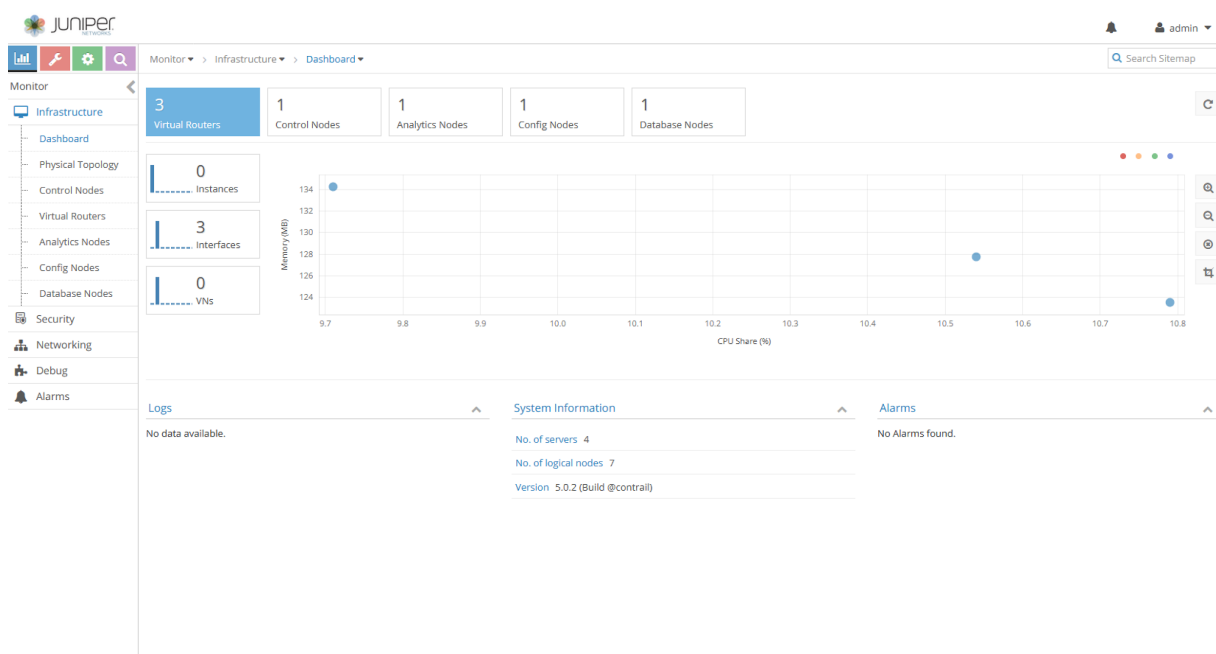
The Contrail UI is an administrator's user interface. It provides a view of all components managed by the Contrail controller.

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.

Upon successful login, the Contrail vCenter user interface is displayed.



## Feature Configuration for Contrail vCenter

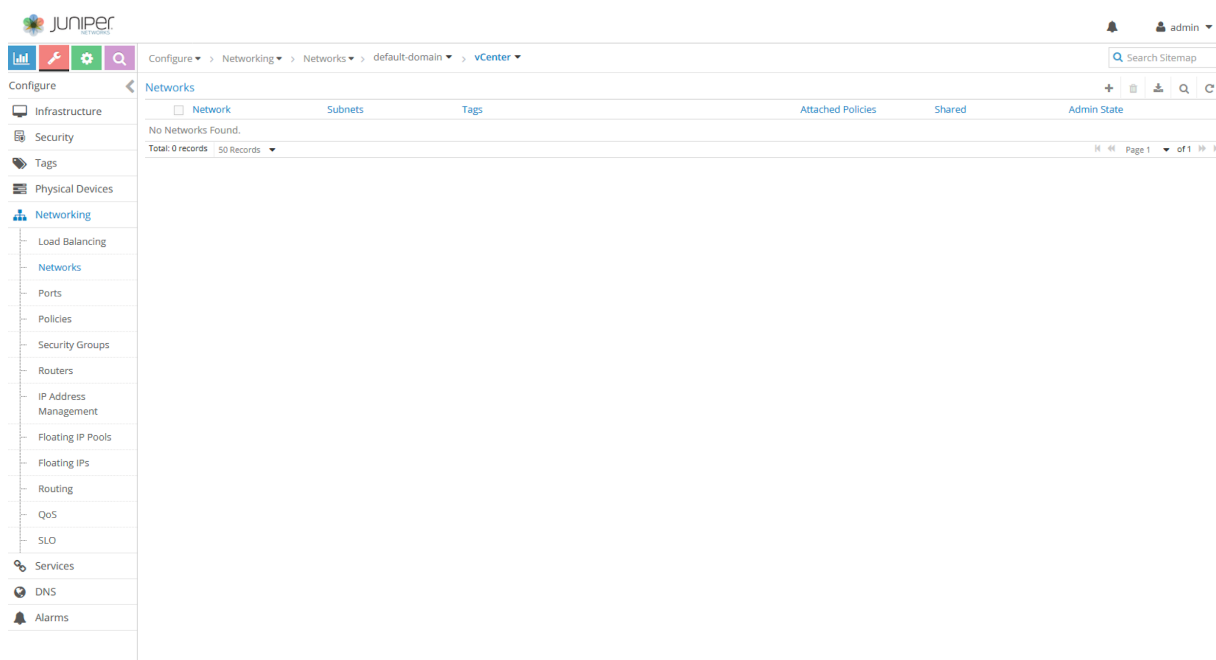
This section shows how to use the Contrail UI for the Contrail vCenter integrated system.

### 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. Ensure you are in the vCenter project.



At Networks, click the plus icon (+) to access the **Create Network** window.

The following figure shows the creation of a virtual network named Green-VN. Click the plus icon (+) next to IPAM to add a subnet that is part of this virtual network.

Click **Save** to create the virtual network.

## Delete Virtual Network – Contrail UI

You can delete a virtual network in either the Contrail 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 and click the delete icon.



**NOTE:** No VMs or templates must be using this virtual network before it is deleted.

A Confirm window is displayed. Click **Confirm** to delete the selected network.

## Creating a Virtual Machine

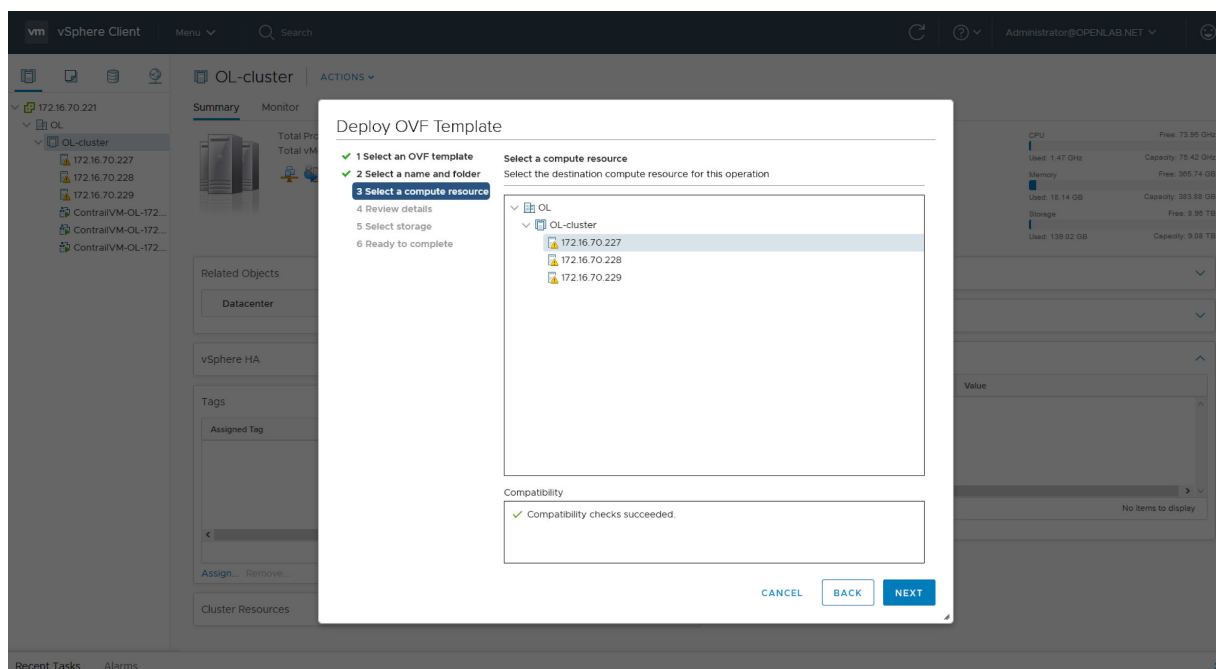
Use the vCenter client interface to create a virtual machine for your VMware vCenter Contrail integrated system. This section describes how to create a virtual machine using a virtual machine template from the vCenter client interface.

## Create a Virtual Machine – vCenter UI

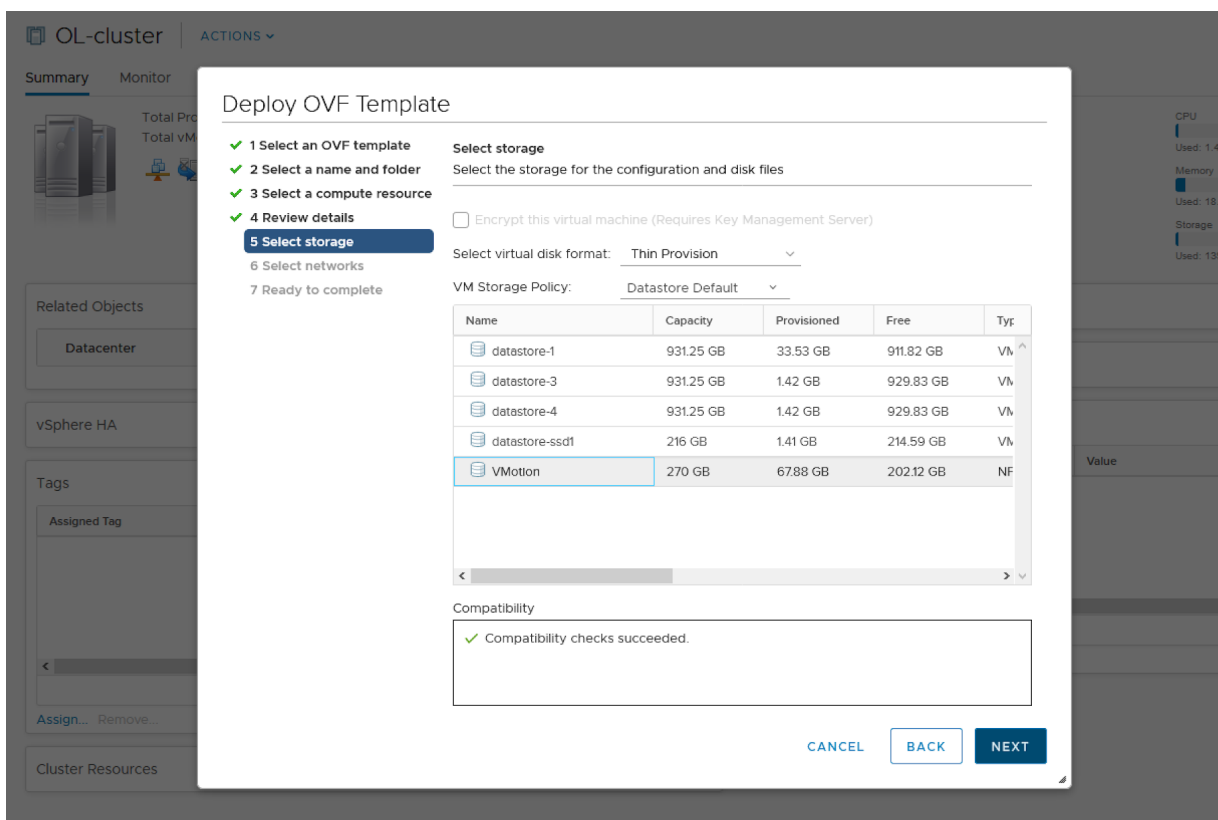
From the vCenter UI, right-click on the cluster and select **Deploy OVF Template**.

Select the virtual machine name. When you are finished, click **Next**.

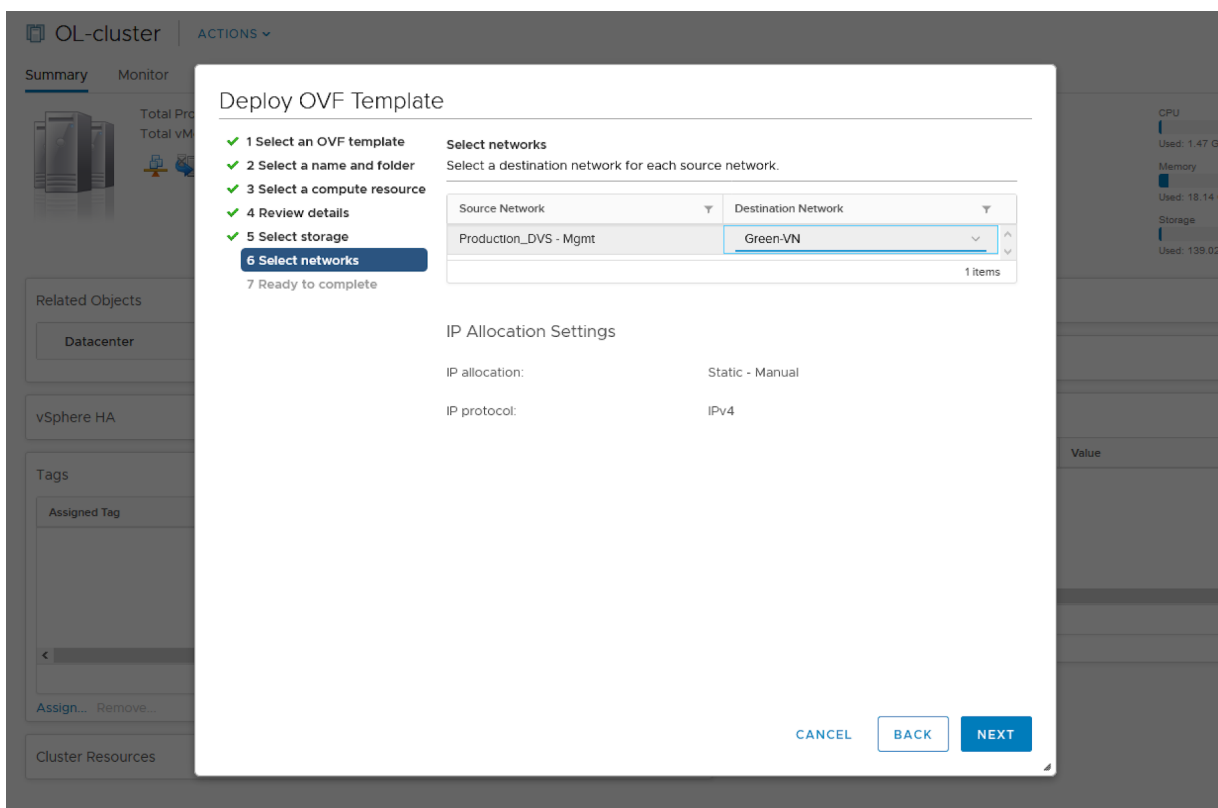
Select the compute resource on which to spawn the virtual machine. When you are finished, click **Next**.



Specify the storage location for the virtual machine. When you are finished, click **Next**.

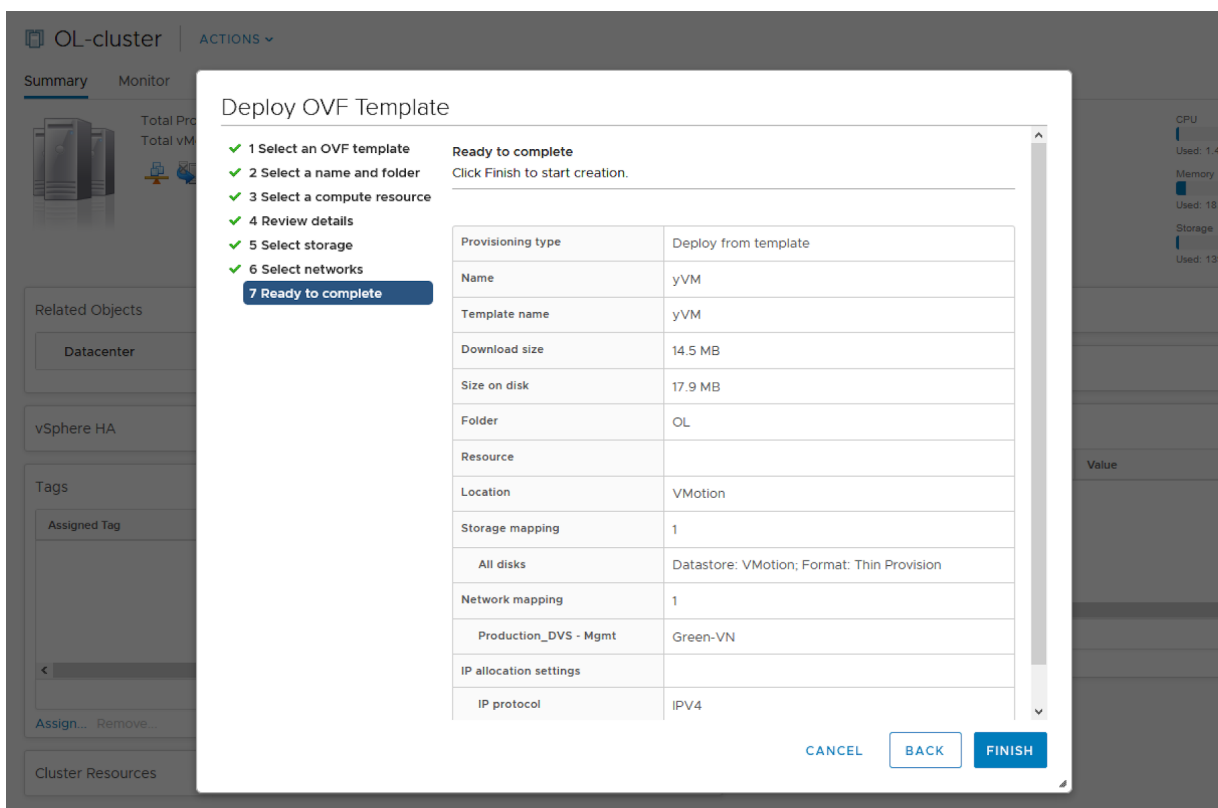


Select the **Destination Network** as Green-VN which you created in the Contrail UI.



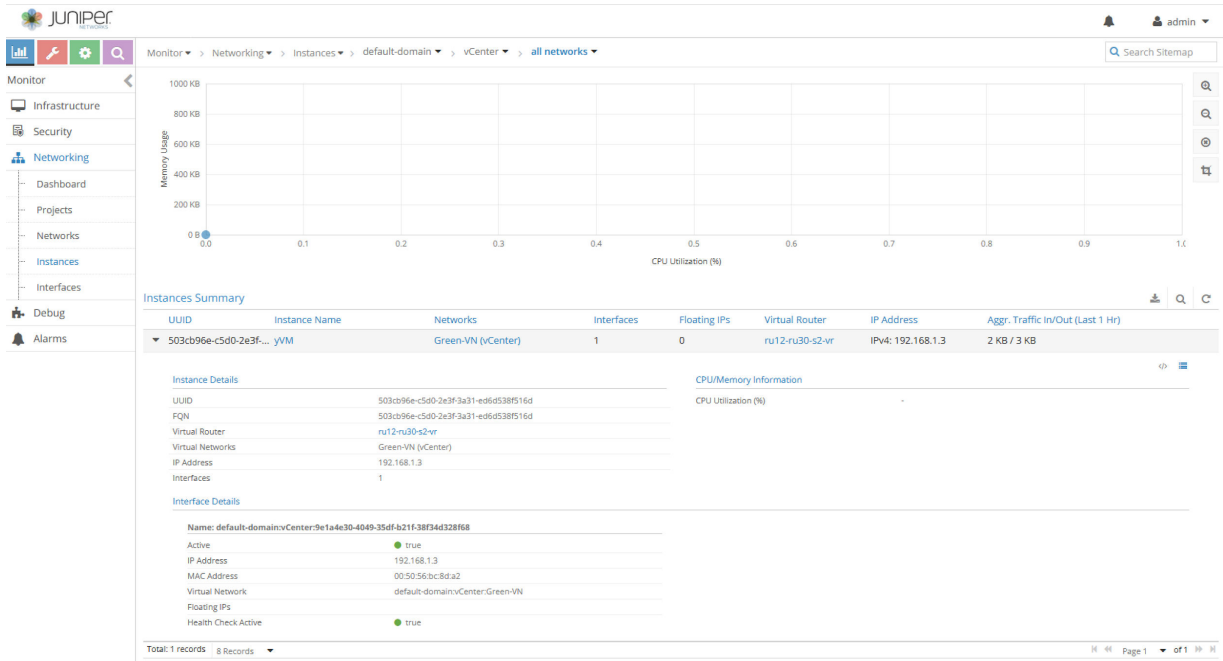
On the **Ready to complete** window, review all of the virtual machine definitions that you have selected for the template.

If all the selections are correct, click **Finish**. This spawns the virtual machine.



You return to the main vCenter UI window. Navigate to the console of the VM (yVM) you just spawned to see that it has been assigned an IP address from the Green-VN you created in the Contrail UI.

Once the virtual machine is launched, you can view the virtual machine from the Contrail UI. Select **Monitor > Networking > Instances**. All the virtual machines are displayed in the Instances Summary window.



## RELATED DOCUMENTATION

[Using the Contrail and VMware vCenter User Interfaces to Manage the Network For Contrail Releases 5.0 and 5.0.1 | 258](#)

[Installing and Provisioning VMware vCenter with Contrail](#)

[vCenter Integration for Contrail Release 5.0 | 233](#)

[vCenter Integration for Contrail Release 5.0.1 | 235](#)

[vCenter Integration for Contrail Release 5.0.2 | 237](#)

[Underlay Network Configuration for ContrailVM | 246](#)

## Integrating Contrail Release 5.0.X with VMware vRealize Orchestrator

### IN THIS SECTION

- [Components of vRealize Orchestrator | 281](#)
- [Contrail Workflows | 281](#)
- [Deploying Contrail vRO Plugin | 285](#)



A dedicated Contrail plugin is used to connect to VMware vRealize Orchestrator (vRO). Contrail Release 5.0 supported a Beta version of the plugin. Starting with Contrail Release 5.0.1, a fully supported version of the plugin is available. The plugin is used to view the Contrail controller configuration in the vRO inventory, and to modify configurations by using vRO workflows. You can also create network policies, security groups, and automate both simple and complex workflows by using vRO.

## Components of vRealize Orchestrator

vRO consists of the following components:

- **vRO Inventory**—The vRO inventory displays the Contrail plugin and the Contrail node or endpoint. All Contrail plugin objects that represent your system are displayed in the vRO Inventory. Objects are displayed in a hierarchical order and are based on the Contrail schema.

With Release 5.0, Contrail inventory objects such as projects, security groups, virtual networks, network IPAMs, network policies, ports, floating IP pools, and service templates are displayed in the vRO inventory. Relevant API objects are also displayed in the vRO inventory.

- **vRO Workflows**—The vRO workflow is a process that manipulates objects in a vRO Inventory. Each plugin has a set of predefined workflows. vRO combines workflows from different plugins to create complex processes and manages them. Multiple workflows are used to create blueprints in vRA.



**NOTE:** VMware vCenter plugin workflows are represented as simple workflows in vRO plugin.

## Contrail Workflows

You must connect to the Contrail controller or an endpoint before you create Contrail workflows. You must use **Create Contrail controller connection** to connect to an endpoint. You must use **Delete Contrail controller connection** to terminate a connection with an endpoint. Once you connect to the Control controller, the vRO plugin accesses the Contrail inventory objects and then updates the vRO inventory with the Contrail inventory objects.

The following workflows are defined for simple networking operations in Contrail Release 5.0:

- Network
  - Create network
  - Delete network



**NOTE:** You must use the **Create network** workflow to create a network on the Contrail controller.

- Network Policy
  - Create network policy
  - Add rule to network policy
  - Remove network policy rule
  - Delete network policy
- Security policy
  - Create security group
  - Add rule to security group
  - Edit security group
  - Remove security group rule
  - Delete security group
- Service Instance
  - Add port tuple to service instance
  - Create service instance
  - Delete service instance
  - Remove port tuple from service instance
- Network IPAM
- Port
- Project
- Service Template
- Virtual Network
- Floating IP
  - Create floating IP

- Delete floating IP
- Floating IP pools
  - Create floating IP pool
  - Delete floating IP pool
  - Edit floating IP pool

Starting with Contrail Release 5.0.1, the following workflows are also defined:

- Tag
  - Create global tag
  - Create tag in project
  - Delete tag
- Tag Type
  - Create tag type
  - Delete tag type
- Network Policy
  - Edit network policy rule
- Security policy
  - Edit security group rule
- Service Health Check
  - Create service health check
  - Add service instance to service health check
  - Remove service instance from service health check
  - Edit service health check
  - Delete service health check
- Project
  - Add application policy set to project
  - Remove application policy set to project

- Add tag to project
- Remove tag from project
- Virtual Network
  - Add tag to virtual network
  - Remove tag from virtual network
- Virtual Machine Interface (VMI) - Port
  - Add tag to port
  - Remove tag from port
- Service Group
  - Create service group in policy management
  - Create service group in project
  - Add service to service group
  - Edit service of service group
  - Remove service from service group
  - Delete service group
- Address Group
  - Create global address group
  - Create address group in project
  - Add subnet to address group
  - Remove subnet from address group
  - Delete address group
  - Add label to address group
  - Remove label from address group
- Application Policy Set
  - Create global application policy set
  - Create application policy set in project

- Add firewall policy to application policy set
- Remove firewall policy from application policy set
- Add tag to application policy set
- Remove tag from application policy set
- Delete application policy set
- Firewall Policy
  - Create global firewall policy
  - Create firewall policy in project
  - Add firewall rule to firewall policy
  - Remove firewall rule from firewall policy
  - Delete firewall policy
- Firewall Rule
  - Create global firewall rule
  - Create firewall rule in project
  - Edit firewall rule
  - Delete firewall rule

## Deploying Contrail vRO Plugin

You can deploy the Contrail plugin in any Java Virtual Machine (JVM) compatible environment and load it on an active vRO instance.

For more information, see ["Installing and Provisioning Contrail VMware vRealize Orchestrator Plugin" on page 286](#).

## Installing and Provisioning Contrail VMware vRealize Orchestrator Plugin

### IN THIS SECTION

- [Accessing vRO Control Center | 286](#)
- [Installing vRO Plugin | 289](#)
- [Accessing vRO Desktop Client | 291](#)
- [Connecting to vRO using the Desktop Client | 291](#)
- [Connecting to Contrail Controller | 292](#)

A dedicated Contrail plugin is used to connect to VMware vRealize Orchestrator (vRO). Contrail Release 5.0 supported a Beta version of the plugin. Starting with Contrail Release 5.0.1, a fully supported version of the plugin is available.

You must install the Contrail VMware vRealize Orchestrator (vRO) plugin to connect to the vRO server.

Before you begin installation, ensure the following:

- You have administrator-level access to the Control Center of a deployed vRO appliance.
- You know the host name (*{vRO}*) of the deployed vRO Appliance.
- You have the login credentials of the vCenter SSO service.
- You have downloaded the vRO plugin package file to your local system.

You can download the plugin from <https://www.juniper.net/support/downloads/?p=contrail>.

The following topics describe how to install and provision the Contrail vRO plugin.

### Accessing vRO Control Center

Follow the steps given below to access and log in to vRO Control Center:

1. To access vRO Control Center through a Web browser, navigate to the **`https://{vRO}:8283/vco-controlcenter`** URL.

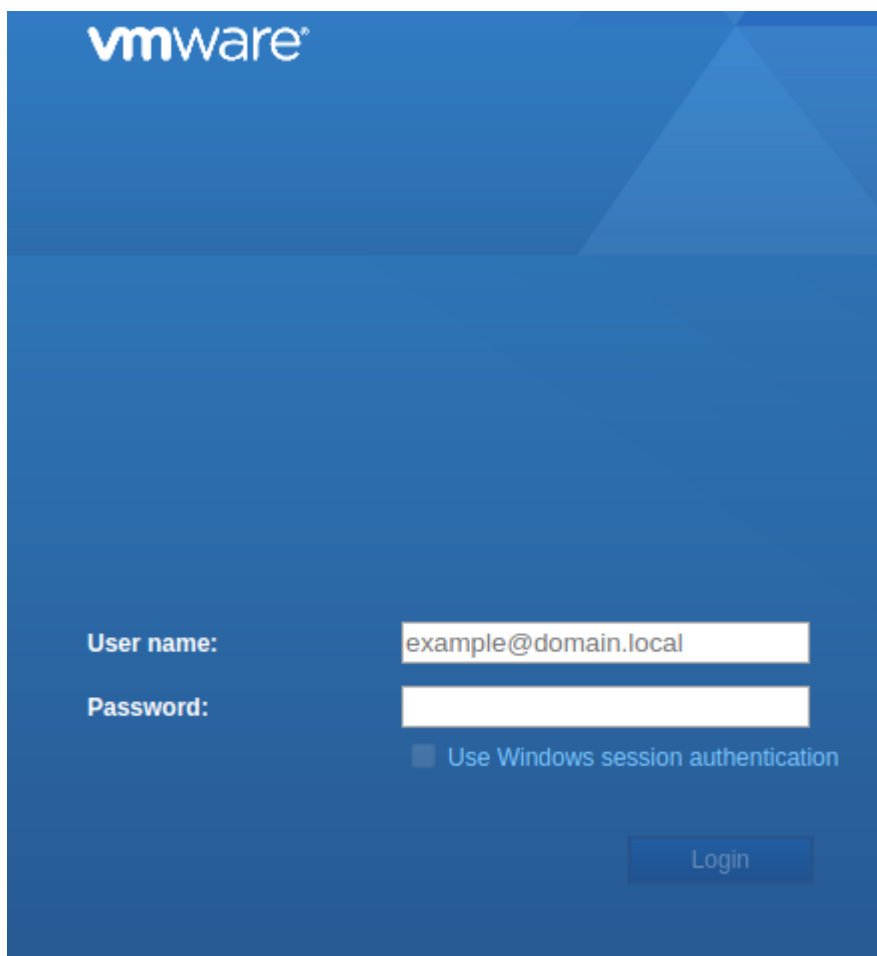


**NOTE:** Replace *{vRO}* given in the URL with the *host name* of the deployed vRO Appliance.

The *host name* is the IP address or the FQDN of the vRO node.

The **vCenter SSO** service page is displayed.

Figure 32: vCenter SSO service page

The image shows the vCenter SSO service page login interface. It has a blue background with the VMware logo at the top left. Below the logo, there are two input fields: 'User name:' and 'Password:'. The 'User name:' field contains the text 'example@domain.local'. Below the 'Password:' field, there is a checkbox labeled 'Use Windows session authentication'. At the bottom right, there is a 'Login' button.

vmware®

User name: example@domain.local

Password:

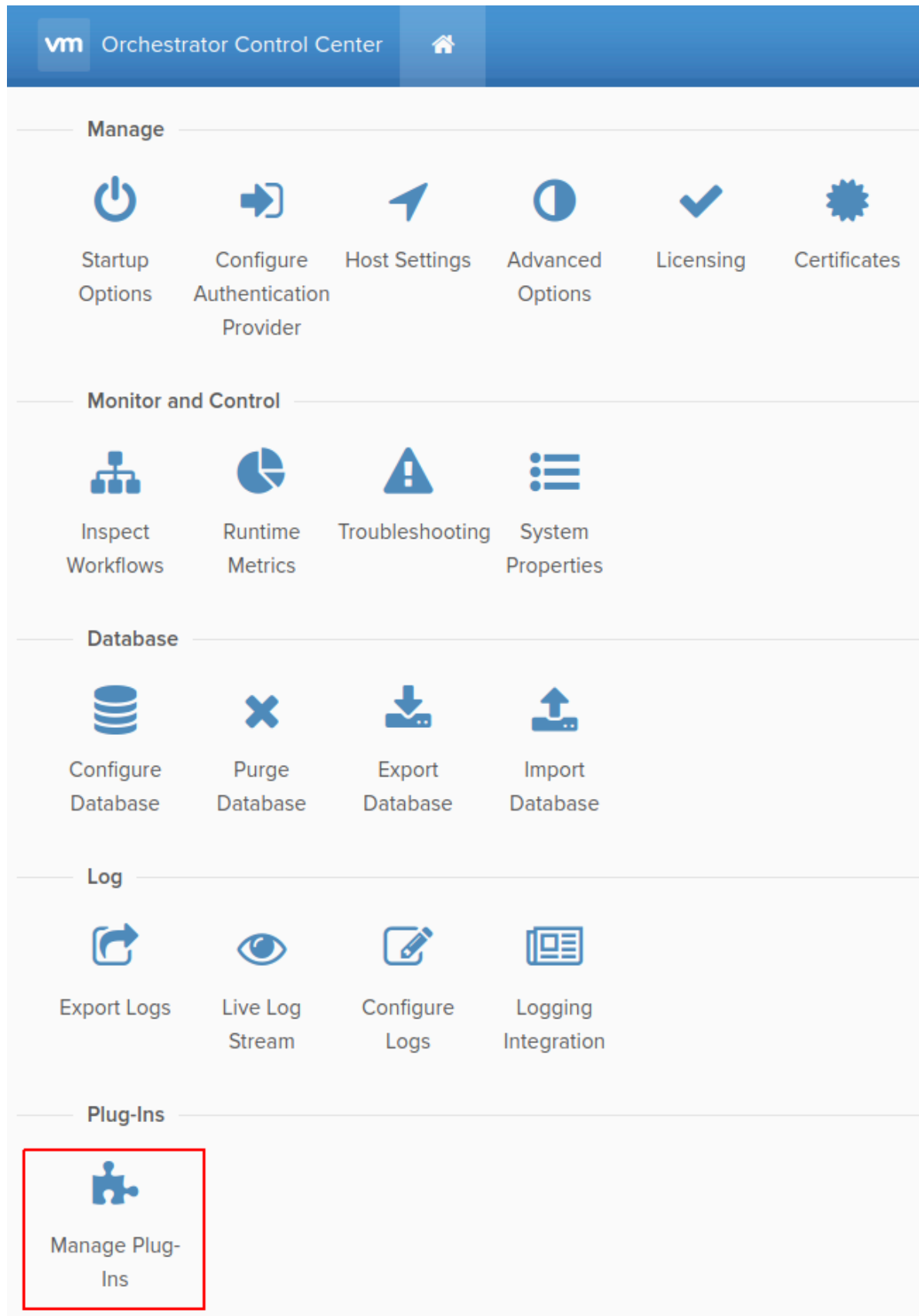
☐ Use Windows session authentication

Login

2. On the vCenter SSO service page, enter the **User name** and **Password** in the respective fields and click **Login**. See [Figure 32 on page 287](#).

The **Orchestrator Control Center** home page is displayed.

Figure 33: Orchestrator Control Center





## Installing vRO Plugin

Perform the following steps to install the vRO plugin:

1. Upload vRO plugin package.

To upload vRO plugin package:

- From the Orchestrator Control Center home page, click **Manage Plug-Ins** under the **Plug-Ins** section.

The **Manage Plug-Ins** page is displayed.

Figure 34: Manage Plug-Ins page

**Manage Plug-Ins**

Install a new plug-in or manage already installed plug-ins. The preferred plug-in installation file format is .VMOAPP, but plug-ins can also be installed as .DAR files. When 'DEFAULT' logging level is selected for a specific plug-in the log level is inherited from the log level set in [Configure Logs](#) page.

**Install plug-in**

Plugin file (\*.dar or \*.vmoapp) **BROWSE** **INSTALL**

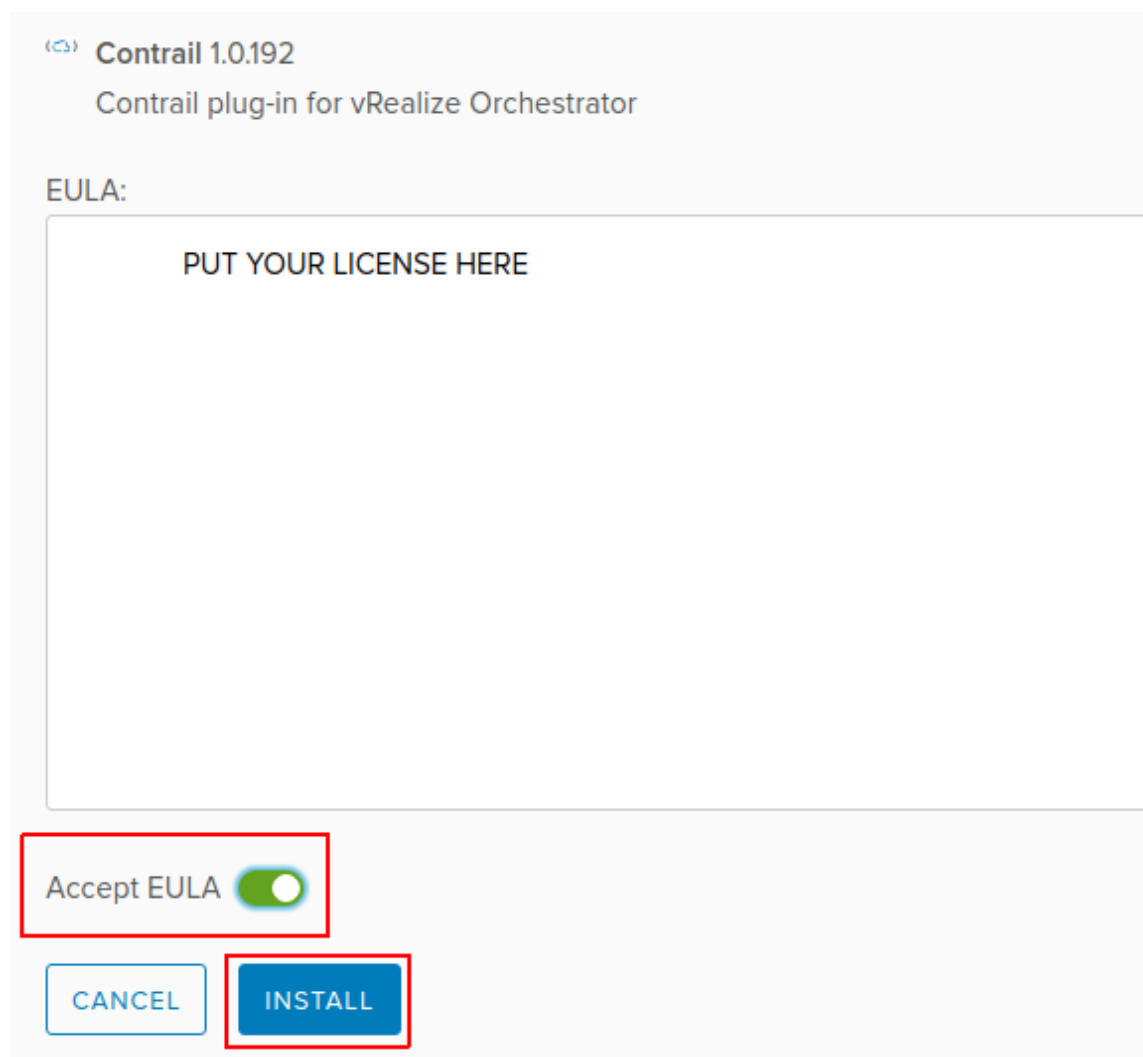
**NOTE:** You can install a new plugin or manage an already installed plugin from the Manage Plug-Ins page.

**NOTE:** \*.vmoapp or \*.dar file format can be used. Also, the version in this example may be different from the version you have downloaded.

- Click **Browse** in the **Install plug-in** pane and select the downloaded vRO plugin package file on your local system.
- After you select vRO plugin package file, click **Install** to upload the vRO plugin package to the vRO server.

The **EULA** page is displayed.

Figure 35: EULA page



## 2. Install vRO plugin.

After you upload the vRO plugin package, select **Accept EULA** on the **EULA** page and then click **Install**.



**NOTE:** If you use \*.vmoapp file format, you are directed to the Accept EULA page before you proceed with the installation.

If you use \*.dar file format, you can directly proceed with installation.

The vRO plugin is installed.

## Accessing vRO Desktop Client

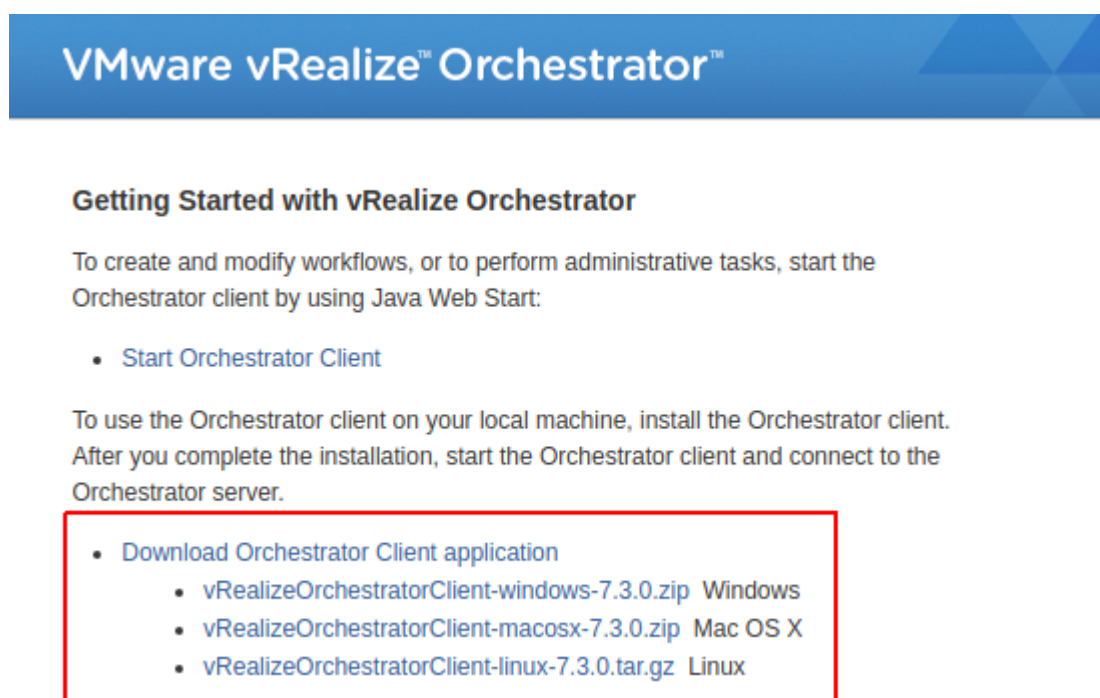
After you install the vRO plugin, you access the vRO server using the vRO desktop client.

To download and install the vRO desktop client application, click <https://{vRO}:8281/vco/>.



**NOTE:** Replace *{vRO}* given in the URL with the *host name* of the deployed vRO Appliance.

Figure 36: Getting Started with vRealize Orchestrator



You can download vRO desktop client applications for Windows, Mac OS X, and Linux operating systems.

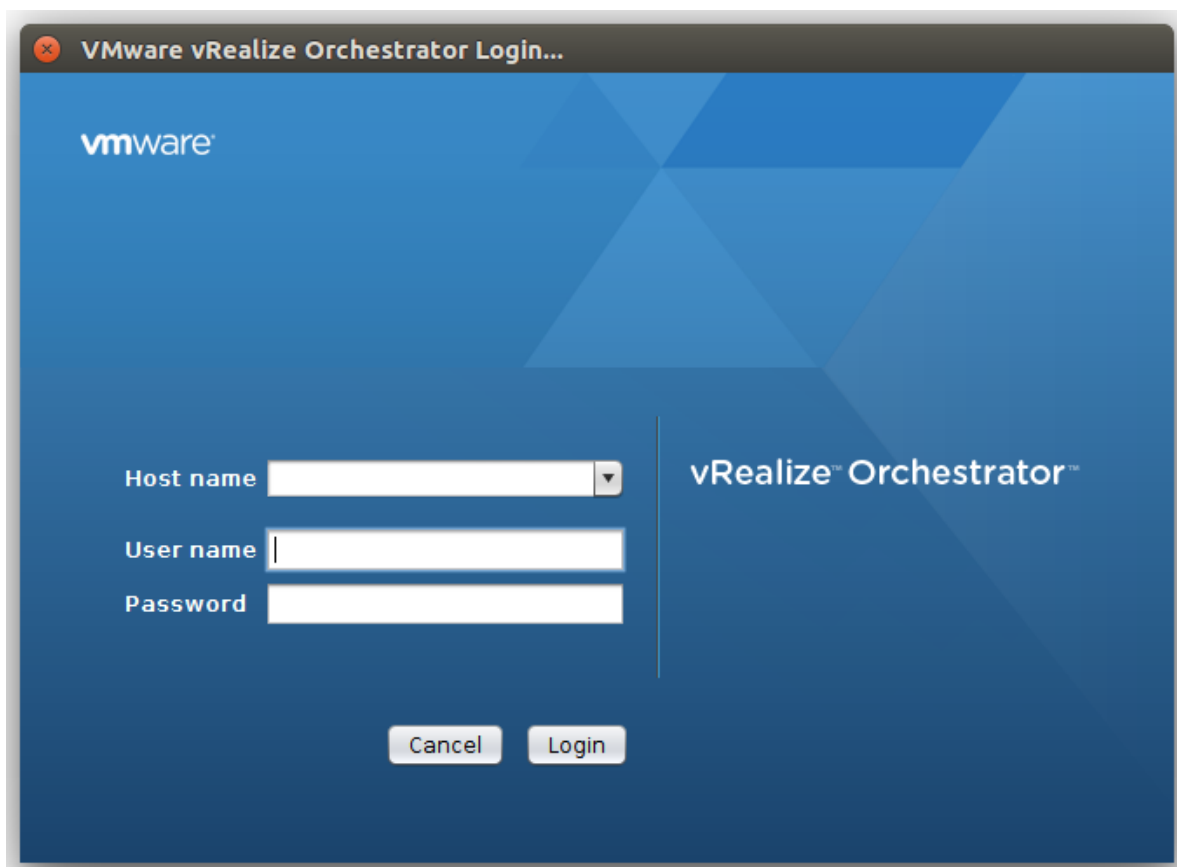
## Connecting to vRO using the Desktop Client

You connect to the vRO server by using the vRO desktop client.

1. Start the vRO desktop client.

The **VMware vRealize Orchestrator Login** page is displayed.

Figure 37: VMware vRealize Orchestrator Login page



2. In the VMware vRealize Orchestrator Login page, enter **Host name**, **User name**, and **Password**.



**NOTE:** The **Host name** also includes the port number and must be in the {vRO}:8281 format.

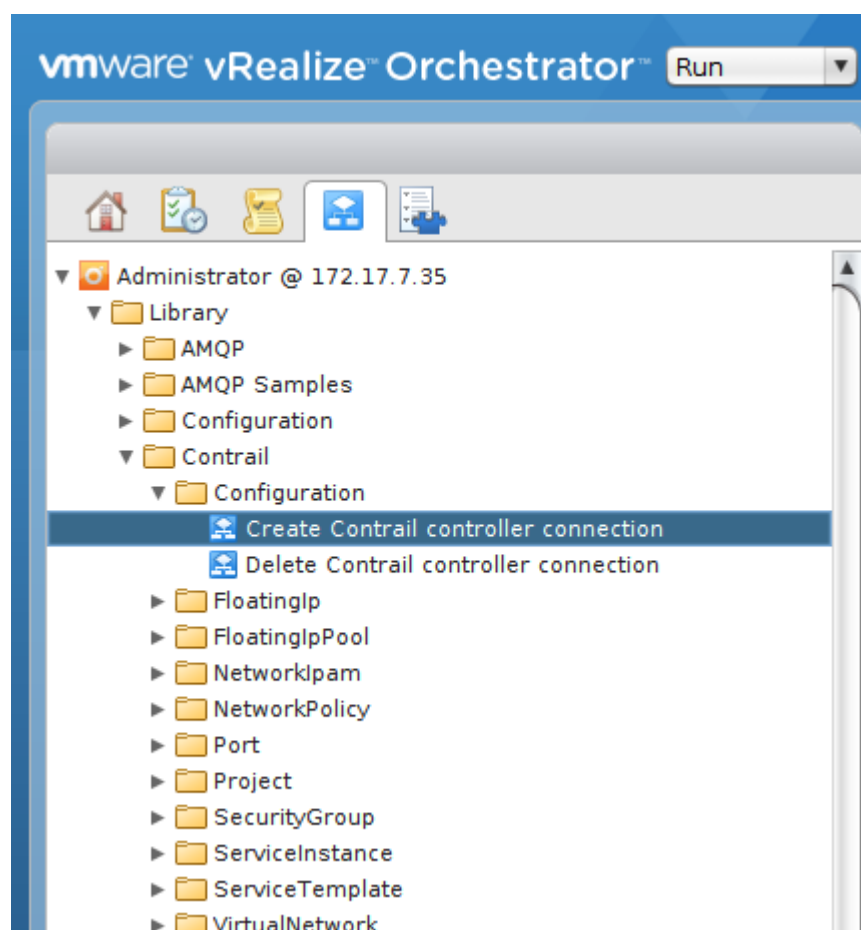
3. Click **Login** to connect to the vRO server. See [Figure 37 on page 292](#).

## Connecting to Contrail Controller

To connect Contrail vRO to the Contrail Controller:

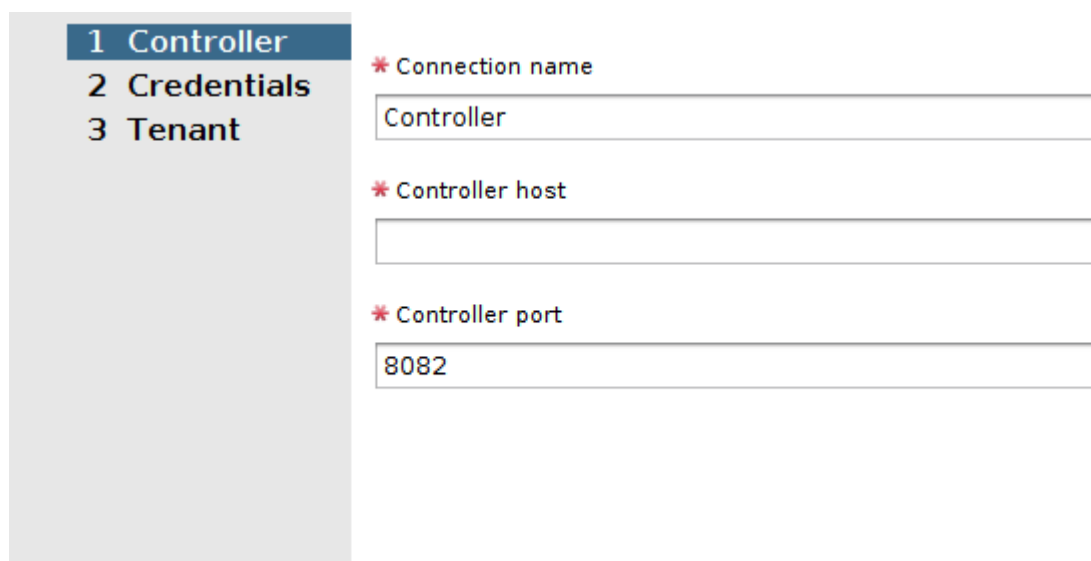
1. Navigate to the **Contrail > Configuration** folder in the workflow library. See [Figure 38 on page 293](#).
2. Select **Create Contrail controller connection**.

Figure 38: Workflow Library



3. Click the **Controller** tab and enter the following information:
  - **Connection name**—a unique name to identify the connection
  - **Controller host**—host name of the Contrail Connector
  - **Controller port**—port used to access the Contrail Controller

Figure 39: Controller Tab



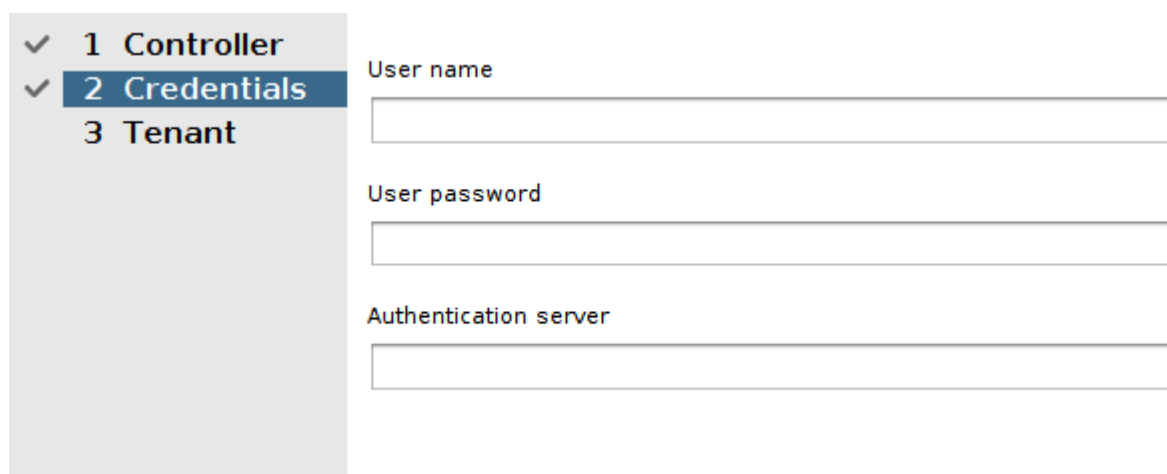
The screenshot shows a configuration interface with a sidebar on the left containing three tabs: **1 Controller** (selected), **2 Credentials**, and **3 Tenant**. The main area displays three required fields, each marked with a red asterisk:

- \* Connection name**: A text input field containing the value "Controller".
- \* Controller host**: An empty text input field.
- \* Controller port**: A text input field containing the value "8082".

4. Click the **Credentials** tab and enter the following credentials to manage the Contrail Controller:

- **User name**—user name to access the Contrail Controller
- **User password**—password to access the Contrail Controller
- **Authentication server**—URL of the authentication server

Figure 40: Credentials Tab



The screenshot shows the same configuration interface, but the **2 Credentials** tab is now selected in the sidebar. The main area displays three input fields:

- User name**: An empty text input field.
- User password**: An empty text input field.
- Authentication server**: An empty text input field.

5. Click the **Tenant** tab to define tenant information.

In the **Tenant** field, enter the name of the Contrail tenant.

Figure 41: Tenant Tab



6. Click **Submit** to establish connection.

Once you connect Contrail vRO to the Contrail Controller, you use Contrail workflows to make configuration changes to Contrail.

RELATED DOCUMENTATION

| [Integrating Contrail Release 5.0.X with VMware vRealize Orchestrator](#) | 280

# Using Contrail with Red Hat

## IN THIS CHAPTER

- [Deploying Contrail with Red Hat OpenStack Platform Director 13 | 296](#)
- [Provisioning Red Hat OpenShift Container Platform Clusters Using Ansible Deployer | 351](#)

## Deploying Contrail with Red Hat OpenStack Platform Director 13

### IN THIS SECTION

- [Overview | 296](#)
- [OSPd Features | 297](#)
- [Composable Roles | 297](#)
- [Preparing the Environment for Deployment | 298](#)
- [Creating Infrastructure | 301](#)
- [undercloud Configuration | 303](#)
- [undercloud Post Configuration | 308](#)
- [overcloud Configuration | 309](#)

This document explains how to integrate a Contrail 5.0.1 installation with Red Hat OpenStack Platform Director 13.

### Overview

Red Hat 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.



TripleO can be used to deploy a RDO based OpenStack environment integrated with Tungsten Fabric. Red Hat OpenStack Platform Director (RHOSPd) can be used to deploy a RHOSP based OpenStack environment integrated with Contrail.

## OSPd Features

OSPd uses the concepts of undercloud and overcloud. OSPd 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.

OSPd leverages existing core components of OpenStack including Nova, Ironi, Neutron, Heat, Glance, and Ceilometer to deploy OpenStack on bare metal hardware.

- Nova and Ironi 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 OSPd architecture, see [OSPd documentation](#)

## Composable Roles

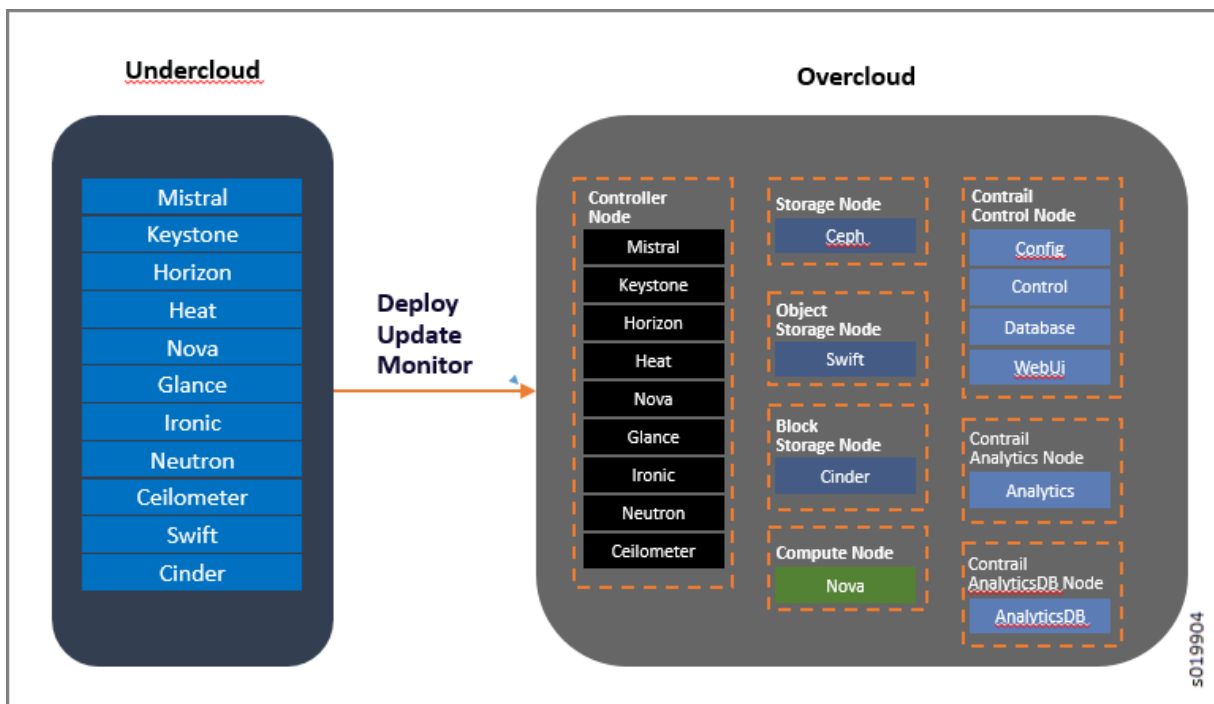
OSPd enables composable roles. Each role is a group of services that are defined in Heat templates. Composable roles gives the operator the flexibility to add and modify roles as needed.

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 42 on page 298 shows the relationship and components of an undercloud and overcloud architecture for Contrail.

Figure 42: undercloud and overcloud with Roles



## 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](#).

## Preparing for the Contrail Roles

Ensure the following requirements are met for the Contrail nodes per role.

- Non-high availability: A minimum of 4 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:

- [Planning Networks](#)
- [Networking Requirements](#)

At a high level, every overcloud node must support IPMI.

## 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 vSwitch 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

OSPd 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

## Supported Combinations

The following combinations of Operating System/OpenStack/Deployer/Contrail are supported:

**Table 4: Compatibility Matrix**

Operating System	OpenStack	Deployer	Contrail
RHEL 7.5	OSP13	OSPd13	Contrail 5.0.1

**Table 4: Compatibility Matrix *(Continued)***

Operating System	OpenStack	Deployer	Contrail
CentOS 7.5	RDO queens/stable	tripleo queens/stable	Tungsten Fabric latest

## Creating Infrastructure

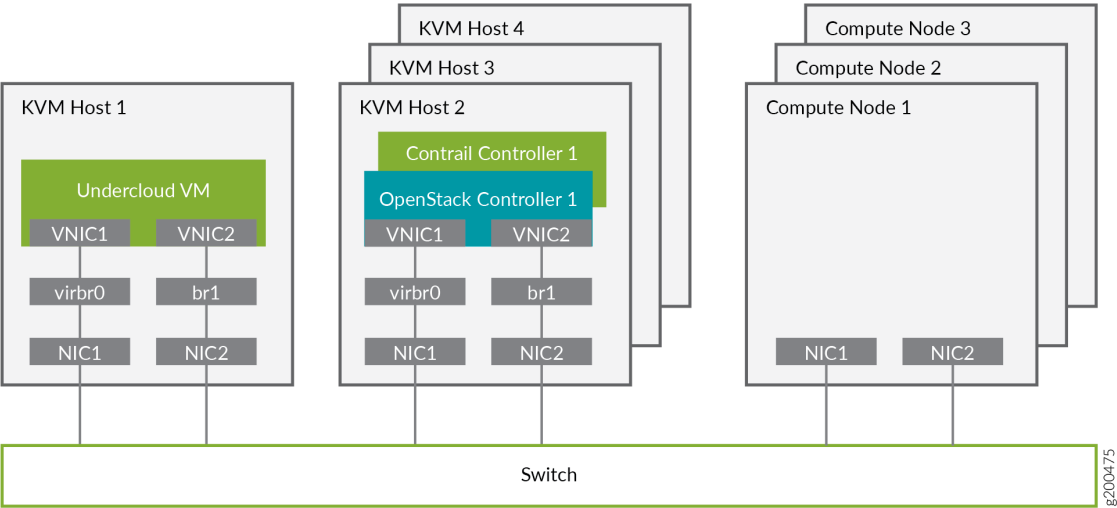
There are many different ways on how to create the infrastructure providing the control plane elements. The following example illustrates all control plane functions as Virtual Machines hosted on KVM hosts.

**Table 5: Control Plane Functions**

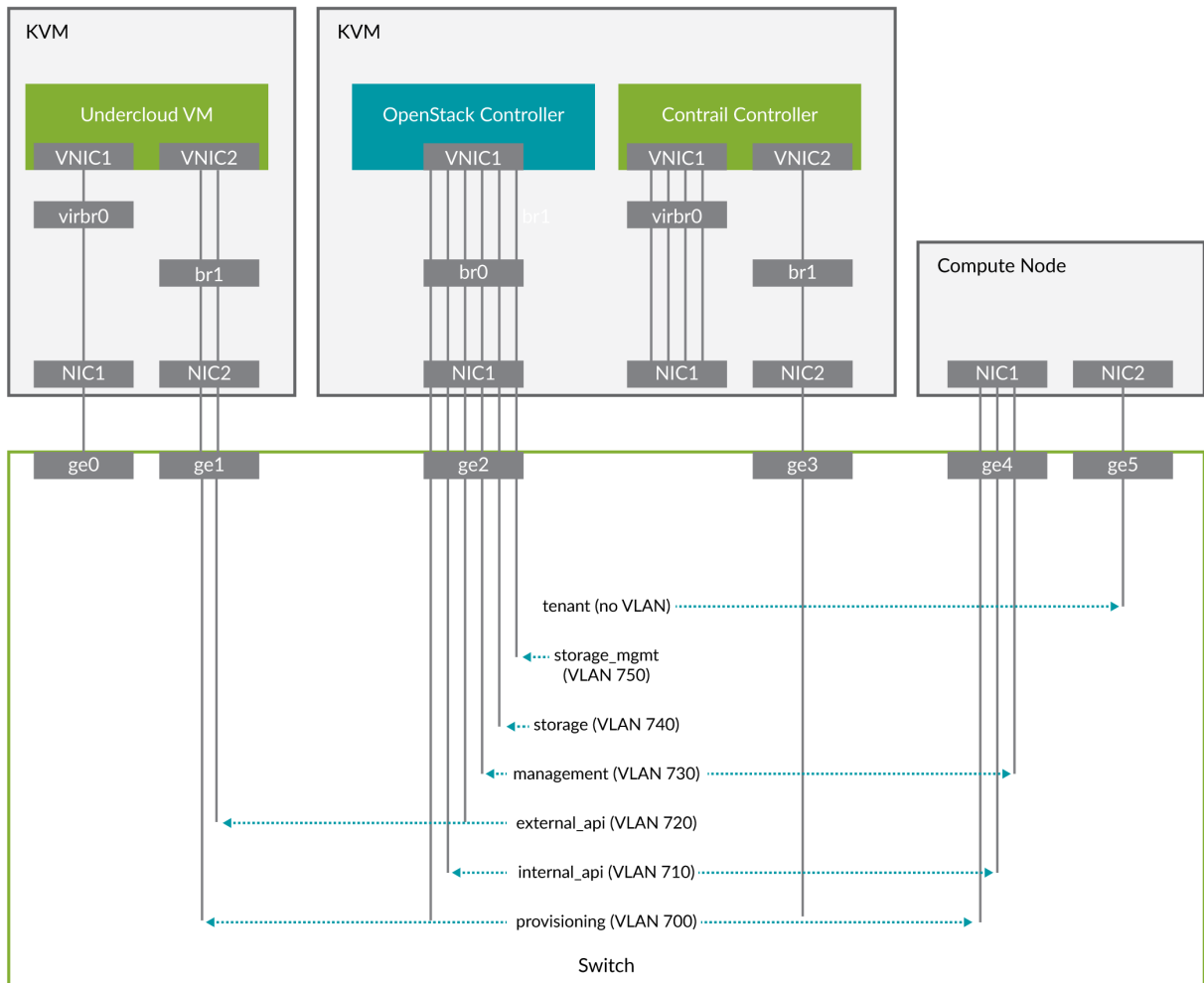
KVM Host	Virtual Machines
KVM1	undercloud
KVM2	OpenStack Controller 1, Contrail Controller 1
KVM3	OpenStack Controller 2, Contrail Controller 2
KVM4	OpenStack Controller 2, Contrail Controller 2

Sample Topology

Layer 1: Physical Layer



## Layer 2: Logical Layer



g200476

## undercloud Configuration

### Physical Switch

Use the following information to create ports and Trunked VLANs

**Table 6: Physical Switch**

Port	Trunked VLAN	Native VLAN
ge0	-	-
ge1	700, 720	-
ge2	700, 710, 720, 730, 740, 750	-
ge3	-	-
ge4	710, 730	700
ge5	-	-

## undercloud and overcloud KVM Host Configuration

undercloud and overcloud KVM hosts will need virtual switches and virtual machine definitions configured. You can deploy any KVM host operating system version which supports KVM and OVS. The following example shows a RHEL/CentOS based system. If you are using RHEL, the system must be subscribed.

- *Install Basic Packages*

```
yum install -y libguestfs \ libguestfs-tools \ openvswitch \ virt-install \ kvm libvirt \ libvirt-python \
python-virtualbmc \ python-virtinst
```

- *Start libvirtd and ovs*

```
systemctl start libvirtd systemctl start openvswitch
```

- *Configure vSwitch*



**Table 7: Configure vSwitch**

Bridge	Trunked VLAN	Native VLAN
br0	710, 720, 730 740, 750	700
br1	-	-

*Create bridges*

```

ovs-vsctl add-br br0 ovs-vsctl add-br br1 ovs-vsctl add-port br0 NIC1 ovs-vsctl add-port br1 NIC2 cat << EOF >
br0.xml <network> <name>br0</name> <forward mode='bridge'/> <bridge name='br0'/> <virtualport
type='openvswitch'/> <portgroup name='overcloud'/> <vlan trunk='yes'> <tag id='700' nativeMode='untagged'/>
<tag id='710'/> <tag id='720'/> <tag id='730'/> <tag id='740'/> <tag id='750'/> </vlan> </portgroup> </network>
EOF cat << EOF > br1.xml <network> <name>br1</name> <forward mode='bridge'/> <bridge name='br1'/> <virtualport
type='openvswitch'/> </network> EOF virsh net-define br0.xml virsh net-start br0 virsh net-autostart br0 virsh
net-define br1.xml virsh net-start br1 virsh net-autostart br1

```

- *Create overcloud VM Definitions on the overcloud KVM Hosts (KVM2-KVM4)*



**NOTE:** overcloud VM definition is required to create on each overcloud KVM host.



**NOTE:** Use the following formula to create the number of roles per overcloud KVM host:

ROLES=compute:2,contrail-controller:1,control:1

The following example defines:

2x compute nodes 1x cotrail controller node 1x openstack controller node

```

num=0 ipmi_user=<user> ipmi_password=<password> libvirt_path=/var/lib/libvirt/images port_group=overcloud
prov_switch=br0 /bin/rm ironic_list IFS=',' read -ra role_list <<< "${ROLES}" for role in ${role_list[@]}; do
role_name=`echo $role|cut -d ":" -f 1` role_count=`echo $role|cut -d ":" -f 2` for count in `seq 1 $
{role_count}`; do echo $role_name $count qemu-img create -f qcow2 ${libvirt_path}/${role_name}_${count}.qcow2
99G virsh define /dev/stdin <<EOF $(virt-install --name ${role_name}_${count} \ --disk ${libvirt_path}/${
role_name}_${count}.qcow2 \ --vcpus=4 \ --ram=16348 \ --network network=br0,model=virtio,portgroup=$
{port_group} \ --network network=br1,model=virtio \ --virt-type kvm \ --cpu host \ --import \ --os-variant
rhel7 \ --serial pty \ --console pty,target_type=virtio \ --graphics vnc \ --print-xml) EOF vbmc add $
{role_name}_${count} --port 1623${num} --username ${ipmi_user} --password ${ipmi_password} / vbmc start $
{role_name}_${count} prov_mac=`virsh domiflist ${role_name}_${count}|grep ${prov_switch}|awk '{print $5}'`

```

```
vm_name=${role_name}-${count}-${hostname} -s` kvm_ip=`ip route get 1 |grep src |awk '{print $7}'` echo $
{prov_mac} ${vm_name} ${kvm_ip} ${role_name} 1623${num}>> ironic_list num=$(expr $num + 1) done done
```



**CAUTION:** One *ironic\_list* file per KVM host will be created. You need to combine all the *ironic\_list* files from each KVM host on the undercloud.

The following output shows combined list from all the three Overcloud KVM hosts:

```
52:54:00:e7:ca:9a compute-1-5b3s31 10.87.64.32 compute 16230 52:54:00:30:6c:3f compute-2-5b3s31
10.87.64.32 compute 16231 52:54:00:9a:0c:d5 contrail-controller-1-5b3s31 10.87.64.32 contrail-
controller 16232 52:54:00:cc:93:d4 control-1-5b3s31 10.87.64.32 control 16233 52:54:00:28:10:d4
compute-1-5b3s30 10.87.64.31 compute 16230 52:54:00:7f:36:e7 compute-2-5b3s30 10.87.64.31
compute 16231 52:54:00:32:e5:3e contrail-controller-1-5b3s30 10.87.64.31 contrail-controller
16232 52:54:00:d4:31:aa control-1-5b3s30 10.87.64.31 control 16233 52:54:00:d1:d2:ab
compute-1-5b3s32 10.87.64.33 compute 16230 52:54:00:ad:a7:cc compute-2-5b3s32 10.87.64.33
compute 16231 52:54:00:55:56:50 contrail-controller-1-5b3s32 10.87.64.33 contrail-controller
16232 52:54:00:91:51:35 control-1-5b3s32 10.87.64.33 control 16233
```

- *Create undercloud VM Definitions on the undercloud KVM host (KVM1)*



**NOTE:** undercloud VM definitions is required to create only on undercloud KVM.

1. Create images directory

```
mkdir ~/images cd images
```

2. Retrieve the image



**NOTE:** The image must be retrieved based on the operating system:

- CentOS

```
curl https://cloud.centos.org/centos/7/images/CentOS-7-x86_64-GenericCloud-1802.qcow2.xz
\ -o CentOS-7-x86_64-GenericCloud-1802.qcow2.xz zx -d images/CentOS-7-x86_64-
GenericCloud-1802.qcow2.xz cloud_image=~/.images/CentOS-7-x86_64-
GenericCloud-1804_02.qcow2
```

- RHEL

```
Download rhel-server-7.5-update-1-x86_64-kvm.qcow2 from Red Hat portal to ~/.images
cloud_image=~/.images/rhel-server-7.5-update-1-x86_64-kvm.qcow2
```

### 3. Customize the undercloud image

```
undercloud_name=queensa undercloud_suffix=local root_password=<password>
stack_password=<password> export LIBGUESTFS_BACKEND=direct qemu-img create -f
qcow2 /var/lib/libvirt/images/${undercloud_name}.qcow2 100G virt-resize --expand /dev/sda1 $
{cloud_image} /var/lib/libvirt/images/${undercloud_name}.qcow2 virt-customize -a /var/lib/
libvirt/images/${undercloud_name}.qcow2 \ --run-command 'xfs_growfs /' \ --root-password
password:${root_password} \ --hostname ${undercloud_name}.${undercloud_suffix} \ --run-
command 'useradd stack' \ --password stack:password:${stack_password} \ --run-command 'echo
"stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack' \ --chmod 0440:/etc/
sudoers.d/stack \ --run-command 'sed -i "s/PasswordAuthentication no/PasswordAuthentication
yes/g" /etc/ssh/sshd_config' \ --run-command 'systemctl enable sshd' \ --run-command 'yum
remove -y cloud-init' \ --selinux-relabel
```

### 4. Define the undercloud virsh template

```
vcpus=8 vram=32000 virt-install --name ${undercloud_name} \ --disk /var/lib/libvirt/images/$
{undercloud_name}.qcow2 \ --vcpus=${vcpus} \ --ram=${vram} \ --network
network=default,model=virtio \ --network network=br0,model=virtio,portgroup=overcloud \ --
virt-type kvm \ --import \ --os-variant rhel7 \ --graphics vnc \ --serial pty \ --
noautoconsole \ --console pty,target_type=virtio
```

### 5. Start the undercloud VM

```
virsh start ${undercloud_name}
```

### 6. Retrieve the undercloud IP

It might take several seconds before the IP is available.

```
undercloud_ip=`virsh domifaddr ${undercloud_name} |grep ipv4 |awk '{print $4}' |awk -F"/"
'{print $1}'` ssh-copy-id ${undercloud_ip}
```

## undercloud Configuration

### 1. Login to the undercloud VM from the undercloud KVM host

```
ssh ${undercloud_ip}
```

### 2. Configure Hostname

```
undercloud_name=`hostname -s` undercloud_suffix=`hostname -d` hostnamectl set-hostname ${undercloud_name}.${
undercloud_suffix} hostnamectl set-hostname --transient ${undercloud_name}.${undercloud_suffix}
```



**NOTE:** Make sure to set undercloud IP in the host file located at **etc\hosts**.

The commands will be as follows assuming the mgmt NIC is eth0:

```
undercloud_ip=`ip addr sh dev eth0 |grep "inet " |awk '{print $2}' |awk -F"/" '{print $1}'` echo
${undercloud_ip} ${undercloud_name}.${undercloud_suffix} ${undercloud_name} >> /etc/hosts`
```

### 3. Setup Repositories



**NOTE:** The repository must be setup based on the operating system:

- CentOS

```
tripleo_repos=`python -c 'import requests;r = requests.get("https://trunk.rdoproject.org/centos7-queens/
current"); print r.text ' |grep python2-tripleo-repos|awk -F"href=\"" '{print $2}'|awk -F"\"" '{print $1}'`
yum install -y https://trunk.rdoproject.org/centos7-queens/current/${tripleo_repos} tripleo-repos -b queens
current
```

- RHEL

```
#Register with Satellite (can be done with CDN as well) satellite_fqdn=device.example.net act_key=xxx
org=example yum localinstall -y http://${satellite_fqdn}/pub/katello-ca-consumer-latest.noarch.rpm
subscription-manager register --activationkey=${act_key} --org=${org}
```

### 4. Install Tripleo Client

```
yum install -y python-tripleoclient tmux
```

### 5. Copy *undercloud.conf*

```
su - stack cp /usr/share/instack-undercloud/undercloud.conf.sample ~/undercloud.conf
```

## undercloud Installation

Run the following command to install the undercloud:

```
openstack undercloud install source stackrc
```

## undercloud Post Configuration

Complete the following configurations post undercloud installation:

- Configure forwarding:

```
sudo iptables -A FORWARD -i br-ctlplane -o eth0 -j ACCEPT sudo iptables -A FORWARD -i eth0 -o br-ctlplane -m
state --state RELATED,ESTABLISHED -j ACCEPT sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

- Add external API interface:

```
sudo ip link add name vlan720 link br-ctlplane type vlan id 720 sudo ip addr add 10.2.0.254/24 dev vlan720
sudo ip link set dev vlan720 up
```

- Add stack user to the docker group:

```
newgrp docker exit su - stack source stackrc
```

## overcloud Configuration

### Configuration

- Configure nameserver for overcloud nodes

```
undercloud_nameserver=8.8.8.8 openstack subnet set 'openstack subnet show ctlplane-subnet -c id -f value' --
dns-nameserver ${undercloud_nameserver}
```

- overcloud images

#### 1. Create image directory

```
mkdir images cd images
```

#### 2. Get overcloud images

- TripleO

```
curl -O https://images.rdoproject.org/queens/rdo_trunk/current-tripleo-rdo/ironic-python-agent.tar curl
-O https://images.rdoproject.org/queens/rdo_trunk/current-tripleo-rdo/overcloud-full.tar tar xvf
ironic-python-agent.tar tar xvf overcloud-full.tar
```

- OSP13

```
sudo yum install -y rhosp-director-images rhosp-director-images-ipa for i in /usr/share/rhosp-director-
images/overcloud-full-latest-13.0.tar /usr/share/rhosp-director-images/ironic-python-agent-
latest-13.0.tar ; do tar -xvf $i; done
```

#### 3. Upload overcloud images

```
cd openstack overcloud image upload --image-path /home/stack/images/
```

- Prepare Ironic

OpenStack bare metal provisioning a.k.a Ironic is an integrated OpenStack program which aims to provision bare metal machines instead of virtual machines, forked from the Nova baremetal driver. It

is best thought of as a bare metal hypervisor API and a set of plugins which interact with the bare metal hypervisors



**NOTE:** Make sure to combine *ironic\_list* files from the three overcloud KVM hosts.

## 1. Add the overcloud VMs to Ironic

```
ipmi_password=<password>
ipmi_user=<user>
while IFS= read -r line; do
    mac=`echo $line|awk '{print $1}'`
    name=`echo $line|awk '{print $2}'`
    kvm_ip=`echo $line|awk '{print $3}'`
    profile=`echo $line|awk '{print $4}'`
    ipmi_port=`echo $line|awk '{print $5}'`
    uuid=`openstack baremetal node create --driver ipmi \
        --property cpus=4 \
        --property memory_mb=16348 \
        --property local_gb=100 \
        --property cpu_arch=x86_64 \
        --driver-info ipmi_username=${ipmi_user} \
        --driver-info ipmi_address=${kvm_ip} \
        --driver-info ipmi_password=${ipmi_password} \
        --driver-info ipmi_port=${ipmi_port} \
        --name=${name} \
        --property capabilities=profile:$
{profile},boot_option:local \
        -c uuid -f value`
    openstack baremetal port create --node ${uuid} ${mac}
done < <(cat ironic_list)

DEPLOY_KERNEL=$(openstack image show bm-deploy-kernel -f value -c id)
DEPLOY_RAMDISK=$(openstack image show bm-deploy-ramdisk -f value -c id)

for i in `openstack baremetal node list -c UUID -f value`; do
    openstack baremetal node set $i --driver-info deploy_kernel=$DEPLOY_KERNEL --driver-info
    deploy_ramdisk=$DEPLOY_RAMDISK
done

for i in `openstack baremetal node list -c UUID -f value`; do
```

```
openstack baremetal node show $i -c properties -f value
done
```

## 2. Introspect overcloud node

```
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
```

## 3. Add Baremetal Server (BMS) to Ironic

- Automated profiling

Evaluate the attributes of the physical server. The server will be automatically profiled based on the rules.

The following example shows how to create a rule for system manufacturer as “Supermicro” and memory greater or equal to 128GByte

```
cat << EOF > ~/rule_compute.json
[
{
    "description": "set physical compute",
    "conditions": [
        {"op": "eq", "field": "data://auto_discovered", "value": true},
        {"op": "eq", "field": "data://inventory.system_vendor.manufacturer",
            "value": "Supermicro"},
        {"op": "ge", "field": "memory_mb", "value": 128000}
    ],
    "actions": [
        {"action": "set-attribute", "path": "driver_info/ipmi_username",
            "value": "<user>"},
        {"action": "set-attribute", "path": "driver_info/ipmi_password",
            "value": "<password>"},
        {"action": "set-capability", "name": "profile", "value": "compute"},
        {"action": "set-attribute", "path": "driver_info/ipmi_address", "value":
            "{data[inventory][bmc_address]}" }
    ]
}
```

```
]
EOF
```

You can import the rule by:

```
openstack baremetal introspection rule import ~/rule_compute.json
```

- Scanning of BMC ranges

Scan the BMC IP range and automatically add new servers matching the above rule by:

```
ipmi_range=10.87.122.25/32
ipmi_password=<password>
ipmi_user=<user>
openstack overcloud node discover --range ${ipmi_range} \
  --credentials ${ipmi_user}:${ipmi_password} \
  --introspect --provide
```

- Create Flavor

```
for i in compute-dpdk \
compute-sriov \
contrail-controller \
contrail-analytics \
contrail-database \
contrail-analytics-database; do
  openstack flavor create $i --ram 4096 --vcpus 1 --disk 40
  openstack flavor set --property "capabilities:boot_option"="local" \
    --property "capabilities:profile"="${i}" ${i}
done
```

- Create TripleO-Heat-Template Copy

```
cp -r /usr/share/openstack-tripleo-heat-templates/ tripleo-heat-templates
git clone https://github.com/juniper/contrail-tripleo-heat-templates -b stable/queens
cp -r contrail-tripleo-heat-templates/* tripleo-heat-templates/
```

- Create and Upload Containers



- OpenStack Contrainers

1. Create OpenStack container file



**NOTE:** The container must be created based on the OpenStack program:

- TripleO

```
openstack overcloud container image prepare \
  --namespace docker.io/tripleoqueens \
  --tag current-tripleo \
  --tag-from-label rdo_version \
  --output-env-file=~/.overcloud_images.yaml

tag=`grep "docker.io/tripleoqueens" docker_registry.yaml |tail -1 |awk -F":"
'{print $3}'`

openstack overcloud container image prepare \
  --namespace docker.io/tripleoqueens \
  --tag ${tag} \
  --push-destination 192.168.24.1:8787 \
  --output-env-file=~/.overcloud_images.yaml \
  --output-images-file=~/.local_registry_images.yaml
```

- OSP13

```
openstack overcloud container image prepare \
  --push-destination=192.168.24.1:8787 \
  --tag-from-label {version}-{release} \
  --output-images-file ~/.local_registry_images.yaml \
  --namespace=registry.access.Red Hat.com/rhosp13 \
  --prefix=openstack- \
  --tag-from-label {version}-{release} \
  --output-env-file ~/.overcloud_images.yaml
```

2. Upload OpenStack Containers

```
openstack overcloud container image upload --config-file ~/.local_registry_images.yaml
```

- Contrail Containers

1. Create Contrail container file



**NOTE:** This step is optional. The Contrail containers can be downloaded from external registries later.

```
cd ~/tripleo-heat-templates/tools/contrail
./import_contrail_container.sh -f container_outputfile -r registry -t tag [-i
insecure] [-u username] [-p password] [-c certificate pat
```

Here are few examples of importing Contrail containers from different sources:

- Import from password protected public registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r hub.juniper.net/
contrail -u USERNAME -p PASSWORD -t 1234
```

- Import from Dockerhub:

```
./import_contrail_container.sh -f /tmp/contrail_container -r docker.io/
opencontrailnightly -t 1234
```

- Import from private secure registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r
device.example.net:5443 -c http://device.example.net/pub/device.example.net.crt -t
1234
```

- Import from private insecure registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r 10.0.0.1:5443 -i 1 -t
1234
```

## 2. Upload Contrail containers to undercloud registry

```
openstack overcloud container image upload --config-file /tmp/contrail_container
```

### Templates

Different YAML templates can be used to customize the overcloud

- Contrail Services customization

```
vi ~/tripleo-heat-templates/environments/contrail-services.yaml
parameter_defaults:
  ContrailSettings:
    VROUTER_GATEWAY: 10.0.0.1
    # KEY1: value1
    # KEY2: value2
```

- Contrail registry settings

```
vi ~/tripleo-heat-templates/environments/contrail-services.yaml
```

Here are few examples of default values for various registries:

- Public Juniper registry

```
parameter_defaults:
  ContrailRegistry: hub.juniper.net/contrail
  ContrailRegistryUser: <USER>
  ContrailRegistryPassword: <PASSWORD>
```

- Insecure registry

```
parameter_defaults:
  ContrailRegistryInsecure: true
  DockerInsecureRegistryAddress: 10.87.64.32:5000,192.168.24.1:8787
  ContrailRegistry: 10.87.64.32:5000
```

- Private secure registry

```
parameter_defaults:
  ContrailRegistryCertUrl: http://device.example.net/pub/device.example.net.crt
  ContrailRegistry: device.example.net:5443
```

- Contrail Container image settings

```
parameter_defaults:
  ContrailImageTag: queens-5.0-104-rhel-queens
```

- Network customization

In order to customize the network, define different networks and configure the overcloud nodes NIC layout. TripleO supports a flexible way of customizing the network.

The following networking customization example uses network as:

**Table 8: Network Customization**

Network	VLAN	overcloud Nodes
provisioning	-	All
internal_api	710	All
external_api	720	OpenStack CTRL
storage	740	OpenStack CTRL, Computes
storage_mgmt	750	OpenStack CTRL
tenant	-	Contrail CTRL, Computes

- Network activation in roles\_data

The networks must be activated per role in the roles\_data file:

```
vi ~/tripleo-heat-templates/roles_data_contrail_aio.yaml
```

- OpenStack Controller

```
#####
# Role: Controller                                     #
#####
- name: Controller
  description: |
    Controller role that has all the controller services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  tags:
    - primary
    - controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
```

- Compute Node

```
#####
# Role: Compute                                       #
#####
- name: Compute
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi
    - Tenant
    - Storage
```

- Contrail Controller

```
#####
# Role: ContrailController                                     #
#####
- name: ContrailController
  description: |
    ContrailController role that has all the Contrail controller services loaded
    and handles config, control and webui functions
  CountDefault: 1
  tags:
    - primary
    - contrailcontroller
  networks:
    - InternalApi
    - Tenant
```

- Compute DPDK

```
#####
# Role: ContrailDpdk                                         #
#####
- name: ContrailDpdk
  description: |
    Contrail Dpdk Node role
  CountDefault: 0
  tags:
    - contraildpdk
  networks:
    - InternalApi
    - Tenant
    - Storage
```

- Compute SRIOV

```
#####
# Role: ContrailSriov
#####
- name: ContrailSriov
  description: |
```

```

    Contrail Sriov Node role
CountDefault: 0
tags:
  - contrailsriov
networks:
  - InternalApi
  - Tenant
  - Storage

```

- Compute CSN

```

#####
# Role: ContrailTsn
#####
- name: ContrailTsn
  description: |
    Contrail Tsn Node role
  CountDefault: 0
  tags:
    - contrailtsn
  networks:
    - InternalApi
    - Tenant
    - Storage

```

- Network parameter configuration

```

cat ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml
resource_registry:
  OS::TripleO::Controller::Net::SoftwareConfig: ../../network/config/contrail/controller-
nic-config.yaml
  OS::TripleO::ContrailController::Net::SoftwareConfig: ../../network/config/contrail/
contrail-controller-nic-config.yaml
  OS::TripleO::ContrailControlOnly::Net::SoftwareConfig: ../../network/config/contrail/
contrail-controller-nic-config.yaml
  OS::TripleO::Compute::Net::SoftwareConfig: ../../network/config/contrail/compute-nic-
config.yaml
  OS::TripleO::ContrailDpdk::Net::SoftwareConfig: ../../network/config/contrail/contrail-
dpdk-nic-config.yaml
  OS::TripleO::ContrailSriov::Net::SoftwareConfig: ../../network/config/contrail/contrail-
sriov-nic-config.yaml

```

```
OS::TripleO::ContrailTsn::Net::SoftwareConfig: ../../network/config/contrail/contrail-
tsn-nic-config.yaml
```

```
parameter_defaults:
  # Customize all these values to match the local environment
  TenantNetCidr: 10.0.0.0/24
  InternalApiNetCidr: 10.1.0.0/24
  ExternalNetCidr: 10.2.0.0/24
  StorageNetCidr: 10.3.0.0/24
  StorageMgmtNetCidr: 10.4.0.0/24
  # CIDR subnet mask length for provisioning network
  ControlPlaneSubnetCidr: '24'
  # Allocation pools
  TenantAllocationPools: [{'start': '10.0.0.10', 'end': '10.0.0.200'}]
  InternalApiAllocationPools: [{'start': '10.1.0.10', 'end': '10.1.0.200'}]
  ExternalAllocationPools: [{'start': '10.2.0.10', 'end': '10.2.0.200'}]
  StorageAllocationPools: [{'start': '10.3.0.10', 'end': '10.3.0.200'}]
  StorageMgmtAllocationPools: [{'start': '10.4.0.10', 'end': '10.4.0.200'}]
  # Routes
  ControlPlaneDefaultRoute: 192.168.24.1
  InternalApiDefaultRoute: 10.1.0.1
  ExternalInterfaceDefaultRoute: 10.2.0.1
  # Vlan
  InternalApiNetworkVlanID: 710
  ExternalNetworkVlanID: 720
  StorageNetworkVlanID: 730
  StorageMgmtNetworkVlanID: 740
  TenantNetworkVlanID: 3211
  # Services
  EC2MetadataIp: 192.168.24.1 # Generally the IP of the undercloud
  DnsServers: ["172.x.x.x"]
  NtpServer: 10.0.0.1
```

- Network interface configuration

There are NIC configuration files per role.

```
cd ~/tripleo-heat-templates/network/config/contrail
```



- OpenStack Controller

```

heat_template_version: queens

description: >
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role. This is an example for a Nova compute node using
  Contrail vrouter and the vhost0 interface.
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::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
          params:
            $network_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.x.x.x/32
                    next_hop:
                      get_param: EC2MetadataIp
                  - default: true
                    next_hop:
                      get_param: ControlPlaneDefaultRoute
                - type: vlan
                  vlan_id:
                    get_param: InternalApiNetworkVlanID
                  device: nic1
                  addresses:
                    - ip_netmask:
                        get_param: InternalApiIpSubnet
                - type: vlan

```

```

        vlan_id:
            get_param: ExternalNetworkVlanID
        device: nic1
        addresses:
            - ip_netmask:
                get_param: ExternalIpSubnet
            - type: vlan
              vlan_id:
                  get_param: StorageNetworkVlanID
              device: nic1
              addresses:
                  - ip_netmask:
                      get_param: StorageIpSubnet
            - type: vlan
              vlan_id:
                  get_param: StorageMgmtNetworkVlanID
              device: nic1
              addresses:
                  - ip_netmask:
                      get_param: StorageMgmtIpSubnet
    outputs:
        OS::stack_id:
            description: The OsNetConfigImpl resource.
            value:
                get_resource: OsNetConfigImpl

```

- Contrail Controller

```

heat_template_version: queens
description: >
    Software Config to drive os-net-config to configure multiple interfaces
    for the compute role. This is an example for a Nova compute node using
    Contrail vrouter and the vhost0 interface.

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::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
          params:
            $network_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.x.x.x/32
          next_hop:
            get_param: EC2MetadataIp
        - default: true
          next_hop:
            get_param: ControlPlaneDefaultRoute
    - type: vlan
      vlan_id:
        get_param: InternalApiNetworkVlanID
      device: nic1
      addresses:
      - ip_netmask:
          get_param: InternalApiIpSubnet
    - type: interface
      name: nic2
      use_dhcp: false
      addresses:
      - ip_netmask:
          get_param: TenantIpSubnet

outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl

```

- Compute Node

```

heat_template_version: queens
description: >
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role. This is an example for a Nova compute node using

```

```

    Contrail vrouter and the vhost0 interface.
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::SoftwareConfig
    properties:

```

```

group: script
config:
  str_replace:
    template:
      get_file: ../../scripts/run-os-net-config.sh
  params:
    $network_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.x.x.x/32
                next_hop:
                  get_param: EC2MetadataIp
              - default: true
                next_hop:
                  get_param: ControlPlaneDefaultRoute
        - type: vlan
          vlan_id:
            get_param: InternalApiNetworkVlanID
          device: nic1
          addresses:
            - ip_netmask:
                get_param: InternalApiIpSubnet
        - type: vlan
          vlan_id:
            get_param: StorageNetworkVlanID
          device: nic1
          addresses:
            - ip_netmask:
                get_param: StorageIpSubnet
        - type: contrail_vrouter
          name: vhost0
          use_dhcp: false

```

```

        members:
          -
            type: interface
            name: nic2
            use_dhcp: false
        addresses:
          - ip_netmask:
              get_param: TenantIpSubnet

    outputs:
      OS::stack_id:
        description: The OsNetConfigImpl resource.
        value:
          get_resource: OsNetConfigImpl

```

- Advanced Network Configuration
- Advanced vRouter Kernel Mode Configurations

In addition to the standard NIC configuration, the vRouter kernel mode supports the following modes:

- VLAN
- Bond
- Bond + VLAN

#### NIC Template Configurations

The snippets below only shows the relevant section of the NIC configuration for each mode.

- VLAN

```

- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: nic2
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:

```

```

      str_replace:
        template: vlanVLANID
        params:
          VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
    addresses:
      - ip_netmask:
          get_param: TenantIpSubnet

```

- Bond

```

- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
    -
      type: interface
      name: nic2
    -
      type: interface
      name: nic3
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name: bond0
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet

```

- Bond + VLAN

```

- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:

```

```

-
  type: interface
  name: nic2
-
  type: interface
  name: nic3
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: bond0
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
        str_replace:
          template: vlanVLANID
        params:
          VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet

```

- Advanced vRouter DPDK Mode Configurations

In addition to the standard NIC configuration, the vRouter DPDK mode supports the following modes:

- Standard
- VLAN
- Bond
- Bond + VLAN

Network Environment Configuration

Enable the number of hugepages:

```
parameter_defaults:
  ContrailDpdkHugepages1GB: 10
```

## NIC Template Configurations

- Standard

```
- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
```

- VLAN

```
- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  vlan_id:
    get_param: TenantNetworkVlanID
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
```

- Bond

```
- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  bond_mode: 4
  bond_policy: layer2+3
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
    -
      type: interface
      name: nic3
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
```

- Bond + VLAN

```
- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  vlan_id:
    get_param: TenantNetworkVlanID
  bond_mode: 4
  bond_policy: layer2+3
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
    -
      type: interface
      name: nic3
```

```

        use_dhcp: false
    addresses:
    - ip_netmask:
        get_param: TenantIpSubnet

```

- Advanced vRouter SRIOV Mode Configurations

vRouter SRIOV can be used in the following combinations:

- SRIOV + Kernel mode

- Standard
- VLAN
- Bond
- Bond + VLAN

- SRIOV + DPDK mode

- Standard
- VLAN
- Bond
- Bond + VLAN

Network environment configuration

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages

- SRIOV + Kernel mode

```

parameter_defaults:
    ContrailSriovHugepages1GB: 10

```

- SRIOV + DPDK mode

```

parameter_defaults:
    ContrailSriovMode: dpdk

```



```
ContrailDpdkHugepages1GB: 10
ContrailSriovHugepages1GB: 10
```

### SRIOV PF/VF settings

```
NovaPCIPassthrough:
- devname: "ens2f1"
  physical_network: "sriov1"
ContrailSriovNumVFs: ["ens2f1:7"]
```

### NIC template configurations:

The SRIOV NICs are not configured in the NIC templates. However, vRouter NICs must still be configured.

See following NIC Template Configurations for vRouter kernel mode.

The snippets below only shows the relevant section of the NIC configuration for each mode.

- VLAN

```
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: nic2
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
        str_replace:
          template: vlanVLANID
        params:
          VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
```

- Bond

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
    -
      type: interface
      name: nic2
    -
      type: interface
      name: nic3
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name: bond0
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
```

- Bond + VLAN

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
    -
      type: interface
      name: nic2
    -
      type: interface
      name: nic3
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
```

```

    device: bond0
  - type: contrail_vrouter
    name: vhost0
    use_dhcp: false
    members:
      -
        type: interface
        name:
          str_replace:
            template: vlanVLANID
          params:
            VLANID: {get_param: TenantNetworkVlanID}
        use_dhcp: false
    addresses:
      - ip_netmask:
          get_param: TenantIpSubnet

```

See following NIC Template Configurations for vRouter DPDK mode:

- Standard

```

  - type: contrail_vrouter_dpdk
    name: vhost0
    use_dhcp: false
    driver: uio_pci_generic
    cpu_list: 0x01
    members:
      -
        type: interface
        name: nic2
        use_dhcp: false
    addresses:
      - ip_netmask:
          get_param: TenantIpSubnet

```

- VLAN

```

  - type: contrail_vrouter_dpdk
    name: vhost0
    use_dhcp: false
    driver: uio_pci_generic

```

```

cpu_list: 0x01
vlan_id:
  get_param: TenantNetworkVlanID
members:
  -
    type: interface
    name: nic2
    use_dhcp: false
addresses:
  - ip_netmask:
      get_param: TenantIpSubnet

```

- Bond

```

- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  bond_mode: 4
  bond_policy: layer2+3
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
    -
      type: interface
      name: nic3
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet

```

- Bond + VLAN

```

- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01

```

```

vlan_id:
  get_param: TenantNetworkVlanID
bond_mode: 4
bond_policy: layer2+3
members:
  -
    type: interface
    name: nic2
    use_dhcp: false
  -
    type: interface
    name: nic3
    use_dhcp: false
addresses:
  - ip_netmask:
    get_param: TenantIpSubnet

```

- Advanced Scenarios

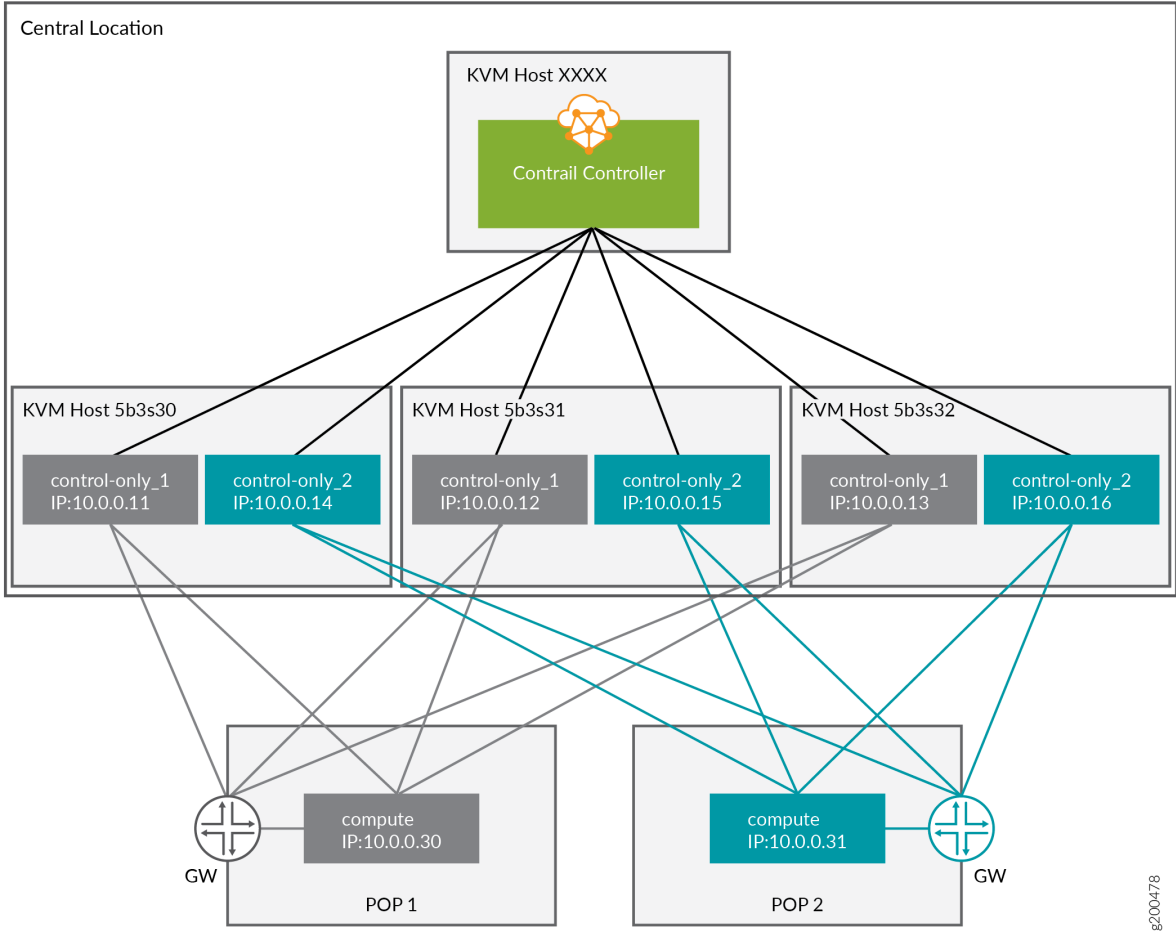
#### Remote Compute

Remote Compute extends the data plane to remote locations (POP) whilst keeping the control plane central. Each POP will have its own set of Contrail control services, which are running in the central location. The difficulty is to ensure that the compute nodes of a given POP connect to the Control nodes assigned to that POP. The Control nodes must have predictable IP addresses and the compute nodes have to know these IP addresses. In order to achieve that the following methods are used:

- Custom Roles
- Static IP assignment
- Precise Node placement
- Per Node hieradata

Each overcloud node has a unique DMI UUID. This UUID is known on the undercloud node as well as on the overcloud node. Hence, this UUID can be used for mapping node specific information. For each POP, a Control role and a Compute role has to be created.

#### Overview



g200478

Mapping Table

Table 9: Mapping Table

Nova Name	Ironic Name	UUID	KVM	IP Address	POP
overcloud - contrailcontrolonly-0	control-only-1-5b3s30	<p>Ironic UUID: 7d758dce-2784-45fd-be09-5a41eb53e764</p> <p>DMI UUID: 73F8D030-E896-4A95-A9F5-E1A4FEBE322D</p>	5b3s30	10.0.0.11	POP1

Table 9: Mapping Table (Continued)

Nova Name	Ironic Name	UUID	KVM	IP Address	POP
overcloud - contrailcontrolonly -1	control-only-2- 5b3s30	Ironic UUID: d26abdeb- d514- 4a37-a7fb-2cd2511c351f  DMI UUID: 14639A66- D62C- 4408-82EE- FDDC4E509687	5b3s30	10.0.0.14	POP2
overcloud - contrailcontrolonly -2	control-only-1- 5b3s31	Ironic UUID: 91dd9fa9- e8eb- 4b51-8b5e-bbaffb6640e4  DMI UUID: 28AB0B57- D612- 431E- B177-1C578AE0FEA4	5b3s31	10.0.0.12	POP1
overcloud - contrailcontrolonly -3	control-only-2- 5b3s31	Ironic UUID: 09fa57b8-580f- 42ec-bf10-a19573521ed4  DMI UUID: 09BEC8CB-77E9- 42A6- AFF4-6D4880FD87D0	5b3s31	10.0.0.15	POP2
overcloud - contrailcontrolonly -4	control-only-1- 5b3s32	Ironic UUID: 4766799-24c8- 4e3b-af54-353f2b796ca4  DMI UUID: 3993957A- ECBF- 4520-9F49-0AF6EE1667A7	5b3s32	10.0.0.13	POP1

Table 9: Mapping Table *(Continued)*

Nova Name	Ironic Name	UUID	KVM	IP Address	POP
overcloud - contrailcontrolonly -5	control-only-2- 5b3s32	Ironic UUID: 58a803ae- a785- 470e-9789-139abbfa74fb  DMI UUID: AF92F485- C30C- 4D0A-BDC4- C6AE97D06A66	5b3s32	10.0.0.16	POP2

ControlOnly preparation

Add ControlOnly overcloud VMs to overcloud KVM host



**NOTE:** This has to be done on the overcloud KVM hosts

Two ControlOnly overcloud VM definitions will be created on each of the overcloud KVM hosts.

```

ROLES=control-only:2
num=4
ipmi_user=<user>
ipmi_password=<password>
libvirt_path=/var/lib/libvirt/images
port_group=overcloud
prov_switch=br0

/bin/rm ironic_list
IFS=',' read -ra role_list <<< "${ROLES}"
for role in ${role_list[@]}; do
    role_name=`echo $role|cut -d ":" -f 1`
    role_count=`echo $role|cut -d ":" -f 2`
    for count in `seq 1 ${role_count}`; do
        echo $role_name $count
        qemu-img create -f qcow2 ${libvirt_path}/${role_name}_${count}.qcow2 99G
        virsh define /dev/stdin <<EOF
$(virt-install --name ${role_name}_${count} \
--disk ${libvirt_path}/${role_name}_${count}.qcow2 \
--vcpus=4 \

```



```

--ram=16348 \
--network network=br0,model=virtio,portgroup=${port_group} \
--network network=br1,model=virtio \
--virt-type kvm \
--cpu host \
--import \
--os-variant rhel7 \
--serial pty \
--console pty,target_type=virtio \
--graphics vnc \
--print-xml)
EOF
    vbmc add ${role_name}_${count} --port 1623${num} --username ${ipmi_user} --password $
{ipmi_password}
    vbmc start ${role_name}_${count}
    prov_mac=`virsh domiflist ${role_name}_${count}|grep ${prov_switch}|awk '{print $5}'`
    vm_name=${role_name}-${count}-`hostname -s`
    kvm_ip=`ip route get 1 |grep src |awk '{print $7}'`
    echo ${prov_mac} ${vm_name} ${kvm_ip} ${role_name} 1623${num}>> ironic_list
    num=$((expr $num + 1))
done
done

```



**NOTE:** The generated *ironic\_list* will be needed on the undercloud to import the nodes to Ironic.

Get the *ironic\_lists* from the overcloud KVM hosts and combine them.

```

cat ironic_list_control_only
52:54:00:3a:2f:ca control-only-1-5b3s30 10.87.64.31 control-only 16234
52:54:00:31:4f:63 control-only-2-5b3s30 10.87.64.31 control-only 16235
52:54:00:0c:11:74 control-only-1-5b3s31 10.87.64.32 control-only 16234
52:54:00:56:ab:55 control-only-2-5b3s31 10.87.64.32 control-only 16235
52:54:00:c1:f0:9a control-only-1-5b3s32 10.87.64.33 control-only 16234
52:54:00:f3:ce:13 control-only-2-5b3s32 10.87.64.33 control-only 16235

```

Import:

```

ipmi_password=<password>
ipmi_user=<user>

```

```

DEPLOY_KERNEL=$(openstack image show bm-deploy-kernel -f value -c id)
DEPLOY_RAMDISK=$(openstack image show bm-deploy-ramdisk -f value -c id)

num=0
while IFS= read -r line; do
    mac=`echo $line|awk '{print $1}'`
    name=`echo $line|awk '{print $2}'`
    kvm_ip=`echo $line|awk '{print $3}'`
    profile=`echo $line|awk '{print $4}'`
    ipmi_port=`echo $line|awk '{print $5}'`
    uuid=`openstack baremetal node create --driver ipmi \
        --property cpus=4 \
        --property memory_mb=16348 \
        --property local_gb=100 \
        --property cpu_arch=x86_64 \
        --driver-info ipmi_username=${ipmi_user} \
        --driver-info ipmi_address=${kvm_ip} \
        --driver-info ipmi_password=${ipmi_password} \
        --driver-info ipmi_port=${ipmi_port} \
        --name=${name} \
        --property capabilities=boot_option:local \
        -c uuid -f value`

    openstack baremetal node set ${uuid} --driver-info deploy_kernel=$DEPLOY_KERNEL --driver-
info deploy_ramdisk=$DEPLOY_RAMDISK
    openstack baremetal port create --node ${uuid} ${mac}
    openstack baremetal node manage ${uuid}
    num=$((expr $num + 1))
done < <(cat ironic_list_control_only)

```

### ControlOnly node introspection

```
openstack overcloud node introspect --all-manageable --provide
```

### Get the ironic UUID of the ControlOnly nodes

```

openstack baremetal node list |grep control-only
| 7d758dce-2784-45fd-be09-5a41eb53e764 | control-only-1-5b3s30 | None | power off |
available | False |
| d26abdeb-d514-4a37-a7fb-2cd2511c351f | control-only-2-5b3s30 | None | power off |
available | False |

```

```
| 91dd9fa9-e8eb-4b51-8b5e-bbaffb6640e4 | control-only-1-5b3s31 | None | power off |
available | False |
| 09fa57b8-580f-42ec-bf10-a19573521ed4 | control-only-2-5b3s31 | None | power off |
available | False |
| f4766799-24c8-4e3b-af54-353f2b796ca4 | control-only-1-5b3s32 | None | power off |
available | False |
| 58a803ae-a785-470e-9789-139abbfa74fb | control-only-2-5b3s32 | None | power off |
available | False |
```

The first ControlOnly node on each of the overcloud KVM hosts will be used for POP1, the second for POP2, and so and so forth.

Get the ironic UUID of the POP compute nodes:

```
openstack baremetal node list |grep compute
| 91d6026c-b9db-49cb-a685-99a63da5d81e | compute-3-5b3s30 | None | power off | available |
False |
| 8028eb8c-e1e6-4357-8fcf-0796778bd2f7 | compute-4-5b3s30 | None | power off | available |
False |
| b795b3b9-c4e3-4a76-90af-258d9336d9fb | compute-3-5b3s31 | None | power off | available |
False |
| 2d4be83e-6fcc-4761-86f2-c2615dd15074 | compute-4-5b3s31 | None | power off | available |
False |
```

The first two compute nodes belong to POP1 the second two compute nodes belong to POP2.

Create an input YAML using the ironic UUIDs:

```
~/subcluster_input.yaml
---
- subcluster: subcluster1
  asn: "65413"
  control_nodes:
    - uuid: 7d758dce-2784-45fd-be09-5a41eb53e764
      ipaddress: 10.0.0.11
    - uuid: 91dd9fa9-e8eb-4b51-8b5e-bbaffb6640e4
      ipaddress: 10.0.0.12
    - uuid: f4766799-24c8-4e3b-af54-353f2b796ca4
      ipaddress: 10.0.0.13
  compute_nodes:
    - uuid: 91d6026c-b9db-49cb-a685-99a63da5d81e
      vrouter_gateway: 10.0.0.1
```

```

- uuid: 8028eb8c-e1e6-4357-8fcf-0796778bd2f7
  vrouter_gateway: 10.0.0.1
- subcluster: subcluster2
  asn: "65414"
  control_nodes:
    - uuid: d26abdeb-d514-4a37-a7fb-2cd2511c351f
      ipaddress: 10.0.0.14
    - uuid: 09fa57b8-580f-42ec-bf10-a19573521ed4
      ipaddress: 10.0.0.15
    - uuid: 58a803ae-a785-470e-9789-139abbfa74fb
      ipaddress: 10.0.0.16
  compute_nodes:
    - uuid: b795b3b9-c4e3-4a76-90af-258d9336d9fb
      vrouter_gateway: 10.0.0.1
    - uuid: 2d4be83e-6fcc-4761-86f2-c2615dd15074
      vrouter_gateway: 10.0.0.1

```



**NOTE:** Only control\_nodes, compute\_nodes, dpdk\_nodes and sriov\_nodes are supported.

Generate subcluster environment:

```

~/tripleo-heat-templates/tools/contrail/create_subcluster_environment.py -i ~/
subcluster_input.yaml \
-o ~/tripleo-heat-templates/environments/contrail/contrail-subcluster.yaml

```

Check subcluster environment file:

```

cat ~/tripleo-heat-templates/environments/contrail/contrail-subcluster.yaml
parameter_defaults:
  NodeDataLookup:
    041D7B75-6581-41B3-886E-C06847B9C87E:
      contrail_settings:
        CONTROL_NODES: 10.0.0.14,10.0.0.15,10.0.0.16
        SUBCLUSTER: subcluster2
        VROUTER_GATEWAY: 10.0.0.1
    09BEC8CB-77E9-42A6-AFF4-6D4880FD87D0:
      contrail_settings:
        BGP_ASN: '65414'
        SUBCLUSTER: subcluster2

```

```

14639A66-D62C-4408-82EE-FDDC4E509687:
  contrail_settings:
    BGP_ASN: '65414'
    SUBCLUSTER: subcluster2
28AB0B57-D612-431E-B177-1C578AE0FEA4:
  contrail_settings:
    BGP_ASN: '65413'
    SUBCLUSTER: subcluster1
3993957A-ECBF-4520-9F49-0AF6EE1667A7:
  contrail_settings:
    BGP_ASN: '65413'
    SUBCLUSTER: subcluster1
73F8D030-E896-4A95-A9F5-E1A4FEBE322D:
  contrail_settings:
    BGP_ASN: '65413'
    SUBCLUSTER: subcluster1
7933C2D8-E61E-4752-854E-B7B18A424971:
  contrail_settings:
    CONTROL_NODES: 10.0.0.14,10.0.0.15,10.0.0.16
    SUBCLUSTER: subcluster2
    VROUTER_GATEWAY: 10.0.0.1
AF92F485-C30C-4D0A-BDC4-C6AE97D06A66:
  contrail_settings:
    BGP_ASN: '65414'
    SUBCLUSTER: subcluster2
BB9E9D00-57D1-410B-8B19-17A0DA581044:
  contrail_settings:
    CONTROL_NODES: 10.0.0.11,10.0.0.12,10.0.0.13
    SUBCLUSTER: subcluster1
    VROUTER_GATEWAY: 10.0.0.1
E1A809DE-FDB2-4EB2-A91F-1B3F75B99510:
  contrail_settings:
    CONTROL_NODES: 10.0.0.11,10.0.0.12,10.0.0.13
    SUBCLUSTER: subcluster1
    VROUTER_GATEWAY: 10.0.0.1

```

## Deployment

Add contrail-subcluster.yaml, contrail-ips-from-pool-all.yaml and contrail-scheduler-hints.yaml to the OpenStack deploy command:

```
openstack overcloud deploy --templates ~/tripleo-heat-templates \
-e ~/overcloud_images.yaml \
-e ~/tripleo-heat-templates/environments/network-isolation.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-subcluster.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-ips-from-pool-all.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-scheduler-hints.yaml \
--roles-file ~/tripleo-heat-templates/roles_data_contrail_aio.yaml
```

## overcloud Installation

Deployment:

```
openstack overcloud deploy --templates ~/tripleo-heat-templates \
-e ~/overcloud_images.yaml \
-e ~/tripleo-heat-templates/environments/network-isolation.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
--roles-file ~/tripleo-heat-templates/roles_data_contrail_aio.yaml
```

Validation Test:

```
source overcloudrc
curl -O http://download.cirros-cloud.net/0.3.5/cirros-0.3.5-x86_64-disk.img
openstack image create --container-format bare --disk-format qcow2 --file cirros-0.3.5-x86_64-disk.img cirros
openstack flavor create --public cirros --id auto --ram 64 --disk 0 --vcpus 1
openstack network create net1
openstack subnet create --subnet-range 1.0.0.0/24 --network net1 sn1
nova boot --image cirros --flavor cirros --nic net-id=openstack network show net1 -c id -f value --availability-zone nova:overcloud-novacompute-0.localdomain c1
nova list
```

## RELATED DOCUMENTATION

*Provisioning Red Hat OpenShift Container Platform Clusters Using Ansible Deployer*

# Provisioning Red Hat OpenShift Container Platform Clusters Using Ansible Deployer

## IN THIS SECTION

- [Installing a Standalone OpenShift Cluster Using Ansible Deployer | 351](#)
- [Provisioning of Nested OpenShift Clusters Using Ansible Deployer—Beta | 361](#)

Contrail Release 5.0.2 supports the following ways of installing and provisioning standalone and nested Red Hat OpenShift Container Platform version 3.9 clusters. These instructions are valid for systems with Microsoft Azure, Amazon Web Services (AWS), or bare metal server (BMS).

## Installing a Standalone OpenShift Cluster Using Ansible Deployer

### Prerequisites

Ensure the following system requirements.

- **Master Node** (x1 or x3 for high availability)
  - Image: RHEL 7.5
  - CPU/RAM: 4 CPU, 32 GB RAM
  - Disk: 250 GB
  - Security Group: Allow all traffic from everywhere
- **Slave Node** (x*n*)
  - Image: RHEL 7.5
  - CPU/RAM: 8 CPU, 64 GB RAM
  - Disk: 250 GB
  - 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 7.5
  - CPU/RAM: 2 CPU, 16 GB RAM
  - Disk: 100 GB
  - Security Group: Allow all traffic from everywhere



**NOTE:** Ensure that you launch the instances in the same subnet.

### Installing a standalone OpenShift cluster using Ansible deployer

Perform the following steps to install a standalone OpenShift cluster with Contrail as networking provider and provision the cluster using contrail-ansible-deployer.

1. Re-image all the servers.

```
/server-manager reimage --server_id server1 redhat-7.5-minimal
```

2. Set up environment nodes:

- a. You must register all nodes in order to subscribe to OpenShift Container Platform. Register all nodes in the cluster using Red Hat Subscription Manager (RHSM).

```
(all-nodes)# subscription-manager register --username username --password password --force
```

- b. List the available subscriptions.

```
(all-nodes)# subscription-manager list --available --matches '*OpenShift*'
```

- c. From the list of available subscriptions, find and attach the pool ID for the OpenShift Container Platform subscription.

```
(all-nodes)# subscription-manager attach --pool=pool-ID
```

- d. Disable all yum repositories.

```
(all-nodes)# subscription-manager repos --disable="*"
```

- e. Enable only the required repositories.

```
(all-nodes)# subscription-manager repos \
  --enable="rhel-7-server-rpms" \
  --enable="rhel-7-server-extras-rpms" \
  --enable="rhel-7-server-ose-3.9-rpms" \
```



```
--enable="rhel-7-fast-datapath-rpms" \
--enable="rhel-7-server-ansible-2.5-rpms"
```

- f. 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
```

```
(all-nodes)# yum install wget -y && wget -O /tmp/epel-release-latest-7.noarch.rpm
```

- g. Update the system to use the latest packages.

```
(all-nodes)# yum update -y
```

- h. Install the following package which provides OpenShift Container Platform utilities.

```
(all-nodes)# yum install atomic-openshift-excluder atomic-openshift-utils git python-netaddr -y
```

- i. Remove the atomic-openshift packages from the list for the duration of the installation.

```
(all-nodes)# atomic-openshift-excluder unexclude -y
```

- j. 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
```

- k. Log out.

```
(all-nodes)# logout
```

- l. Log in as root user.

```
ssh node-ip -l root
```

- m. Enforce the SELinux security policy.

```
(all-nodes)# vi /etc/selinux/config

SELINUX=enforcing
```

3. Install the supported Ansible version by running the following command:

```
yum install ansible
```

4. Get the files from the latest tar ball. Download the OpenShift Container Platform install package from Juniper software download site and modify the contents of the openshift-ansible inventory file.

- a. Download the Openshift Deployer (contrail-openshift-deployer-5.0.X.tar) installer tar ball from the Juniper software download site: <https://www.juniper.net/support/downloads/?p=contrail#sw>

- b. Copy the install package to the node from where Ansible must be deployed. Ensure that the node has password-free access to the OpenShift master and slave nodes.

```
scp contrail-openshift-deployer-5.0.X.tar openshift-ansible-node:/root/
```

- c. Untar the contrail-openshift-deployer-5.0.X.tar package.

```
tar -xvf contrail-openshift-deployer-5.0.X.tar -C /root/
```

- d. Verify the contents of the **openshift-ansible** directory.

```
cd /root/openshift-ansible/
```

- e. Modify the inventory file to match your OpenShift environment.

Populate the install file with Contrail configuration parameters specific to your system. Refer to the following example.

Add the master nodes in the [nodes] section of the inventory to ensure that the Contrail control pods will come up on the OpenShift master nodes.

```
(ansible-node)# vi /root/openshift-ansible/inventory/ose-install

[OSEv3:vars]
...
contrail_version=5.0
contrail_container_tag=5.0.X-0.X
contrail_registry=hub.juniper.net/contrail
contrail_registry_username=username-for-contrail-container-registry
contrail_registry_password=password-for-contrail-container-registry
...
```

For more information about each of these parameters and for an example for a HA master, see <https://github.com/Juniper/contrail-kubernetes-docs/blob/master/install/openshift/3.9/standalone-openshift.md>.



**NOTE:** Juniper Networks recommends that you obtain the Ansible source files from the latest release.

This procedure assumes that there is one master node, one infra node, and one compute node.

```
master : server1 (1x.xx.xx.11)

infra : server2 (1x.xx.xx.22)

compute : server3 (1x.xx.xx.33)
```

5. Edit `/etc/hosts` to allow all machines to access all nodes.

```
[root@server1]# cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.84.5.100 puppet
1x.xx.xx.11 server1.contrail.juniper.net server1
1x.xx.xx.22 server2.contrail.juniper.net server2
1x.xx.xx.33 server3.contrail.juniper.net server3
```

6. Set up password free SSH access to the Ansible node and all the nodes.

```
ssh-keygen -t rsa
ssh-copy-id root@1x.xx.xx.11
ssh-copy-id root@1x.xx.xx.22
ssh-copy-id root@1x.xx.xx.33
```

7. Run Ansible playbook to install OpenShift Container Platform with Contrail. Before you run Ansible playbook, ensure that you have edited **inventory/ose-install** file as shown below.

```
(ansible-node)# cd /root/openshift-ansible
(ansible-node)# ansible-playbook -i inventory/ose-install playbooks/prerequisites.yml
(ansible-node)# ansible-playbook -i inventory/ose-install playbooks/deploy_cluster.yml
```

8. Verify that Contrail has been installed and is operational.

```
(master)# oc get ds -n kube-system
(master)# oc get pods -n kube-system
```

9. Install the customized web console that should run on the infra nodes. To do this, disable the OpenShift Web console and enable the Contrail Web console and add the following lines in **ose-install**:

```
openshift_web_console_install=false
openshift_web_console_contrail_install=true
```

10. Create a password for the admin user to log in to the UI from the master node.

```
(master-node)# htpasswd /etc/origin/master/htpasswd admin
```



**NOTE:** If you are using a load balancer, you must manually copy the htpasswd file into all your master nodes.

11. Assign cluster-admin role to admin user.

```
(master-node)# oc adm policy add-cluster-role-to-user cluster-admin admin
(master-node)# oc login -u admin
```

12. Open a Web browser and type the entire fqdn name of your master node or load balancer node, followed by :8443/console.

```
https://<your host name from your ose-install inventory>:8443/console
```



**NOTE:** Use the user name and password created above to log into the Web console.



**NOTE:** Your DNS should resolve the host name for access. If the host name is not resolved, modify the /etc/hosts file to route to the above host.

13. Verify the provisioning process.

```
(master-node)# oc get pods -n kube-system
```

The status of all the pods must be displayed as Running.

```
(master-node)# contrail-status
```

All contrail-services must be displayed as active.

**14. Access the Contrail and OpenShift Web user interfaces and attempt to log in to each.**

Contrail: <https://master-node-ip:8143> with <admin/cOntrail123> login credentials.

OpenShift: <https://infra-node-ip:8443> with <admin/password created in step 10> login credentials.

You can test the system by launching pods, services, namespaces, network-policies, ingress, and soon. For more information, see the examples listed in <https://github.com/juniper/openshift-contrail/tree/master/openshift/examples>.

### Sample ose-install File

Use the following sample **ose-install** file for reference.

```
[OSEv3:children]
masters
nodes
etcd
openshift_ca

[OSEv3:vars]
ansible_ssh_user=root
ansible_become=yes
debug_level=2
deployment_type=origin #openshift-enterprise for Redhat
openshift_release=v3.9
#openshift_repos_enable_testing=true
containerized=false
openshift_install_examples=true
openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge':
'true', 'kind': 'HTPasswdPasswordIdentityProvider', 'filename': '/etc/origin/master/htpasswd'}]
osm_cluster_network_cidr=10.32.0.0/12
openshift_portal_net=10.96.0.0/12
openshift_use_dnsmasq=true
openshift_clock_enabled=true
openshift_hosted_manage_registry=false
openshift_hosted_manage_router=false
openshift_enable_service_catalog=false
```

```

openshift_use_openshift_sdn=false
os_sdn_network_plugin_name='cni'
openshift_disable_check=memory_availability,package_availability,disk_availability,package_version,docker_storage
openshift_docker_insecure_registries=opencontrailnightly
openshift_web_console_install=false
#openshift_web_console_nodeselector={'region':'infra'}

openshift_web_console_contrail_install=true
openshift_use_contrail=true
nested_mode_contrail=false
contrail_version=5.0
contrail_container_tag=queens-5.0-156
contrail_registry=opencontrailnightly
# Username /Password for private Docker registries
#contrail_registry_username=test
#contrail_registry_password=test
# Below option presides over contrail masters if set
#vrouter_physical_interface=ens160
#docker_version=1.13.1
ntpserver=10.1.1.1 # a proper ntpserver is required for contrail.

# Contrail_vars
# below variables are used by contrail kubemanager to configure the cluster,
# you can configure all options below. All values are defaults and can be modified.

#kubernetes_api_server=10.84.13.52      # in our case this is the master, which is default
#kubernetes_api_port=8080
#kubernetes_api_secure_port=8443
#cluster_name=myk8s
#cluster_project={}
#cluster_network={}
#pod_subnets=10.32.0.0/12
#ip_fabric_subnets=10.64.0.0/12
#service_subnets=10.96.0.0/12
#ip_fabric_forwarding=false
#ip_fabric_snat=false
#public_fip_pool={}
#vnc_endpoint_ip=20.1.1.1
#vnc_endpoint_port=8082

[masters]
10.84.13.52 openshift_hostname=openshift-master

```

```
[etcd]
10.84.13.52 openshift_hostname=openshift-master

[nodes]
10.84.13.52 openshift_hostname=openshift-master
10.84.13.53 openshift_hostname=openshift-compute
10.84.13.54 openshift_hostname=openshift-infra openshift_node_labels="{ 'region': 'infra' }"

[openshift_ca]
10.84.13.52 openshift_hostname=openshift-master
```

### Sample ose-install File for a HA setup

Use the following sample **ose-install** file for reference.

```
[OSEv3:children]
masters
nodes
etcd
lb
openshift_ca

[OSEv3:vars]
ansible_ssh_user=root
ansible_become=yes
debug_level=2
deployment_type=openshift-enterprise
openshift_release=v3.9
openshift_repos_enable_testing=true
containerized=false
openshift_install_examples=true
openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge':
'true', 'kind': 'HTPasswdPasswordIdentityProvider', 'filename': '/etc/origin/master/htpasswd'}]
osm_cluster_network_cidr=10.32.0.0/12
openshift_portal_net=10.96.0.0/12
openshift_use_dnsmasq=true
openshift_clock_enabled=true
openshift_enable_service_catalog=false
openshift_use_openshift_sdn=false
os_sdn_network_plugin_name='cni'
openshift_disable_check=disk_availability,package_version,docker_storage
```

```

openshift_docker_insecure_registries=ci-repo.englab.juniper.net:5010
openshift_web_console_install=false
openshift_web_console_contrail_install=true
openshift_web_console_nodeselector={'region':'infra'}
openshift_hosted_manage_registry=true
openshift_hosted_registry_selector="region=infra"
openshift_hosted_manage_router=true
openshift_hosted_router_selector="region=infra"
ntpserver=10.84.5.100

# Openshift HA
openshift_master_cluster_method=native
openshift_master_cluster_hostname=lb
openshift_master_cluster_public_hostname=lb

# Below are Contrail variables. Comment them out if you don't want to install Contrail through
# ansible-playbook
contrail_version=5.0
openshift_use_contrail=true
#rhel-queens-5.0-latest
#contrail_container_tag=rhel-queens-5.0-319
#contrail_registry=ci-repo.englab.juniper.net:5010
contrail_registry=hub.juniper.net/contrail
contrail_registry_username=JNPR-Customer200
contrail_registry_password=F*****f
contrail_container_tag=5.0.2-0.309-rhel-queens
contrail_nodes=[10.0.0.7, 10.0.0.8, 10.0.0.13]
vrouters_physical_interface=eth0

[masters]
10.0.0.7 openshift_hostname=master1
10.0.0.8 openshift_hostname=master2
10.0.0.13 openshift_hostname=master3

[lb]
10.0.0.5 openshift_hostname=lb

[etcd]
10.0.0.7 openshift_hostname=master1
10.0.0.8 openshift_hostname=master2
10.0.0.13 openshift_hostname=master3

```



```
[nodes]
10.0.0.7 openshift_hostname=master1
10.0.0.8 openshift_hostname=master2
10.0.0.13 openshift_hostname=master3
10.0.0.10 openshift_hostname=slave1
10.0.0.4 openshift_hostname=slave2
10.0.0.6 openshift_hostname=infra1 openshift_node_labels="{ 'region': 'infra' }"
10.0.0.11 openshift_hostname=infra2 openshift_node_labels="{ 'region': 'infra' }"
10.0.0.12 openshift_hostname=infra3 openshift_node_labels="{ 'region': 'infra' }"

[openshift_ca]
10.0.0.7 openshift_hostname=master1
10.0.0.8 openshift_hostname=master2
10.0.0.13 openshift_hostname=master3
```

## Provisioning of Nested OpenShift Clusters Using Ansible Deployer—Beta

### IN THIS SECTION

- [Configure network connectivity to Contrail configuration and data plane functions | 361](#)
- [Installing Nested OpenShift Cluster using Ansible Deployer | 365](#)

When Contrail provides networking for an OpenShift cluster that is provisioned on a Contrail-OpenStack cluster, it is called a nested OpenShift cluster. Contrail components are shared between the two clusters.

The following steps describe how to provision a nested OpenShift cluster.



**NOTE:** Provisioning of nested OpenShift Clusters is supported only as a Beta feature. Ensure that you have an operational Contrail-OpenStack cluster based on Contrail Release 5.0 before provisioning a nested OpenShift cluster.

### Configure network connectivity to Contrail configuration and data plane functions

A nested OpenShift cluster is managed by the same Contrail control processes that manage the underlying OpenStack cluster. The nested OpenShift cluster needs IP reachability to the Contrail control processes. Because the OpenShift cluster is actually an overlay on the OpenStack cluster, you can use

the link local service feature or a combination of link local service with fabric Source Network Address Translation (SNAT) feature of Contrail to provide IP reachability to and from the OpenShift cluster on the overlay and the OpenStack cluster.

Use *one* of the following options to create link local services.

- **Fabric SNAT with link local service**

To provide IP reachability to and from the Kubernetes cluster using the fabric SNAT with link local service, perform the following steps.

1. Enable fabric SNAT on the virtual network of the VMs.

The fabric SNAT feature must be enabled on the virtual network of the virtual machines on which the Kubernetes master and minions are running.

2. Create one link local service for the Container Network Interface (CNI) to communicate with its vRouter using the Contrail GUI.

The following link local service is required.

Contrail Process	Service IP	Service Port	Fabric IP	Fabric Port
vRouter	<i>Service_IP for the active node</i>	9091	127.0.0.1	9091



**NOTE:** Fabric IP address is 127.0.0.1 since you must make the CNI communicate with the vRouter on its underlay node.

For example, the following link local services must be created:

Link Local Service Name	Service IP	Service Port	Fabric IP	Fabric Port
K8s-cni-to-agent	10.10.10.5	9091	127.0.0.1	9091



**NOTE:** Here 10.10.10.5 is the Service IP address that you chose. This can be any unused IP in the cluster. This IP address is primarily used to identify link local traffic and has no other significance.

- **Link local only**

To configure a Link local service, you need a Service IP address and a Fabric IP address. The fabric IP address is the node IP address on which the Contrail processes are running. Service IP address along with port number is used by the data plane to identify the fabric IP address. Service IP address is required to be a unique and unused IP address in the entire OpenStack cluster. For each node of the OpenStack cluster, one service IP address must be identified.

The following are the link local services are required:

Contrail Process	Service IP	Service Port	Fabric IP	Fabric Port
Contrail Config	<i>Service_IP for the active node</i>	8082	<i>Node_IP for the active node</i>	8082
Contrail Analytics	<i>Service_IP for the active node</i>	8086	<i>Node_IP for the active node</i>	8086
Contrail Msg Queue	<i>Service_IP for the active node</i>	5673	<i>Node_IP for the active node</i>	5673
Contrail VNC DB	<i>Service_IP for the active node</i>	9161	<i>Node_IP for the active node</i>	9161
Keystone	<i>Service_IP for the active node</i>	35357	<i>Node_IP for the active node</i>	35357
vRouter	<i>Service_IP for the active node</i>	9091	127.0.0.1	9091

For example, consider the following hypothetical OpenStack cluster:

```
Contrail Config : 192.168.1.100
Contrail Analytics : 192.168.1.100, 192.168.1.101
Contrail Msg Queue : 192.168.1.100
Contrail VNC DB : 192.168.1.100, 192.168.1.101, 192.168.1.102
Keystone: 192.168.1.200
Vrouter: 192.168.1.300, 192.168.1.400, 192.168.1.500
```

This cluster is made of seven nodes. You must allocate seven unused IP addresses for these nodes:

```
192.168.1.100 --> 10.10.10.1
192.168.1.101 --> 10.10.10.2
192.168.1.102 --> 10.10.10.3
192.168.1.200 --> 10.10.10.4
192.168.1.300 --> 10.10.10.5
192.168.1.400 --> 10.10.10.6
192.168.1.500 --> 10.10.10.7
```

The following link local services must be created:

Link Local Service Name	Service IP	Service Port	Fabric IP	Fabric Port
Contrail Config	10.10.10.1	8082	192.168.1.100	8082
Contrail Analytics	10.10.10.1	8086	192.168.1.100	8086
Contrail Analytics 2	10.10.10.2	8086	192.168.1.101	8086
Contrail Msg Queue	10.10.10.1	5673	192.168.1.100	5673
Contrail VNC DB 1	10.10.10.1	9161	192.168.1.100	9161
Contrail VNC DB 2	10.10.10.2	9161	192.168.1.101	9161
Contrail VNC DB 3	10.10.10.3	9161	192.168.1.102	9161
Keystone	10.10.10.4	35357	192.168.1.200	35357
VRouter-192.168.1.300	10.10.10.5	9091	127.0.0.1	9091
VRouter-192.168.1.400	10.10.10.6	9091	127.0.0.1	9091

VRouter-192.168.1.500	10.10.10.7	9091	127.0.0.1	9091
-----------------------	------------	------	-----------	------

## Installing Nested OpenShift Cluster using Ansible Deployer

Perform the steps on [No Link Title](#) to continue installing and provisioning the OpenShift cluster.

### Sample ose-install File

Add the following information to the ["No Link Title" on page 357](#).

```
#Nested mode vars
nested_mode_contrail=true
auth_mode=keystone
keystone_auth_host=192.168.24.12
keystone_auth_admin_tenant=admin
keystone_auth_admin_user=admin
keystone_auth_admin_password=MAYffWrX7ZpPrV2AMaA9zAUvG
keystone_auth_admin_port=35357
keystone_auth_url_version=/v3
#k8s_nested_vrouter_vip is a service IP for the running node which we configured above
k8s_nested_vrouter_vip=10.10.10.5
#k8s_vip is kubernetes api server ip
k8s_vip=192.168.1.3
#cluster_network is the one which vm network belongs to
cluster_network="{ 'domain': 'default-domain', 'project': 'admin', 'name': 'net1' }"
```

For more information, see <https://github.com/Juniper/contrail-kubernetes-docs/tree/master/install/openshift/3.9>.

## RELATED DOCUMENTATION

| *Deploying Contrail with Red Hat OpenStack Platform Director 13*

# Contrail and AppFormix Kolla/Ocata OpenStack Deployment

## IN THIS CHAPTER

- [Contrail and AppFormix Deployment Requirements | 366](#)
- [Preparing for the Installation | 367](#)
- [Run the Playbooks | 371](#)
- [Accessing Contrail in AppFormix Management Infrastructure in UI | 372](#)
- [Notes and Caveats | 373](#)
- [Example Instances.yml for Contrail and AppFormix OpenStack Deployment | 373](#)
- [Installing AppFormix for OpenStack | 377](#)
- [Installing AppFormix for OpenStack in HA | 382](#)

## Contrail and AppFormix Deployment Requirements

### IN THIS SECTION

- [Software Requirements | 367](#)
- [Hardware Requirements | 367](#)

Starting with Contrail Release 5.0.1, the combined installation of Contrail and AppFormix allows Contrail monitoring by AppFormix. The following topics are referenced for the deployment.

- *Installing Contrail with OpenStack and Kolla Ansible*
- ["Installing AppFormix for OpenStack" on page 377](#)
- ["Installing AppFormix for OpenStack in HA" on page 382](#)

The following software and hardware requirements apply to the combined Contrail and AppFormix deployment.

## Software Requirements

- Contrail Release 5.0.x Targets: Centos 7.5 with kernel 3.10.0-862.3.2.el7.x86\_64.
- AppFormix Targets: Refer to “Software requirements” in ["Installing AppFormix for OpenStack" on page 377](#).
- Targets running both Contrail and AppFormix: CentOS 7.5 Ansible 2.4.2 for the installer.
- AppFormix 2.18.x and later.

## Hardware Requirements

- It is strongly recommended that the AppFormix Controller and Contrail services be installed on separate targets.
- See *Deploying Contrail Cluster using Contrail-Command and instances.yml* and ["Installing AppFormix for OpenStack" on page 377](#) for specifics about requirements for installation.

## RELATED DOCUMENTATION

*Deploying Contrail Cluster using Contrail-Command and instances.yml*

[Installing AppFormix for OpenStack | 377](#)

## Preparing for the Installation

### IN THIS SECTION

- [Preparing the Targets | 368](#)
- [Preparing the Base Host using Ansible Installer | 368](#)
- [TCP/IP Port Conflicts Between Contrail and AppFormix | 369](#)
- [Plugins to Enable for Contrail and AppFormix Deployment | 369](#)
- [Configuring Contrail Monitoring in AppFormix | 369](#)
- [Compute Monitoring: Listing IP Addresses to Monitor | 370](#)

- [Configuring Openstack\\_Controller Hosts for AppFormix | 370](#)
- [Other AppFormix group\\_vars That Must be Enabled in instances.yaml | 370](#)
- [AppFormix License | 370](#)

In Contrail Release 5.0.1, nodes on which Contrail, AppFormix, or both are installed are referred to as *targets*. The host from which Ansible is run is referred to as the *base host*. A *base host* can also be a *target*, meaning you can install either Contrail, AppFormix, or both on a *base host*.

## Preparing the Targets

Workflow for preparing the targets consists of the following steps:

1. Image all the Contrail targets with CentOS 7.5 kernel 3.10.0-862.3.2.el7.x86\_64.
2. Install the necessary platform software on the targets on which AppFormix Controller or AppFormix Agent is going to be installed. See the instructions in ["Installing AppFormix for OpenStack" on page 377](#).

## Preparing the Base Host using Ansible Installer

Workflow for preparing the base host consists of the following steps:

1. Install Ansible 2.4.2 on the base host. See "Set Up the Bare Host" in *Installing Contrail with OpenStack and Kolla Ansible*.
2. Set-up the base host. See "Set Up the Base Host" in *Installing Contrail with OpenStack and Kolla Ansible*. This section includes information about creating the Ansible `instances.yaml` file.
3. On the base host, create a single Ansible `instances.yaml` file that lists inventory for both Contrail and AppFormix deployments. An example of the single `instances.yaml` file is provided later in this section.
  - The Contrail inventory section of the `instances.yaml` file is configured according to guidelines in the section "Set Up the Base Host" in *Installing Contrail with OpenStack and Kolla Ansible*.
  - The AppFormix inventory section of the `instances.yaml` file is configured according to guidelines in ["Installing AppFormix for OpenStack" on page 377](#).



## TCP/IP Port Conflicts Between Contrail and AppFormix

It is strongly recommended that AppFormix Controller and Contrail services be installed on separate target nodes. However, if AppFormix Controller and Contrail services are installed on the same target, the following configuration is required to resolve port conflicts.

The following AppFormix ports must be reconfigured in the AppFormix group-vars section of the `instances.yaml` file.

- `appformix_datamanager_port_http`
- `appformix_datamanager_port_https`
- `appformix_haproxy_datamanager_port_http`
- `appformix_haproxy_datamanager_port_https`
- `appformix_datamanager_port_http:8200`

## Plugins to Enable for Contrail and AppFormix Deployment

Enable the following plugins by including them in the AppFormix group-vars section of the `instances.yaml` file.

```
appformix_plugins: '{{ appformix_contrail_factory_plugins }}' + '{{
  appformix_openstack_factory_plugins }}'
```

## Configuring Contrail Monitoring in AppFormix

Connections to Contrail are configured by providing complete URLs by which to access the analytics and configuration API services.

- `contrail_cluster_name:` **Contrail\_Clusterxxx**

A name by which the Contrail instance will be displayed in the Dashboard. If not specified, this variable has a default value of `default_contrail_cluster`.

- `contrail_analytics_url:` **http://analytics-api-node-ip-address:8081**

URL for the Contrail analytics API. The URL should only specify the protocol, address, and optionally port.

- `contrail_config_url:` **http://contrail-config-api-server-api-address:8082**

URL for the Contrail configuration API. The URL should only specify the protocol, address, and optionally port.



**NOTE:** The IP address specified for contrail monitoring corresponds to one of the IPs listed in the Contrail roles for *config* and *analytics*. Typically, the first active IP address is selected.

## Compute Monitoring: Listing IP Addresses to Monitor

The IP addresses to monitor can be added in the `compute` section of AppFormix in the `instances.yaml` file. A list of IP addresses with a *vrouter* role in the `instances.yaml` file.

## Configuring Openstack\_Controller Hosts for AppFormix

The `Openstack_controller` hosts section must be configured with at least one host. An example section is shown.

```
openstack_controller:
  hosts:
    <ip-address>:
      ansible_connection: ssh
      ansible_ssh_user: <root user>
      ansible_sudo_pass: <contrail password>
```

## Other AppFormix group\_vars That Must be Enabled in instances.yaml

The following `group_vars` must be enabled in `instances.yaml`:

- `openstack_platform_enabled`: **true**
- `appformix_remote_host_monitoring_enabled`: **true**

## AppFormix License

You must have an appropriate license that supports the combined deployment of Contrail with AppFormix for OpenStack. To obtain a license, send an email to “AppFormix-Key-Request@juniper.net”. Also, the following `group_vars` in the `instances.yaml` file must point to this license.

- `appformix_license`: `/path/appformix-contrail-license-file.sig`

This is the path where the license is placed on the *bare host* so that the license can be deployed on the target.

## RELATED DOCUMENTATION

*Installing Contrail with OpenStack and Kolla Ansible*

*Deploying Contrail Cluster using Contrail-Command and instances.yml*

[Installing AppFormix for OpenStack | 377](#)

[Example Instances.yml for Contrail and AppFormix OpenStack Deployment | 373](#)

## Run the Playbooks

Refer to section “Install Contrail and Kolla requirements” and section “Deploying contrail and Kolla containers” in *Installing Contrail with OpenStack and Kolla Ansible* and execute the ansible-playbook.

Following are examples listing the Contrail play-book invocation from the contrail-ansible-deployer directory:

- Configure Contrail OpenStack instances:

```
ansible-playbook -i inventory/ -e config_file=/path/instances.yaml -e
orchestrator=openstack playbooks/configure_instances.yml (-vvv for debug)
```

- Install OpenStack:

```
ansible-playbook -i inventory/ -e config_file=/path/instances.yaml
playbooks/install_openstack.yml
```

- Install Contrail:

```
ansible-playbook -i inventory/ -e config_file=/path/instances.yaml -e
orchestrator=openstack playbooks/install_contrail.yml
```

Source the `/etc/kolla/kolla-toolbox/admin-openrc.sh` file from the OpenStack controller node (`/etc/kolla/kolla-toolbox/ admin-openrc.sh`) to the AppFormix-Controller to authenticate the OpenStack adapter to access admin privileges over controller services. If the OpenStack control node is different from the base

host, either Secure Copy Protocol (SCP) the file over and source it (for example, execute `source /path/admin-openrc.sh`) or manually export the environment enumerated in `/etc/kolla/kolla-toolbox/ admin-openrc.sh` by invoking `export OS_USERNAME=admin` etc. and the remainder as listed in `admin-openrc.sh`

Also at this point, obtain a list of IP addresses to include in the `compute` section of AppFormix in the `instances.yaml` file. Refer to [Compute monitoring: Listing IP addresses to monitor](#) in the `compute` section of AppFormix in the `instances.yaml` file.

Refer to ["Installing AppFormix for OpenStack" on page 377](#) and validate target configuration requirements and inventory parameters for AppFormix Controller and Agent. In place of `-i inventory/use -i /absolute-file-path/instances.yaml`.

Following is an example listing the AppFormix playbook invocation from the AppFormix-2.18.x directory where `appformix_openstack.yaml` is located:

- Install AppFormix:

```
ansible-playbook -i /path/instances.yaml appformix_openstack.yaml (-vvv for debug)
```

## RELATED DOCUMENTATION

*Installing Contrail with OpenStack and Kolla Ansible*

*Deploying Contrail Cluster using Contrail-Command and instances.yaml*

[Installing AppFormix for OpenStack | 377](#)

## Accessing Contrail in AppFormix Management Infrastructure in UI

AppFormix service monitoring Dashboard for a Contrail cluster displays the overall state of the cluster and its components. For more information, see “Dashboard” in “Contrail Monitoring” in the *AppFormix User Guide*.

Open the Dashboard in a Web browser and log in.

`http://controller-IP-address:9000`

## Notes and Caveats

- Versions of AppFormix-2.17 and earlier are not supported with Ansible-2.4.2. The combined Contrail and AppFormix installation is not validated on these earlier releases.
- The installation was validated with AppFormix-2.18 Agent.
- To view and monitor Contrail in the AppFormix Management Infrastructure dashboard, the license used in the deployment must include support for Contrail.
- Verify the datamanager port (re)definitions in the inventory file.
- For AppFormix OpenStack HA installation steps, see ["Installing AppFormix for OpenStack in HA" on page 382](#).

### RELATED DOCUMENTATION

| [Installing AppFormix for OpenStack in HA | 382](#)

## Example Instances.yml for Contrail and AppFormix OpenStack Deployment

See *Installing Contrail with OpenStack and Kolla Ansible* and ["Installing AppFormix for OpenStack" on page 377](#) for specific inventory file details:

The following items are part of the all section in the `instances.yml` file for AppFormix:

```
all:
  children:
    openstack_controller:
      hosts:
        <ip-address>:
          ansible_connection: ssh
          ansible_ssh_user: <ssh-user>
          ansible_sudo_pass: <sudo-password>
```

The following items are part of the vars section in the `instances.yaml` file for AppFormix:

```

openstack_platform_enabled: true
##License must support Contrail and Openstack
appformix_license: /path/license-file.sig
contrail_cluster_name: 'Contrail_Cluster'
contrail_analytics_url: 'http://<contrail-analytics-api-server-ip-address>:8081'
contrail_config_url: 'http://<contrail-config-api-server-ip-address>:8082'
# Defaults from roles/appformix_defaults/defaults/main.yml are overwritten below
appformix_datamanager_port_http: "{{ (appformix_scale_setup_flag|bool) | ternary(28200, 8200) }}"
appformix_datamanager_port_https: "{{ (appformix_scale_setup_flag|bool) | ternary(28201, 8201) }}"
appformix_haproxy_datamanager_port_http: 8200
appformix_haproxy_datamanager_port_https: 8201
appformix_plugins: '{{ appformix_contrail_factory_plugins }} +
{{ appformix_network_device_factory_plugins }}'

```

Following is an example listing of the `instances.yaml`:

There is one `instances.yaml` file for the Contrail and AppFormix combined installation.

```

#Contrail inventory section
provider_config:
  bms:
    ssh_pwd: <ssh-password>
    ssh_user: <ssh-user>
    ntpserver: <ntp-server-ip-address>
    domainsuffix: local
instances:
  bms1:
    provider: bms
    ip: <ip-address>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      vrouter:
      openstack:
      openstack_compute:

```

```

global_configuration:
  CONTAINER_REGISTRY: <ci-repository-URL>:5000
  REGISTRY_PRIVATE_INSECURE: True
contrail_configuration:
  #UPGRADE_KERNEL: true
  CONTRAIL_VERSION: <contrail-version>
  #CONTRAIL_VERSION: latest
  CLOUD_ORCHESTRATOR: openstack
  VROUTER_GATEWAY: <gateway-ip-address>
  RABBITMQ_NODE_PORT: 5673
  PHYSICAL_INTERFACE: <interface-name>
  AUTH_MODE: keystone
  KEYSTONE_AUTH_HOST: <keystone-ip-address>
  KEYSTONE_AUTH_URL_VERSION: /v3
  CONFIG_NODEMGR__DEFAULTS__minimum_diskGB: 2
  DATABASE_NODEMGR__DEFAULTS__minimum_diskGB: 2
kolla_config:
  kolla_globals:
    network_interface: <interface-name>
    kolla_internal_vip_address: <ip-address>
    contrail_api_interface_address: <ip-address>
    enable_haproxy: no
    enable_swift: no
  kolla_passwords:
    keystone_admin_password: <password>

# Appformix inventory section
all:
  children:
    appformix_controller:
      hosts:
        <ip-address>:
          ansible_connection: ssh
          ansible_ssh_user: <ssh-user>
          ansible_sudo_pass: <sudo-password>
    openstack_controller:
      hosts:
        <ip-address>:
          ansible_connection: ssh
          ansible_ssh_user: <ssh-user>
          ansible_sudo_pass: <sudo-password>
  compute:
    hosts:

```

```

#List IP addresses of Contrail roles to be monitored here
<<IP-addresses>>:
  ansible_connection: ssh
  ansible_ssh_user: <ssh-user>
  ansible_sudo_pass: <sudo-password>
bare_host:
  hosts:
    <ip-address>:
      ansible_connection: ssh
      ansible_ssh_user: <ssh-user>
      ansible_sudo_pass: <sudo-password>
    #If host is local
    <ip-address>:
      ansible_connection: local
vars:
  appformix_docker_images:
    - /opt/software/appformix/appformix-platform-images-<version>.tar.gz
    - /opt/software/appformix/appformix-dependencies-images-<version>.tar.gz
    - /opt/software/appformix/appformix-network_device-images-<version>.tar.gz
    - /opt/software/appformix/appformix-openstack-images-<version>.tar.gz
  openstack_platform_enabled: true
  # appformix_license: /opt/software/openstack_appformix/<appformix-contrail-license-file>.sig
  appformix_license: /opt/software/configs/contrail.sig
  appformix_docker_registry: registry.appformix.com/
  appformix_version: <version>      #Must be 2.18.x or above
  appformix_plugins: '{{ appformix_contrail_factory_plugins }}' +
  '{{ appformix_network_device_factory_plugins }}' + '{{ appformix_openstack_factory_plugins }}'
  appformix_kvm_instance_discovery: true
  # For enabling pre-requisites for package installation
  appformix_network_device_monitoring_enabled: true
  # For running the appformix-network-device-adapter
  network_device_discovery_enabled: true
  appformix_remote_host_monitoring_enabled: true
  appformix_jti_network_device_monitoring_enabled: true
  contrail_cluster_name: 'Contrail_Cluster'
  contrail_analytics_url: 'http://<contrail-analytics-api-server-IP-address>:8081'
  contrail_config_url: 'http://<contrail-config-api-server-IP-address>:8082'
  # Defaults overwritten below were defined in roles/appformix_defaults/defaults/main.yml
  appformix_datamanager_port_http: '{{ (appformix_scale_setup_flag|bool) | ternary(28200,
8200) }}'
  appformix_datamanager_port_https: '{{ (appformix_scale_setup_flag|bool) | ternary(28201,
8201) }}'

```



```
appformix_haproxy_datamanager_port_http: 8200
appformix_haproxy_datamanager_port_https: 8201
```



**NOTE:** Replace `<contrail_version>` with the correct `contrail_container_tag` value for your Contrail release. The respective `contrail_container_tag` values are listed in [README Access to Contrail Registry](#).

## RELATED DOCUMENTATION

*Installing Contrail with OpenStack and Kolla Ansible*

*Deploying Contrail Cluster using Contrail-Command and instances.yml*

[Installing AppFormix for OpenStack | 377](#)

## Installing AppFormix for OpenStack

### IN THIS SECTION

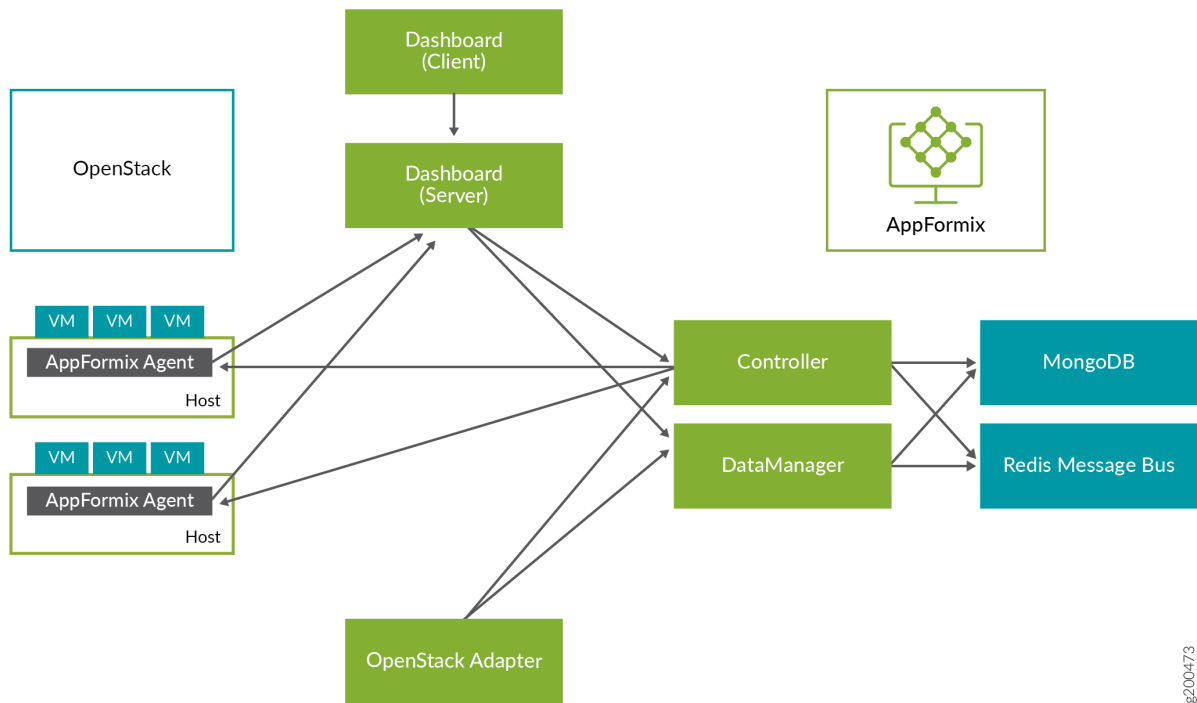
- [Architecture | 377](#)
- [Installing AppFormix | 378](#)
- [Removing a Node from AppFormix | 382](#)

AppFormix provides resource control and visibility for hosts and virtual machines in an OpenStack environment. This topic explains how to install AppFormix for OpenStack.

### Architecture

AppFormix provides resource control and visibility for hosts, containers, and virtual machines in your cloud infrastructure. [Figure 43 on page 378](#) shows the AppFormix architecture with OpenStack.

Figure 43: AppFormix Architecture with OpenStack



8200-473

- Agent monitors resource usage on the compute nodes.
- Controller offers REST APIs to configure the system.
- DataManager stores data from multiple Agents.
- Dashboard provides a Web-based user interface.
- An adapter discovers platform-specific resources and configures the AppFormix Controller.
- Adapters exist for OpenStack, Kubernetes, and Amazon EC2.

## Installing AppFormix

To install AppFormix:

1. Install Ansible on the AppFormix Controller node. Ansible will install docker and docker-py on the controller.

```

apt-get install python-pip python-dev      #Installs Pip

pip install ansible==2.3                  #Installs Ansible 2.3

```

```
sudo apt-get install build-essential libssl-dev libffi-dev    #Dependencies

pip install markupsafe httpplib2                            #Dependencies
```

2. On the vRouter compute nodes where AppFormix Agent runs verify that python virtualenv is installed.

```
apt-get install -y python-pip

pip install virtualenv
```

3. Enable passwordless login to facilitate AppFormix Controller node with Ansible to install agents on the nodes. Run the same command on the AppFormix Controller node also.

```
ssh-keygen -t rsa    #Creates Keys

ssh-copy-id -i ~/.ssh/id_rsa.pub <target_host>    #Copies key from the node to other hosts
```

4. Use the Sample\_Inventory file as a template to create a host file.

```
# Example naming schemes are as below:
#   hostname ansible_ssh_user='username' ansible_sudo_pass='password'

# List all Compute Nodes
[compute]
203.0.113.5
203.0.113.17

# AppFormix controller host
#
# Host variables can be defined to control AppFormix configuration parameters
# for particular host. For example, to specify the directory in which MongoDB
# data is stored on hostname1 (the default is /opt/appformix/mongo/data):
#
#   hostname1 appformix_mongo_data_dir=/var/lib/appformix/mongo
#
# For variables with same value for all AppFormix controller hosts, set group
# variables below.
#
```

```
[appformix_controller]
203.0.113.119
```

5. Verify that all the hosts listed in the inventory file are reachable from the AppFormix Controller.

```
export ANSIBLE_HOST_KEY_CHECKING=False # Eliminates interactive experience prompting for
Known_Hosts

ansible -i inventory -m ping all          # Pings all the hosts in the inventory file
```

6. At the top-level of the distribution, create a directory named `group_vars`.

```
mkdir group_vars
```

7. Every installation requires an authorized license file and Docker images. In `group_vars` directory, create a file named `all`. Add the following:

```
openstack_platform_enabled: true

appformix_version: <version>
appformix_manager_version: <version>
appformix_license: path/to/appformix-license-file.sig          # Location of License
Provided

appformix_docker_images:
  - /path/to/appformix-platform-images-<version>.tar.gz
  - /path/to/appformix-dependencies-images-<version>.tar.gz
  - /path/to/appformix-openstack-images-<version>.tar.gz
```

8. Enable the Appformix Certified\_Plugins in the file named `all` in the `group_vars` directory and add the following:

```
appformix_plugins: '{{ appformix_contrail_factory_plugins }}' +
{{ appformix_network_device_factory_plugins }} + {{ appformix_openstack_factory_plugins }}
+ {{ appformix_application_factory_plugins }} + {{ appformix_remote_host_factory_plugins }}
+ {{ appformix_network_device_factory_juniper_plugins }}'

appformix_network_device_monitoring_enabled: true
```

```
appformix_remote_host_monitoring_enabled: true
appformix_jti_network_device_monitoring_enabled: true
```

9. Source the openrc file from the OpenStack controller node (/etc/contrail/openstackrc) to the AppFormix Controller to authenticate the adapter to access admin privileges over the controller services.

```
export OS_USERNAME=<admin user>
export OS_PASSWORD=<password>
export OS_AUTH_URL=http://<openstack-auth-URL>/v2.0/
export OS_NO_CACHE=1
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

10. Add the username and password for credentials-based login.

Constraints for creating Username:

- Should not be more than 30 characters
- Can have anything mentioned below:
  1. alphanumeric character
  2. '\_' or '.'

Constraints for creating Password:

- 8 characters length or more
- 1 digit or more
- 1 uppercase letter or more
- 1 lowercase letter or more

```
export APPFORMIX_USERNAME=<username>
export APPFORMIX_PASSWORD=<password>
```

11. Run Ansible with the created inventory file.

```
ansible-playbook -i inventory appformix_openstack.yml
```

## Removing a Node from AppFormix

To remove a node from AppFormix:

1. Edit the inventory file and add `appformix_state=absent` to each node that you want to remove from AppFormix.

```
# Example naming schemes are as below:
#  hostname ansible_ssh_user='username' ansible_sudo_pass='password'

# List all Compute Nodes
[compute]
203.0.113.5 appformix_state=absent
203.0.113.17
```

2. Run Ansible with the edited inventory file. This will remove the node and all its resources from AppFormix.

```
ansible-playbook -i inventory appformix_openstack.yml
```

## Installing AppFormix for OpenStack in HA

### IN THIS SECTION

- [HA Design Overview | 382](#)
- [Requirements | 383](#)
- [Installing AppFormix for High Availability | 384](#)

### HA Design Overview

AppFormix Platform can be deployed to multiple hosts for high availability. Platform services continue to communicate using an API proxy that listens on a virtual IP address. Only one host will have the virtual IP at a time, and so only one API proxy will be the “active” API proxy at a time.

The API proxy is implemented by HAProxy. HAProxy is configured to use services in active-standby or load-balanced active-active mode, depending on the service.

At most, one host will be assigned the virtual IP at any given time. This host is considered the “active” HAProxy. The virtual IP address is assigned to a host by keepalived, which uses VRRP protocol for election.

Services are replicated in different modes of operation. In the “active-passive” mode, HAProxy sends all requests to a single “active” instance of a service. If the service fails, then HAProxy will select a new “active” from the other hosts, and begin to send requests to the new “active” service. In the “active-active” mode, HAProxy load balances requests across hosts on which a service is operational.

AppFormix Platform can be deployed in a 3-node, 5-node, or 7-node configuration for high availability.

## Requirements

Each host, on which AppFormix Platform is installed, has the following requirements.

### Hardware Requirements

- CPU: 8 cores (virtual or physical)
- Memory: 16 GB
- Storage: 100 GB (recommended)

### Software Requirements

- docker 17.03.1-ce
- docker-py 1.3.1
- Ansible 1.9.6, or 2.3, httpplib2

### Connectivity

- One virtual IP address to be shared among all the Platform Hosts. This IP address should not be used by any host before installation. It should have reachability from all the Platform Hosts after installation.
- Dashboard client (in browser) must have IP connectivity to the virtual IP.
- IP addresses for each Platform Host for installation and for services running on these hosts to communicate.

- `keepalived_vrrp_interface` for each Platform Host which would be used for assigning virtual IP address. Details on how to configure this interface is described in the `sample_inventory` section.
- The installer node needs to download the following packages from <https://www.juniper.net/support/downloads/?p=appformix#sw>.
  - `appformix-openstack-images- <version>.tar.gz`
  - `appformix-platform-images- <version>.tar.gz`
  - `appformix-dependencies-images- <version>.tar.gz`

## AppFormix Agent Supported Platforms

AppFormix Agent runs on a host to monitor resource consumption of the host itself and the virtual machines and containers executing on that host.

- Ubuntu 14.04
- Red Hat Enterprise Linux 7.1
- Red Hat Enterprise Linux 6.5, 6.6
- CentOS 7.1
- CentOS 6.5, 6.6

## Installing AppFormix for High Availability

To install AppFormix to multiple hosts for high availability:

1. Install Ansible on the installer node. Ansible will install docker and docker-py on the `appformix_controller`.

```
# sudo apt-get install python-pip python-dev build-essential libssl-dev libffi-dev
# sudo pip install ansible==1.9.6 markupsafe httpplib2
```

For Ansible 2.3:

```
# sudo pip install ansible==2.3 markupsafe httpplib2 cryptography==1.5
```



2. Install python and python-pip on all the Platform Hosts so that Ansible can run between the installer node and the appformix\_controller node.

```
# sudo apt-get install -y python python-pip
```

3. Install python pip package on the hosts where AppFormix Agents run.

```
# apt-get install -y python-pip
```

4. To enable passwordless login to all Platform Hosts by Ansible, create an SSH public key on the node where Ansible playbooks are run and then copy the key to all the Platform Hosts.

```
# ssh-keygen -t rsa                                #Creates Keys
# ssh-copy-id -i ~/.ssh/id_rsa.pub <platform_host_1>.....#Copies key from the node to all
platform hosts
# ssh-copy-id -i ~/.ssh/id_rsa.pub <platform_host_2>.....#Copies key from the node to all
platform hosts
# ssh-copy-id -i ~/.ssh/id_rsa.pub <platform_host_3>.....#Copies key from the node to all
platform hosts
```

5. Use the sample\_inventory file as a template to create a host file. Add all the Platform Hosts and compute hosts details.

```
# List all compute hosts which needs to be monitored by AppFormix
[compute]
203.0.113.5
203.0.113.17
# AppFormix controller hosts
[appformix_controller]
203.0.113.119 keepalived_vrrp_interface=eth0
203.0.113.120 keepalived_vrrp_interface=eth0
203.0.113.121 keepalived_vrrp_interface=eth0
```



**NOTE:** Note: In the case of 5-node or 7-node deployment, list all the nodes under appformix\_controller.

6. At top-level of the distribution, create a directory named `group_vars` and then create a file named `all` inside this directory.

```
# mkdir group_vars
# touch group_vars/all
```

Add the following entries to the newly created `all` file:

```
appformix_vip: <ip-address>
appformix_docker_images:
- /path/to/appformix-platform-images-<version>.tar.gz
- /path/to/appformix-dependencies-images-<version>.tar.gz
- /path/to/appformix-openstack-images-<version>.tar.gz
```

7. Copy and source the `openrc` file from the OpenStack controller node (`/etc/contrail/openrc`) to the AppFormix Controller to authenticate the adapter to access admin privileges over the controller services.

```
root@installer_node:~# cat /etc/contrail/openrc
export OS_USERNAME=<admin user>
export OS_PASSWORD=<password>
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://<openstack-auth-URL>/v2.0/
export OS_NO_CACHE=1
root@installer_node:~# source /etc/contrail/openrc
```

8. Run Ansible with the created inventory file.

```
ansible-playbook -i inventory appformix_openstack.yml
```

9. If running the playbooks as root user then this step can be skipped. As a non-root user (for example, “ubuntu”), the user “ubuntu” needs access to the docker user group. The following command adds the user to the docker group.

```
sudo usermod -aG docker ubuntu
```



**NOTE:** If step 8. is being done with offline installation and failed due to step 8. not being done, then the appformix \*.tar.gz need to be removed from the /tmp/ folder on the appformix\_controller node. This is the workaround required as of version 2.11.1.

# Using Contrail with Juju Charms

## IN THIS CHAPTER

- [Deploying Contrail by Using Juju Charms | 388](#)

## Deploying Contrail by Using Juju Charms

### IN THIS SECTION

- [Preparing to Deploy Contrail by Using Juju Charms | 389](#)
- [Deploying Contrail Charms | 391](#)
- [Options for Juju Charms | 403](#)

You can deploy Contrail by using Juju Charms. Juju helps you deploy, configure, and efficiently manage applications on private clouds and public clouds. Juju accesses the cloud with the help of a Juju controller. A Charm is a module containing a collection of scripts and metadata and is used with Juju to deploy Contrail.

Contrail supports the following charms:

- `contrail-agent`
- `contrail-analytics`
- `contrail-analyticsdb`
- `contrail-controller`
- `contrail-keystone-auth`
- `contrail-openstack`

These topics describe how to deploy Contrail by using Juju Charms.

## Preparing to Deploy Contrail by Using Juju Charms

Follow these steps to prepare for deployment:

### 1. Install Juju.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install juju
```

### 2. Configure Juju.

You can add a cloud to Juju, identify clouds supported by Juju, and also manage clouds already added to Juju.

- **Adding a cloud**—Juju recognizes a wide range of cloud types. You can use any one of the following methods to add a cloud to Juju:
  - **Adding a Cloud by Using Interactive Command**

*Example: Adding an MAAS cloud to Juju*

```
juju add-cloud
Cloud Types
  maas
  manual
  openstack
  oracle
  vsphere

Select cloud type: maas

Enter a name for your maas cloud: maas-cloud

Enter the API endpoint url: http://<ip-address>:<node>/MAAS

Cloud "maas-cloud" successfully added
You may bootstrap with 'juju bootstrap maas-cloud'
```



**NOTE:** Juju 2.x is compatible with MAAS series 1.x and 2.x.

- **Adding a Cloud Manually**

You use a YAML configuration file to add a cloud manually. Enter the following command:

```
juju add-cloud <cloud-name>
juju add-credential <cloud name>
```

For an example, to add the cloud *junmaas*, assuming that the name of the configuration file in the directory is *maas-clouds.yaml*, you run the following command:

```
juju add-cloud junmaas maas-clouds.yaml
```

The following is the format of the YAML configuration file:

```
clouds:
  <cloud_name>:
    type: <type_of_cloud>
    auth-types: [<authentication_types>]
    regions:
      <region-name>:
        endpoint: <http://<ip-address>:<node>/MAAS>
```



**NOTE:** The auth-types for a MAAS cloud type is *oauth1*.

- **Identifying a supported cloud**

Juju recognizes the cloud types given below. You use the `juju clouds` command to list cloud types that are supported by Juju.


```
$ juju clouds
```

Cloud	Regions	Default	Type	Description
aws	15	us-east-1	ec2	Amazon Web Services
aws-china	1	cn-north-1	ec2	Amazon China
aws-gov	1	us-gov-west-1	ec2	Amazon (USA Government)
azure	26	centralus	azure	Microsoft Azure
azure-china	2	chinaeast	azure	Microsoft Azure China
cloudsigma	5	hnl	cloudsigma	CloudSigma Cloud
google	13	us-east1	gce	Google Cloud Platform
joyent	6	eu-ams-1	joyent	Joyent Cloud

oracle	5	uscom-central-1	oracle	Oracle Cloud
rackspace	6	dfw	rackspace	Rackspace Cloud
localhost	1	localhost	lxd	LXD Container Hypervisor

3. Create a Juju controller.



```
juju bootstrap --bootstrap-series=xenial <cloud name> <controller name>
```



**NOTE:** A Juju controller manages and keeps track of applications in the Juju cloud environment.

### Deploying Contrail Charms

**IN THIS SECTION**

- 
[Deploying Contrail Charms in a Bundle | 391](#)
- 
[Deploying Juju Charms Manually | 398](#)

You can deploy Contrail Charms in a bundle or manually.

#### Deploying Contrail Charms in a Bundle

Follow these steps to deploy Contrail Charms in a bundle.

1. Deploy Contrail Charms.

To deploy Contrail Charms in a bundle, use the `juju deploy <bundle_yaml_file>` command.

The following example shows you how to use **bundle\_yaml\_file** to deploy Contrail on Amazon Web Services (AWS) Cloud.

```
series: xenial
services:
  ubuntu:
    charm: cs:xenial/ubuntu
    num_units: 3
    to: [ "1", "2", "3" ]
```

```

ntp:
  charm: cs:xenial/ntp
  num_units: 0
  options:
    source: ntp.juniper.net
mysql:
  charm: cs:xenial/percona-cluster
  options:
    dataset-size: 15%
    max-connections: 10000
    root-password: password
    sst-password: password
    vip: ip-address
    vip_cidr: 24
  num_units: 3
  to: [ "lxd:1", "lxd:2", "lxd:3" ]
rabbitmq-server:
  charm: cs:xenial/rabbitmq-server
  num_units: 3
  to: [ "lxd:1", "lxd:2", "lxd:3" ]
heat:
  charm: cs:xenial/heat
  num_units: 3
  options:
    vip: ip-address
    vip_cidr: 24
  to: [ "lxd:1", "lxd:2", "lxd:3" ]
keystone:
  charm: cs:xenial/keystone
  options:
    admin-password: password
    admin-role: admin
    openstack-origin: cloud:xenial-newton
    vip: ip-address
    vip_cidr: 24
  num_units: 3
  to: [ "lxd:1", "lxd:2", "lxd:3" ]
nova-cloud-controller:
  charm: cs:xenial/nova-cloud-controller
  options:
    network-manager: Neutron
    openstack-origin: cloud:xenial-newton
    vip: ip-address

```



```

    vip_cidr: 24
    num_units: 3
    to: [ "lxd:1", "lxd:2", "lxd:3" ]
neutron-api:
    charm: cs:xenial/neutron-api
    series: xenial
    options:
        manage-neutron-plugin-legacy-mode: false
        openstack-origin: cloud:xenial-newton
        vip: ip-address
        vip_cidr: 24
    num_units: 3
    to: [ "lxd:1", "lxd:2", "lxd:3" ]
glance:
    charm: cs:xenial/glance
    options:
        openstack-origin: cloud:xenial-newton
        vip: ip-address
        vip_cidr: 24
    num_units: 3
    to: [ "lxd:1", "lxd:2", "lxd:3" ]
openstack-dashboard:
    charm: cs:xenial/openstack-dashboard
    options:
        openstack-origin: cloud:xenial-newton
        vip: ip-address
        vip_cidr: 24
    num_units: 3
    to: [ "lxd:1", "lxd:2", "lxd:3" ]
nova-compute:
    charm: cs:xenial/nova-compute
    options:
        openstack-origin: cloud:xenial-newton
    num_units: 3
    to: [ "4", "5", "6" ]
mysql-hacluster:
    charm: cs:xenial/hacluster
    options:
        cluster_count: 3
    num_units: 0
keystone-hacluster:
    charm: cs:xenial/hacluster
    options:

```

```

        cluster_count: 3
        num_units: 0
ncc-hacluster:
    charm: cs:xenial/hacluster
    options:
        cluster_count: 3
        num_units: 0
neutron-hacluster:
    charm: cs:xenial/hacluster
    options:
        cluster_count: 3
        num_units: 0
glance-hacluster:
    charm: cs:xenial/hacluster
    options:
        cluster_count: 3
        num_units: 0
dashboard-hacluster:
    charm: cs:xenial/hacluster
    options:
        cluster_count: 3
        num_units: 0
heat-hacluster:
    charm: cs:xenial/hacluster
    options:
        cluster_count: 3
        num_units: 0
contrail-openstack:
    charm: cs:~juniper-os-software/contrail-openstack
    series: xenial
    num_units: 0
contrail-agent:
    charm: cs:~juniper-os-software/contrail-agent
    num_units: 0
    series: xenial
    options:
        log-level: "SYS_DEBUG"
contrail-analytics:
    charm: cs:~juniper-os-software/contrail-analytics
    num_units: 3
    series: xenial
    to: [ "1", "2", "3" ]
contrail-analyticsdb:

```

```

charm: cs:~juniper-os-software/contrail-analyticsdb
num_units: 3
series: xenial
options:
  log-level: "SYS_DEBUG"
  cassandra-minimum-diskgb: 4
  cassandra-jvm-extra-opts: "-Xms1g -Xmx2g"
  to: [ "1", "2", "3" ]
contrail-controller:
charm: cs:~juniper-os-software/contrail-controller
series: xenial
options:
  vip: ip-address
  log-level: "SYS_DEBUG"
  cassandra-minimum-diskgb: 4
  cassandra-jvm-extra-opts: "-Xms1g -Xmx2g"
  to: [ "1", "2", "3" ]
contrail-keystone-auth:
charm: cs:~juniper-os-software/contrail-keystone-auth
series: xenial
num_units: 1
to: [ "lxd:1" ]

contrail-keepalived:
charm: cs:~boucherv29/keepalived-19
series: xenial
options:
  virtual_ip: ip-address
contrail-haproxy:
charm: haproxy
series: xenial
expose: true
options:
  peering_mode: "active-active"
  to: [ "1", "2", "3" ]

relations:
# openstack
- [ "ubuntu", "ntp" ]
- [ mysql, mysql-hacluster ]
- [ "keystone", "mysql" ]
- [ keystone, keystone-hacluster ]
- [ "glance", "mysql" ]

```

```

- [ "glance", "keystone" ]
- [ glance, glance-hacluster ]
- [ "nova-cloud-controller", "mysql" ]
- [ "nova-cloud-controller", "rabbitmq-server" ]
- [ "nova-cloud-controller", "keystone" ]
- [ "nova-cloud-controller", "glance" ]
- [ nova-cloud-controller, ncc-hacluster ]
- [ "neutron-api", "mysql" ]
- [ "neutron-api", "rabbitmq-server" ]
- [ "neutron-api", "nova-cloud-controller" ]
- [ "neutron-api", "keystone" ]
- [ neutron-api, neutron-hacluster ]
- [ "nova-compute:amqp", "rabbitmq-server:amqp" ]
- [ "nova-compute", "glance" ]
- [ "nova-compute", "nova-cloud-controller" ]
- [ "nova-compute", "ntp" ]
- [ "openstack-dashboard:identity-service", "keystone" ]
- [ openstack-dashboard, dashboard-hacluster ]
- [ "heat", "mysql" ]
- [ "heat", "rabbitmq-server" ]
- [ "heat", "keystone" ]
- [ "heat", "heat-hacluster" ]

#contrail
- [ "contrail-keystone-auth", "keystone" ]
- [ "contrail-controller", "contrail-keystone-auth" ]
- [ "contrail-analytics", "contrail-analyticsdb" ]
- [ "contrail-controller", "contrail-analytics" ]
- [ "contrail-controller", "contrail-analyticsdb" ]
- [ "contrail-openstack", "nova-compute" ]
- [ "contrail-openstack", "neutron-api" ]
- [ "contrail-openstack", "heat" ]
- [ "contrail-openstack", "contrail-controller" ]
- [ "contrail-agent:juju-info", "nova-compute:juju-info" ]
- [ "contrail-agent", "contrail-controller" ]

#haproxy
- [ "haproxy:juju-info", "keepalived:juju-info" ]
- [ "contrail-analytics", "haproxy" ]
- [ "contrail-controller:http-services", "haproxy" ]
- [ "contrail-controller:https-services", "haproxy" ]

```

machines:

```

"1":
  series: xenial
  #constraints: mem=15G root-disk=40G
  constraints: tags=contrail-controller-vm-1
"2":
  series: xenial
  #constraints: mem=15G root-disk=40G
  constraints: tags=contrail-controller-vm-2
"3":
  series: xenial
  #constraints: mem=15G root-disk=40G
  constraints: tags=contrail-controller-vm-3
"4":
  series: xenial
  #constraints: mem=4G root-disk=20G
  constraints: tags=compute-storage-1
"5":
  series: xenial
  #constraints: mem=4G root-disk=20G
  constraints: tags=compute-storage-2
"6":
  series: xenial
  #constraints: mem=4G root-disk=20G
  constraints: tags=compute-storage-3

```

You can create or modify the Contrail Charm deployment bundle YAML file to:

- Point to machines or instances where the Contrail Charms must be deployed.
- Include the options you need.

Each Contrail Charm has a specific set of options. The options you choose depend on the charms you select. For more information on the options that are available, see *Options for Juju Charms*.

## 2. (Optional) Check the status of deployment.

You can check the status of the deployment by using the `juju status` command.

## 3. Enable configuration statements.

Based on your deployment requirements, you can enable the following configuration statements:

- `contrail-agent`

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-agent/>.

- `contrail-analytics`

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-analytics>.

- contrail-analyticsdb

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-analyticsdb>.

- contrail-controller

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-controller>.

- contrail-keystone-auth

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-keystone-auth>.

- contrail-openstack

For more information see, <https://jaas.ai/u/juniper-os-software/contrail-openstack>.

## Deploying Juju Charms Manually

Before you begin deployment, ensure that you have:

- Installed and configured Juju
- Created a Juju controller
- Ubuntu 16.04 or Ubuntu 18.04 installed

Follow these steps to deploy Juju Charms manually:

1. Create machine instances for OpenStack, compute, and Contrail.

```
juju add-machine --constraints mem=8G cores=2 root-disk=40G --series=xenial #for openstack
machine(s) 0
juju add-machine --constraints mem=7G cores=4 root-disk=40G --series=xenial #for compute
machine(s) 1,(3)
juju add-machine --constraints mem=15G cores=2 root-disk=300G --series=xenial #for contrail
machine 2
```

2. Deploy OpenStack services.

You can deploy OpenStack services by using any one of the following methods:

- **By specifying the OpenStack parameters in a YAML file**

The following is an example of a YAML-formatted (`nova-compute-config.yaml`) file.

```
nova-compute:
  openstack-origin: cloud:xenial-ocata
  virt-type: qemu
  enable-resize: True
  enable-live-migration: True
  migration-auth-type: ssh
```

Use this command to deploy OpenStack services by using a YAML-formatted file:

```
juju deploy cs:xenial/nova-compute --config ./nova-compute-config.yaml
```

- **By using CLI**

To deploy OpenStack services through the CLI:

```
juju deploy cs:xenial/nova-cloud-controller --config console-access-protocol=novnc --
config openstack-origin=cloud:xenial-ocata
```

- **By using a combination of YAML-formatted file and CLI**

To deploy OpenStack services by using a combination of YAML-formatted file and CLI:



**NOTE:** Use the `--to <machine number>` command to point to a machine or container where you want the application to be deployed.

```
juju deploy cs:xenial/ntp
juju deploy cs:xenial/rabbitmq-server --to lxd:0
juju deploy cs:xenial/percona-cluster mysql --config root-password=<root-password> --
config max-connections=1500 --to lxd:0
juju deploy cs:xenial/openstack-dashboard --config openstack-origin=cloud:xenial-ocata --
to lxd:0
juju deploy cs:xenial/nova-cloud-controller --config console-access-protocol=novnc --
config openstack-origin=cloud:xenial-ocata --config network-manager=Neutron --to lxd:0
juju deploy cs:xenial/neutron-api --config manage-neutron-plugin-legacy-mode=false --
config openstack-origin=cloud:xenial-ocata --config neutron-security-groups=true --to lxd:0
juju deploy cs:xenial/glance --config openstack-origin=cloud:xenial-ocata --to lxd:0
```

```
juju deploy cs:xenial/keystone --config admin-password=<admin-password> --config admin-
role=admin --config openstack-origin=cloud:xenial-ocata --to lxd:0
```



**NOTE:** You set OpenStack services on different machines or on different containers to prevent HAProxy conflicts from applications.

### 3. Deploy and configure nova-compute.

```
juju deploy cs:xenial/nova-compute --config ./nova-compute-config.yaml --to 1
```



**NOTE:** You can deploy nova-compute to more than one compute machine.

(Optional) To add additional computes:

```
juju add-unit nova-compute --to 3 # Add one more unit
```

### 4. Deploy and configure Contrail services.

```
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-keystone-auth --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-controller --config
auth-mode=rbac --config cassandra-minimum-diskgb=4 --config cassandra-jvm-extra-opts="-Xms1g -
Xmx2g" --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-analyticsdb cassandra-
minimum-diskgb=4 --config cassandra-jvm-extra-opts="-Xms1g -Xmx2g" --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-analytics --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-openstack
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-agent
```

### 5. Enable applications to be available to external traffic:

```
juju expose openstack-dashboard
juju expose nova-cloud-controller
juju expose neutron-api
juju expose glance
juju expose keystone
```



6. Enable contrail-controller and contrail-analytics services to be available to external traffic if you do not use HAProxy.

```
juju expose contrail-controller
juju expose contrail-analytics
```

7. Apply SSL.

You can apply SSL if needed. To use SSL with Contrail services, deploy easy-rsa service and add-relation command to create relations to contrail-controller service and contrail-agent services.

```
juju deploy cs:~containers/xenial/easyrsa --to 0
juju add-relation easyrsa contrail-controller
juju add-relation easyrsa contrail-agent
```

8. (Optional) HA configuration.

If you use more than one controller, follow the HA solution given below:

- a. Deploy HAProxy and Keepalived services.

HAProxy charm is deployed on machines with Contrail controllers. HAProxy charm must have `peering_mode` set to active-active. If `peering_mode` is set to active-passive, HAProxy creates additional listeners on the same ports as other Contrail services. This leads to port conflicts.

Keepalived charm does not require to option.

```
juju deploy cs:xenial/haproxy --to <first contrail-controller machine> --config
peering_mode=active-active
juju add-unit haproxy --to <another contrail-controller machine>
juju deploy cs:~boucherv29/keepalived-19 --config virtual_ip=<vip>
```

- b. Enable HAProxy to be available to external traffic.

```
juju expose haproxy
```



**NOTE:** If you enable HAProxy to be available to external traffic, do not follow step 6.

c. Add HAProxy and Keepalived relations.

```
juju add-relation haproxy:juju-info keepalived:juju-info
juju add-relation contrail-analytics:http-services haproxy
juju add-relation contrail-controller:http-services haproxy
juju add-relation contrail-controller:https-services haproxy
```

d. Configure contrail-controller service with VIP.

```
juju set contrail-controller vip=<vip>
```

9. Add other necessary relations.

```
juju add-relation keystone:shared-db mysql:shared-db
juju add-relation glance:shared-db mysql:shared-db
juju add-relation keystone:identity-service glance:identity-service
juju add-relation nova-cloud-controller:image-service glance:image-service
juju add-relation nova-cloud-controller:identity-service keystone:identity-service
juju add-relation nova-cloud-controller:cloud-compute nova-compute:cloud-compute
juju add-relation nova-compute:image-service glance:image-service
juju add-relation nova-compute:amqp rabbitmq-server:amqp
juju add-relation nova-cloud-controller:shared-db mysql:shared-db
juju add-relation nova-cloud-controller:amqp rabbitmq-server:amqp
juju add-relation openstack-dashboard:identity-service keystone

juju add-relation neutron-api:shared-db mysql:shared-db
juju add-relation neutron-api:neutron-api nova-cloud-controller:neutron-api
juju add-relation neutron-api:identity-service keystone:identity-service
juju add-relation neutron-api:amqp rabbitmq-server:amqp

juju add-relation contrail-controller ntp
juju add-relation nova-compute:juju info ntp:juju info

juju add-relation contrail-controller contrail-keystone-auth
juju add-relation contrail-keystone-auth keystone
juju add-relation contrail-controller contrail-analytics
juju add-relation contrail-controller contrail-analyticsdb
juju add-relation contrail-analytics contrail-analyticsdb

juju add-relation contrail-openstack neutron-api
juju add-relation contrail-openstack nova-compute
```

```
juju add-relation contrail-openstack contrail-controller

juju add-relation contrail-agent:juju info nova-compute:juju info
juju add-relation contrail-agent contrail-controller
```

## Options for Juju Charms

Each Contrail Charm has a specific set of options. The options you choose depend on the charms you select. The following tables list the various options you can choose:

- Options for **contrail-agent** Charms.

**Table 10: Options for contrail-agent**

Option	Default option	Description
physical-interface		Specify the interface where you want to install vhost0 on. If you do not specify an interface, vhost0 is installed on the default gateway interface.
vhost-gateway	auto	Specify the gateway for vhost0. You can enter either an IP address or the keyword (auto) to automatically set a gateway based on the existing vhost routes.
remove-juju-bridge	true	To install vhost0 directly on the interface, enable this option to remove any bridge created to deploy LXD/LXC and KVM workloads.
dpdk	false	Specify DPDK vRouter.
dpdk-driver	uio_pci_generic	Specify DPDK driver for the physical interface.
dpdk-hugepages	70%	Specify the percentage of huge pages reserved for DPDK vRouter and OpenStack instances.
dpdk-coremask	1	Specify the vRouter CPU affinity mask to determine on which CPU the DPDK vRouter will run.

**Table 10: Options for contrail-agent (Continued)**

Option	Default option	Description
dpgk-main-mempool-size		Specify the main packet pool size.
dpgk-pmd-td-size		Specify the DPDK PMD Tx Descriptor size.
dpgk-pmd-rxd-size		Specify the DPDK PMD Rx Descriptor size.
docker-registry	opencontrailnightly	Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag	latest	Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services.  Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.
no_proxy		Specify the list of destinations that must be directly accessed.

- Options for **contrail-analytics** Charms.

**Table 11: Options for contrail-analytics**

Option	Default option	Description
control-network		Specify the IP address and network mask of the control network.
docker-registry		Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag		Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services.  Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.
no_proxy		Specify the list of destinations that must be directly accessed.

- Options for **contrail-analyticsdb** Charms.

**Table 12: Options for contrail-analyticsdb**

Option	Default option	Description
control-network		Specify the IP address and network mask of the control network.
cassandra-minimum-diskgb	256	Specify the minimum disk requirement.
cassandra-jvm-extra-opts		Specify the memory limit.
docker-registry		Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag		Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services.  Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.
no_proxy		Specify the list of destinations that must be directly accessed.

- Options for **contrail-controller** Charms.

Table 13: Options for contrail-controller

Option	Default option	Description
control-network		Specify the IP address and network mask of the control network.
auth-mode	rbac	Specify the authentication mode.  Options: rbac, cloud-admin, no-auth.  For more information, see <a href="https://github.com/Juniper/contrail-controller/wiki/RBAC">https://github.com/Juniper/contrail-controller/wiki/RBAC</a> .
cassandra-minimum-diskgb	20	Specify the minimum disk requirement.
cassandra-jvm-extra-opts		Specify the memory limit.
cloud-admin-role	admin	Specify the role name in keystone for users who have admin-level access.
global-read-only-role		Specify the role name in keystone for users who have read-only access.
vip		Specify if the Contrail API VIP is used for configuring client-side software. If not specified, private IP of the first Contrail API VIP unit will be used.
use-external-rabbitmq	false	To enable the Charm to use the internal RabbitMQ server, set use-external-rabbitmq to false.  To use an external AMQP server, set use-external-rabbitmq to true.  <b>NOTE:</b> Do not change the flag after deployment.
flow-export-rate	0	Specify how many flow records are exported by vRouter agent to the Contrail Collector when a flow is created or deleted.

**Table 13: Options for contrail-controller (Continued)**

Option	Default option	Description
docker-registry		Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag		Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services.  Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.
no_proxy		Specify the list of destinations that must be directly accessed.

- Options for **contrail-keystone-auth** Charms.

**Table 14: Options for contrail-keystone-auth**

Option	Default option	Description
ssl_ca		Specify if the base64-encoded SSL CA certificate is provided to Contrail keystone clients.  <b>NOTE:</b> This certificate is required if you use a privately signed ssl_cert and ssl_key.



- Options for **contrail-openstack** Charms.

**Table 15: Options for contrail-controller**

Option	Default option	Description
enable-metadata-server	true	Set enable-metadata-server to true to configure metadata and enable nova to run a local instance of nova-api-metadata for virtual machines
use-internal-endpoints	false	Set use-internal-endpoints to true for OpenStack to configure services to use internal endpoints.
heat-plugin-dirs	/usr/lib64/heat,/usr /lib/heat/usr/lib/ python2.7/dist-packages/vnc_api/gen/heat/resources	Specify the heat plugin directories.
docker-registry		Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag		Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services.  Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.

Table 15: Options for contrail-controller *(Continued)*

Option	Default option	Description
no_proxy		Specify the list of destinations that must be directly accessed.

SEE ALSO

[Understanding Juju Charms](#)

Preparing to Deploy Contrail by Using Juju Charms

Deploying Contrail Charms

# Contrail Command

## IN THIS CHAPTER

- [Configuring Contrail Command | 411](#)
- [Deploying Contrail Cluster using the Contrail Command UI | 418](#)
- [Deploying Contrail Cluster using Contrail-Command and instances.yml | 431](#)
- [Importing Contrail Cluster Data using Contrail Command | 439](#)

## Configuring Contrail Command

### IN THIS SECTION

- [Requirements | 411](#)
- [Overview | 412](#)
- [Configuration | 412](#)
- [Sample command\\_servers.yml File | 414](#)

The Contrail Command user interface (UI) is supported starting with Contrail Release 5.0.1. Contrail Command is an intuitive, wizard-based UI which provides automated work flows such as the following:

- Contrail cluster deployment (Kolla-based OpenStack cluster)
- Automating the data center IP fabric
- Orchestrating virtual machines and bare metal servers

### Requirements

The system requirements to install the Contrail Command server are:

- A VM or physical server with:
  - 8 vCPUs
  - 64 GB RAM
  - 300 GB disk out of which 256 GB is allocated to **/root** directory.
- Internet access to and from the physical server, hereafter referred to as the Contrail Command server
- (Recommended) x86 server with CentOS 7.5 as the base OS to install Contrail Command

## Overview

Contrail Command is an intuitive, wizard-based user interface (UI) to manage private and public clouds, physical and virtual workloads and devices.

## Configuration

### IN THIS SECTION

- [Procedure | 412](#)

## Prerequisite

`docker-py` is obsolete in Contrail Release 5.0.2. You must remove `docker-py` and `docker` Python packages from all the nodes where you want to install the Contrail Command UI.

```
pip uninstall docker-py docker
```

## Procedure

### Step-by-Step Procedure

Perform the following steps to configure and install Contrail Command.

1. Install Docker on the Contrail Command server. These packages are necessary to automate the deployment of Contrail Command software.

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

```
yum install -y docker-ce
```

```
systemctl start docker
```

2. Download the contrail-command-deployer Docker container image to deploy contrail-command (contrail\_command, contrail\_mysql containers) from hub.juniper.net. Allow Docker to connect to the private secure registry.

```
docker login hub.juniper.net --username <container_registry_username> --password <container_registry_password>
```

Pull Contrail-Command-Deployer Container from the private secure registry.

```
docker pull hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
```

Example, for container\_tag: 5.0.1-0.214, use the following command:

```
docker pull hub.juniper.net/contrail/contrail-command-deployer:5.0.1-0.214
```

3. Create the input configuration **command\_servers.yml** file.

Use the ["Sample command\\_servers.yml File" on page 414](#) to create the **command\_servers.yml** file.

4. Start the Contrail\_Command\_Deployer container to deploy the Contrail-Command server.

```
docker run -t --net host -v ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE:/command_servers.yml -d --privileged --name  
contrail_command_deployer hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
```

*ABSOLUTE\_PATH\_TO\_COMMAND\_SERVERS\_FILE*—path to the **command\_servers.yml** file that you created in step ["3" on page 413](#).

Example, for container\_tag: 5.0.1-0.214, use the following command:

```
docker run -t --net host -v /root/command_servers.yml:/command_servers.yml -d --privileged --name  
contrail_command_deployer hub.juniper.net/contrail/contrail-command-deployer:5.0.1-0.214
```

The contrail\_command and contrail\_mysql Contrail Command containers are deployed.

5. (Optional) You can also upgrade Contrail-Command UI without deleting existing database information. To update contrail\_command container and not make changes to the database container, use the following command.

```
docker run -t --net host -e delete_db=no -v <ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE>:/command_servers.yml -d --  
privileged --name contrail_command_deployer hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
```



**NOTE:** Code changes that involve schema modifications require updating the database container as well. Step ["5" on page 413](#) is recommended only if the UI application requires an update.

6. (Optional) Track the progress of Step "4" on page 413.

```
docker logs -f contrail_command_deployer
```

7. Once the playbook execution completes, log in to Contrail Command using [https:// Contrail-Command-Server-IP-Address:9091](https://Contrail-Command-Server-IP-Address:9091). Use the same user name and password that was entered in "3" on page 413. Default username is admin and password is contrail123.

## Sample command\_servers.yml File

```
---
command_servers:
  server1:
    ip: <IP Address>
    connection: ssh
    ssh_user: root
    ssh_pass: <contrail command server password>
    sudo_pass: <contrail command server root password>
    ntpserver: <NTP Server address>

    # Specify either container_path
    #container_path: /root/contrail-command-051618.tar
    # or registry details and container_name
    # registry_insecure: true
    # container_registry: ci-repo.englab.juniper.net:5010
    registry_insecure: false
    container_registry: hub.juniper.net/contrail
    container_name: contrail-command
    container_tag: 5.0.2-0.349
    container_registry_username: <registry username>
    container_registry_password: <registry password>
    config_dir: /etc/contrail

    # contrail command container configurations given here go to /etc/contrail/contrail.yml
    contrail_config:
      # Database configuration. MySQL/PostgreSQL supported
      database:
        # MySQL example
        type: mysql
        dialect: mysql
        host: localhost
        user: root
        password: contrail123
```

```

    name: contrail_test
    # Postgres example
    #connection: "user=root dbname=contrail_test sslmode=disable"
    #type: postgres
    #dialect: postgres

    # Max Open Connections for DB Server
    max_open_conn: 100
    connection_retries: 10
    retry_period: 3s

# Log Level
log_level: debug

# Server configuration
server:
    enabled: true
    read_timeout: 10
    write_timeout: 5
    log_api: true
    address: ":9091"

# TLS Configuration
tls:
    enabled: true
    key_file: /usr/share/contrail/ssl/cs-key.pem
    cert_file: /usr/share/contrail/ssl/cs-cert.pem

# Enable GRPC or not
enable_grpc: false

# Static file config
# key: URL path
# value: file path. (absolute path recommended in production)
static_files:
    /: /usr/share/contrail/public

# API Proxy configuration
# key: URL path
# value: String list of backend host
#proxy:
#    /contrail:
#        - http://localhost:8082

```

```

notify_etcd: false

# Keystone configuration
keystone:
  local: true
  assignment:
    type: static
  data:
    domains:
      default: &default
      id: default
      name: default
    projects:
      admin: &admin
      id: admin
      name: admin
      domain: *default
      demo: &demo
      id: demo
      name: demo
      domain: *default
    users:
      admin:
        id: admin
        name: Admin
        domain: *default
        password: contrail123
        email: admin@juniper.nets
        roles:
          - id: admin
            name: Admin
            project: *admin
      bob:
        id: bob
        name: Bob
        domain: *default
        password: bob_password
        email: bob@juniper.net
        roles:
          - id: Member
            name: Member
            project: *demo

```



```

    store:
        type: memory
        expire: 36000
        insecure: true
        authurl: https://localhost:9091/keystone/v3

# disable authentication with no_auth true and comment out keystone configuraion.
#no_auth: true
insecure: true

etcd:
    endpoints:
        - localhost:2379
    username: ""
    password: ""
    path: contrail

watcher:
    enabled: false
    storage: json

client:
    id: admin
    password: contrail123
    project_name: admin
    domain_id: default
    schema_root: /
    endpoint: https://localhost:9091

compilation:
    enabled: false
    # Global configuration
    plugin_directory: 'etc/plugins/'
    number_of_workers: 4
    max_job_queue_len: 5
    msg_queue_lock_time: 30
    msg_index_string: 'MsgIndex'
    read_lock_string: "MsgReadLock"
    master_election: true

# Plugin configuration
plugin:
    handlers:

```

```

        create_handler: 'HandleCreate'
        update_handler: 'HandleUpdate'
        delete_handler: 'HandleDelete'

    agent:
        enabled: true
        backend: file
        watcher: polling
        log_level: debug

    # The following are optional parameters used to patch/cherrypick
    # revisions into the contrail-ansible-deployer sandbox. These configs
    # go into the /etc/contrail/contrail-cluster.tpl file
    #   cluster_config:
    #       ansible_fetch_url: "https://review.opencontrail.org/Juniper/contrail-ansible-
    #       deployer refs/changes/80/40780/20"
    #       ansible_cherry_pick_revision: FETCH_HEAD
    #       ansible_revision: GIT_COMMIT_HASH

```

## RELATED DOCUMENTATION

*Installing Contrail Cluster using the Contrail Command UI*

*Installing Contrail Cluster using Contrail Command and instances.yml*

*Importing Contrail Cluster Data using Contrail Command*

## Deploying Contrail Cluster using the Contrail Command UI

### IN THIS SECTION

- [Requirements | 419](#)
- [Overview | 419](#)
- [Configuration | 421](#)

This example topic describes how to use the Contrail Command User interface (UI) to deploy a Contrail Cluster starting with Contrail Release 5.0.1.

## Requirements

- Contrail Controller – 8 vCPU, 64G memory, 300G storage
- OpenStack Controller – 4 vCPU , 32G memory, 100G storage
- Contrail Server Node (CSN) – 4 vCPU, 16G memory, 100G storage
- Compute nodes– Dependent on the workloads

## Overview

### IN THIS SECTION

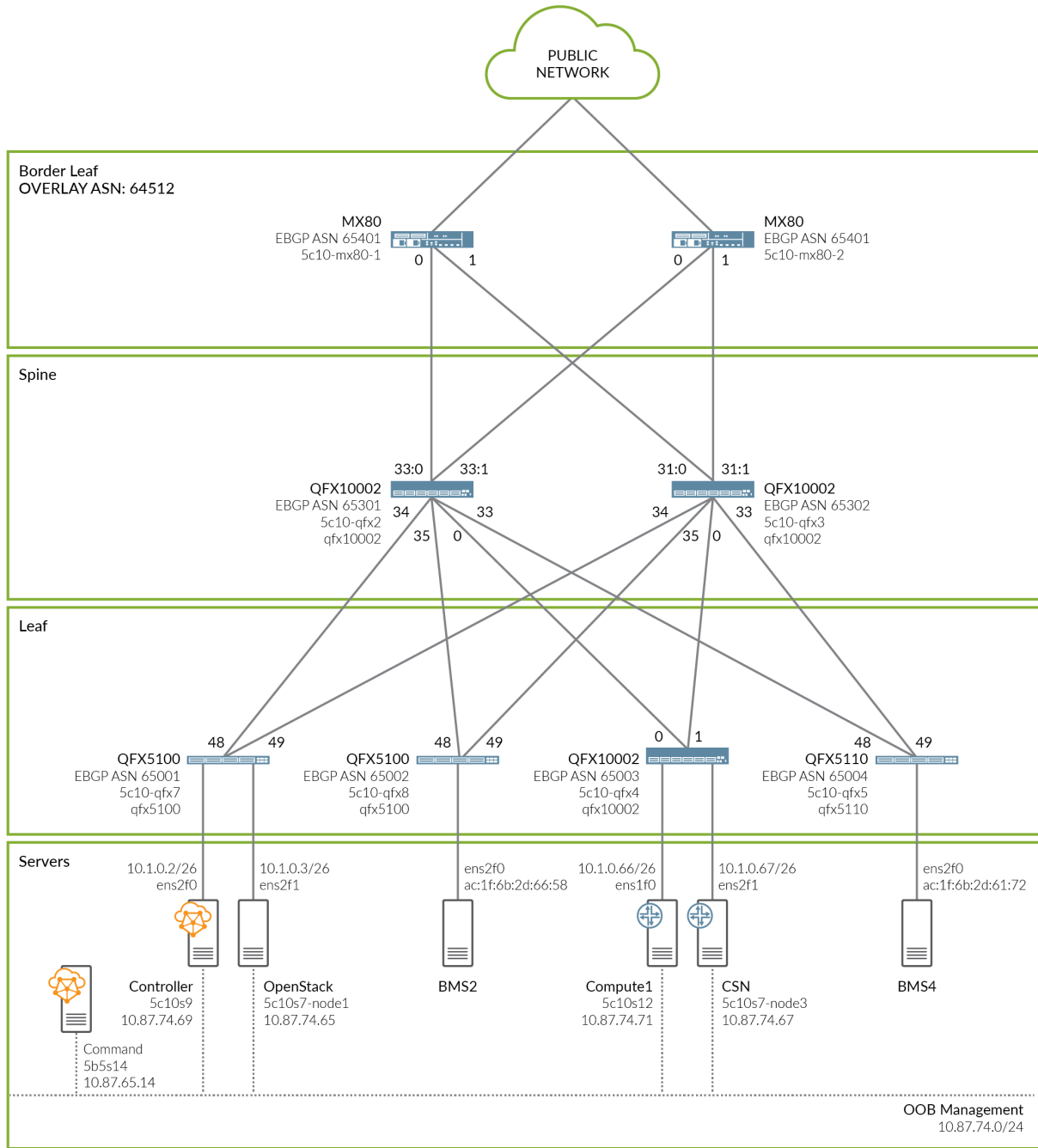
- [Topology | 419](#)

Contrail Cluster is an OpenStack orchestration coupled with the Contrail Networking plugin.

### Topology

Consider a sample cluster topology, with a non-HA environment of one Contrail Controller and one OpenStack Controller, one compute node and one CSN, as displayed in [Figure 44 on page 420](#).

Figure 44: Sample Contrail Cluster Topology



g200484

## Configuration

### IN THIS SECTION

- [Deploying a Contrail Cluster | 421](#)

## Deploying a Contrail Cluster

### Step-by-Step Procedure

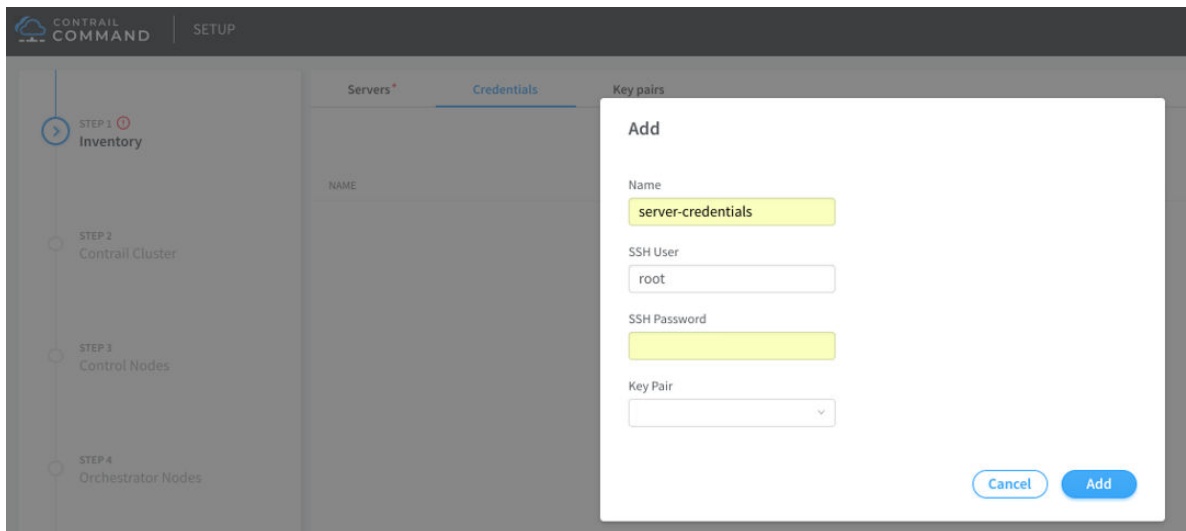
To deploy a Contrail Cluster using Contrail Command, perform the following steps.

1. Add physical servers. You can add a server in the following two ways:

- One by one
- CSV file bulk import



**NOTE:** Create server login credentials before adding the servers.



- **One by one:** You can add each server one by one. To add servers one by one, the following mandatory parameters should be entered.

Hostname, Management IP address and the Management Interface.

- **CSV file bulk import:** You can upload a CSV file which contains the details of each server. This CSV file must conform to the Contrail Command format. You can download the CSV template from Contrail Command and reuse the template as per your requirements. To download a sample CSV file, navigate to **Infrastructure > Servers > Add Servers** in the Contrail Command UI.

Use the Bulk Import option for large deployments. Bulk Import option requires a CSV file input. For large deployments, use the **Bulk Import (csv)** option. Click **Bulk Import (csv)**, to download the template. A sample CSV file is shown here:

```
Workload Type,HostName,Management IP,Disk Partition,Network Interface,MAC address,IPMI
Driver,IPMI Address,IPMI UserName,IPMI Password,Memory mb,CPU's,CPU Arch,Local
gb,Capabilities,Number of Network Interfaces,Interface Name,Interface MAC
Address,Interface IP,Enable PXE,Interface Name,Interface MAC Address,Interface IP,Enable
PXE

physical,5c10s9,10.87.74.69,,enp4s0f0,,,,,,,,,2,enp4s0f0,,10.87.74.69,,ens2f0,,10.1.0.2,

physical,5c10s7-node1,10.87.74.65,,eno1,,,,,,,,,2,eno1,,10.87.74.65,,ens2f1,,10.1.0.3,

physical,5c10s7-node3,10.87.74.67,,eno1,,,,,,,,,2,eno1,,10.87.74.67,,ens2f1,,10.1.0.67,

physical,5c10s12,10.87.74.71,,eno1,,,,,,,,,2,eno1,,10.87.74.71,,ens1f0,,10.1.0.66,
```



**NOTE:** The demo topology above has only one compute node. If you are deploying additional compute nodes, you must include them in the CSV file.

Figure 45: Add Server

## 2. Create a cluster.

If **Container registry** = `hub.juniper.net/contrail`. This registry is secure. Unselect the **Insecure** box. Also, **Contrail version** = `contrail_container_tag` for your release of Contrail as listed in [README Access for Contrail](#).

**Default vRouter Gateway** = Default gateway for the compute nodes. If any one of the compute nodes has a different default gateway than the one provided here, enter that gateway in "5" on page 427 and "6" on page 428 for service nodes.

Set the order of **Encapsulation Priority** for the EVPN supported methods - MPLS over UDP, MPLS over GRE And VxLAN.

VXLAN, MPLSoUDP, MPLSoGRE

Select **Show Advanced Options**. Add the following configuration parameters:

- **CONTROLLER\_NODES** = List of comma separated mgmt interface IP addresses of the Contrail controller

- CONTROL\_NODES = List of comma separated data interface IP addresses of the Contrail controller
- TSN\_NODES = List of comma separated data interface IP addresses of the Contrail service nodes
- CONTRAIL\_CONTAINER\_TAG = *contrail\_container\_tag-ocata* or *container\_tag-queens*

Figure 46: Create Cluster

The screenshot displays the 'Contrail Command' setup interface. On the left, a vertical progress bar indicates the steps: STEP 1 Inventory, STEP 2 Contrail Cluster (active), STEP 3 Control Nodes, STEP 4 Orchestrator Nodes, STEP 5 (optional) Compute Nodes, STEP 6 (optional) Contrail Service Nodes, STEP 7 Summary, and STEP 8 Provisioning. The main area is titled 'SETUP' and contains the following fields:

- Cluster Name:** Demo-cluster
- Container Registry:** hub.juniper.net/contrail
- Container Registry Username:** username
- Container Registry Password:** password
- Contrail Version:** 5.0.1-0.214
- Provisioner Type:** Ansible
- Domain Suffix:** local
- NTP Server:** 10.84.5.100
- Default Vrouter Gateway:** 10.1.0.254
- Encapsulation Priority:** VXLAN,MPLS,UDP,MPLS...
- Enable ZTP:** ☐
- Show Advanced Options:** ☒
  - Contrail Configuration:**

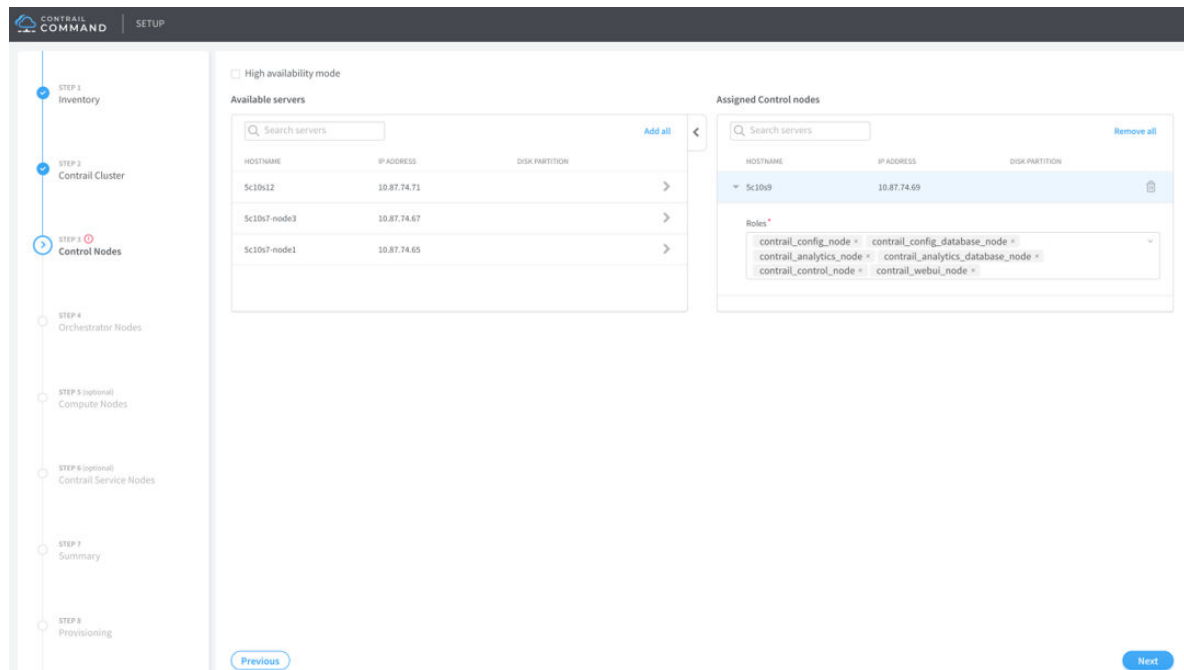
Key	Value
CONTROLLER_NODES	10.87.74.69
CONTROL_NODES	10.1.0.2
TSN_NODES	10.1.0.67
CONTRAIL_CONTAINER_TAG	5.0.1-0.214-ocata

At the bottom, there are 'Previous' and 'Next' buttons.

### 3. Select the contrail-control node.



Figure 47: Select Control Nodes



#### 4. Select the orchestration node.

Select **Show Advanced** option to customize your deployment and then select **Orchestration Nodes**.

In order to run Openstack services on the control data network, set the following parameters:

- **Control & Data Network Virtual IP address:** It is an internal VIP. e.g. - 10.87.74.100
- **Management Network Virtual IP address:** It is an external VIP. e.g. - 10.1.0.100
- **keepalived\_virtual\_router\_id:** It is to be added as a key value pair. It can be set to any value between 0-255

Add the following under custom configuration for VM based setup:

```
nova.conf: |
    [libvirt]
    virt_type=qemu
    cpu_mode=none
```



**NOTE:** Minimum 8 indent spaces are required for lines following the nova.conf.

In Contrail Command, key-value pairs handle parameters that be enabled or disabled in Kolla Global.

Set the following in Kolla Globals:

```
enable_ironic = yes
enable_swift = yes
upgrade_kernel = yes
```

If Ironic is not needed and if you are not going to use Life Cycle Management in Contrail Command then you need not deploy Ironic.

Swift can be enabled for Image management uses case, this parameter is disabled by default.

Compute node kernel version = kernel-3.10.0-862.3.2.el7.x86\_64; Else kernel upgrade is required.

To configure kolla password add keystone\_admin\_password <password> key-value pair in the kolla passwords section. This password will be used for logging onto the contrail command UI after the provisioning completes.

**Figure 48: Select Orchestrator Nodes**

The screenshot shows the 'SETUP' page in the Contrail Command UI. The left sidebar lists steps from 'Inventory' to 'Provisioning'. Step 4, 'Orchestrator Nodes', is the current step. The main area contains configuration fields for the orchestrator type (Openstack), container registry (default), Openstack release (ocata), and network IP addresses. Below these are sections for 'Kolla Globals' and 'Kolla Passwords'. In the 'Kolla Globals' section, 'enable\_ironic' and 'enable\_swift' are both set to 'yes'. In the 'Kolla Passwords' section, 'keystone\_admin\_password' is set to 'Juniper123'. At the bottom, there are 'Previous' and 'Next' buttons.

Select the Openstack (orchestration) node.

The screenshot shows the CONTRAIL COMMAND SETUP interface. On the left, a vertical sidebar lists steps: STEP 1: Inventory, STEP 2: Contrail Cluster, STEP 3: Control Nodes, STEP 4: Orchestrator Nodes (highlighted), STEP 5 (optional): Compute Nodes, STEP 6 (optional): Contrail Service Nodes, STEP 7: Summary, and STEP 8: Provisioning.

The main content area is divided into several sections:

- Kolla Globals:** A table with two rows:
 

Key	Value
enable_ironic	yes
enable_swift	yes

 Below the table is a "+ Add" button.
- Kolla Passwords:** A table with one row:
 

Key	Value
keystone_admin_password	Juniper123

 Below the table is a "+ Add" button.
- Available servers:** A table with columns: HOSTNAME, IP ADDRESS, and DISK PARTITION. It lists three servers:
 

HOSTNAME	IP ADDRESS	DISK PARTITION
Sc10n12	10.87.74.71	>
Sc10n7-node3	10.87.74.67	>
Sc10n8	10.87.74.69	>

 Below the table is a "Previous" button.
- Assigned Openstack nodes:** A table with columns: HOSTNAME, IP ADDRESS, and DISK PARTITION. It lists one server:
 

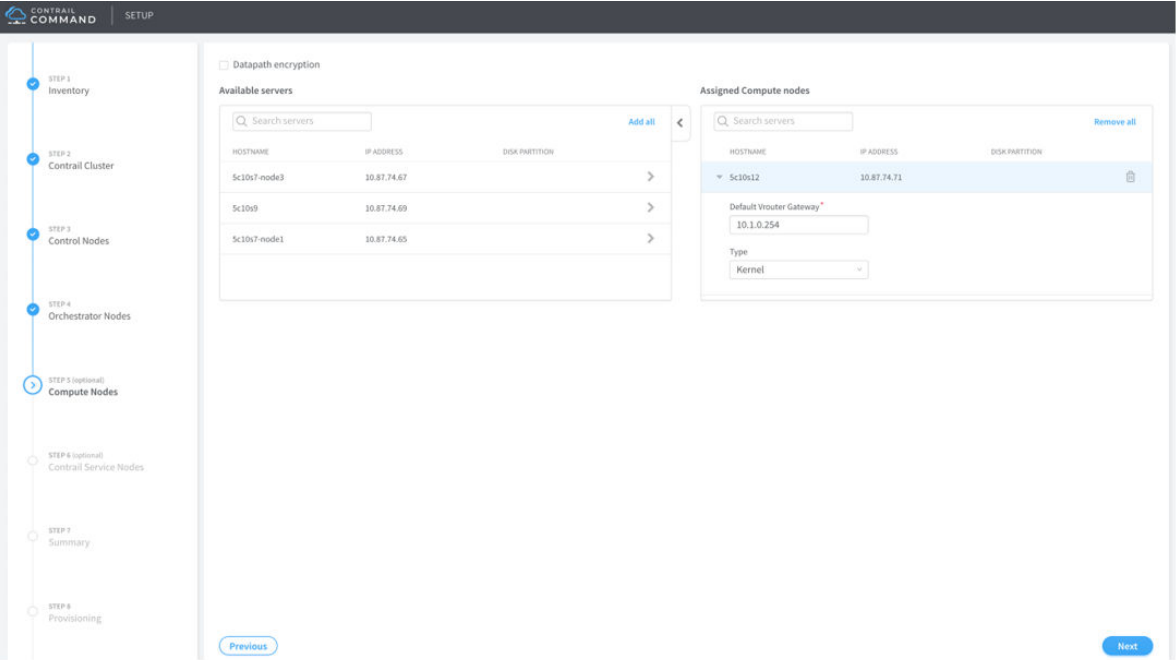
HOSTNAME	IP ADDRESS	DISK PARTITION
Sc10n7-node1	10.87.74.65	>

 Below the table, there is a "Roles" section with a dropdown menu showing selected roles:
  - openstack\_control\_node
  - openstack\_network\_node
  - openstack\_storage\_node
  - openstack\_monitoring\_node
 Below the roles is a "Next" button.

5. Select the Compute Nodes. For each compute node, enter the gateway, if it is different from what was added in the global parameters in "2" on page 423. Then set the mode.

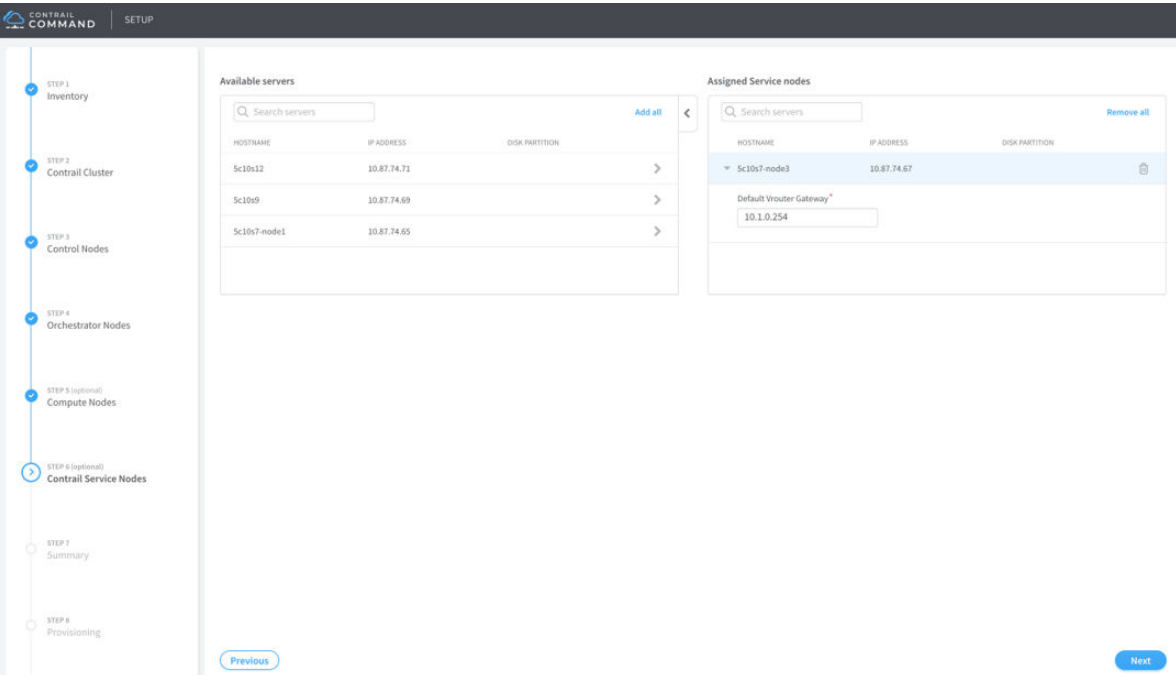
As of now, only *Kernel* mode is supported

Figure 49: Select Compute Nodes



6. Select the Contrail Service Node.

Figure 50: Select Contrail Service Nodes



7. Verify the summary of your cluster configuration and click **Provision**.

Figure 51: Verify Summary

Cluster overview

Display name

Container registry

Container registry username

Container registry password

Contrail version

Provisioner type

Domain Suffix

NTP server

Default Vrouter Gateway

Encapsulation priority

Enable ZTP

Contrail configuration

CONTROLLER\_NODES

CONTROL\_NODES

TSN\_NODES

CONTRAIL\_CONTAINER\_TAG

High availability mode

Orchestrator

Openstack release

Openstack internal virtual IP

Openstack external virtual IP

Kolla globals

enable\_ironic

enable\_swift

Kolla passwords

keystone\_admin\_password

Demo-cluster

hub.juniper.net/contrail

username

password

5.0.1-0.214

ansible

local

10.84.5.100

10.1.0.254

VXLAN,MPLSoUDP,MPLSoGRE

false

10.87.74.69

10.1.02

10.1.0.67

5.0.1-0.214-ocata

false

openstack

ocata

-

-

yes

yes

Juniper123

CONTRAIL  
COMMAND

SETUP

STEP 1  
Inventory

STEP 2  
Contrail Cluster

STEP 3  
Control Nodes

STEP 4  
Orchestrator Nodes

STEP 5 (optional)  
Compute Nodes

STEP 6 (optional)  
Contrail Service Nodes

STEP 7  
Summary

STEP 8  
Provisioning

Cluster overview

Display name

Container registry

Container registry username

Container registry password

Contrail version

Provisioner type

Domain Suffix

NTP server

Default Vrouter Gateway

Encapsulation priority

Enable ZTP

Contrail configuration

High availability mode

Orchestrator

Openstack release

Openstack internal virtual IP

Openstack external virtual IP

Kolla globals

Kolla passwords

Demo-cluster

hub.juniper.net/contrail

username

password

5.0.1-0.214

ansible

local

10.84.5.100

10.1.0.254

VXLAN,MPLSoUDP,MPLSoGRE

false

10.87.74.69

10.1.02

10.1.0.67

5.0.1-0.214-ocata

false

openstack

ocata

-

-

yes

yes

Juniper123

Nodes overview

All cluster nodes

Control nodes

Compute nodes

Openstack nodes

Service nodes

NAME	TYPE	IP ADDRESS	ROLES
Sc10a12	physical/virtual node	10.87.74.71	Compute node
Sc10a7-node1	physical/virtual node	10.87.74.65	Openstack node
Sc10a7-node3	physical/virtual node	10.87.74.67	Service node
Sc10a9	physical/virtual node	10.87.74.69	Control node

Previous

Provision

## RELATED DOCUMENTATION

*Configuring Contrail Command*

*Installing Contrail Cluster using Contrail Command and instances.yml*

*Importing Contrail Cluster Data using Contrail Command*

## Deploying Contrail Cluster using Contrail-Command and instances.yml

Contrail Release 5.0.1 supports deploying a Contrail cluster using Contrail Command and the **instances.yml** file.

### System Requirements

- A VM or physical server with:
  - 8 vCPUs
  - 64 GB RAM
  - 300 GB disk out of which 256 GB is allocated to **/root** directory.
- Internet access to and from the physical server, hereafter referred to as the Contrail Command server
- (Recommended) x86 server with CentOS 7.5 (Minimal ISO) as the base OS to install Contrail Command

### Prerequisite

`docker-py` is obsolete in Contrail Release 5.0.2. You must remove `docker-py` and `docker` Python packages from all the nodes where you want to install the Contrail Command UI.

```
pip uninstall docker-py docker
```

### Configuration

Perform the following steps to deploy a Contrail cluster using Contrail Command and the **instances.yml** file.

1. Install Docker on the Contrail Command server. These packages are necessary to automate the deployment of Contrail Command software.

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

```
yum install -y docker-ce
```

```
systemctl start docker
```

2. Download Contrail-Command-Deployer docker container image from hub.juniper.net. To download these containers and for access to hub.juniper.net, refer to the *Access to Contrail Registry* topic on the [Contrail software download](#) page. Allow Docker to connect to the private secure registry.

```
docker login hub.juniper.net --username < container_registry_username > --password <
container_registry_password >
```

Pull Contrail-Command-Deployer Container from the private secure registry.

```
docker pull hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
```

Example, for container\_tag: 5.0.1-0.214, use the following command:

```
docker pull hub.juniper.net/contrail/contrail-command-deployer:5.0.1-0.214
```

3. Edit the input configuration **instances.yml** file. See "[No Link Title](#)" on page 432 for a sample **instances.yml** file.
4. Start the Contrail\_Command\_Deployer container to deploy the Contrail-Command (UI) server and provision Contrail-Cluster using the **instances.yml** file provided.

```
docker run -t --net host -e action=provision_cluster -v <ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE>:/
command_servers.yml -v <ABSOLUTE_PATH_TO_INSTANCES_FILE>:/instances.yml -d --privileged --name
contrail_command_deployer hub.juniper.net/contrail/contrail-command-deployer: <container_tag>
```

The contrail\_command and contrail\_mysql Contrail Command containers are deployed. Contrail cluster is also provisioned using the given **instances.yml** file.

5. (Optional) Track the progress of 4.

```
docker logs -f contrail_command_deployer
```

6. Once the playbook execution completes, log in to Contrail Command using [https:// Contrail-Command-Server-IP-Address:9091](https://Contrail-Command-Server-IP-Address:9091). Use the same user name and password that was entered in 3. Default username is admin and password is contrail123.

### Sample instances.yml File

```
global_configuration:
  CONTAINER_REGISTRY: hub.juniper.net/contrail
  CONTAINER_REGISTRY_USERNAME: < container_registry_username >
  CONTAINER_REGISTRY_PASSWORD: < container_registry_password >
provider_config:
  bms:
    ssh_pwd: <Pwd>
    ssh_user: root
    ntpserver: <NTP Server>
```



```

    domainsuffix: local
instances:
  bms1:
    provider: bms
    ip: <BMS IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      vrouter:
      openstack:
      openstack_compute:
  bms2:
    provider: bms
    ip: <BMS2 IP>
    roles:
      openstack:
  bms3:
    provider: bms
    ip: <BMS3 IP>
    roles:
      openstack:
  bms4:
    provider: bms
    ip: <BMS4 IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
  bms5:
    provider: bms
    ip: <BMS5 IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:

```

```

    analytics:
    webui:
bms6:
  provider: bms
  ip: <BMS6 IP>
  roles:
    config_database:
    config:
    control:
    analytics_database:
    analytics:
    webui:
bms7:
  provider: bms
  ip: <BMS7 IP>
  roles:
    vrouter:
      PHYSICAL_INTERFACE: <Interface name>
      VROUTER_GATEWAY: <Gateway IP>
    openstack_compute:
bms8:
  provider: bms
  ip: <BMS8 IP>
  roles:
    vrouter:
      # Add following line for TSN Compute Node
      TSN_EVPN_MODE: True
    openstack_compute:
contrail_configuration:
  CLOUD_ORCHESTRATOR: openstack
  CONTRAIL_VERSION: latest or <contrail_container_tag>
  CONTRAIL_CONTAINER_TAG: <contrail_container_tag>-queens
  RABBITMQ_NODE_PORT: 5673
  VROUTER_GATEWAY: <Gateway IP>
  ENCAP_PRIORITY: VXLAN,MPLSoUDP,MPLSoGRE
  AUTH_MODE: keystone
  KEYSTONE_AUTH_HOST: <Internal VIP>
  KEYSTONE_AUTH_URL_VERSION: /v3
  CONTROLLER_NODES: < list of mgmt. ip of control nodes >
  CONTROL_NODES: <list of control-data ip of control nodes>
  OPENSTACK_VERSION: queens
kolla_config:
  kolla_globals:

```

```
openstack_release: queens
kolla_internal_vip_address: <Internal VIP>
kolla_external_vip_address: <External VIP>
openstack_release: queens
enable_haproxy: "no"      ("no" by default, set "yes" to enable)
enable_ironic: "no"       ("no" by default, set "yes" to enable)
enable_swift: "no"        ("no" by default, set "yes" to enable)
keepalived_virtual_router_id: <Value between 0-255>
kolla_passwords:
  keystone_admin_password: <Keystone Admin Password>
```

The following [Table 16 on page 435](#) defines the function of each variable present in the instances.yml file.

**Table 16: Description of Variables in Instances.yml file**

Instances	Description
CONTROLLER_NODES	<p>Enter the management interface IP address of the controller node.</p> <p>The default value is either populated from the IP address in the instances.yml file or the IP address provided while adding the server from contrail command UI.</p> <p>CONTROLLER_NODES is a meta variable that is used to populate other unpopulated nodes like CONFIG_NODES, CONTROL_NODES, ANALYTICS_NODES, and others.</p>
CONTROL_NODES	<p>Enter the data interface IP address of the controller node.</p> <p>The default value is populated from the CONTROLLER_NODES variable.</p> <p>The CONTROL_NODES variable also indicates the interfaces on which the control services can function.</p>

**Table 16: Description of Variables in Instances.yml file (Continued)**

Instances	Description
kolla_internal_vip_address	<p>Enter the ctrl-data-network IP address.</p> <p>kolla_internal_vip_address is a virtual IP address that separates internal communication requests.</p>
kolla_external_vip_address	<p>Enter the management network IP address.</p> <p>kolla_external_vip_address is a virtual IP address that separates external communication requests.</p>
KEYSTONE_AUTH_HOST	<p>Enter the IP address of the host, which has AUTH_MODE set to keystone.</p> <p>The default value is populated automatically.</p>
CLOUD_ORCHESTRATOR	<p>Enter the name of orchestrator you are using.</p> <p>Default value: none.</p> <p>A CLOUD_ORCHESTRATOR is the platform that automates provisioning of cloud services.</p>
CONTRAIL_VERSION	<p>Enter the Contrail version number.</p> <p>Default value: latest.</p>
RABBITMQ_NODE_PORT	<p>Enter the port number assigned to RabbitMQ node.</p> <p>Default value: 5672.</p> <p>RabbitMQ is a software that provides message queuing service, which is an efficient method of exchanging data between applications and servers.</p>

**Table 16: Description of Variables in Instances.yml file (Continued)**

Instances	Description
VROUTER_GATEWAY	<p>Enter the gateway IP address of the virtual router. The default gateway assigned to the virtual router is known as virtual router gateway.</p> <p>The default value is the default gateway IP address of management subnet in case of single interface setup.</p> <p>The VROUTER_GATEWAY is default gateway IP address for the compute nodes in the contrail cluster.</p>
ENCAP_PRIORITY	<p>ENCAP_PRIORITY is used to set the order of Encapsulation Priority for the EVPN supported methods - MPLS over UDP, MPLS over GRE And VxLAN.</p> <p>Default value: MPLSoUDP, MPLSoGRE, VXLAN.</p>

**Table 16: Description of Variables in Instances.yml file (Continued)**

AUTH_MODE	<p>Enter the desired keystone authentication mode.</p> <p>Default value: noauth.</p>
KEYSTONE_AUTH_URL_VERSION	<p>Enter the URL of the Keystone authentication server.</p> <p>Default value: /v2.0.</p> <p>Keystone is a Opentack identity service. All Openstack operations are authenticated via the keystone server.</p>
OPENSTACK_VERSION	<p>Enter the software version number of the Openstack platform.</p> <p>Default value: queens.</p>

enable_haproxy	<p>Enter yes to enable haproxy.</p> <p>Enter no to disable haproxy.</p> <p>Default value: no.</p> <p>High Availability Proxy (Haproxy) provides high availability load balancing and proxy servers in cloud networking.</p>
enable_irony	<p>Enter yes to deploy Ironic notification manager service container.</p> <p>Default value: no.</p> <p>Deploy Ironic if you want to perform Life Cycle Management in Contrail Command.</p>
enable_swift	<p>Enter yes to deploy Swift.</p> <p>Default value: no.</p> <p>Deploy Swift if you want to perform Image management in Contrail Command.</p>
keepalived_virtual_router_id	<p>Enter value between 0-255.</p> <p>Default value: 51.</p> <p>keepalived_virtual_router_id is used to configure keepalives. In Openstack HA, the keepalives are used to run and elect a master based on VRRP protocol.</p>
keystone_admin_password	<p>Enter keystone admin password.</p> <p>Default value: contrail123.</p> <p>keystone_admin_password is used to set a password for the admin to access keystone. Keystone is a Openstack identity service.</p>

## RELATED DOCUMENTATION

*Configuring Contrail Command*

*Deploying Contrail Cluster using the Contrail Command UI*

*Importing Contrail Cluster Data using Contrail Command*

## Importing Contrail Cluster Data using Contrail Command

Contrail Release 5.0.1 supports importing of Contrail cluster data using Contrail Command.

### Before you begin

`docker-py` is obsolete in Contrail Release 5.0.2. You must remove `docker-py` and `docker` Python packages from all the nodes where you want to install the Contrail Command UI.

```
pip uninstall docker-py docker
```

### System Requirements

- A VM or physical server with:
  - 8 vCPUs
  - 64 GB RAM
  - 300 GB disk out of which 256 GB is allocated to `/root` directory.
- Internet access to and from the physical server, hereafter referred to as the Contrail Command server
- (Recommended) x86 server with CentOS 7.5 as the base OS to install Contrail Command

### Configuration

Perform the following steps to import Contrail cluster data.

1. Install Docker on the Contrail Command server. These packages are necessary to automate the deployment of Contrail Command software.

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

```
yum install -y docker-ce
```

```
systemctl start docker
```

2. Download the `contrail-command-deployer` Docker container image to deploy `contrail-command` (`contrail_command`, `contrail_mysql` containers) from `hub.juniper.net`. Allow Docker to connect to the private secure registry.

```
docker login hub.juniper.net --username <container_registry_username> --password <container_registry_password>
```

Pull `Contrail-Command-Deployer` Container from the private secure registry.

```
docker pull hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
```

Example, for container\_tag: 5.0.1-0.214, use the following command:

```
docker pull hub.juniper.net/contrail/contrail-command-deployer:5.0.1-0.214
```

3. Get the **instances.yml** file that was used to provision the Contrail cluster.
4. Start the Contrail-Command-Deployer container to deploy the Contrail Command (UI) server and import Contrail cluster data to Contrail Command (UI) server using the **instances.yml** file provided.

- To import a Contrail cluster using OpenStack:

```
docker run -t --net host -e orchestrator=openstack -e action=import_cluster -v <
ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE>:/command_servers.yml -v < ABSOLUTE_PATH_TO_INSTANCES_FILE>:/
instances.yml -d --privileged --name contrail_command_deployer hub.juniper.net/contrail/contrail-command-
deployer: <container_tag>
```

- To import a Contrail cluster using Kubernetes:

```
docker run -t --net host -e orchestrator=kubernetes -e action=import_cluster -v <
ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE>:/command_servers.yml -v < ABSOLUTE_PATH_TO_INSTANCES_FILE>:/
instances.yml -d --privileged --name contrail_command_deployer hub.juniper.net/contrail/contrail-command-
deployer: <container_tag>
```



**NOTE:** These steps explain how to import Contrail cluster data provisioned using OpenStack and Kubernetes. If your orchestrator is different, use `-e orchestrator=<YOUR_ORCHESTRATOR>` in the above command. The following orchestrators are supported:

- OpenStack—Use `orchestrator=openstack`
- Kubernetes—Use `orchestrator=kubernetes`
- Red Hat OpenShift—Use `orchestrator=openshift`
- VMware vCenter—Use `orchestrator=vcenter`

### Sample instances.yml File

```
global_configuration:
  CONTAINER_REGISTRY: hub.juniper.net/contrail
  CONTAINER_REGISTRY_USERNAME: < container_registry_username >
  CONTAINER_REGISTRY_PASSWORD: < container_registry_password >
provider_config:
  bms:
    ssh_pwd: <Pwd>
    ssh_user: root
    ntpserver: <NTP Server>
```



```

    domainsuffix: local
instances:
  bms1:
    provider: bms
    ip: <BMS1 IP>
    roles:
      openstack:
  bms2:
    provider: bms
    ip: <BMS2 IP>
    roles:
      openstack:
  bms3:
    provider: bms
    ip: <BMS3 IP>
    roles:
      openstack:
  bms4:
    provider: bms
    ip: <BMS4 IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
  bms5:
    provider: bms
    ip: <BMS5 IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
  bms6:
    provider: bms
    ip: <BMS6 IP>
    roles:
      config_database:
      config:

```

```

    control:
    analytics_database:
    analytics:
    webui:
bms7:
  provider: bms
  ip: <BMS7 IP>
  roles:
    vrouter:
      PHYSICAL_INTERFACE: <Interface name>
      VROUTER_GATEWAY: <Gateway IP>
    openstack_compute:
bms8:
  provider: bms
  ip: <BMS8 IP>
  roles:
    vrouter:
      # Add following line for TSN Compute Node
      TSN_EVPN_MODE: True
    openstack_compute:
contrail_configuration:
  CLOUD_ORCHESTRATOR: openstack
  CONTRAIL_VERSION: latest or <contrail_container_tag>
  CONTRAIL_CONTAINER_TAG: <contrail_container_tag>-queens
  RABBITMQ_NODE_PORT: 5673
  VROUTER_GATEWAY: <Gateway IP>
  ENCAP_PRIORITY: VXLAN,MPLSoUDP,MPLSoGRE
  AUTH_MODE: keystone
  KEYSTONE_AUTH_HOST: <Internal VIP>
  KEYSTONE_AUTH_URL_VERSION: /v3
  CONTROLLER_NODES: < list of mgmt. ip of control nodes >
  CONTROL_NODES: <list of control-data ip of control nodes>
  OPENSTACK_VERSION: queens
kolla_config:
  kolla_globals:
    openstack_release: queens
    kolla_internal_vip_address: <Internal VIP>
    kolla_external_vip_address: <External VIP>
    openstack_release: queens
    enable_haproxy: "no"          ("no" by default, set "yes" to enable)
    enable_ironic: "no"          ("no" by default, set "yes" to enable)
    enable_swift: "no"           ("no" by default, set "yes" to enable)
    keepalived_virtual_router_id: <Value between 0-255>

```

```
kolla_passwords:  
  keystone_admin_password: <Keystone Admin Password>
```

## RELATED DOCUMENTATION

*Configuring Contrail Command*

---

*Deploying Contrail Cluster using the Contrail Command UI*

---

*Deploying Contrail Cluster using Contrail-Command and instances.yml*

# Extending Contrail to Physical Routers, Bare Metal Servers, Switches, and Interfaces

## IN THIS CHAPTER

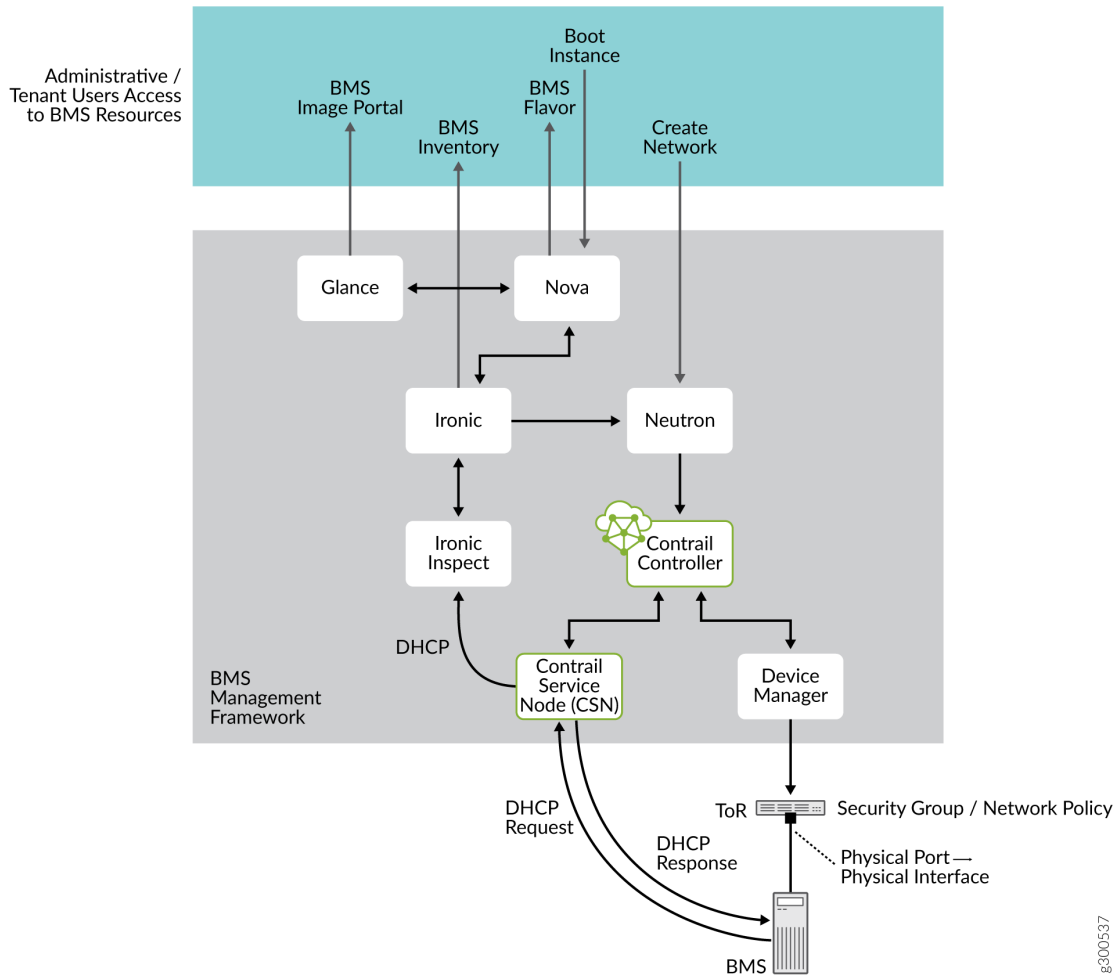
- [Understanding Bare Metal Server Management | 444](#)
- [Configuring High Availability for the Contrail OVSDb ToR Agent | 446](#)
- [Using Device Manager to Manage Physical Routers | 453](#)
- [SR-IOV VF as the Physical Interface of vRouter | 485](#)
- [Using Gateway Mode to Support Remote Instances | 487](#)
- [REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces | 489](#)

## Understanding Bare Metal Server Management

In Contrail Networking, you can manage the life cycle of bare metal servers (BMS) by using a backend framework, which acts as a bare metal server (BMS) manager. The BMS management framework in Contrail uses the functionality provided by the following OpenStack services: Ironi, Nova, and Glance. The BMS Management framework or the BMS framework manages the bare metal workload within a fabric. It includes BMS server life cycle management, onboarding of bare metal servers, bare metal image management, flavor management, inventory management, IP address management, security management, monitoring and reporting of life cycle management events, and discovery of bare metal servers.

An administrative user can configure the BMS framework and a tenant user can avail the services provided by the BMS framework. [Figure 52 on page 445](#) shows an architectural view of the BMS Management framework.

Figure 52: BMS Management- Detailed Architecture View



8300537



**NOTE:** In single-tenant environments, administrative and tenant workflows are performed by the same user.

To avail the functionalities of the BMS framework, you must first deploy a Contrail cluster with OpenStack. After this, the administrative user needs to specify the details of the server or node to be added to the BMS available in nodes database, from the Contrail Command UI. The BMS framework then creates a record of this new node and adds them to the available nodes database.

The administrative user creates images, nodes, and flavors, which the tenant users use to deploy bare metal servers in their network. A tenant user selects one of these flavors and images that suit their need to deploy a bare metal server. The BMS framework monitors the state of the deployed servers and provides this information to analytics DB by using Sandesh, which is an XML-based protocol for reporting analytics information. All the nodes onboarded or registered with BMS manager are in

**Available** state. After the tenant user has completed using the bare metal server and remove it, the server is then unprovisioned by the BMS framework and moved to the list of available nodes. Alternatively, the tenant user can remove the BMS instance from the tenant's network. For example, if you want to rent a BMS from a service provider, the service provider deploys a BMS instance and gives you an IP address of the BMS instance, which you can use to access the BMS. Once you have completed using the BMS, you can delete the instance and the service provider reclaims the BMS. After reclaiming the BMS the service provider cleans it and rents it to the next client. The BMS framework in Contrail Networking manages all these tasks. If the service provider wants to remove the BMS instance from the service, they can delete it from the available servers and the next tenant will get a new BMS instance from a server.

The BMS framework can install tenant user-specific software images on BMS and attach them to the tenant user network in a multi-tenant cloud. It provides a single-click solution for the tenant users to manage the bare metal servers in their network.

## Configuring High Availability for the Contrail OVSDb ToR Agent

### IN THIS SECTION

- [Overview: High Availability for a ToR Switch | 446](#)
- [High Availability Solution for Contrail ToR Agent | 447](#)
- [Failover Methodology Description | 448](#)
- [Failure Scenarios | 448](#)
- [Redundancy for HAProxy | 450](#)
- [Configuration for ToR Agent High Availability | 451](#)

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

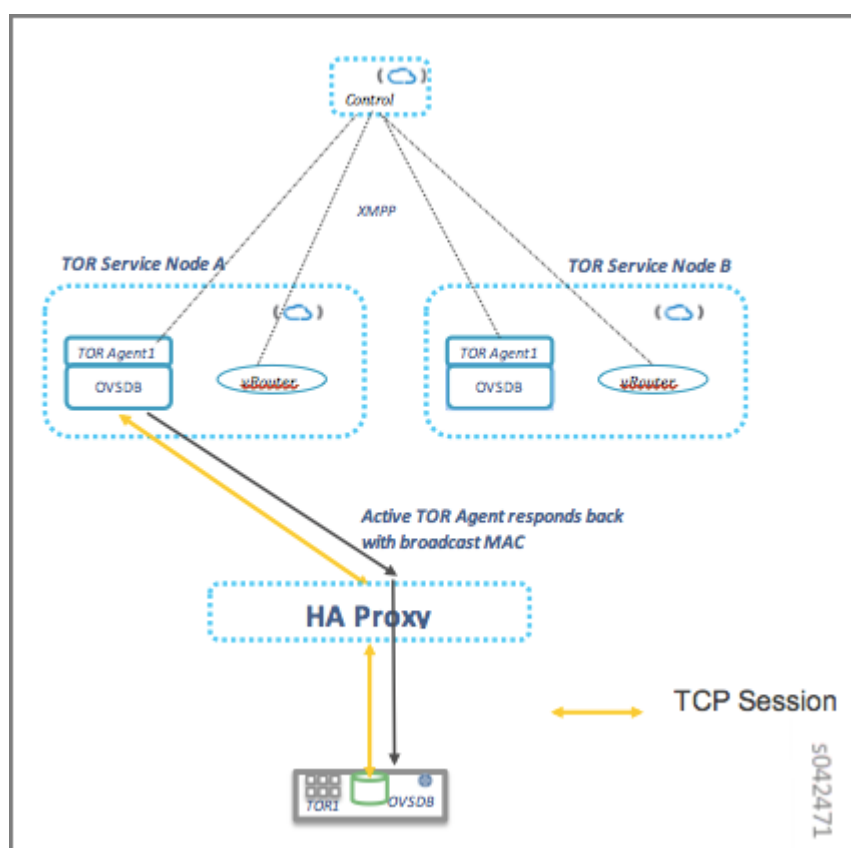
as the vRouter agent for the ToR switch, due to any failure condition in the network or the node, then another ToR agent takes over and manages the ToR switch.

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 53: 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

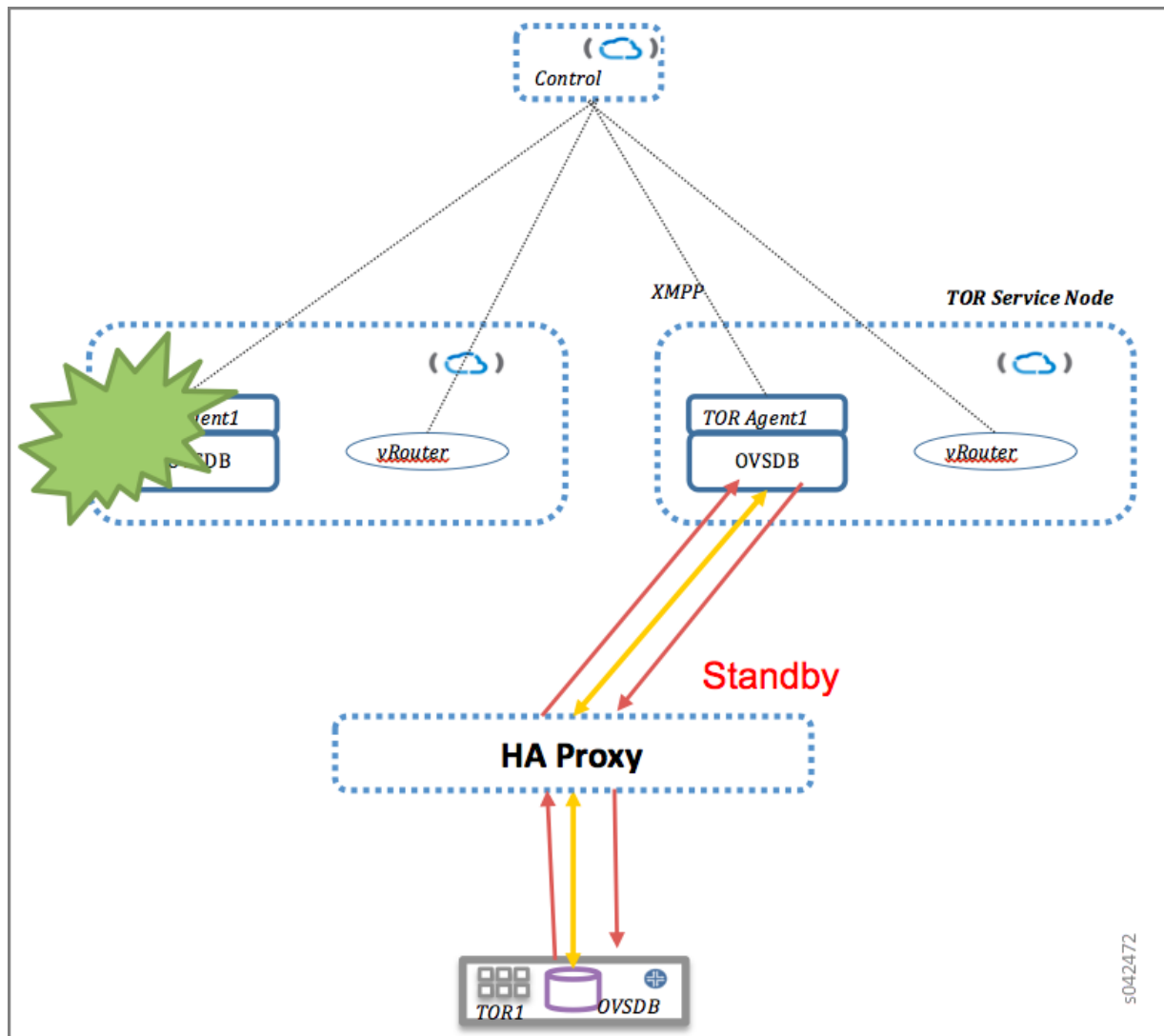
Whenever the HAProxy cannot communicate with the ToR agent, a new SSL connection from the ToR switch is established to the other ToR agent.

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 54: 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 OVSDDB local table to the control node.

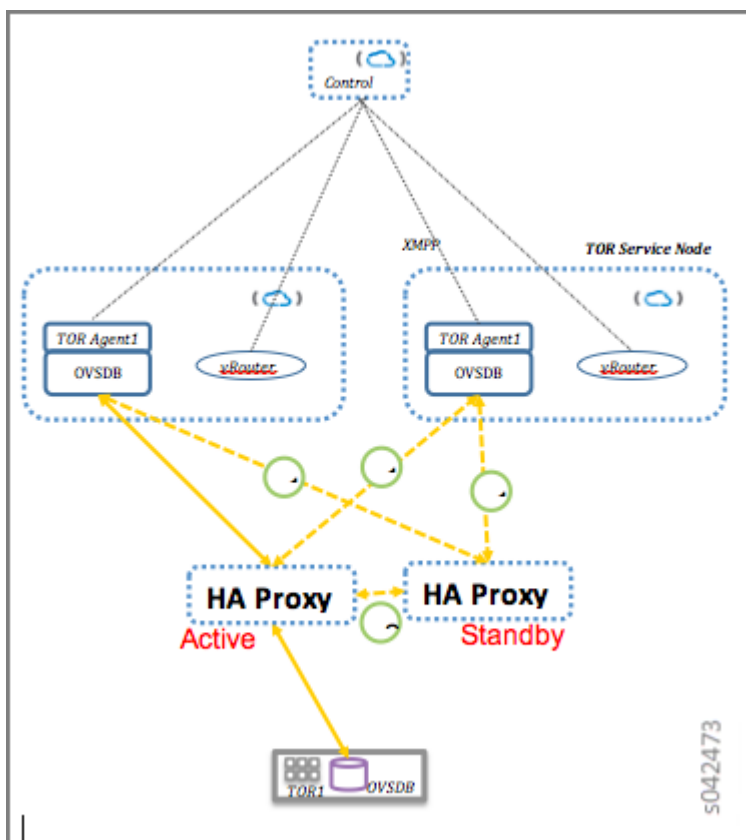
Because the configuration and routes from the control node are already synchronized to the new ToR Services Node (TSN), the new TSN can immediately act on the broadcast traffic from the ToR switch. Any impact to the service is only for the time needed for the SSL connection to be set up and for programming the multicast and unicast routes in the OVSDDB.

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 OVSDDB 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 primary 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 55: Redundancy for HAProxy



If the HAProxy node fails, a standby node becomes the virtual IP primary 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 56 on page 452](#).

Figure 56: Add OVSDB Managed ToR Window

The screenshot shows a window titled "Add OVSDB Managed ToR" with a close button (X) in the top right corner. The window contains the following fields and controls:

- 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 checkbox that is checked.
- SNMP Settings:** A section with a dropdown arrow and the text "SNMP Settings".
  - Version:** Two radio buttons, "2" (selected) and "3".
- Buttons:** "Cancel" and "Save" buttons at the bottom right.

A vertical scrollbar is visible on the right side of the window, with the text "s042502" near the bottom.

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](#)

## Using Device Manager to Manage Physical Routers

### IN THIS SECTION

- [Support for Physical Routers Overview | 453](#)
- [Configuration Model | 453](#)
- [Alternate Ways to Configure a Physical Router | 456](#)
- [Device Manager Configurations | 456](#)
- [Prerequisite Configuration Required on MX Series Device | 457](#)
- [Configuration Scenarios | 457](#)
- [Device Manager Functionality | 458](#)
- [Dynamic Tunnels | 458](#)
- [Extending the Public Network | 466](#)
- [Ethernet VPN Configuration | 468](#)
- [Floating IP Addresses and Source Network Address Translation for Guest Virtual Machines and Bare Metal Servers | 469](#)
- [Samples of Generated Configurations for an MX Series Device | 479](#)

### Support for Physical Routers Overview

A configuration node daemon named Device Manager can be used to manage physical routers in the Contrail system.

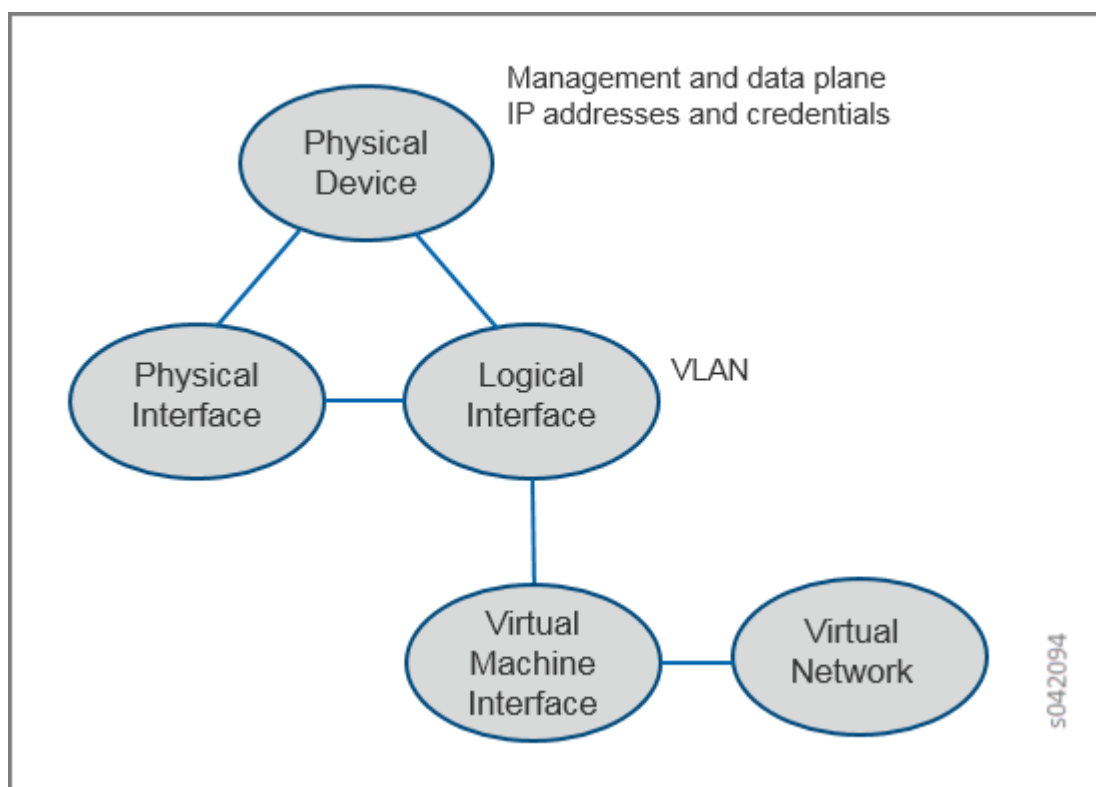
The Device Manager daemon listens to configuration events from the API server, creates any necessary configurations for all physical routers it is managing, and programs those physical routers.

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 57 on page 454](#) depicts the configuration model used in the system. The Physical Router, Physical Interface, and Logical Interface all represent physical router entities.

Figure 57: 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 58: Add Physical Router Window

Add Netconf Managed Physical Router

Name

Vendor

Model

Management IP

Netconf Username

Password

Junos Service Ports

☒

Port

+

SNMP Monitored

☒

Cancel

Save

s042493

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 59: Add Interface Window

**Add Interface**

Type	Name	Parent
Logical		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 - +

Cancel Save

s042092

## 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 489.

## 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 489.

### 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: `_contrail_{l2|l3}_{vn-id}_{vn-name}`
- NAT VRF name: `_contrail_{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 60 on page 459 shows the web user interface used to configure dynamic tunnels.

Figure 60: Edit Global Config Window

**Edit Global Config**

**Encapsulation Priority Order**

VxLAN + -

MPLS Over GRE + -

MPLS Over UDP + -

**BGP Options**

Global ASN 64512

Enable iBGP Auto Mesh ☒

**IP Fabric Subnets** +

s042474 + -

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 <router-ip-address>;

route-distinguisher-id <route-distinguisher-ip-address>;

autonomous-system 64512;

dynamic-tunnels {
    __contrail__ {
        source-address <source-ip-address>

        gre;

        destination-networks {
            <destination-network-1>/24;

            <destination-network-2>/32;

            <destination-network-3>/32;

            <destination-network-4>/32;

            <destination-network-5>/32;

            <destination-network-6>/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 61 on page 461 shows the web user interface used to configure BGP groups.

Figure 61: Edit BGP Router Window

Edit BGP Router

Hostname

tasman

Router Type

Control Node

BGP Router

Vendor ID

mx

IP Address

172.16.184.200

Router ID

10.87.140.107

Autonomous System

64512

Address Families

route-target

inet-vpn

e-vpn

inet6-vpn

Advanced Options

Peer(s) Selection

Available Peer(s)

Configured Peer(s)

Cancel

Save

Figure 62 on page 462 shows the web user interface used to configure the physical router.

Figure 62: Edit Physical Router Window for BGP Groups

Name	tasman		
Vendor	juniper	Model	mx
Management IP	10.87.140.107	VTEP Address	172.16.184.200
BGP Gateway	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 <ip-address>;
    hold-time 90;
    keep all;
    family inet-vpn {
        unicast;
    }
    family inet6-vpn {
        unicast;
    }
    family evpn {
        signaling;
    }
    family route-target;
    neighbor <ip-address>;
    neighbor <ip-address>;
    neighbor <ip-address>;
    neighbor <ip-address>;
}

group __contrail_external__ {
    type external;
    multihop;

```

```
local-address <ip-address>;
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 63 on page 464](#) shows the web user interface for configuring the physical router for extending the private network.

Figure 63: Edit Physical Router Window for Extending Private Networks

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) × vn_private-x2-17 (default-domain:default-project) ×		

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 <ip-address>/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 _contrail_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 _contrail_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 64 on page 467](#) shows the web user interface for extending the public network.

Figure 64: Edit Network Gateway Window

The screenshot shows the 'Edit Network GW1' window with the 'Advanced Options' tab selected. The configuration includes:

- Admin State:** A dropdown menu set to 'Up'.
- Shared:** An unchecked checkbox.
- External:** A checked checkbox.
- DNS Servers:** A section with a 'DNS Servers' label and a '+' icon to add servers.
- VxLAN Identifier:** A dropdown menu set to 'Automatic'.
- Allow Transit:** An unchecked checkbox.
- Flood unknown unicast:** An unchecked checkbox.
- Extend To Physical Router(s):** A checked checkbox.
- Physical Router(s):** A text field containing 'bali' with a close icon (x) to the right.

A vertical ID 's042478' 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.

```
/* 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 {

            <ip-address>/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

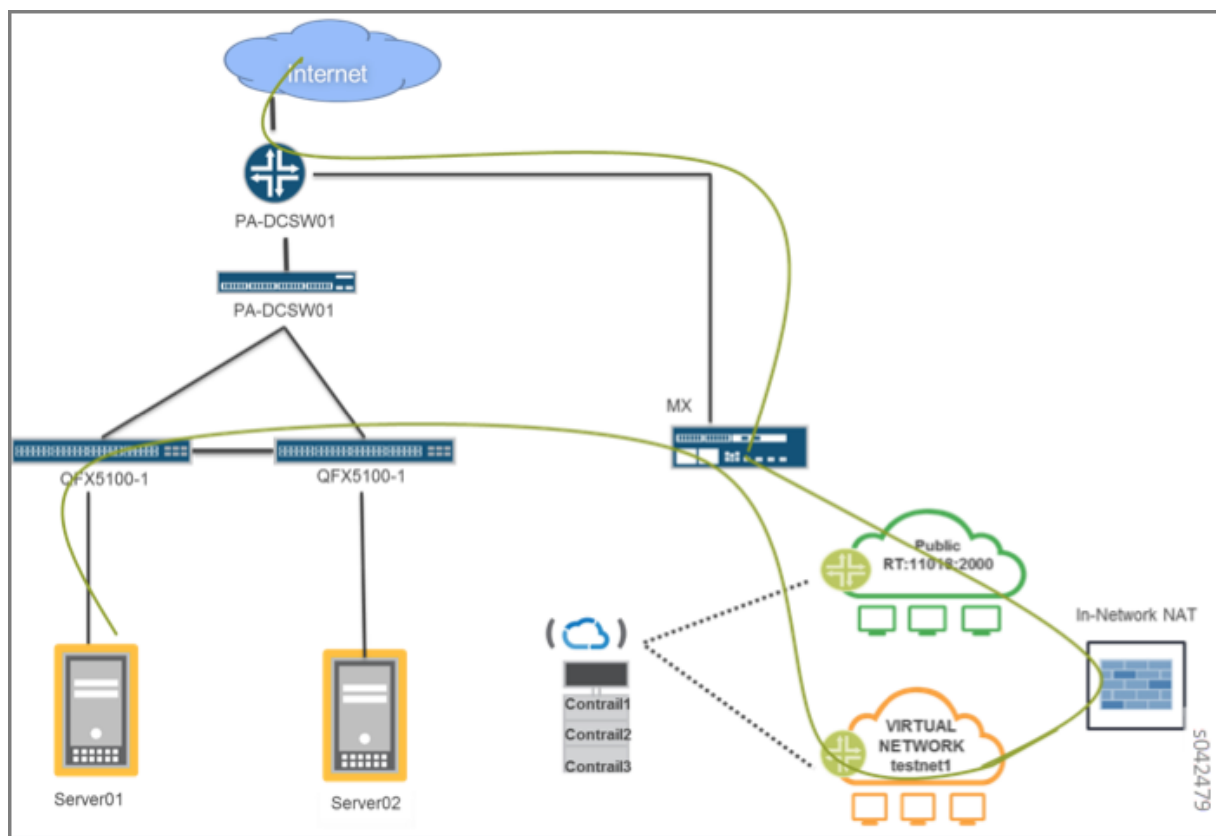
This section describes a bare metal server deployment scenario in which servers are connected to a TOR QFX device inside a private network and an MX Series router is the gateway for the public network connection.

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 65 on page 470](#) 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 65: 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 65 on page 470](#), you must create the necessary physical device, interface, and virtual network configuration that is pushed 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_l3_147_vn_private-x1-63-nat

instance-type vrf;

interface si-2/0/0.293;

vrf-import _contrail_l3_147_vn_private-x1-63-nat-import;

vrf-export _contrail_l3_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 <ip-address>/32 next-hop si-2/0/0.294;

        route <ip-address>/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_l3_147_vn_-sn-rule

match-direction input;

term term<_x_x_x_x> {

    from {

        source-address {

            <ip-address>/32;

        }

    }

    then {

        translated {

            source-prefix <ip-address>/32;

            translation-type {

                basic-nat44;

            }

        }

    }

}

term term<_x_x_x_x> {

    from {

        source-address {

```

```

        <ip-address>/32;

    }

}

then {

    translated {

        source-prefix <ip-address>/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<_x_x_x_x> {

    from {

        destination-address {

            <ip-address>/32;

        }

    }

}

```

```

then {

    translated {

        destination-prefix <ip-address>/32;

        translation-type {

            dnat-44;

        }

    }

}

term term <_x_x_x_x> {

    from {

        destination-address {

            <ip-address>/32;

        }

    }

    then {

        translated {

            destination-prefix <ip-address>/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 {

            <ip-address>/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 {

        <ip-address>/32;

        <ip-address>/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 <ip-address>/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 <ip-address> --api_server_port 8082 --
admin_user <admin user> --admin_password <password> --admin_tenant_name default-domain --op
add_basic
```

Generated configuration for MX Series device:

```
root@host# show groups __contrail__
routing-options {
    route-distinguisher-id <route-distinguisher-ip-address>;
    autonomous-system 64512;
}
protocols {
    bgp {
        group __contrail__ {
            type internal;
            multihop;
            local-address <ip-address>;
            keep all;
            family inet-vpn {
                unicast;
            }
            family inet6-vpn {
                unicast;
            }
            family evpn {
                signaling;
            }
            family route-target;
        }
        group __contrail_external__ {
            type external;
            multihop;
            local-address <ip-address>;
            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_80000008;
                accept;
            }
        }
    }
    policy-statement __contrail__default-domain_default-project_vn1-import {
        term t1 {
            from community target_64200_80000008;
            then accept;
        }
        then reject;
    }
    community target_64200_80000008 members target:64200:80000008;
}
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 <ip-address>/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 <ip-address> --api_server_port 8082 --
admin_user <admin user> --admin_password <password> --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 {
          <ip-address>/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 <ip-address> --api_server_port 8082 --
admin_user <admin user> --admin_password <password> --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 {
                <ip-address>/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 user> --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](#) | 489

## 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': '<ip-address>/24', 'gw' : '<ip-address>', 'device': 'p6p2_1' },
host2 : { 'ip': '<ip-address>/24', 'gw' : '<ip-address>', '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 | 487](#)
- [Provisioning Gateway Mode | 487](#)
- [Configuring Gateway Mode | 488](#)
- [Configuring Gateway Mode and High Availability | 488](#)
- [Scaling | 489](#)

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 | 489](#)
- [REST API for Physical Routers | 490](#)
- [REST API for Physical Interfaces | 492](#)
- [REST API for Logical Interfaces | 493](#)

### 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'<ip-address>',
    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'<ip-address>'  
  
}  
  
}
```

## 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 Device Manager to Manage Physical Routers](#) | 453

# Contrail for Data Center Automation and Fabric Management

## IN THIS CHAPTER

- [Understanding Underlay Management | 496](#)
- [Support for Intent Driven Automation Functionality using Ansible | 497](#)
- [Providing Intent Driven Automation Capabilities on Physical Network Elements | 498](#)
- [Configuring Data Center Gateway | 506](#)

## Understanding Underlay Management

### IN THIS SECTION

- [Benefits of Underlay Management | 496](#)

An underlay network is the physical infrastructure on which an overlay network is built. An overlay network is built on top of and is supported by an underlay network's physical infrastructure. Starting with release 5.0.1, Contrail supports underlay network management. The existing Contrail configuration node can provide intent driven automation capabilities on physical network elements such as ToR and EoR switches, Spines, SDN gateway, and VPN gateways in the data center. In addition, you can perform basic device management functions such as device discovery, device import, image upgrade, device underlay configuration, and topology discovery from the node.

### Benefits of Underlay Management

- Enables basic device management functions from the Contrail configuration node.
- Enables underlay network automation.



- Supports zero touch provisioning (ZTP) of factory-default devices to form an IP Clos network.



**NOTE:** ZTP allows you to provision new devices in your network automatically, with minimal manual intervention.

## RELATED DOCUMENTATION

[Support for Intent Driven Automation Functionality using Ansible | 497](#)

[Providing Intent Driven Automation Capabilities on Physical Network Elements | 498](#)

## Support for Intent Driven Automation Functionality using Ansible

Contrail Release 5.0.1 supports the intent driven automation functionality using Ansible. Basic device management functions such as device discovery, device import, image upgrade, device underlay configuration, and device overlay configuration are implemented as Ansible playbooks and are triggered by an action URL ([/execute-job](#)) on the API server. Custom Ansible python modules such as Contrail Ansible Modules and Vendor Specific Ansible Modules are developed for interactions with the intent driven automation functionality. These python modules are also developed to interact with Virtualized Network Services (VNS) data models and User-Visible Entity (UVE) or Object Log operational data models.

The intent driven automation functionality using Ansible provides the following benefits:

- Multi-vendor support
- Extensible configuration management and automation with a set of plugins
- Option to customize underlay and overlay configuration templates



**NOTE:** All configurations are added by using Jinja templates and therefore can be customized.

## RELATED DOCUMENTATION

[Understanding Underlay Management | 496](#)

[Providing Intent Driven Automation Capabilities on Physical Network Elements | 498](#)

## Providing Intent Driven Automation Capabilities on Physical Network Elements

### IN THIS SECTION

- [Image Management | 498](#)
- [Fabric Management | 500](#)

This topic describes the design and implementation of extending the existing Contrail configuration node to provide intent driven automation capabilities on physical network elements such as ToR and EoR switches, Spines, SDN gateway, and VPN gateways in the data center.

### Image Management

#### IN THIS SECTION

- [Upload a New Device Image | 498](#)
- [Upgrade an Existing Device Image | 499](#)

You can upload a new device image to the Contrail fabric or upgrade an existing device image of any device in the Contrail fabric.

#### Upload a New Device Image

##### Upload a New Device Image

Follow these steps to upload a new device image:

1. Click **Infrastructure>Fabrics>Images**.  
The Device Images page is displayed.
2. Click **Upload**.  
The Upload Image pop-up is displayed.
3. Enter the following information given in Table 1.

Table 17: Upload Image Details

Field	Action
<b>Name</b>	Enter a name for the device image.
<b>Pick a file</b>	Click <b>Upload</b> and navigate to the local directory and select the device image file. Click <b>Open</b> to confirm selection.
<b>Vendor Name</b>	Displays the name of the vendor.
<b>Device Family</b>	Select the device family from the list.
<b>Supported platforms</b>	Select the supported platforms from the list.
<b>OS version</b>	Enter the OS version.
<b>Image MD5</b>	(Optional) Enter MD5 image details.
<b>Image SHA1</b>	(Optional) Enter SHA1 image details.

- Click **Upload** to begin uploading the device image file.

You are redirected to the Device Images page. When the image upload is complete, the device image is listed in Device Images page.

### Upgrade an Existing Device Image

Follow these steps to upgrade an existing device image:

- Click the **Upgrade** icon.  
The Image Upgrade pop-up is displayed.
- From the Compatible Devices pane, select the device you want to upgrade by clicking the device display name.



**NOTE:** You can select more than one device.

The device you select is then moved to the Selected devices to image upgrade pane.



**NOTE:** Devices that are compatible, based on device name and device family, are displayed in the Compatible Devices pane.

3. Click **Upgrade** to start device image upgrade.

## Fabric Management

### IN THIS SECTION

- [Create a Fabric | 501](#)
- [Delete a Fabric | 504](#)
- [Discover a Device | 504](#)
- [Assign Role to a Device | 505](#)
- [Manage Device Configuration | 506](#)
- [View Node Profile Information | 506](#)

You can manage a set of devices, bare metal servers (BMS), and physical network functions (PNF) in Contrail as a fabric. A fabric is a set of devices, and BMS and PNFs that fall under the same data center admin responsibility area. The fabric is linked to different role-based access control (RBAC) profiles for ease of administration and management.

You can provision greenfield devices and brownfield devices by using the Contrail Command user interface (UI).

### Greenfield devices

You can provision new devices to form an IP Clos network. These devices are connected to a management network that is provisioned before device onboarding. The greenfield fabric workflow then zero-touch-provisions all factory-default devices to form an operational IP Clos network with underlay connectivity.

This greenfield fabric workflow includes playbooks that automate the fabric data model creation in the database, DHCP server configuration, generating device bootstrap configuration, uploading device bootstrap configuration to TFTP server, device discovery, node profile auto-assignment, device role assignment, and role-based auto configuration.

### Brownfield devices

You can provision legacy devices or existing devices to form an IP Clos network. Unlike greenfield devices, brownfield devices are manually provisioned before device onboarding. The brownfield fabric workflow includes playbooks that automate the fabric data model creation in the database. You can perform basic device management

functions such as image upgrade, device discovery, device underlay configuration, assign roles to devices, and view node profile information.

You can use the Contrail Command UI to:

Create a Fabric

You can create a new fabric by using the Contrail Command UI. Follow these steps to create a new fabric:

1. Click **Infrastructure>Fabrics**.

The Fabrics page is displayed.

2. Click **Create**.

You are prompted to select a provisioning option.

- Click **New Fabric (Beta)** to deploy new (greenfield) devices.
- Click **Existing Fabric** to import existing (brownfield) devices by discovery.

Click **Provision**.

The Create Fabric page is displayed.

3. If you select **New Fabric (Beta)** as the provisioning option, enter the information as given in Table 2.

Table 18: Provisioning Option - New Fabric (Beta)

Field	Action
Name	Enter a name for the fabric.
Device credentials	Enter root user password.
Overlay ASN (iBGP)	Enter an autonomous system (AS) number.  The AS number can be in the range from 1 through 65535.
Minimum devices to be ZTP'ed	Enter the minimum number of devices you want zero-touch-provisioned (ZTP'ed).

Table 18: Provisioning Option - New Fabric (Beta) *(Continued)*

Field	Action
<b>Node profiles</b>	<p>Add node profiles.</p> <p>You can add more than one node profile.</p> <p>All preloaded node profiles are added to the fabric by default. You can remove a node profile by clicking <b>X</b> on the node profile. For more information, see View Node Profile Information .</p>
<b>Management subnets</b>	<p>Enter the following information to auto-assign management IP addresses to devices:</p> <p><b>CIDR</b>—Enter CIDR network address.</p> <p><b>Gateway</b>—Enter gateway address.</p>
<b>Underlay ASNs (eBGP)</b>	<p>Enter autonomous system (AS) number in the range from 1 through 65535.</p> <ul style="list-style-type: none"> <li>• Enter minimum value in <b>ASN From</b> field.</li> <li>• Enter maximum value in <b>ASN To</b> field.</li> </ul>
<b>Fabric subnets (CIDR)</b>	<p>Enter fabric CIDR address.</p> <p><b>NOTE:</b> Fabric subnets are used to assign IP addresses to interfaces that connect to leaf or spine devices.</p>
<b>Loopback subnets (CIDR)</b>	<p>Enter loopback subnet address.</p> <p><b>NOTE:</b> Loopback subnets are used to auto-assign loopback IP addresses to the fabric devices.</p>

4. If you select **Existing Fabric** as the provisioning option, enter the information as given in Table 3.

Table 19: Provisioning Option - Existing Fabric

Field	Action
<b>Name</b>	Enter a name for the fabric.
<b>Username</b>	Enter a username for the device.
<b>Password</b>	Enter a password for the device.
<b>Overlay ASN (iBGP)</b>	<p>Enter an autonomous system (AS) number.</p> <p>The AS number can be in the range from 1 through 65535.</p>
<b>Node profiles</b>	<p>Add node profiles.</p> <p>You can add more than one node profile.</p> <p>All preloaded node profiles are added to the fabric by default. You can remove a node profile by clicking <b>X</b> on the node profile. For more information, see <a href="#">View Node Profile Information</a>.</p>
<b>Management subnets</b>	<p>Enter the following information:</p> <p><b>CIDR</b>—Enter CIDR network address.</p> <p><b>Gateway</b>—Enter gateway address.</p> <p><b>NOTE:</b> You enter the CIDR address range in the <b>Management subnets</b> field to search for devices. Any device that has a previously configured management IP on the subnet is discovered.</p>
<b>Underlay ASNs (eBGP)</b>	<p>Enter autonomous system (AS) number in the range from 1 through 65535.</p> <ul style="list-style-type: none"> <li>• Enter minimum value in <b>ASN From</b> field.</li> <li>• Enter maximum value in <b>ASN To</b> field.</li> </ul>

Table 19: Provisioning Option - Existing Fabric *(Continued)*

Field	Action
Fabric subnets (CIDR)	Enter fabric CIDR address.  <b>NOTE:</b> Fabric subnets are used to assign IP addresses to interfaces that connect to leaf or spine devices.
Loopback subnets (CIDR)	Enter loopback address.  <b>NOTE:</b> Loopback subnets are used to auto-assign loopback IP addresses to the fabric devices.

5. Click **Next**.

The Discovered devices page is displayed.

### Delete a Fabric

You can delete a fabric by using the Contrail Command UI. Follow these steps to delete a fabric:

1. Click **Fabrics**.

The Fabrics page is displayed.

2. Select the fabric you want removed by selecting the check box next to the name of fabric.



**NOTE:** Contrail Release 5.0.1 does not support bulk deletion of fabric.

3. Click the **Delete** icon at the end of the row to delete a fabric.

The Delete confirmation pop-up is displayed.

4. Click **Delete** to confirm.

### Discover a Device

Device discovery is initiated as soon as you click **Next** on the Fabrics page. The **Device discovery progress** bar on the Discovered devices page displays the progress of the device discovery job. The list of devices discovered is listed in the Discovered devices page.

You can add a discovered device to the fabric by following these steps:

1. Select the device you want to add by selecting the check box next to the device name.





**NOTE:** You can select more than one device.

2. Click **Add**.

The device is added to the fabric.

Click **Next** to assign roles.

The Assign to devices page is displayed.

### Assign Role to a Device

You can assign roles to the devices from the Assign to devices page.

Follow these steps to assign roles to devices:

1. Select the device you want to assign a role to by selecting the check box next to the device name.
2. Click the **Assign** icon at the end of the row to assign roles.



**NOTE:** To assign roles to more than one device at a time, select the devices by selecting the check box next to the device name and then click **Assign Role**.

The Assign role to devices pop-up is displayed.

3. Select a physical role type from the **Physical Role** list.



**NOTE:** Contrail Release 5.0.1 supports only leaf and spine physical underlay roles.

4. Select a routing bridging role from the **Routing Bridging Role** list.



**NOTE:** Contrail Release 5.0.1 supports CRB-Access, CRB-Gateway, and DC-Gateway overlay roles. Contrail Release 5.1 supports the ERB-UCAST-Gateway and CRB-MCAST-Gateway roles. For more information, see [Centrally-Routed Bridging Overlay Design and Implementation](#).

5. Click **Assign** to confirm.
6. Click **Autoconfigure** to initiate the auto-configuration job.

The Autoconfigure page is displayed.

## Manage Device Configuration

After you assign device roles, you initiate the auto-configuration job by clicking **Autoconfigure** on the Assign to devices page. The **Autoconfigure progress** bar on the Discovered devices page displays the progress of the auto-configuration job.

Once the auto-configuration job is completed, click **Finish**.

## View Node Profile Information

You can view basic device information, vendor information, vendor hardware information, supported routing bridging roles, supported physical roles, assigned devices, and node permission information of a node on the Node Profiles page of the Contrail Command UI.

Follow these steps to view node profiles:

1. Click **Infrastructure>Fabrics>Node Profiles**.

The Node Profiles page is displayed.

2. Select the node profile you want to view by clicking the arrow next to the node profile name.

The details and permissions of the node profile are displayed.

By default, all preloaded node profiles are available for devices in a fabric.

## RELATED DOCUMENTATION

[Understanding Underlay Management | 496](#)

[Support for Intent Driven Automation Functionality using Ansible | 497](#)

## Configuring Data Center Gateway

### IN THIS SECTION

- [Configuring QFX Series Devices as Data Center Gateway | 507](#)
- [Configuring MX Series Routers as Data Center Gateway | 520](#)

You can configure a QFX series device and an MX series router as a Data Center Gateway (DC-GW). DC-GW is an overlay role that is assigned to a QFX series switch or an MX series router to:

- Extend private network
- Extend public routable network

You can extend private network and extend public routable network with EVPN Type 5.

For more information on supported QFX series and MX series devices, see [Contrail Networking Supported Hardware Platforms and Associated Roles And Node Profiles](#).

## Configuring QFX Series Devices as Data Center Gateway

### IN THIS SECTION

- [Onboard Brownfield Devices | 507](#)
- [Add Bare Metal Server | 508](#)
- [Create Tenant Virtual Network | 509](#)
- [Add CSN Nodes | 517](#)
- [Create Logical Routers | 518](#)
- [Verification | 520](#)

You can configure a QFX series device as a DC-GW. For more information on supported QFX series devices, see [Contrail Networking Supported Hardware Platforms and Associated Roles And Node Profiles](#).

As an example, follow these steps to configure a QFX10000 device as a DC-GW.

### Onboard Brownfield Devices

Follow the steps provided in the [Onboard Brownfield Devices](#) topic to onboard fabric devices and assign roles to devices.

See [Table 20 on page 508](#) for an example configuration of how you can assign roles to a device.

**Table 20: Assign Roles to Devices**

Device	Physical Role	Routing-Bridging Role
<b>Spine devices</b> <b>QFX10000</b>	spine	CRB-Gateway, Route-Reflector, CRB-MCAST-Gateway, DC-Gateway
<b>Leaf devices</b>	leaf	CRB-Access

Ensure that you assign the DC-Gateway role to the QFX10000 device as shown in [Table 20 on page 508](#).

### Add Bare Metal Server

Follow these steps to add an existing bare metal server (BMS) by using the Contrail Command UI:

1. Click **Workloads>Instances**.  
The Instances page is displayed.
2. Click **Create** to create a new instance.  
The Create Instance page is displayed.
3. Select **Existing Baremetal Server** as the Server Type.
4. Enter the following information in the Create Existing Baremetal Server pane:

**Table 21: Add Existing Bare Metal Server Information**

Field	Action
<b>Instance Name</b>	Displays the name of the BMS instance.
<b>Baremetal Node</b>	Select a bare metal node.
<b>Interface</b>	Select an interface from the list.
<b>IP Address</b>	Enter IP address of the instance.
<b>Virtual Network</b>	Select a virtual network from the list.

Table 21: Add Existing Bare Metal Server Information (*Continued*)

Field	Action
<b>VLAN ID</b>	Enter VLAN ID.
<b>Select Security Groups</b>	Select <b>default</b> security group from the list.
<b>Port Profile</b>	Select a port profile from the list.
<b>Native/Untagged</b>	Select this check box to receive untagged packets without native VLAN ID.

Figure 66: Existing Bare Metal Server

Server Type ⓘ

☐ Virtual Machine
 ☐ New Baremetal Server
 ☒ Existing Baremetal Server

Create Existing Baremetal Server

Instance Name\*

Baremetal Node\*

Associate interfaces

Interface	IP Address	Virtual Network*	VLAN ID*
<input type="text"/>	<input type="text" value="Enter valid IPv4"/>	<input type="text"/>	<input type="text" value="1"/>
Select Security Groups ⓘ	Port Profile	<input type="checkbox"/> Native/Untagged	
<input type="text"/>	<input type="text"/>		

+ Add

+ Add

Create Cancel

5. Click **Create** to confirm.

## Create Tenant Virtual Network

A virtual network is a collection of endpoints, such as virtual machine instances, that can communicate with each other. You can also connect virtual networks to your on-premises network. A virtual network in a EVPN VXLAN data center corresponds to a bridge domain for one tenant in a multi-tenant data center fabric.

Follow these steps to create a tenant virtual network from the Contrail Command user interface (UI).

1. Navigate to **Overlay>Virtual Networks**.

The All Networks page is displayed.

2. Click **Create** to create a network.

The Create Virtual Network page is displayed.

3. Enter a name for the network in the **Name** field.

4. Select VN Fabric Type.

Select **Routed** to enable routed virtual network functionality. A routed virtual network represents a layer 3 subnet between the fabric (border gateway) and the third-party physical network device.

For more information, see [Using Static, eBGP, PIM, and OSPF Protocols to Connect to Third-Party Network Devices](#).

Select **Switched** (default option) for tenant virtual network on leaf, bare metal server, or vRouter.

5. Select network policies from the **Network Policies** list. You can select more than one network policy.

Network policies provide connectivity between virtual networks by allowing or denying specified traffic. They define the access control lists to virtual networks. To create a new network policy, navigate to **Overlay>Network Policies**.

For more information on creating network policies, see [Create Network Policy](#).



**NOTE:** You can attach a network policy to the virtual network after you have created the virtual network.

6. Select any one of the following preferred allocation mode.

- Flat subnet only
- Flat subnet preferred
- (Default) User defined subnet only
- User defined subnet preferred

An allocation mode indicates how you choose a subnet. You select **Flat subnet only** or **Flat subnet preferred** allocation mode when the subnet is shared by multiple virtual networks. However, you select **(Default) User defined subnet only** or **User defined subnet preferred** allocation mode when you want to define a subnet range.

7. Enter subnet information as given in [Table 22 on page 511](#).

Table 22: Subnet Information

Field	Action
<b>Network IPAM</b>	Select the IP address management method that controls IP address allocation, DNS, and DHCP for the subnet.
<b>CIDR</b>	Enter the overlay subnet CIDR.
<b>Allocation Pools</b>	Enter a list of ranges of IP addresses for vRouter-specific allocation.
<b>Gateway</b>	Enter the gateway IP address of the overlay subnet. This field is disabled by default. To configure this field, uncheck Auto Gateway.
<b>Service Address</b>	Specify the user configured IP address for DNS Service instead of the default system allocated one.
<b>Auto Gateway</b>	This check box is enabled by default and gateway address is allocated by the system. When this box is unchecked, gateway address is user configurable.
<b>DHCP</b>	Select this check box if you want Contrail to provide DHCP service.
<b>DNS</b>	Select this check box if you want the vRouter agent to provide DNS service.

8. Enter host route information.

Host routes are a list of prefixes and next hops that are passed to the virtual machine through DHCP.

a. **Route Prefix**—Enter a full CIDR value with an IP address and a subnet mask. For example, 10.0.0.0/24.

b. **Next Hop**—Enter next hop address.

9. Enter floating IP pool information.

A floating IP address is an IP address (typically public) that can be dynamically assigned to a running virtual instance. You can configure floating IP address pools in project networks, then allocate floating IP addresses from the pool to virtual machine instances in other virtual networks.

a. **Pool Name**—Enter pool name.

b. **Projects**—Select project from the list.

10. Enter fat flows information. See [Table 23 on page 512](#).

You can apply fat flows to all VMIs under the configured VN. Fat flows help reduce the number of flows that are handled by Contrail.

**Table 23: Configure Fat Flow**

Field		Action
<b>Protocol</b>		Select the application protocol.
<b>Port</b>		<p>Enter a value between 0 through 65,535. Enter 0 to ignore both source and destination port numbers.</p> <p><b>NOTE:</b> If you select ICMP as the protocol, the <b>Port</b> field is not enabled.</p>
<b>Ignore Address</b>		<p>Configure fat flows to support aggregation of multiple flows into a single flow by ignoring source and destination ports or IP addresses. If you select Destination, only the Prefix Aggregation Source fields are enabled. If you select Source, only the Prefix Aggregation Destination fields are enabled. If you select the None (selected by default), both Prefix Aggregation Source and Prefix Aggregation Destination fields are enabled.</p>
<b>Prefix Aggregation Source</b>	<b>Source Subnet</b>	<p>Enter the source IP address.</p> <p>Ensure that the source subnet of the flows match. For example, enter 10.1.0.0/24 to create fat flows with 10.1.0.0/24 as the subnet. The valid subnet mask range is /8 through /32.</p> <p><b>NOTE:</b> For packets from the local virtual machine, source refers to the source IP of the packet. For packets from the physical interface, source refers to the destination IP of the packet.</p>
	<b>Prefix</b>	<p>Enter source subnet prefix length.</p> <p>The prefix length you enter is used to aggregate flows matching the source subnet. For example, when the source subnet is 10.1.0.0/16 and prefix length is 24, the flows matching the source subnet is aggregated to 10.1.x.0/24 flows. The valid the prefix length range is /(subnet mask of the source subnet) through /32.</p>



Table 23: Configure Fat Flow (Continued)

Field		Action
Prefix Aggregation Destination	Destination Subnet	<p>Enter the destination IP address.</p> <p>Ensure that the destination subnet of the flows match. Enter 10.1.0.0/24 to create fat flows with 10.1.0.0/24 as the subnet. The valid subnet mask range is /8 through /32.</p> <p><b>NOTE:</b> For packets from the local virtual machine, destination refers to the destination IP of the packet. For packets from the physical interface, destination refers to the source IP of the packet.</p>
	Prefix	<p>Enter the destination subnet prefix length.</p> <p>The prefix length you enter is used to aggregate flows matching the destination subnet. For example, when the source subnet is 10.1.0.0/16 and prefix length is 24, the flows matching the source subnet is aggregated to 10.1.x.0/24 flows. The valid prefix length range is /(subnet mask of the destination subnet) through /32.</p>

**11.** Enter routing policy and bridge domain information as given below.

**a.** Select routing policy from the **Routing Policies** list.

To create a routing policy, navigate to **Overlay>Routing>Routing Policy**.

**b.** Define a list of route target prefixes.

Enter an IP address in the ASN field and Target in the range 0 through 65,535, or ASN in the range 1 through 65,535 and Target in the range 1 through 4,294,967,295 if 4-byte ASN is disabled. If 4-byte ASN is enabled, enter ASN in the range 1 through 4,294,967,295 and Target in the range 0 through 65,535.

**c.** Define export route targets.

You can advertise the matched routes from the local virtual routing and forwarding (VRF) table to the MPLS routing table.

Enter an IP address in the ASN field and Target in the range 0 through 65,535, or ASN in the range 1 through 65,535 and Target in the range 1 through 4,294,967,295 if 4-byte ASN is disabled. If 4-byte ASN is enabled, enter ASN in the range 1 through 4,294,967,295 and Target in the range 0 through 65,535.

**d.** Define import route targets.

Import the matched routes from the MPLS routing table and to the local virtual routing and forwarding (VRF) table.

Enter an IP address in the ASN field and Target in the range 0 through 65,535, or ASN in the range 1 through 65,535 and Target in the range 1 through 4,294,967,295 if 4-byte ASN is disabled. If 4-byte ASN is enabled, enter ASN in the range 1 through 4,294,967,295 and Target in the range 0 through 65,535.

- e. Enter bridge domain information. See [Table 24 on page 514](#).

A bridge domain is a set of logical interfaces that share the same flooding or broadcast characteristics.

**Table 24: Bridge Domains**

Field	Action
<b>Name</b>	Enter a name for the Layer 2 or Layer 3 bridge domain.
<b>I-SID</b>	Enter a Service Identifier in the range from 1 through 16777215.
<b>MAC Learning</b>	<p>Enable or disable MAC learning.</p> <p>MAC learning is the process of obtaining the MAC addresses of all the nodes in a virtual network. It is enabled by default.</p>
<b>MAC Limit</b>	Configure the maximum number of MAC addresses that can be learned.
<b>MAC Move Limit</b>	<p>Configure the maximum number of times a MAC address move occurs in the MAC move time window.</p> <p>A MAC move is when a MAC address appears on a different physical interface or within a different unit of the same physical interface.</p>
<b>Time Window (secs)</b>	<p>Configure the period of time over which the MAC address move occurs.</p> <p>The default period is 10 seconds.</p>
<b>Aging Time (secs)</b>	<p>Configure the MAC table aging time, the maximum time that an entry can remain in the Ethernet Switching table before it is removed.</p> <p>The default time period is 300 seconds.</p>

12. Enter advanced configuration information as given in [Table 25 on page 515](#).

**Table 25: Advanced Configuration**

Field	Action
<b>Admin State</b>	Select the administrative state of the virtual network.
<b>Reverse Path Forwarding</b>	Enable or disable Reverse Path Forwarding (RPF) check for the virtual network.
<b>Shared</b>	Select to share the virtual network with all tenants.
<b>External</b>	Select the check box to make the virtual networks reachable externally.
<b>Allow Transit</b>	Select to enable the transitive property for route imports.
<b>Mirroring</b>	Select to mark the virtual network as a mirror destination network.
<b>Flood Unknown Unicast</b>	<p>Select to flood the network with packets with unknown unicast MAC address.</p> <p>By default, the packets are dropped.</p>
<b>Multiple Service Chains</b>	Select to allow multiple service chains within two networks in a cluster.
<b>IP Fabric Forwarding</b>	Select to enable fabric based forwarding.
<b>Forwarding Mode</b>	Select the packet forwarding mode for the virtual network.
<b>Extend to Physical Router(s)</b>	<p>Select the physical router to which you want to extend the logical router.</p> <p>The physical router provides routing capability to the logical router.</p>
<b>Static Route(s)</b>	Select the static routes to be added to this virtual network.

Table 25: Advanced Configuration (*Continued*)

Field	Action
<b>QoS</b>	Select the QoS to be used for this forwarding class.
<b>Security Logging Object(s)</b>	Select the security logging object configuration for specifying session logging criteria.
<b>ECMP Hashing Fields</b>	<p>Configure one or more ECMP hashing fields.</p> <p>When configured all traffic destined to that VN will be subject to the customized hash field selection during forwarding over ECMP paths by vRouters.</p>
<b>PBB Encapsulation</b>	Select to enable Provider Backbone Bridging (PBB) EVPN tunneling on the network.
<b>PBB ETree</b>	<p>Select to enable PBB ETREE mode on the virtual network which allows L2 communication between two end points connected to the vRouters.</p> <p>When the check box is deselected, end point communication happens through an L3 gateway provisioned in the remote PE site.</p>
<b>Layer2 Control Word</b>	Select to enable adding control word to the Layer 2 encapsulation.
<b>SNAT</b>	Select to provide connectivity to the underlay network by port mapping.
<b>MAC Learning</b>	<p>Enable or disable MAC learning.</p> <p>MAC learning is the process of obtaining the MAC addresses of all the nodes in a virtual network. It is enabled by default.</p>
<b>Provider Network</b>	<p>Select the provider network.</p> <p>The provider network specifies VLAN tag and the physical network name.</p>

**Table 25: Advanced Configuration (Continued)**

Field	Action
<b>IGMP enable</b>	Enable or disable IGMP.
<b>Multicast Policies</b>	Select the multicast policies.  To create a policy, navigate to <b>Overlay&gt;Multicast Policies</b> .
<b>Max Flows</b>	Enter the maximum number of flows permitted on each virtual machine interface of the virtual network.

**13. Click Create.**

The All Networks page is displayed. The virtual network that you created is displayed on this page.

**Add CSN Nodes**

Follow these steps to add CSN Nodes to the fabric by using the Contrail Command UI:

Navigate to the EVPN fabric you provisioned.

**1. Click the fabric name, and then click the fabric device.**

The Fabric Device page is displayed.

**2. Enter the following information:****Table 26: Add CSN Node to Fabric Device Information**

Field	Action
<b>Management IP</b>	Enter management IP address.
<b>VTEP Address</b>	Enter VTEP address.
<b>Loopback IP</b>	Enter loopback IP address.
<b>BGP Router</b>	Select BGP router from the list.
<b>Virtual Router Type</b>	Select virtual router type from the list.

Table 26: Add CSN Node to Fabric Device Information *(Continued)*

Field	Action
Existing CSN	Select existing CSN from the list.

3. Click **Save** to confirm changes to the fabric.

### Create Logical Routers

A logical router replicates the functions of a physical router. It connects multiple virtual networks. A logical router performs a set of tasks that can be handled by a physical router, and contains multiple routing instances and routing tables.

Follow these steps to create a logical router (LR).

1. Click **Overlay>Logical Routers**.

The Logical Routers page is displayed.

2. Click **Create**.

The Create Logical Router page is displayed.

3. Enter the following information.

Field	Action
<b>Name</b>	Enter a name for the Logical Router.
<b>Admin State</b>	Select the administrative state that you want the device to be in when the router is activated.  <b>Up</b> is selected by default.
<b>Extend to Physical Router</b>	Select the physical router(s) to which you want to extend virtual networks or routed virtual networks to, from the Extend to Physical Router list.  A physical router provides routing capability to the logical router.
<b>Logical Router Type</b>	Select <b>SNAT Routing</b> or <b>VXLAN Routing</b> from the list.

*(Continued)*

Field	Action
<b>Connected Networks</b>	Select the networks that you want to connect this logical router to.
<b>Public Logical Router</b>	(Optional) Select this check box if you want the logical router to function as a public logical router.
<b>VxLAN Network Identifier</b>	Enter VXLAN network identifier in the range from 1 through 16,777,215.  This field is disabled by default.
<b>Route Target(s)</b>	<p>Click <b>+Add</b> to add route targets.</p> <p>Enter Autonomous System (AS) number in the <b>ASN</b> field.</p> <ul style="list-style-type: none"> <li>Enter ASN in the range of 1-4,294,967,295, when <b>4 Byte ASN</b> is enabled in <b>Global Config</b>.</li> <li>Enter ASN in the range of 1-65,535, when <b>4 Byte ASN</b> is disabled.</li> <li>You can also add suffix <i>L</i> or <i>l</i> (<i>lower-case L</i>) at the end of a value in the <b>ASN</b> field to assign an AS number in 4-byte range. Even if the value provided in the <b>ASN</b> field is in the range of 1-65,535, adding <i>L</i> or <i>l</i> (<i>lower-case L</i>) at the end of the value assigns the AS number in 4-byte range. If you assign the <b>ASN</b> field a value in the 4-byte range, you must enter a value in the range of 0-65,535 in the <b>Target</b> field.</li> </ul> <p>Enter route target in the <b>Target</b> field.</p> <ul style="list-style-type: none"> <li>Enter route target in the range of 0-65,535, when <b>4 Byte ASN</b> is enabled and <b>ASN</b> field is assigned a 4-byte value.</li> <li>Enter route target in the range of 0-4,294,967,295, when the <b>ASN</b> field is assigned a 2-byte value.</li> </ul>

4. Click **Create** to create the logical router.

The Logical Routers page is displayed.

5. Repeat Step 3 and Step 4 to create another logical router.



**NOTE:** The router\_interface object (Virtual Port) is created as part of the LR creation. While planning the Virtual Network IP address scheme, you must be aware that an extra one IP address is required for the router\_interface object which gets created automatically.

## Verification

EVPN type 5 configuration is pushed to QFX10000 switch as a DC-GW.

Figure 67: EVPN Type 5 Configuration

```
set groups _contrail_overlay_evpn_ interfaces irb unit 5 proxy-macip-advertisement
set groups _contrail_overlay_evpn_ interfaces irb unit 5 family inet address 10.x7.x8.xx/28 virtual-gateway-address 10.x7.7x.1xx
set groups _contrail_overlay_evpn_ protocols evpn vni-options vni 5 vrf-target target:64512:8000004
set groups _contrail_overlay_evpn_ protocols evpn encapsulation vxlan
set groups _contrail_overlay_evpn_ policy-options policy-statement _contrail_vn-public-l2-5-import term t1 from community target_64512_8000004
set groups _contrail_overlay_evpn_ policy-options policy-statement _contrail_vn-public-l2-5-import term t1 then accept
set groups _contrail_overlay_evpn_ policy-options policy-statement _contrail_vn-public-l2-5-export term t1 then community add target_64512_8000004
set groups _contrail_overlay_evpn_ policy-options policy-statement _contrail_vn-public-l2-5-export term t1 then accept
set groups _contrail_overlay_evpn_ policy-options policy-statement _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6-import term t1 from community target_64512_8000005
set groups _contrail_overlay_evpn_ policy-options policy-statement _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6-import term t1 then accept
set groups _contrail_overlay_evpn_ policy-options policy-statement _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6-export term t1 then community add target_64512_8000005
set groups _contrail_overlay_evpn_ policy-options policy-statement _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6-export term t1 then accept
set groups _contrail_overlay_evpn_ policy-options community target_64512_8000004 members target:64512:8000004
set groups _contrail_overlay_evpn_ policy-options community target_64512_8000005 members target:64512:8000005
set groups _contrail_overlay_evpn_ switch-options vtep-source-interface lo0.0
set groups _contrail_overlay_evpn_ switch-options route-distinguisher x.5.x.5:1
set groups _contrail_overlay_evpn_ switch-options vrf-import _contrail_vn-public-l2-5-import
set groups _contrail_overlay_evpn_ switch-options vrf-import _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6-import
set groups _contrail_overlay_evpn_ switch-options vrf-export _contrail_vn-public-l2-5-export
set groups _contrail_overlay_evpn_ switch-options vrf-export _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6-export
set groups _contrail_overlay_evpn_ switch-options vrf-target Target:64512:1
set groups _contrail_overlay_evpn_ vlans bd-5 vlan-id 5
set groups _contrail_overlay_evpn_ vlans bd-5 l3-interface irb.5
set groups _contrail_overlay_evpn_ vlans bd-5 vlan vni 5
set groups _contrail_overlay_evpn_type5_ interfaces lo0 unit 1006 family inet address 12x.x.0.1/32
set groups _contrail_overlay_evpn_type5_ forwarding-options family inet filter input redirect_to_public_vrf_filter
set groups _contrail_overlay_evpn_type5_ protocols evpn default-gateway no-gateway-community
set groups _contrail_overlay_evpn_type5_ firewall family inet filter redirect_to_public_vrf_filter term term-100 then routing-instance _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6
set groups _contrail_overlay_evpn_type5_ firewall family inet filter redirect_to_public_vrf_filter term default-term then accept
set groups _contrail_overlay_evpn_type5_ routing-instances _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6 instance-type vrf
set groups _contrail_overlay_evpn_type5_ routing-instances _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6 interface lo0.1006
set groups _contrail_overlay_evpn_type5_ routing-instances _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6 interface irb.5
set groups _contrail_overlay_evpn_type5_ routing-instances _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6 vrf-import _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6-import
set groups _contrail_overlay_evpn_type5_ routing-instances _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6 vrf-export _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6-export
set groups _contrail_overlay_evpn_type5_ routing-instances _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6 routing-options static route 0.0.0.0/0 next-hop e_inet.0
set groups _contrail_overlay_evpn_type5_ routing-instances _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6 protocols evpn ip-prefix-routes advertise direct
set groups _contrail_overlay_evpn_type5_ routing-instances _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6 protocols evpn ip-prefix-routes encapsulation vxlan
set groups _contrail_overlay_evpn_type5_ routing-instances _contrail__contrail_lr_internal_vn_23640071-2302-4728-8424-5528df330ae8_-l3-6 protocols evpn ip-prefix-routes vni 100
```

## Configuring MX Series Routers as Data Center Gateway

### IN THIS SECTION

- Onboard Brownfield Devices | 521
- Create Virtual Network | 521

You can configure an MX series router as a DC-GW. You must ensure that you assign the DC-Gateway routing-bridging role to the MX series router during device onboarding. For more information on



supported MX series routers, see [Contrail Networking Supported Hardware Platforms and Associated Roles And Node Profiles](#).

Follow these steps to configure an MX series router as a DC-GW.

### Onboard Brownfield Devices

Follow the steps provided in the [Onboard Brownfield Devices](#) topic to onboard fabric devices and assign roles to devices.

Ensure that you also assign DC-Gateway routing-bridging role to the MX series router (spine device) while assigning device roles.

### Create Virtual Network

After you have onboarded fabric devices and assigned roles to devices, you create a virtual network and extend it to the MX series router.

Follow these steps to create a virtual network and extend it to MX series router.

1. Navigate to **Overlay>Virtual Networks** and click **Create**.

The Create Virtual Network page is displayed.

2. Enter a name for the network in the **Name** field.

3. Select VN Fabric Type.

Select **Routed** to enable routed virtual network functionality. A routed virtual network represents a layer 3 subnet between the fabric (border gateway) and the third-party physical network device. For more information, see [Using Static, eBGP, PIM, and OSPF Protocols to Connect to Third-Party Network Devices](#).

Select **Switched** (default option) for tenant virtual network on leaf, bare metal server, or vRouter.

4. Enter subnet information as given in [Table 27 on page 521](#).

**Table 27: Subnet Information**

Field	Action
<b>Network IPAM</b>	Select the IP address management method that controls IP address allocation, DNS, and DHCP for the subnet.
<b>CIDR</b>	Enter the overlay subnet CIDR.

5. Click **Advanced** to view the advance configuration section.

6. Select the **External** check box to make the virtual network reachable externally.

7. Select the MX series router from the Extend to Physical Router(s) list.
8. Click **Create** to save configuration.

The MX series router is now configured as a DC-GW.

After you configure an MX series router as a DC-GW, you can enable DNAT. For more information on enabling DNAT in a DC-GW, see [Destination Network Address Translation for Bare Metal Servers](#).

## RELATED DOCUMENTATION

*Fabric Overview*

---

[Edge-Routed Bridging for QFX Series Switches](#)

---

[Destination Network Address Translation for Bare Metal Servers](#)

# 3

PART

## Configuring Contrail

---

- [Configuring Virtual Networks | 524](#)
  - [Example of Deploying a Multi-Tier Web Application Using Contrail | 558](#)
  - [Configuring Services | 573](#)
  - [Configuring Service Chaining | 606](#)
  - [Examples: Configuring Service Chaining | 658](#)
  - [Adding Physical Network Functions in Service Chains | 684](#)
  - [QoS Support in Contrail | 694](#)
  - [BGP as a Service | 706](#)
  - [Load Balancers | 712](#)
  - [Optimizing Contrail | 744](#)
-

# Configuring Virtual Networks

## IN THIS CHAPTER

- Creating Projects in OpenStack for Configuring Tenants in Contrail | 524
- Creating a Virtual Network with Juniper Networks Contrail | 526
- Creating a Virtual Network with OpenStack Contrail | 530
- Creating an Image for a Project in OpenStack Contrail | 532
- Creating a Floating IP Address Pool | 536
- Using Security Groups with Virtual Machines (Instances) | 538
- Support for IPv6 Networks in Contrail | 543
- Configuring EVPN and VXLAN | 547
- Support for EVPN Route Type 5 | 556

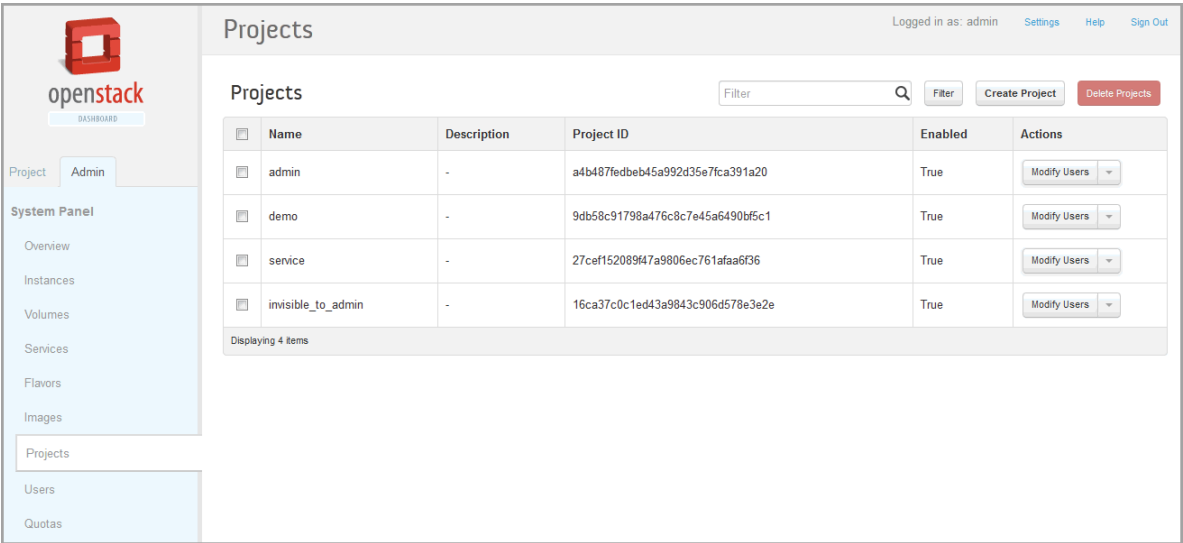
## Creating Projects in OpenStack for Configuring Tenants in Contrail

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.

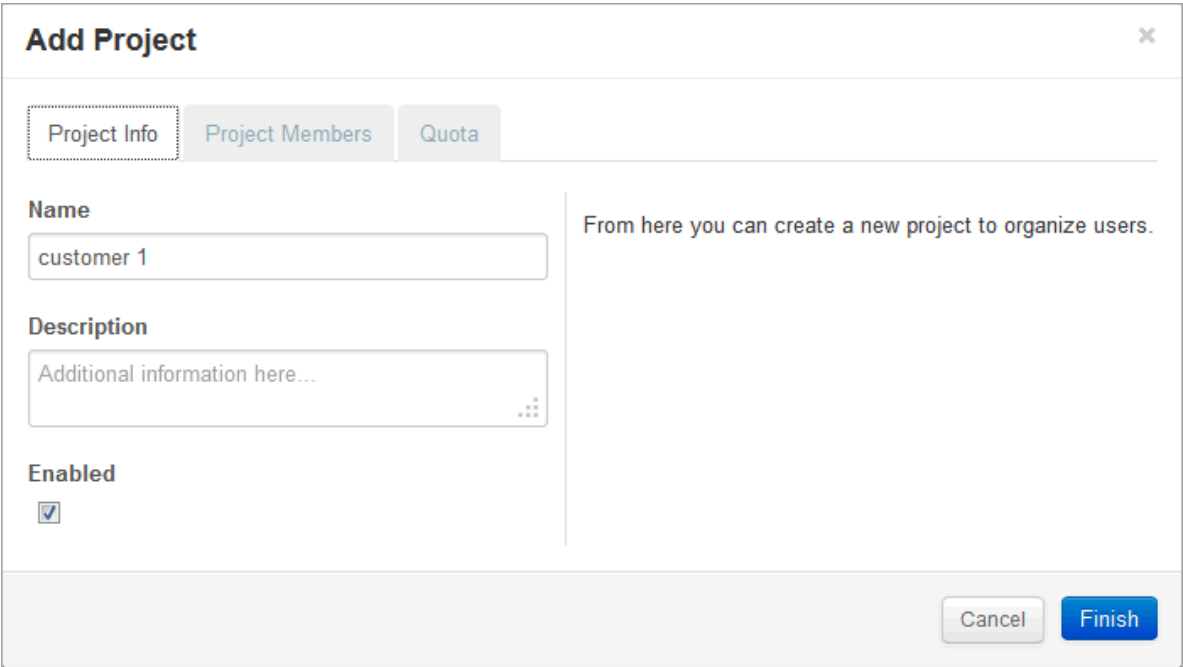
1. Click the **Admin** tab on the OpenStack dashboard, then click the **Projects** link to access the **Projects** page; see [Figure 68 on page 525](#).

Figure 68: OpenStack Projects



2. In the upper right, click the **Create Project** button to access the **Add Project** window; see [Figure 69 on page 525](#).

Figure 69: Add Project



3. In the **Add Project** window, on the **Project Info** tab, enter a **Name** and a **Description** for the new project, and select the **Enabled** check box to activate this project.

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 70 on page 527](#).

Figure 70: Add IP Address Management

Add IP Address Management

Name

IPAM Name

DNS Method

Default

NTP Server IP

Domain Name

Cancel

Save

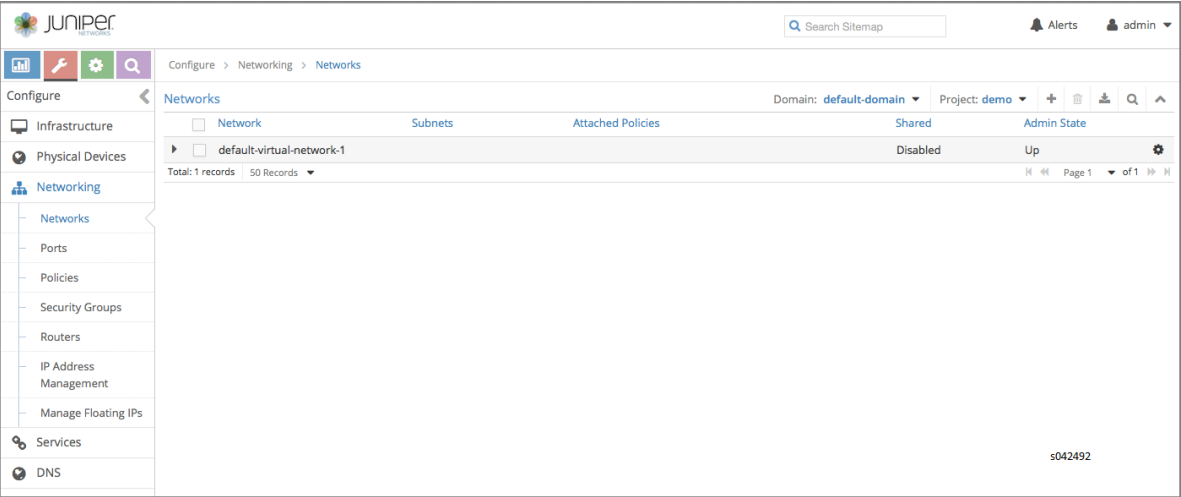
2. Complete the fields in **Add IP Address Management**: The fields are described in [Table 28 on page 527](#).

Table 28: Add IP Address Management Fields

Field	Description
Name	Enter a name for the IPAM you are creating.
DNS Method	Select from a list the domain name server method for this IPAM: <b>Default</b> , <b>Virtual DNS</b> , <b>Tenant</b> , or <b>None</b> .
NTP Server IP	Enter the IP address of an NTP server to be used for this IPAM.
Domain Name	Enter a domain name to be used for this IPAM.

3. You must create a network policy first, as you need a network policy to create a virtual network. Follow the steps described in [Creating a Network Policy—Juniper Networks Contrail](#) to create a network policy.
4. Select **Configure > Networking > Networks** to access the **Configure Networks** page; see [Figure 71 on page 528](#).

Figure 71: Configure Networks




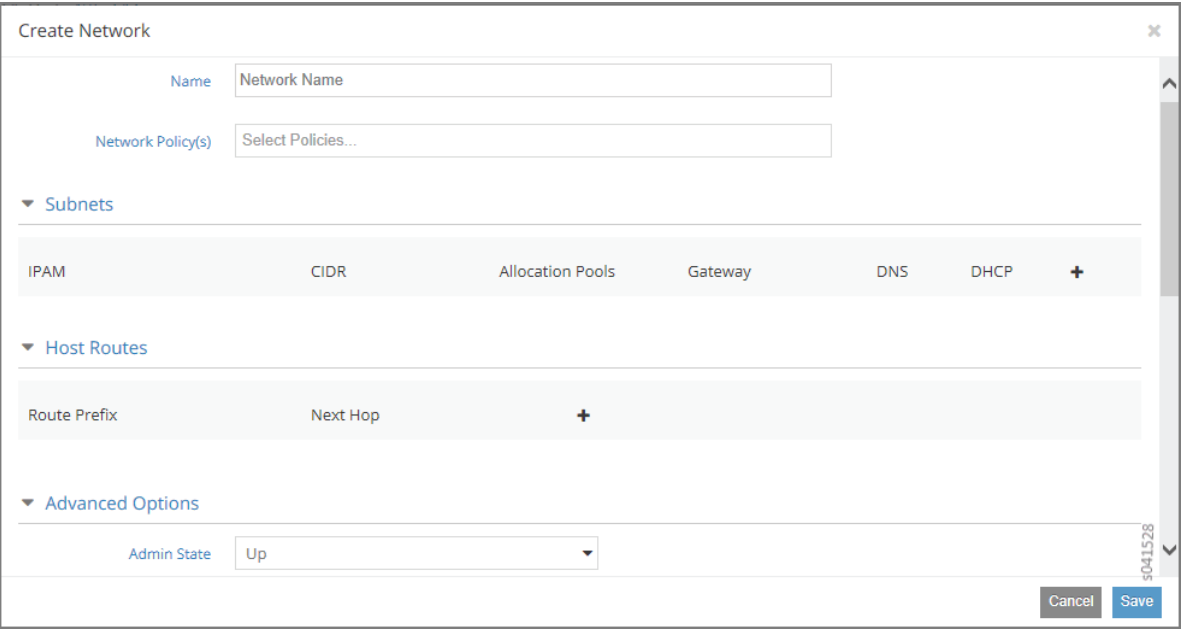
5. Verify that your project is displayed as active in the upper-right field, then click the  icon. The **Create Network** window is displayed. See [Figure 72 on page 528](#). Use the scroll bar to access all sections of this window.

Figure 72: Create Network



6. 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 29 on page 529](#).



Table 29: 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.

7. To save your network, click the **Save** button, or click **Cancel** to discard your work and start over.

## RELATED DOCUMENTATION

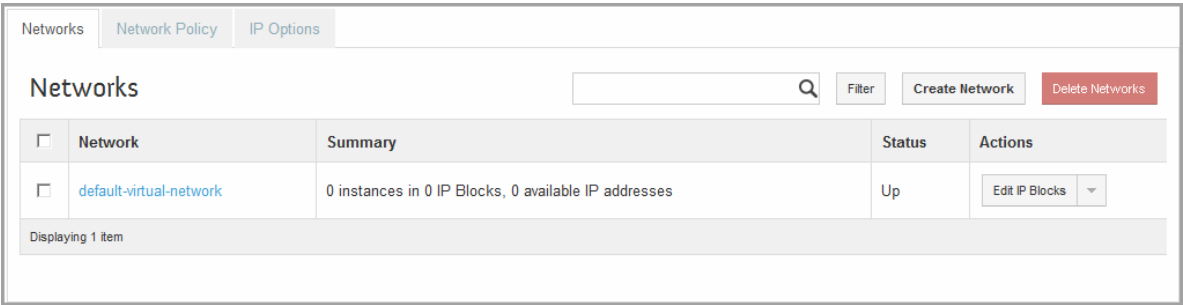
*Creating an Image for a Project in OpenStack Contrail*

## 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 73 on page 530](#).

Figure 73: 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 74 on page 530](#) and [Figure 75 on page 531](#).

Figure 74: Create Network Window

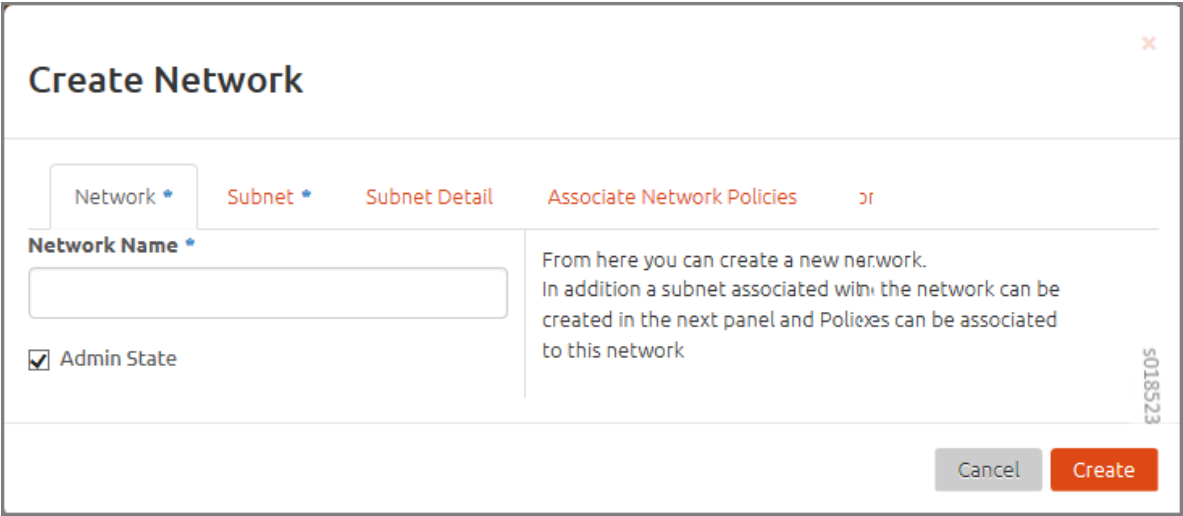


Figure 75: Create Network Window Subnet Tab

Create Network

Network \*

Subnet \*

Subnet Detail

Associate Network Policies

☒ Create Subnet

**Subnet Name**

**IPAM**

default-network-ipam (default-projec

+

**Network Address**

**IP Version** \*

IPv4

**Gateway IP**

☐ Disable Gateway

You can create a subnet associated with the new network, in which case "Network Address" must be specified. If you wish to create a network WITHOUT a subnet, uncheck the "Create Subnet" checkbox.

5018524

Cancel

Create

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 30 on page 531](#).

Table 30: Create Network Fields

Field	Description
Network Name	Enter a name for the network.
Subnet Name	Enter a name for the subnetwork.

Table 30: Create Network Fields *(Continued)*

Field	Description
<b>IPAM</b>	Select the IPAM associated with the IP block.  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.

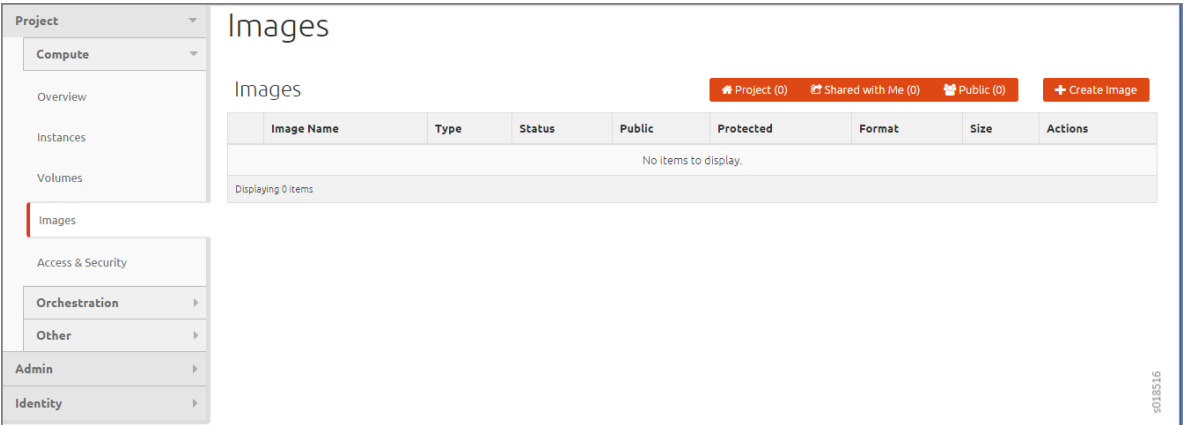
4. Click the **Subnet Details** tab to specify the Allocation Pool, DNS Name Servers, and Host Routes.
5. Click the **Associate Network Policies** tab to associate policies to the network.
6. To save your network, click **Create Network**, or click **Cancel** to discard your work and start over.

## 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 76 on page 533](#).

Figure 76: 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 77 on page 534](#).

Figure 77: OpenStack Create An Image Window

Create An Image

Name \*

Description

Image Source

Image Location ▼

Image Location ?

http://example.com/image.iso

Format \*

Select format ▼

Architecture

Minimum Disk (GB) ?

Minimum RAM (MB) ?

☐ Public

☐ Protected

Description:

Currently only images available via an HTTP URL are supported. The image location must be accessible to the Image Service. Compressed image binaries are supported (.zip and .tar.gz.)

**Please note:** The Image Location field MUST be a valid and direct URL to the image binary. URLs that redirect or serve error pages will result in unusable images.

s018515

Cancel

Create Image

4. Complete the fields to specify your image. [Table 31 on page 535](#) 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 31: 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> .  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: AKI- Amazon Kernel Image AMI- Amazon Machine Image ARI- Amazon Ramdisk Image ISO- Optical Disk Image QCOW2- QEMU Emulator Raw- An unstructured image format VDI- Virtual Disk Image VHD- Virtual Hard Disk VMDK- Virtual Machine Disk
<b>Architecture</b>	Enter the architecture.
<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).

Table 31: Create an Image Fields *(Continued)*

Field	Description
<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.

- When you are finished, click **Create Image**.

## Creating a Floating IP Address Pool

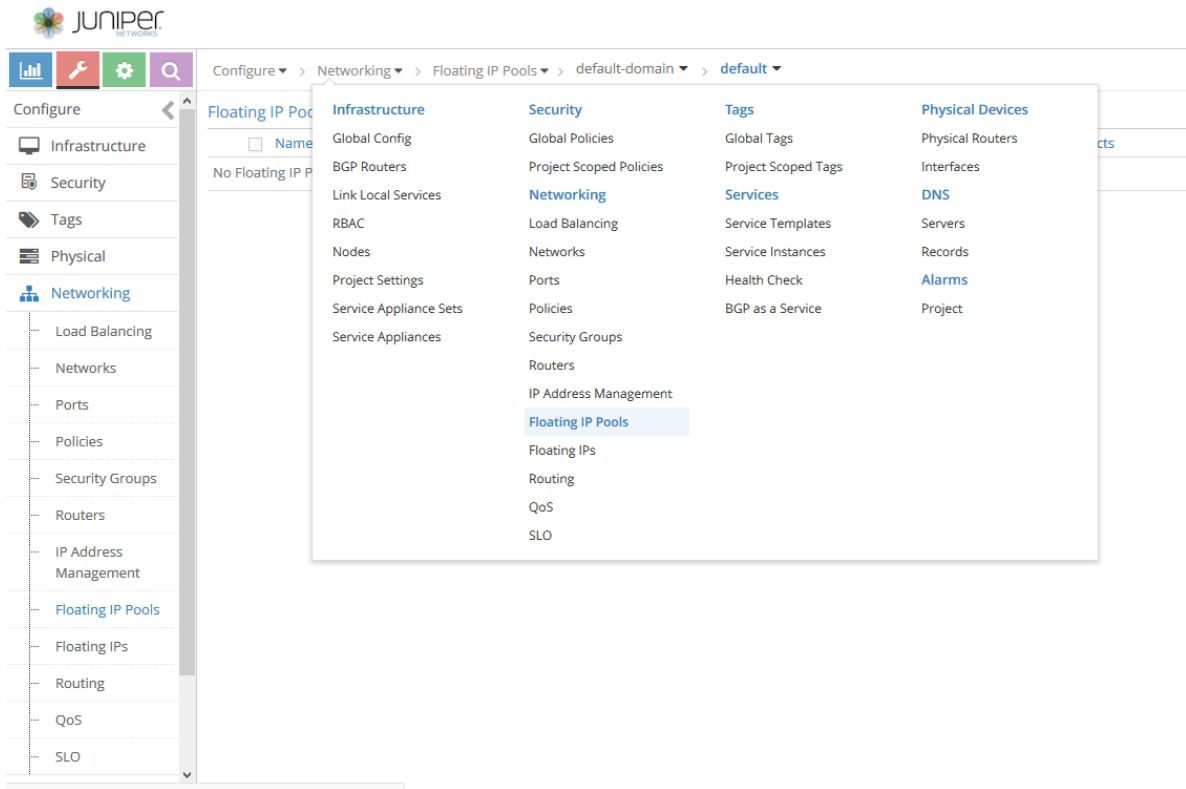
A floating IP address is an IP address (typically public) that can be dynamically assigned to a running virtual instance.

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:

- Select **Configure > Networking > Floating IP Pools**.

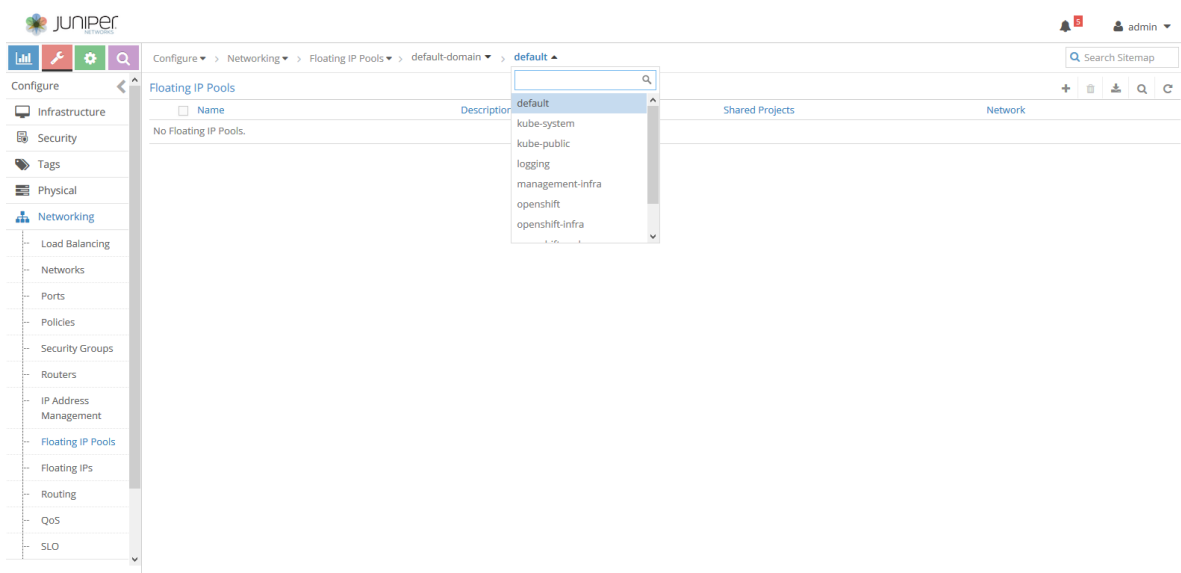


Figure 78: Floating IP Pools Selection



2. Select the network you want to associate with a floating IP pool.

Figure 79: Network Selection

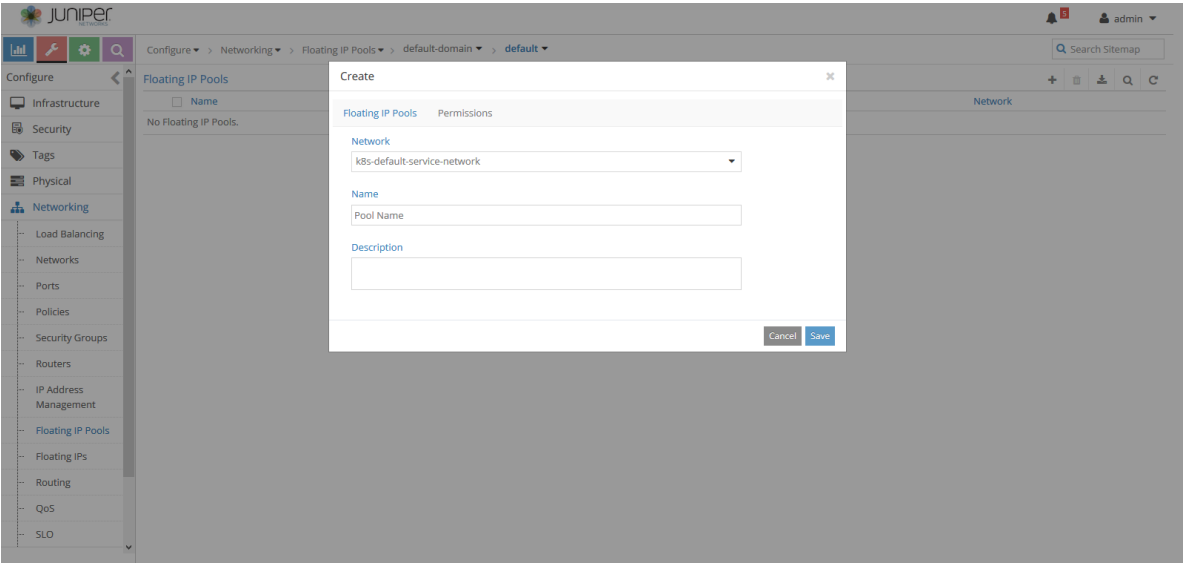


3. Click the add icon (+) to create a floating IP pool.

4. Add a **Name** and **Description** in the **Floating IP Pools** tab.

Click the **Permissions** tab to set **Owner Permissions** and **Global Share Permissions** for the floating IP pool. To associate the floating IP pool with multiple projects, click the add icon (+) in the **Share List**.

Figure 80: Create the Floating IP Pool



5. Click **Save** to create the floating IP address pool, or click **Cancel** to discard your changes and start over.

## Using Security Groups with Virtual Machines (Instances)

### IN THIS SECTION

- [Security Groups Overview | 538](#)
- [Creating Security Groups and Adding Rules | 539](#)

### 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

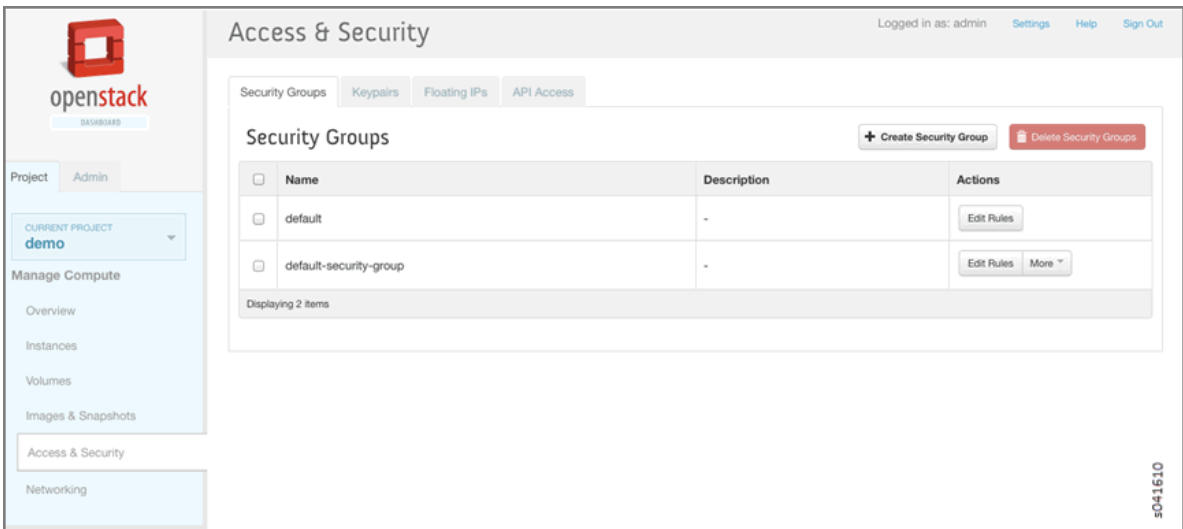
A default security group is created for each project. You can add security rules to the default security group and you can create additional security groups and add rules to them. The security groups are then associated with a VM, when the VM is launched or at a later date.

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 81 on page 539](#).

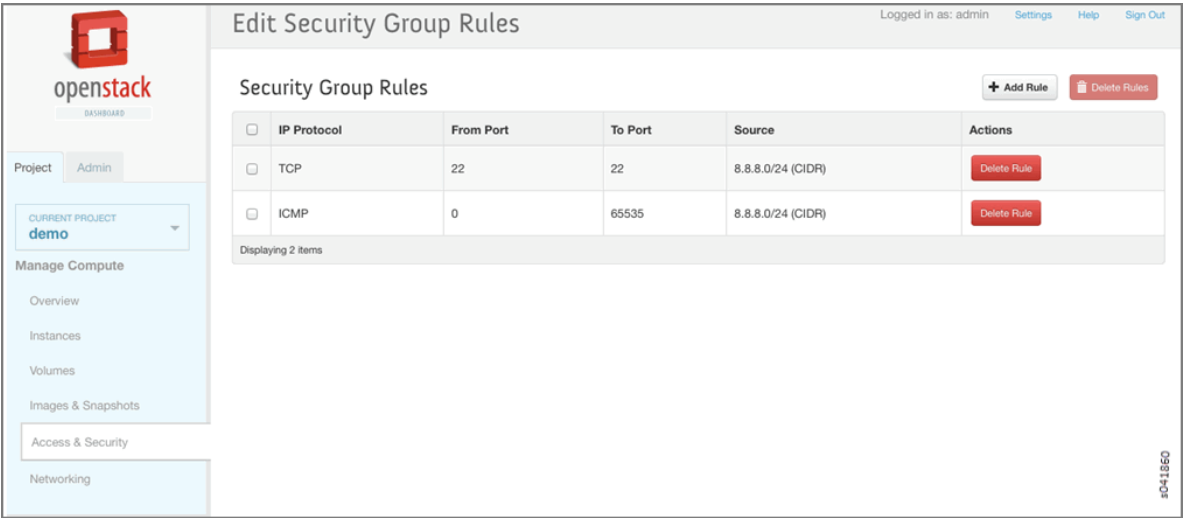
**Figure 81: Security Groups**



2. Select the **default-security-group** and click **Edit Rules** in the **Actions** column.

The **Edit Security Group Rules** window is displayed; see [Figure 82 on page 540](#). Any rules already associated with the security group are listed.

Figure 82: Edit Security Group Rules



3. Click **Add Rule** to add a new rule; see [Figure 83 on page 541](#).

Figure 83: Add Rule

Add Rule

IP Protocol

ICMP

Type

0

Code

0

Source

CIDR

CIDR

Security Group

0.0.0.0/0

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

Table 32: Add Rule Fields

Column	Description
IP Protocol	Select the IP protocol to apply for this rule: TCP, UDP, ICMP.
From Port	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.
To Port	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 32: Add Rule Fields *(Continued)*

Column	Description
Source	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 84 on page 542](#).

Each new security group has a unique 32-bit security group ID and an ACL is associated with the configured rules.

Figure 84: Create Security Group

**Create Security Group**

Name: SG1

Description: From here you can create a new security group

Description: Security Group 1

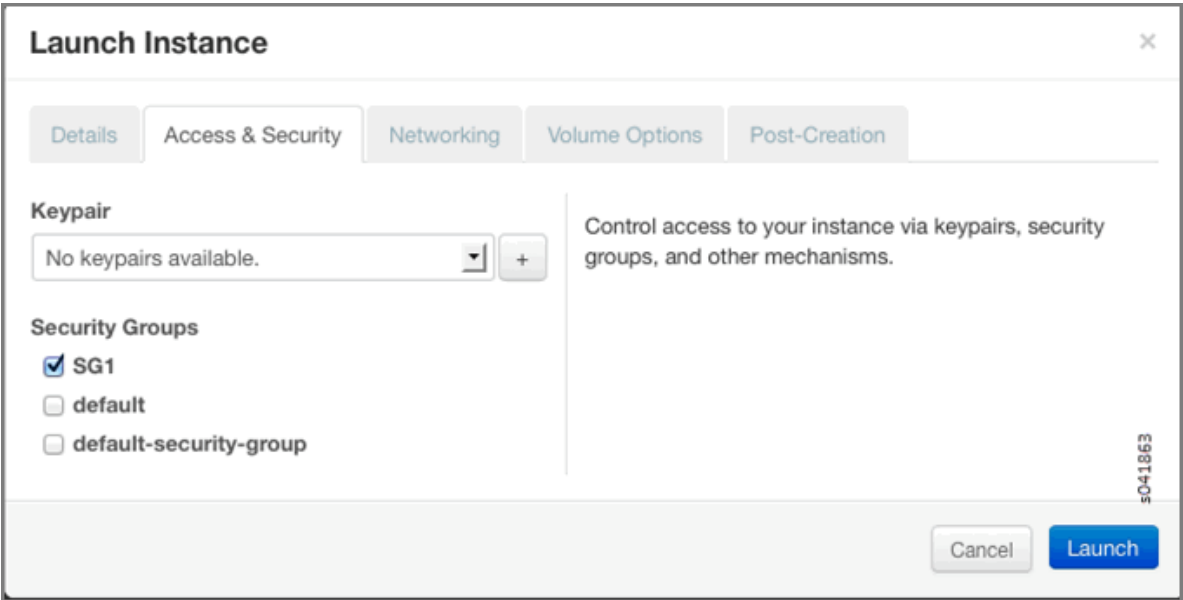
Cancel Create Security Group

sc41861

5. When an instance is launched, there is an opportunity to associate a security group; see [Figure 85 on page 543](#).

In the **Security Groups** list, select the security group name to associate with the instance.

Figure 85: 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`.

## Support for IPv6 Networks in Contrail

### IN THIS SECTION

- [Overview: IPv6 Networks in Contrail | 543](#)
- [Creating IPv6 Virtual Networks in Contrail | 544](#)
- [Adding IPv6 Peers | 546](#)

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.



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.



openstack  
SECURITY

# Instances

Logged in as: admin [Settings](#) [Help](#) [Sign Out](#)

Project

Admin

CURRENT PROJECT

admin

Manage Compute

Overview

Instances

Volumes

Images & Snapshots

Access & Security

Other

Routers

Network Topology

Load Balancers

Networking

Filter

Filter

+ Launch Instance

Soft Reboot Instances

Terminate Instances

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Size	Keypair	Status	Task	Power State	Uptime	Actions
<input type="checkbox"/>	<a href="#">Test-6dba4281-ada9-41fc-8609-bcd899f78ee3</a>	ubuntu-jdaff	<a href="#">data</a> 66.1.1.251 <a href="#">fd66::fff::ff</a> <a href="#">vn-jdaff</a> 76.1.1.252	m1.medium   4GB RAM   2 VCPU   40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<div>Create Snapshot</div> <div>More ~</div>
<input type="checkbox"/>	<a href="#">Test-7a3b7c5b-e5a5-48b3-9346-29079a1abdba</a>	ubuntu-jdaff	<a href="#">data</a> 66.1.1.250 <a href="#">fd66::fff::ff</a> <a href="#">vn-jdaff</a> 76.1.1.250	m1.medium   4GB RAM   2 VCPU   40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<div>Create Snapshot</div> <div>More ~</div>
<input type="checkbox"/>	<a href="#">Test-663309b7-1765-4cc4-9edc-9f025ecd4ae5</a>	ubuntu-jdaff	<a href="#">data</a> 66.1.1.245 <a href="#">fd66::fff::ff</a> <a href="#">vn-jdaff</a> 76.1.1.244	m1.medium   4GB RAM   2 VCPU   40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<div>Create Snapshot</div> <div>More ~</div>
<input type="checkbox"/>	<a href="#">Test-a20ce6d7-3d2b-447e-8894-d794eaa620ab</a>	ubuntu-jdaff	<a href="#">data</a> 66.1.1.252 <a href="#">fd66::fff::ff</a> <a href="#">vn-jdaff</a> 76.1.1.251	m1.medium   4GB RAM   2 VCPU   40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<div>Create Snapshot</div> <div>More ~</div>
<input type="checkbox"/>	<a href="#">Test-43345608-455f-47e6-9346-5cb1f5be2197</a>	ubuntu-jdaff	<a href="#">data</a> 66.1.1.247 <a href="#">fd66::fff::ff</a> <a href="#">vn-jdaff</a> 76.1.1.247	m1.medium   4GB RAM   2 VCPU   40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<div>Create Snapshot</div> <div>More ~</div>

## 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:

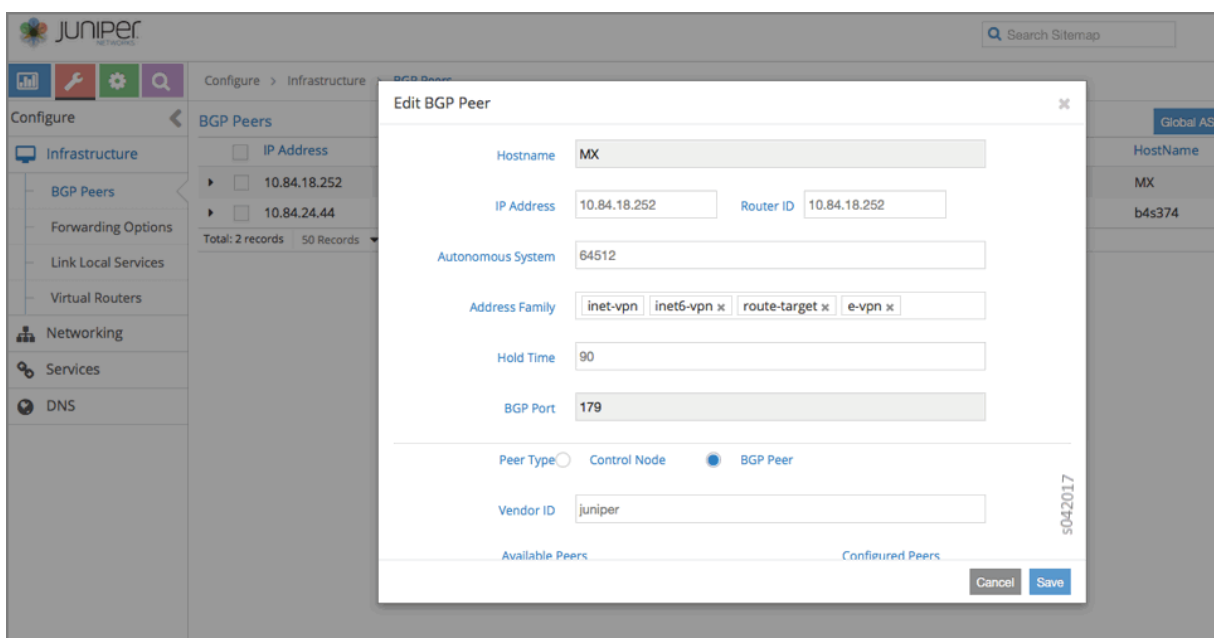
```
iface eht0 inet6 dhcp
```

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 | 549](#)
- [Configuring Forwarding | 551](#)
- [Configuring the VXLAN Identifier | 552](#)
- [Configuring Encapsulation Methods | 553](#)

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

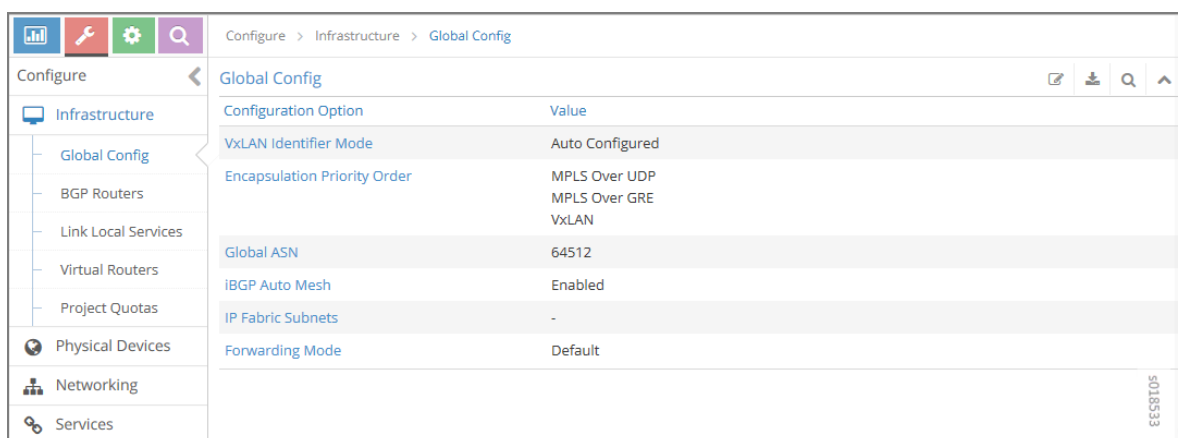
You can configure the global VXLAN identifier mode to select an auto-generated VNID or a user-generated VXLAN ID, either through the Contrail Web UI or by modifying a python file.

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 86: Global Config Window for VXLAN ID**



2. Click the edit icon



.

The Edit Global Config window is displayed as shown in [Figure 87 on page 550](#).

Figure 87: Edit Global Config Window for VXLAN Identifier Mode

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/utils/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 88 on page 551](#).

**Figure 88: Edit Network Window**

Edit Network TestProjectC5Ca5C-VN2D5D41B

Name: TestProjectC5Ca5C-VN2D5D41B

Network Policy(s): Select Policies...

▼ Subnets

IPAM	CIDR	Allocation Pools	Gateway	DNS	DHCP	+
TestProjectC5Ca5C-ipam655...	31.222.172.0/24		<input checked="" type="checkbox"/> 31.222.172.1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	+ -

▶ Host Routes

▶ Advanced Options

▶ Floating IP Pools

▶ Route Targets

Cancel Save

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 89 on page 553](#).

**Figure 89: Edit Network Window for VXLAN Identifier**

Edit Network default-virtual-network-1

▼ Advanced Options

Admin State: Up

☐ Shared ☐ External

DNS Servers: DNS Servers +

Forwarding Mode: L2 and L3

VxLAN Identifier: 0-1048575

☐ Allow Transit

☐ Flood unknown unicast

☐ Extend To Physical Router(s)

Cancel Save

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 90 on page 555](#).

Figure 90: Edit Global Config Window for Encapsulation Priority Order

**Edit Global Config**

▼ **Forwarding Options**

Forwarding Mode: Default

VxLAN Identifier Mode: ☐ Auto Configured ☒ User Configured

Encapsulation Priority Order +

MPLS Over UDP + -

MPLS Over GRE + -

VxLAN + -

▼ **BGP Options**

Global ASN: 64512

Cancel Save

5018508

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.

## Support for EVPN Route Type 5

Contrail Release 5.0.1 supports EVPN Route Type 5 messages as defined in the IETF specification *IP Prefix Advertisement in EVPN*. EVPN Route Type 5 is an extension of EVPN Route Type 2, which carries MAC addresses along with their associated IP addresses. EVPN Route Type 5 facilitates in inter-subnet routing.

Type 5 network layer reachability information (NLRI) contains information in the following format:

```
+-----+
|      RD      (8 octets)      |
+-----+
|Ethernet Segment Identifier (10 octets)|
+-----+
| Ethernet Tag ID (4 octets)    |
+-----+
| IP Prefix Length (1 octet)    |
+-----+
| IP Prefix (4 or 16 octets)    |
+-----+
| GW IP Address (4 or 16 octets)|
+-----+
| MPLS Label (3 octets)        |
+-----+
```

When Type-5 EVPN prefix is received from a BGP peer, it is first installed into **bgp.evpn.0** like all other routes. From here, based on matching route targets, the route gets replicated into all **\*.evpn.0** tables as

applicable. From there, the routes are advertised over Extensible messaging and presence protocol (XMPP) to all interested agents.



**NOTE:** In Release 5.0.1, policy based route-leaking among different L3VRFs is not supported. Hence, service chaining for Type 5 L3VRFs is also not supported.

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

## IN THIS CHAPTER

- [Example: Deploying a Multi-Tier Web Application | 558](#)
- [Sample Network Configuration for Devices for Simple Tiered Web Application | 566](#)

## Example: Deploying a Multi-Tier Web Application

### IN THIS SECTION

- [Multi-Tier Web Application Overview | 558](#)
- [Example: Setting Up Virtual Networks for a Simple Tiered Web Application | 559](#)
- [Verifying the Multi-Tier Web Application | 562](#)
- [Sample Addressing Scheme for Simple Tiered Web Application | 562](#)
- [Sample Physical Topology for Simple Tiered Web Application | 563](#)
- [Sample Physical Topology Addressing | 564](#)

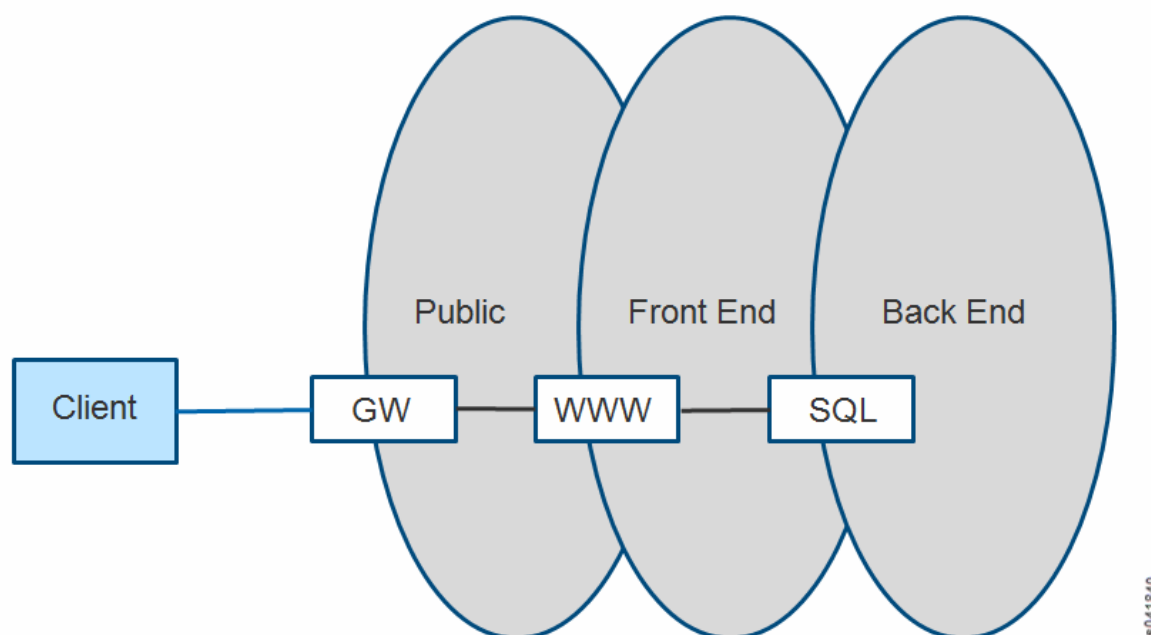
## Multi-Tier Web Application Overview

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 91 on page 559](#).

**Figure 91: 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 92 on page 561](#).



Figure 92: Create Floating IP Pool

Edit Network public

Network Name

public

Network Policy(s)

Select Policies...

Address Management

default-network... ▾

xxx.xxx.xxx.xxx/xx

+

-

IPAM	IP Block
default-network-ipam	10.84.41.0/24

Floating IP Pools

public\_pool

demo ×

+

-

Pool Name

admin

Cancel

Save

4. Allocate the floating IP pool **public\_pool** to the **demo** project; see [Figure 93 on page 561](#).

Figure 93: Allocate Floating IP

Allocate Floating IP

Floating IP Pool

public:public\_pool ▾

Cancel

Save

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 33 on page 563](#) as a guide for addressing devices in the simple tiered web example.

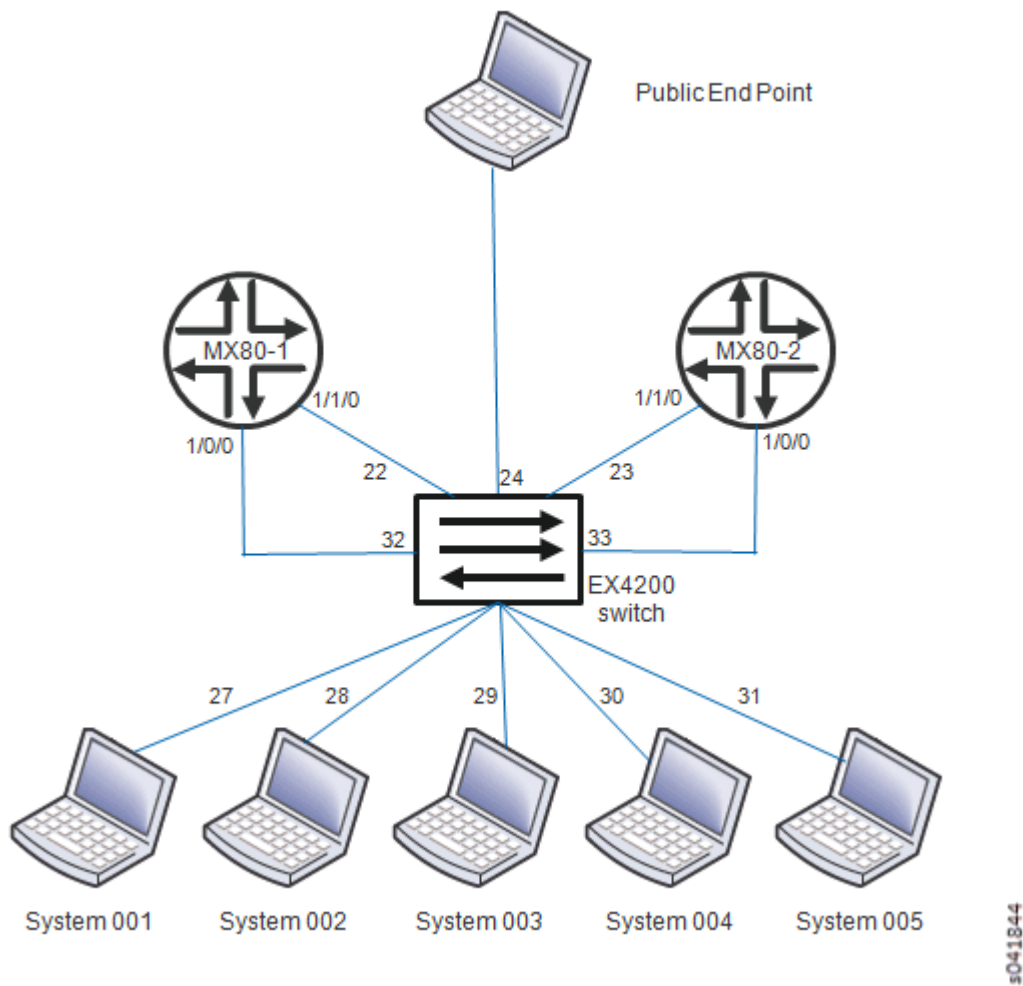
**Table 33: Sample Addressing Scheme for Example**

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 10.84.45.1 (public connection)
MX80-2	10.84.11.252 10.84.45.2 (public connection)
EX4200	10.84.11.254 10.84.45.254 (public connection) 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 94 on page 564 provides a guideline diagram for the physical topology for the simple tiered web application example.

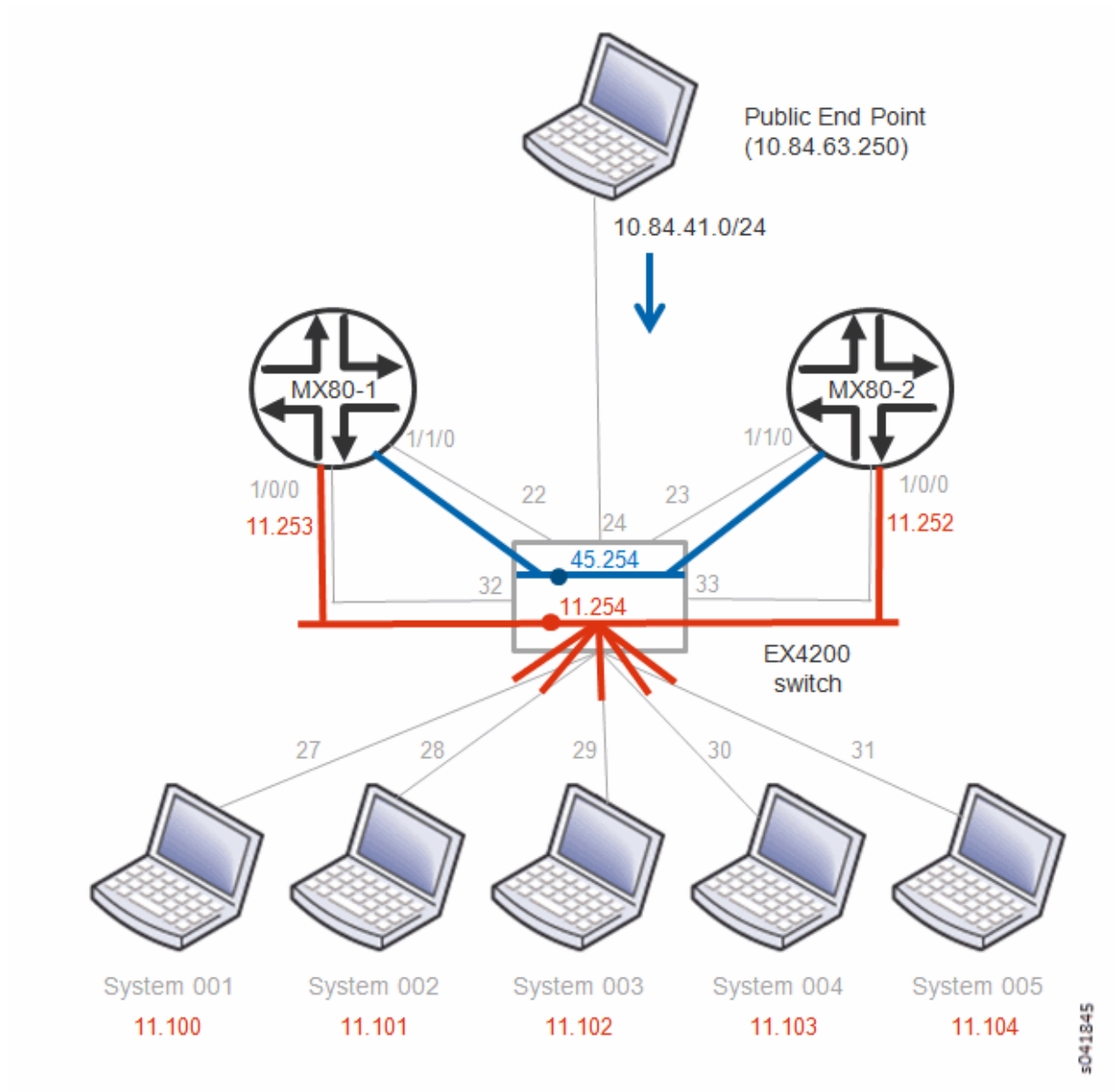
Figure 94: Sample Physical Topology for Simple Tiered Web Application



### Sample Physical Topology Addressing

Figure 95 on page 565 provides a guideline diagram for addressing the physical topology for the simple tiered web application example.

Figure 95: 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 "xxxxxxxxx"; ## 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 | 573](#)
- [Distributed Service Resource Allocation with Containerized Contrail | 586](#)
- [Support for Multicast | 596](#)
- [Using Static Routes with Services | 599](#)
- [Configuring Metadata Service | 604](#)

## Configuring DNS Servers

### IN THIS SECTION

- [DNS Overview | 573](#)
- [Defining Multiple Virtual Domain Name Servers | 574](#)
- [IPAM and Virtual DNS | 575](#)
- [DNS Record Types | 575](#)
- [Configuring DNS Using the Interface | 576](#)
- [Configuring DNS Using Scripts | 584](#)

### 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:

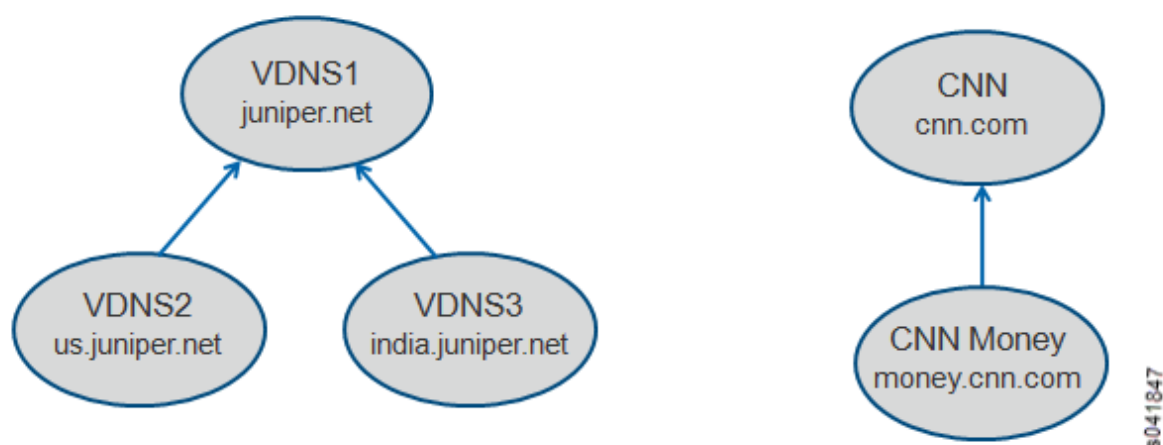
- It should be possible to configure multiple domain name servers to provide name resolution service for the virtual machines spawned in the system.
- It should be possible to configure the domain name servers to form DNS server hierarchies required by each tenant.
  - The hierarchies can be independent 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

Contrail 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 96 on page 574](#) shows examples of virtual DNS servers defined in **default-domain**, providing the name service for the DNS domains indicated.

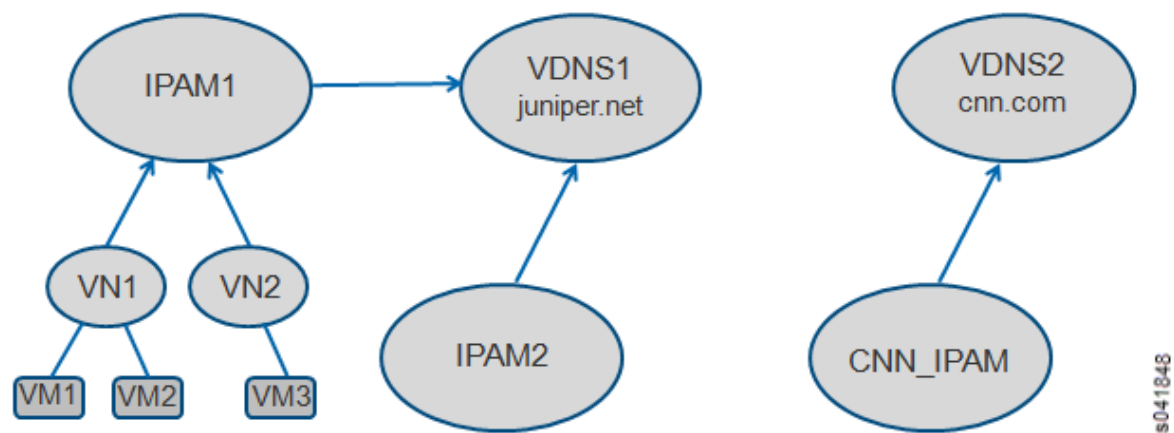
**Figure 96: DNS Servers Examples**



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 97 on page 575](#)

Figure 97: IPAM and Virtual DNS



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 34 on page 575](#) for descriptions of the record types.

Table 34: DNS Record Types Supported

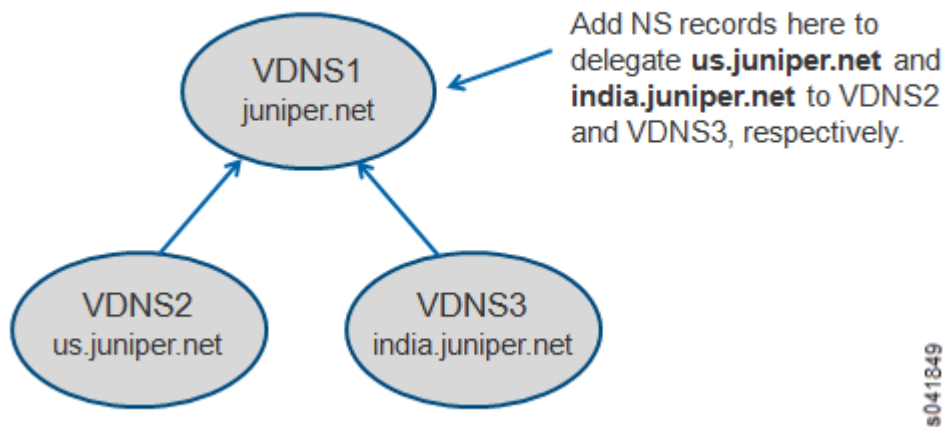
DNS Record Type	Description
<b>A</b>	Used for mapping hostnames to IPv4 addresses. Name refers to the name of the virtual machine, and data is the IPv4 address of the virtual machine.
<b>CNAME</b>	Provides an alias to a name. Name refers to the name of the virtual machine, and data is the new name (alias) for the virtual machine.

Table 34: 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 98 on page 576 shows an example usage for the DNS record type of NS.

Figure 98: Example Usage for NS Record Type



### 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 99 on page 577](#).



Figure 99: Configure DNS Records

Configure > DNS > Servers

Search

Configure DNS Records

default-domainadmin

CreateDelete

Configure Virtual DNS

Virtual DNS Name	DNS Domain Name	Next DNS Server
No Data Found		

DNS RecordsAssociated IPAMs

DNS Records of {{dnsname}}

Add RecordDelete

Name	Type : Data	TTL (secs)	Class
------	-------------	------------	-------

2. To add a new DNS server, click the **Create** button.
- Enter DNS server information in the **Add DNS** window; see [Figure 100 on page 578](#)

Figure 100: Add DNS

Create DNS Server

Server Name

Domain Name

DNS Forwarder

Enter Forwarder IP or Select a DNS Server

Record Resolution Order

Random

Time To Live

TTL (86400 sec)

Associate IPAMs

Cancel

Save

s041864

Complete the fields for the new server; see [Table 35 on page 578](#).

Table 35: Add DNS Fields

Field	Description
Server Name	Enter a name for this server.
Domain Name	Enter the name of the domain for this server.
Time To Live	Enter the <b>TTL</b> in seconds.
Next DNS Server	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 35: 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.

3. 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 101 on page 580](#).

Figure 101: Add DNS Record

Add DNS Record

Type

A (IP Address Record)

Host Name

Host Name to be resolved

IP Address

Enter an IP Address

Class

IN (Internet)

Time To Live

TTL(86400 secs)

Cancel

Save

4. Complete the fields for the new record; see [Table 36 on page 580](#).

Table 36: Add DNS Record Fields

Field	Description
Record Name	Enter a name for this record.
Type	Select the record type from a list— <b>A</b> , <b>CNAME</b> , <b>PTR</b> , <b>NS</b> .
IP Address	Enter the IP address for the location for this record.
Class	Select the record class from a list— <b>IN</b> is the default.
Time To Live	Enter the <b>TTL</b> in seconds.
OK	Click <b>OK</b> to create the record.

Table 36: Add DNS Record Fields *(Continued)*

Field	Description
Cancel	Click <b>Cancel</b> to clear the fields and start over.

5. 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 102 on page 581](#).

Figure 102: Associate IPAMs to DNS

The screenshot shows a window titled "Edit DNS Server" with a close button (X) in the top right corner. The window contains several input fields and a dropdown menu:

- Server Name:** A text input field containing "vdns1".
- Domain Name:** A text input field containing "juniper.net".
- DNS Forwarder:** A text input field containing "Enter Forwarder IP or Select a DNS Server" and a dropdown arrow.
- Record Resolution Order:** A text input field containing "Random" and a dropdown arrow.
- Time To Live:** A text input field containing "86400".
- Associate IPAMs:** A text input field with a dropdown menu open, showing two options: "admin:ipam1" (highlighted) and "default-project:default-network-ipam".

At the bottom right of the window, there are two buttons: "Cancel" and "Save". A vertical text label "5041854" is visible on the right side of the window.

Complete the IPAM associations, using the field descriptions in [Table 37 on page 581](#).

Table 37: Associate IPAMs to DNS Fields

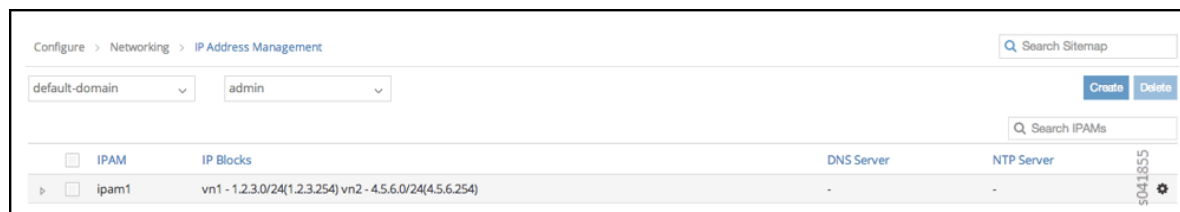
Field	Description
<b>Associate to All IPAMs</b>	Select this box to associate the selected DNS server to all available IPAMs.

Table 37: 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 103 on page 582](#) 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 103: 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 104 on page 583](#).

Figure 104: DNS Server

Add IP Address Management

Name

DNS Method

Virtual DNS

Default

Virtual DNS

Tenant

None

Virtual DNS

NTP Server IP

None

Associate IP Blocks to Networks

fip\_vn

IP Block

Gateway

+ -

Network	IP Block	Gateway
---------	----------	---------

Cancel

Save

8. In the first field, select the **DNS Method** from a list (**None**, **Default DNS**, **Tenant DNS**, **Virtual DNS**; see [Table 38 on page 583](#).

Table 38: DNS Modes

DNS Mode	Description
None	Select <b>None</b> when no DNS support is required for the VMs.
Default	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 38: 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

You can configure DNS by using scripts that are available in the contrail-utils RPM/DEB package in the `/opt/contrail/Utils` directory. The scripts are copied to the `config_api_container` or `config` node when you install the contrail-utils RPM/DEB package. You can execute the scripts from either the `config_api` container or the `config` node. The scripts are described in [Table 39 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 39: 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>
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 | 586](#)
- [New Distributed Resource Allocation Manager | 587](#)
- [Changes in Configuration Files | 587](#)

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 40 on page 587](#) 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 40: Contrail 4.0 Changes in Configuration Files**

Configuration File	Configuration Parameter	Changes
contrail-vrouter-agent.conf	[DISCOVERY]	Section deprecated

Table 40: Contrail 4.0 Changes in Configuration Files (*Continued*)

Configuration File	Configuration Parameter	Changes
	[CONTROL-NODE].servers	Provisioned list of control-node [role=control] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:5269 10.1.1.12:5269
	[DNS].servers	Provisioned list of DNS [role=control] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:53 10.1.1.2:5
	[DEFAULT].collectors	Provisioned list of Collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 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: ip-address:port ip-address2:port Example: 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: ip-address:port ip-address2:port Example: 10.1.1.1:5672 10.1.1.2:5672
	[CONFIGDB].rabbitmq_user	guest (default string)

Table 40: Contrail 4.0 Changes in Configuration Files (*Continued*)

Configuration File	Configuration Parameter	Changes
	[CONFIGDB].rabbitmq_password	guest (default string)
	[CONFIGDB].config_db_server_list	Provisioned list of Config DB [role=database] service providers in the format:  ip-address:port ip-address2:port Example: 10.1.1.1:9042 10.1.1.2:9042  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:  ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086

Table 40: 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:  ip-address:port ip-address2:port Example: 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:  ip-address:port ip-address2:port Example: 10.1.1.1:9042 10.1.1.2:9042 NOTE: 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 40: 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:  ip-address:port  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:  ip-address:port ip-address2:port  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:  ip-address:port  Example: 10.1.1.1:8082 10.1.1.2:8082
	[REDIS].redis_uve_list	Provisioned list of redis instances [role=collector]  Example: 192.168.0.29:6379 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:  ip-address:port ip-address2:port  Example: 10.1.1.1:8086 10.1.1.2:8086

Table 40: Contrail 4.0 Changes in Configuration Files *(Continued)*

Configuration File	Configuration Parameter	Changes
	[REDIS].redis_uve_list	Provisioned list of redis instances [role=collector]  Example: 192.168.0.29:6379 192.168.0.30:6379
contrail-api.conf	[DISCOVERY]	Section deprecated
	[DEFAULTS].collectors	Provisioned list of collector [role=collector] service providers in the format:  ip-address:port ip-address2:port  Example: 10.1.1.1:8086 10.1.1.2:8086
contrail-schema.conf	[DISCOVERY]	Section deprecated
	[DEFAULTS].collectors	Provisioned list of Collector [role=collector] service providers in ip- address:port ip-address2:port  Example: 10.1.1.1:8086 10.1.1.2:8086
contrail-svc-monitor.conf	[DISCOVERY]	Section deprecated
	[DEFAULTS].collectors	Provisioned list of Collector [role=collector] service providers in the format:  ip-address:port ip-address2:port  Example: 10.1.1.1:8086 10.1.1.2:8086
contrail-device-manager.conf	[DISCOVERY]	Section deprecated



**Table 40: Contrail 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  Example: 10.1.1.1:8086 10.1.1.2:8086
contrail-analytics-nodemgr.conf	[DISCOVERY]	Section deprecated
	[COLLECTOR].server_list	Provisioned list of Collector [role=collector] service providers in the format:  ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
contrail-config-nodemgr.conf	[DISCOVERY]	Section deprecated
	[COLLECTOR].server_list	Provisioned list of Collector [role=collector] service providers in the format:  ip-address:port ip-address2:port  Example: 10.1.1.1:8086 10.1.1.2:8086
contrail-control-nodemgr.conf	[DISCOVERY]	Section deprecated
	[COLLECTOR].server_list	Provisioned list of Collector [role=collector] service providers in ip-address:port ip-address2:port  Example: 10.1.1.1:8086 10.1.1.2:8086
contrail-database-nodemgr.conf	[DISCOVERY]	Section deprecated

Table 40: 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:  ip-address:port ip-address2:port  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:  ip-address:port ip-address2:port  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:  ip-address:port ip-address2:port  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:  ip-address:port ip-address2:port  Example: 10.1.1.1:8086 10.1.1.2:8086

Table 40: 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:  ip-address:port  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:  ip-address:port ip-address2:port Example: 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  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  Example: ['10.1.1.1 10.1.1.2']
	config.cnfg.server_port	Server port as a string  Example: '8082'

Table 40: 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  Example: ['10.1.1.1 10.1.1.2']
	<code>config.analytics.server_port</code>	Server port as a string  Example: '8081'
	<code>config.dns.server_ip</code>	Provisioned list of Controller [role=control] service providers as a list of ip-address  Example: ['10.1.1.1 10.1.1.2']
	<code>config.dns.server_port</code>	Server port as a string  Example: '8092'

## Support for Multicast

### IN THIS SECTION

- Subnet Broadcast | 597
- All-Broadcast/Limited-Broadcast and Link-Local Multicast | 597
- Host Broadcast | 598

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 | 599](#)
- [Configuring Static Routes on a Service Instance | 600](#)
- [Configuring Static Routes on Service Instance Interfaces | 601](#)
- [Configuring Static Routes as Host Routes | 603](#)

### Static Routes for Service Instances

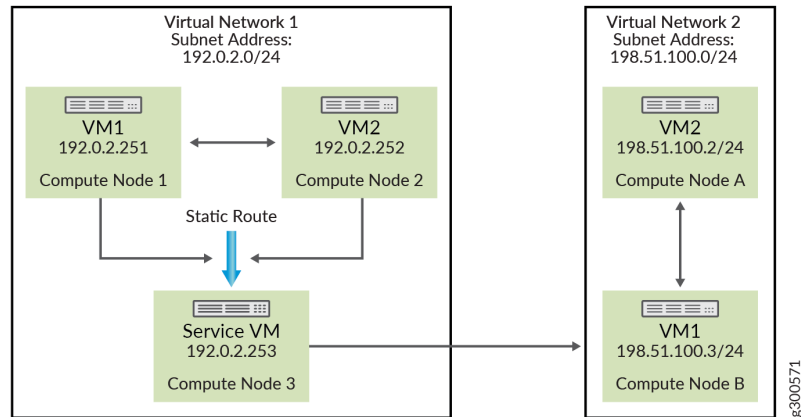
Static routes are manually configured in a network to initiate data transmission between two networks. The traffic generated by a set of devices in a network is directed through a static route, which ensures an efficient flow of traffic towards a specific destination address.

Static routes are used in small networks and networks with simple architecture as a route for direct communication between two networks. Static routes cannot operate in large, dynamic networks due to the frequent change in networks and routes. In such cases, we can use BGPaaS in Contrail for dynamic routing updates. A static route can also be configured as the Default route (a gateway of last resort), which the routers use to send data packets with unknown destination address.

In a virtual network, you can configure static routes towards a service virtual machine (VM) interface to direct all network traffic through the service virtual machine. The configured static routes are advertised to other nodes through BGP, which ensures that traffic is directed through specific virtual machine.

In [Figure 105 on page 600](#), there are three VMs in a virtual network (VN1) with subnet address 192.0.2.0/24. The virtual machines are VM1 (192.0.2.251), VM2 (192.0.2.252), and a Service VM (192.0.2.253). When VM1 or VM2 in VN1 generates traffic targeted towards another virtual network (VN2) with subnet address 198.51.100.0/24, you need to configure a static route. You can configure a static route towards the Service VM interface to direct the traffic generated by VM1 and VM2 destined towards VN2. Once configured, all traffic targeted towards VN2 from VN1 is directed through the static route (192.0.2.253).

Figure 105: Static Route in a Virtual Network



## 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 type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>	+ -
Left <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	+ -
Right <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	+ -

▶ [Advanced options](#)

Cancel
Save

s041915

### 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 (192.0.2.0/24) and VN2 (198.51.100.0/24). The left interface of the service instance is configured with VN1 and the right interface is configured to be VN2 (198.51.100.0/24). The static route 192.0.2.253 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.

Create Service Instances

Instance Name

nat

Services Template

nat - [in-network (management, left, right)]

Interface 1

Management

Auto Configured

Interface 2

Left

vn1

Static Routes

Prefix	Next hop	
11.1.1.0/24	Interface 2	+ -

Interface 3

Right

vn2

Static Routes

Cancel

Save

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.

**Create Service Instances**

Interface 2: Left, vn1

▼ Static Routes

Prefix	Next hop	+
11.1.1.0/24	Interface 2	+ -

Interface 3: Right, vn2

▼ Static Routes

Prefix	Next hop	+
10.1.1.0/24	Interface 3	+ -

Cancel Save

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

## 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](#)

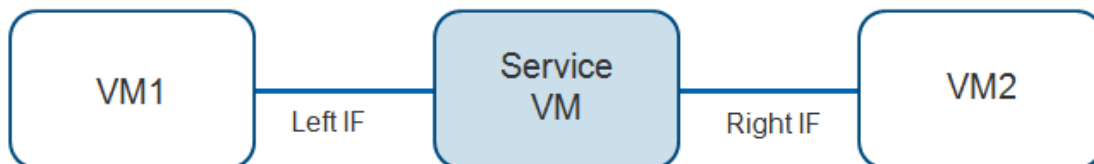
Contrail Controller supports chaining of various Layer 2 through Layer 7 services such as firewall, NAT, IDP, and so on.

### 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.

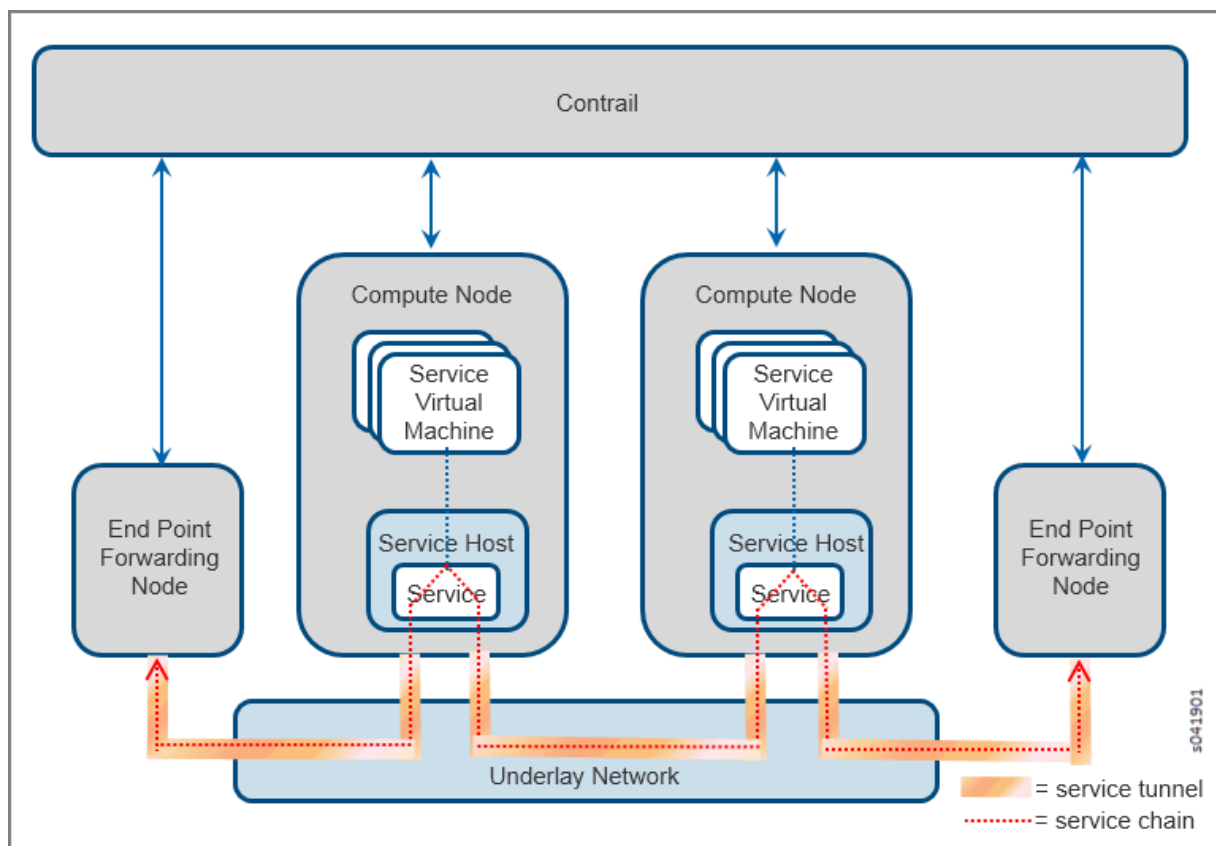
[Figure 106 on page 607](#) shows the basic service chain schema, with a single service. The service VM spawns the service, using the convention of left interface (left IF) and right interface (right IF). Multiple services can also be chained together.

**Figure 106: 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 107 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 107: 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*

*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;
    }
  }
}

```

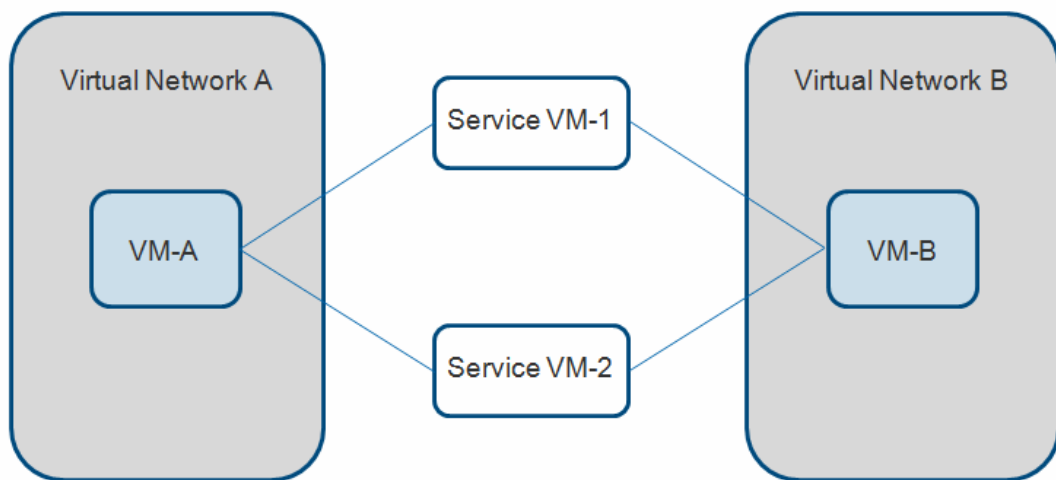
4. The final step is to add target:100:10000 to the public virtual network and target:100:20000 to the enterprise virtual network, using the Contrail Juniper Networks interface.

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 108 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 108: 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.

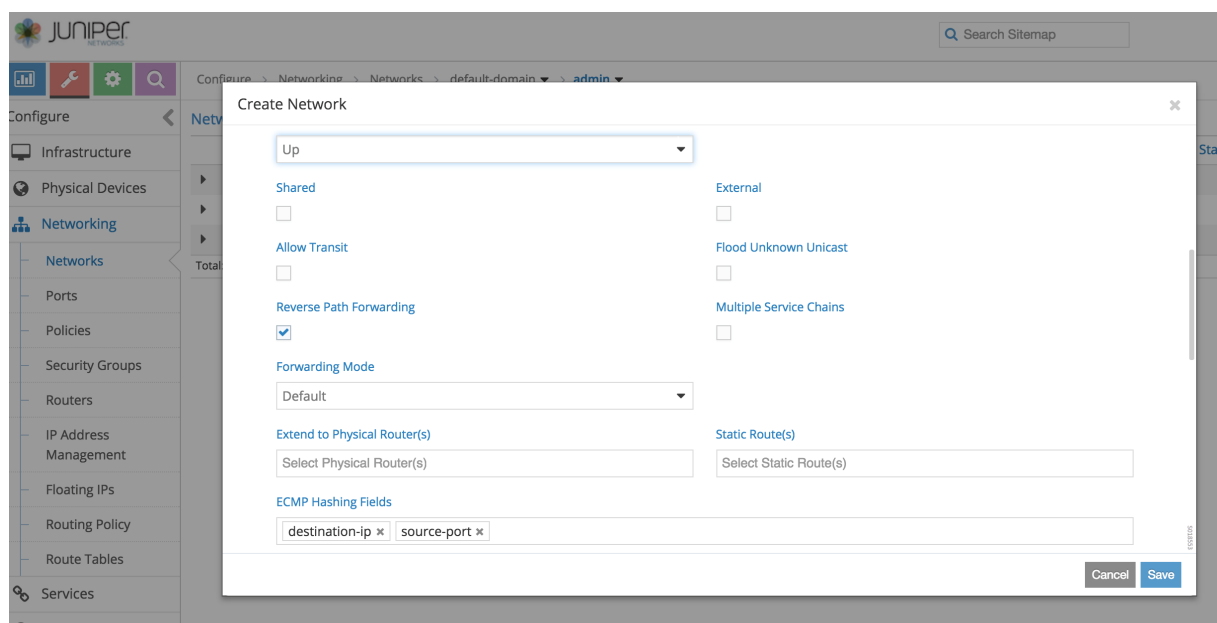
The custom hash configuration can be applied in the following ways:

- globally
- per virtual network (VN)
- per virtual network interface (VMI)

VMI configurations take precedence over VN configurations, and VN configurations take precedence over global level configuration (if present).

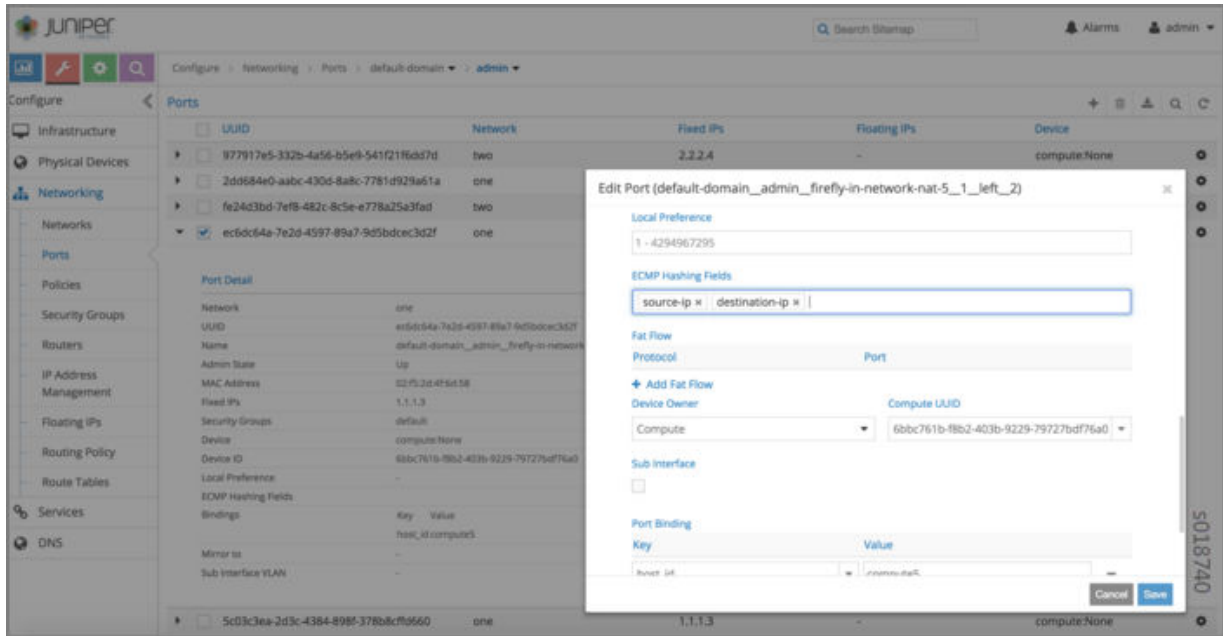
Custom hash is useful whenever packets originating from a particular source and addressed to a particular destination must go through the same set of service instances during transit. This might be required if source, destination, or transit nodes maintain a certain state based on the flow, and the state 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 the desired VMIs 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.

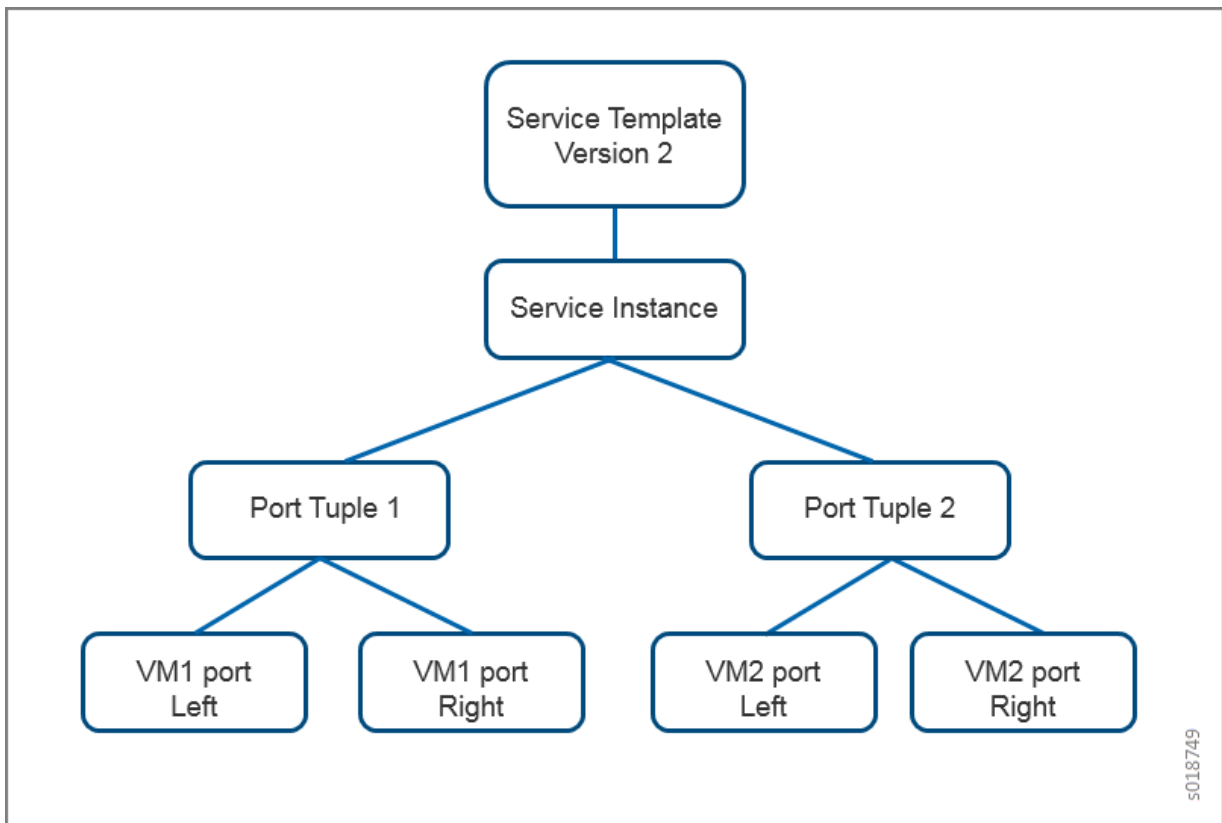
## Overview of Port Tuple

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 109 on page 620](#).

**Figure 109: 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 110 on page 622](#).

**Figure 110: Edit Service Instance, Allowed Address Pair**

Edit

Interface Type

Virtual Machine Interface

left

(1.1.1.4) - f88f8960-7897-4df5-b...

right

(2.2.2.4) - 1fab8367-a4ed-43cb-...

▶ Service Health Check

▶ Routing Policy

▶ Route Aggregate

▼ Allowed Address Pair

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	+ -

Cancel

Save

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](#)

## Service Chain Route Reorigination

### IN THIS SECTION

- [Overview: Service Chaining in Contrail | 628](#)
- [Route Aggregation | 629](#)
- [Routing Policy | 637](#)
- [Control for Route Reorigination | 648](#)

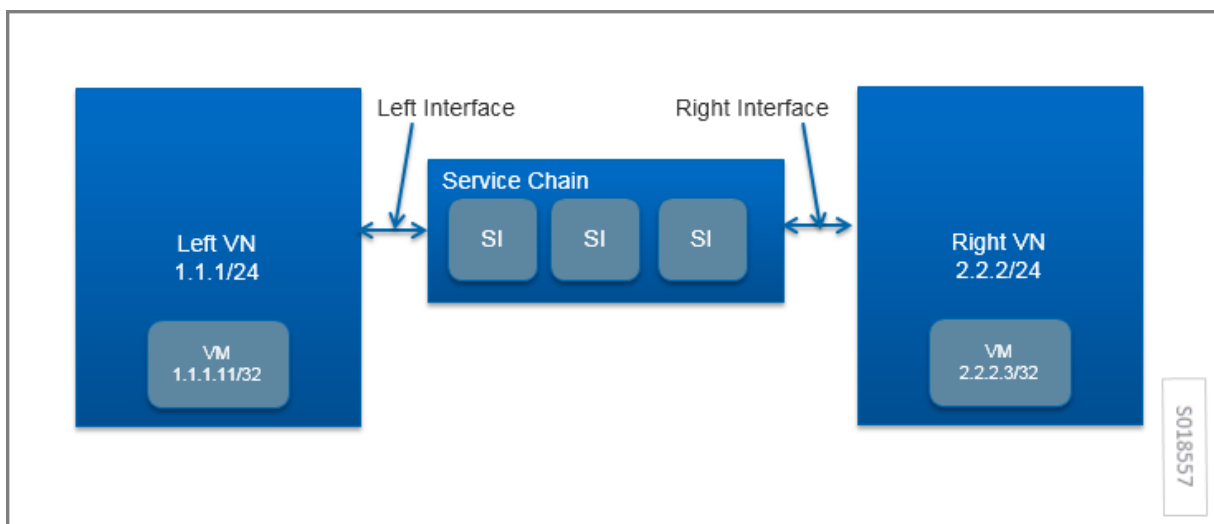
### Overview: Service Chaining in Contrail

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 111 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 111: 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.

Starting in Contrail Release 5.0, when one or more than one service instance in a service chain fails, reorigination of routes on both sides of the service chain is stopped and routes automatically converge to a backup service chain that is part of another Contrail cluster. For more information, see *Service Instance Health Checks*.

## Route Aggregation

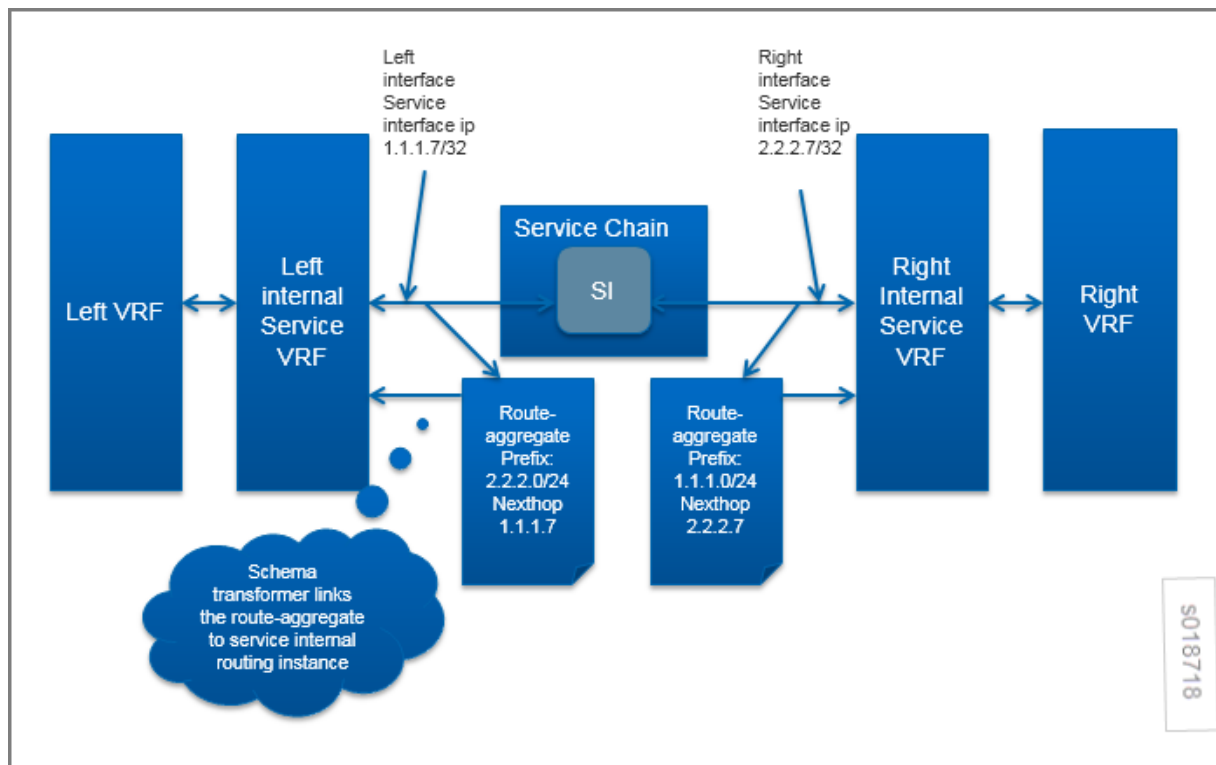
The route aggregation configuration object contains a list of prefixes to aggregate. The next-hop field in the route aggregate object contains the address of the route whose next hop is stitched as a next hop of the aggregate route.

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 112 on page 630](#).

**Figure 112: 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 113 on page 633](#).



Figure 113: Create Route Aggregate

The screenshot shows a 'Create Route Aggregate' window. The 'Name' field contains 'left-to-right'. Below it, under 'Aggregate Route Entries', there is a 'Route' field with '1.1.1.0/24'. A '+ Route Entry' button is located below the route field. On the right side of the window, there is a vertical box containing the number '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 114 on page 634](#).

Figure 114: Create Service Instance

**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

S018720

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

S018721

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

default-network-ip...

CIDR

1.1.1.0/24

Allocation Pools

start-end

Gateway

1.1.1.1

DNS

DHCP

+ -

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.englab.juniper.net:8082/routing-instance/d291a95a-1a5a-4fce-94c8-4abd0968d992",
        attr: null,
        uuid: "d291a95a-1a5a-4fce-94c8-4abd0968d992"
      }
    ],
    parent_href: "http://nodes27.englab.juniper.net:8082/project/6861d89d-a02f-4215-b329-1864084c8a75",
    parent_type: "project",
    + perms2: {(-)},
    href: "http://nodes27.englab.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.englab.juniper.net:8082/service-instance/62accf30-8cc8-4148-b7b8-975573b0d950",
        - attr: {
          interface_type: "right"
        },
        uuid: "62accf30-8cc8-4148-b7b8-975573b0d950"
      }
    ],
    name: "left-to-right"
  }
}

```

s018723

## 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	
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	prefixes 1.1.1.0/24	default-domain:admin:si-aggregate	1.1.1.0/24	1.1.1.3

s018724

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.

service_chain_infos					static_routes aggregate_routes	
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	prefixes 1.1.1.0/24	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 table. 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.

routes									
routes									
prefix	last_modified	paths							
1.1.1.0/24	2016-Feb-18 05:00:29.211876	protocol	last_modified	local_preference	local_as	peer_as	peer_router_id	source_as_path	next_hop
		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.

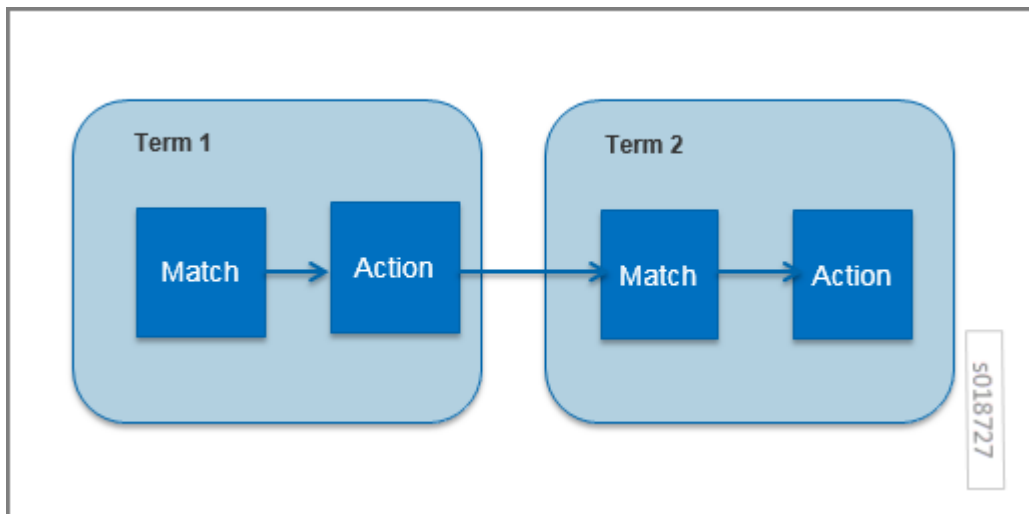
The routing policy contains list terms. A term can be a terminal rule, meaning that upon a match on the specified term, no further terms are evaluated and the route is dropped or accepted, based on the action in that term.

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	Reject the route that matches this term. No more terms are evaluated after hitting this term.
Accept	Yes	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.



*(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.

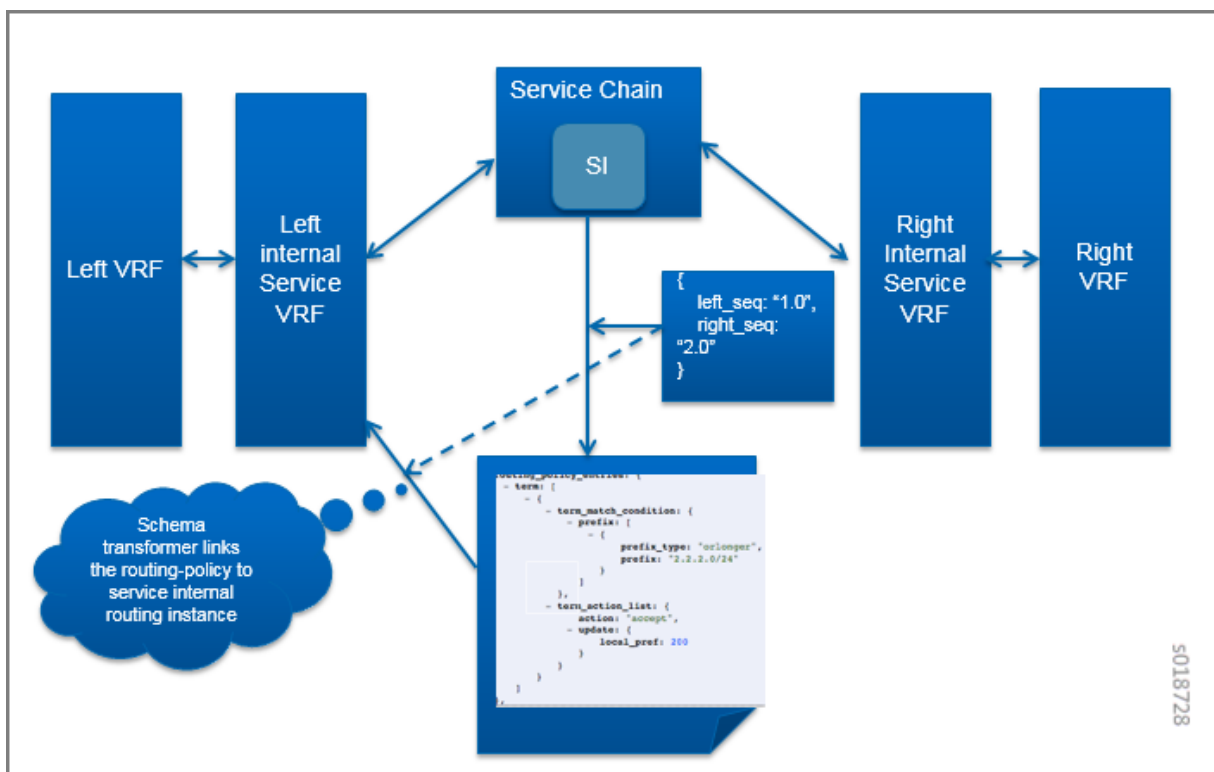
Update Action	User Input	Description
Community	List of community	As part of the policy update, the following actions can be taken for community: <ul style="list-style-type: none"> <li>• Add a list of community to the existing community.</li> <li>• Set a list of community.</li> <li>• Remove a list of 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



**NOTE:** The Contrail UI and REST APIs enable you to configure a BGP routing policy and then assign it to a virtual network, but the routing policy will not be applied if the virtual network is attached to an L3VPN.

### 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

## Configure the Network Policy for the Service Chain

At **Edit Policy**, create a policy for the service chain, see the following example.

**Edit Policy (service-chain-policy)**

Policy Name: service-chain-policy

**Policy Rules**

Action	Protocol	Source	Ports	Direction	Destination	Ports	Log	Services	Mirror
PASS	ANY	left	ANY	<	right	ANY	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Service Instance: si-aggregate ha-chain

+ Add Rule

Cancel Save

## 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.englab.juniper.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.englab.juniper.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.englab.juniper.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	<div> <div>terms</div> <div> <div>terminal</div> <div>true</div> </div> <div> <div>matches</div> <div> <div>prefix [ 2.2.2.0/24 orlonger ]</div> </div> </div> <div> <div>actions</div> <div> <div>accept</div> <div>local-pref 200</div> </div> </div> </div>	false
default-domain:default-project:default-routing-policy 0	0	0	<div> <div>terms</div> </div>	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	<div> <div>terms</div> <div> <div>match</div> <div> <div>from {</div> <div>prefix 2.2.2.0/24 orlonger</div> <div>}</div> </div> <div> <div>action</div> <div> <div>then {</div> <div>local-preference 200</div> <div>accept</div> <div>}</div> </div> </div> </div> </div>	

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 service_chain_info					static_routes	aggregate_routes	routing_policies
family	routing_instance	chain_address	prefixes	service_instance			policy_name sequence
inet	default-domain:admin:right:right	1.1.1.6	prefixes 2.2.2.0/24	default-domain:admin:ha-chain			default-domain:admin:failover 1

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

## 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.

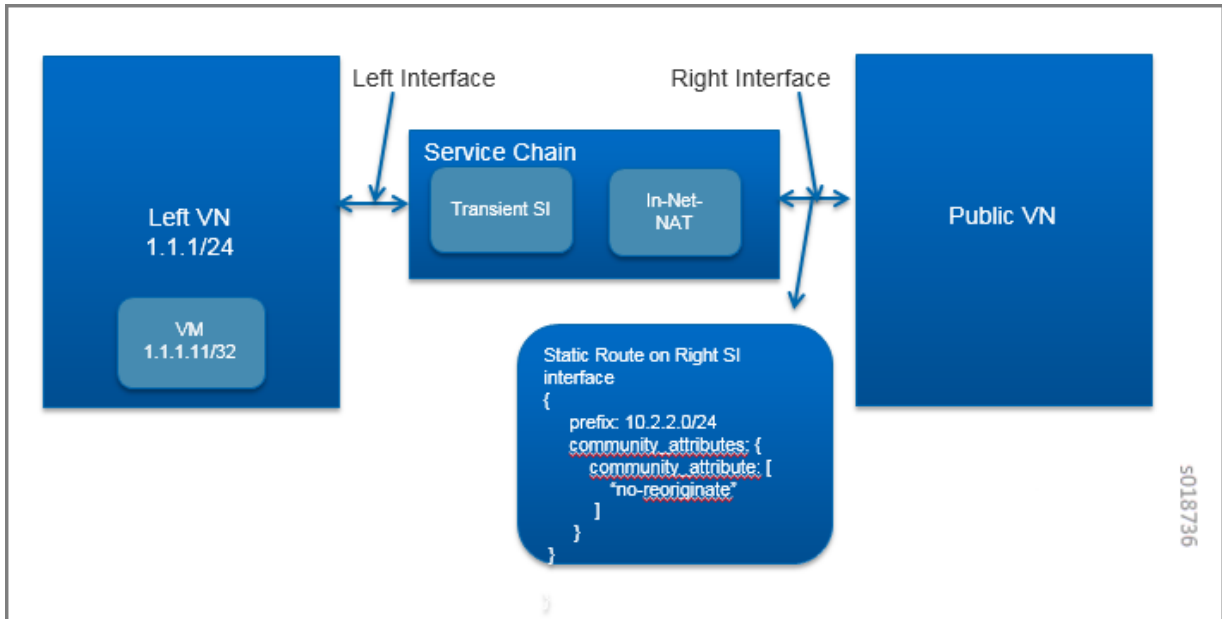
As an example, the following image shows a service chain that has multiple service instances, with an in-net-nat service instance as the last service VM, also with the right VN as the public VN.

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: si-with-static

Service Template: st-with-static - [in-network-nat (left, right)] - v1

Interface Type: left

Virtual Network: Select Virtual Network

Interface Type: right

Virtual Network: Select Virtual Network

+ Add Static Routes

Prefix: 10.2.2.0/24

Next Hop: Interface 2

Community: no-reoriginate

Routing Policy

Interface Type

Routing Policy

Cancel Save

5018737

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: "a6e4e71f-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 | 651](#)
- [Bidirectional Forwarding and Detection Health Check over Virtual Machine Interfaces | 655](#)
- [Bidirectional Forwarding and Detection Health Check for BGPaaS | 656](#)
- [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 liveliness of a service provided by a virtual machine (VM).

# Health Check Object

IN THIS SECTION

- [Health Check Overview | 651](#)
- [Health Check Object Configuration | 651](#)
- [Creating a Health Check with the Contrail User Interface | 653](#)
- [Using the Health Check | 654](#)
- [Health Check Process | 655](#)

## Health Check Overview

The service instance health check is used to determine the liveliness 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 liveliness 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 41 on page 651](#) shows the configurable properties of the health check object.

**Table 41: 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.

**Table 41: Health Check Configurable Parameters (Continued)**

Field	Description
- 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 115 on page 653](#).

**Figure 115: Create Health Check Screen**

2. Complete the fields to define the permissions for the health check, see [Table 42 on page 654](#).

**Table 42: 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.
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 releases prior to Contrail Release 5.0, 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 5.0, when one or more than one SI in a service chain fails, reorigination of routes on both sides of the service chain is stopped and routes automatically converge to a backup



service chain that is part of another Contrail cluster. SI fate sharing brings down the service chain and the gateway nodes automatically reroutes traffic to an alternate cluster.

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 by Using Contrail Command | 658](#)
- [Example: Creating an In-Network-NAT Service Chain by Using Contrail Command | 666](#)
- [Example: Creating a Transparent Service Chain by Using Contrail Command | 675](#)

## Example: Creating an In-Network Service Chain by Using Contrail Command

## 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 Command user interface (UI).

### 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



**NOTE:** For Contrail Networking Release 3.2 through Release 4.1, you use the Contrail Web UI. For more information, see [Example: Creating an In-Network Service Chain by Using Contrail Web UI](#).

## 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 | 660](#)
- [Create Virtual Machine | 661](#)
- [Configure Service Template | 662](#)
- [Add Service Instance | 663](#)
- [Create Service Policy | 664](#)
- [Attach Service Policy | 665](#)
- [Launch Virtual Machine | 665](#)

These topics provide instructions to create an in-network service chain.

## Create Virtual Network

### Step-by-Step Procedure

Use the Contrail Command UI to create a left virtual network, right virtual network, and management virtual network.

To create a left virtual network:

1. Click **Overlay>Virtual Networks**.

The All Networks page is displayed.

2. Click **Create** to create a network.

The Create Virtual Network page is displayed.

3. In the **Name** field enter **test-left-VN** for the left virtual network.

4. Select **(Default) User defined subnet only** from the **Allocation Mode** list.

5. Click **+Add** in the Subnets section to add subnets.

### Step-by-Step Procedure

In the row that is displayed,

- a. Click the arrow in the Network IPAM field and select **left-ipam** for the left virtual network.

For the right virtual network, select **right-ipam** and for the management network, select **mgmt-ipam**.



**NOTE:** Management network is not used to route packets. This network is used to help debug issues with the virtual machine.

6. Enter **192.0.2.0/24** in the **CIDR** field.

7. Click **Create**.

The All Networks page is displayed. All virtual networks that you created are displayed in this page.

Repeat steps "2" on page 660 through "7" on page 660 to create the right virtual network (**test-right-VN**) and management virtual network (**test-mgmt-VN**).

## Create Virtual Machine

### Step-by-Step Procedure

Follow these steps to create a left virtual machine by using the Contrail Command UI.

1. Click **Workloads > Instances**.

The Instances page is displayed.

2. Click **Create**.

The Create Instance page is displayed.

3. Select **Virtual Machine** option button as the serve type.

4. Enter **test-left-VM** for the left virtual machine in the **Instance Name** field.

5. Select **Image** as the boot source from the **Select Boot Source** list.



**NOTE:** vSRX image with M1.large flavor is recommended for in-network virtual machine.

6. Select **vSRX image** file from the **Select Image** list.

7. Select **M1.large** flavor from the **Select Flavor** list.

8. Select the network you want to associate with the left virtual machine by clicking > next to the name of the virtual machine listed in the Available Networks table.

For the left virtual machine, select **test-left-VN**. For the right virtual machine, select **test-right-VN**. For the management virtual machine, select **test-mgmt-VN**.

The network is added to the Allocated Networks table.

9. Select **nova** from the **Availability Zone** list.



**NOTE:** You can choose any other availability zone.

10. Select **5** from the **Count (1-10)** list.



**NOTE:** You can choose any value from 1 through 10.

11. Click **Create** to launch the left virtual machine instance.

The Instances page is displayed. The virtual machine instances that you created are listed on the Instances page.

Repeat steps "2" on page 661 through "11" on page 661 to create right virtual machine instance (**test-right-VM**) and management virtual machine instance (**test-mgmt-VM**).

## Configure Service Template

### Step-by-Step Procedure

Follow these steps to create a service template by using the Contrail Command UI:

1. Click **Services>Catalog**.

The VNF Service Templates page is displayed.

2. Click **Create**.

The Create VNF Service Template page is displayed.

3. Enter **test-service-template** in the **Name** field.

4. Select **v2** as the version type.



**NOTE:** Starting with Release 3.2, Contrail supports only *Service Chain Version 2 (v2)*.

5. Select **Virtual Machine** as the virtualization type.

6. Select **In-Network** as the service mode.

7. Select **Firewall** as the service type.

8. From the Interface section,

- Select **left** as the interface type from the **Interface Type** list.
- Click **+ Add**.

The Interface Type list is added to the table.

Select **right** as the interface type.

- Click **+ Add** again.

Another Interface Type list is added to the table.

Select **management** as the interface type.



**NOTE:** The interfaces created on the virtual machine must follow the same sequence as that of the interfaces in the service template.

9. Click **Create** to create the service template.

The VNF Service Templates page is displayed. The service template that you created is displayed in the VNF Service Templates page.

## Add Service Instance

### Step-by-Step Procedure

Follow these steps to add a service instance by using the Contrail Command UI:

1. Click **Services>Deployments**.

The VNF Service Instances page is displayed.

2. Click **Create**.

The Create VNF Service Instance page is displayed.

3. Enter **test-service-instance** in the **Name** field.

4. Select **test-service-template - [in-network, (left, right, management)] - v2** from the **Service Template** list.

The **Interface Type** and **Virtual Network** fields are displayed.

5. Select the virtual network for each interface type as given below.

- **left**—Select the left virtual network (**test-left-VN**) that you created.
- **right**—Select the right virtual network (**test-right-VN**) that you created.
- **management**—Select the management virtual network (**test-management-VN**) that you created.

6. Click the **Port Tuples** section and click **+Add**.

Select the virtual machine instance for each interface type as given below.

- **left**—Select the left virtual machine instance that you created.
- **right**—Select the right virtual machine instance that you created.
- **management**—Select the management virtual machine instance that you created.

7. Click **Create** to create the service instance.

The VNF Service Instances page is displayed. The service instance that you created is displayed in the VNF Service Instances page.

## Create Service Policy

### Step-by-Step Procedure

Follow these steps to create a service policy by using the Contrail Command UI.

**1. Click **Overlay** > **Network Policies**.**

The Network Policies page is displayed.

**2. Click **Create**.**

The Network Policy tab of the Create Network Policy page is displayed.

**3. Enter **test-network-policy** in the **Policy Name** field.**

**4. In the **Policy Rule(s)** section,**

- Select **pass** from the **Action** list.
- Select **ANY** from the **Protocol** list.
- Select **Network** from the **Source Type** list.
- Select the **test-left-VN** from the **Source** list.
- In the **Source Port** field, leave the default option, **Any**, as is.
- Select **< >** from the **Direction** list.
- Select **Network** from the **Destination Type** list.
- Select the **test-right-VN** from the **Destination** list.
- In the **Destination Ports** field, leave the default option, **Any**, as is.

**5. Click **Create** to create the service policy.**

The Network Policies page is displayed. All policies that you created are displayed in the Network Policies page.



## Attach Service Policy

### Step-by-Step Procedure

Follow these steps to attach a service policy:

1. Click **Overlay>Virtual Networks**.

The All networks page is displayed.

2. Attach service policy to the left virtual network (**test-left-VN**) and right virtual network (**test-right-VN**) that you created.

### Step-by-Step Procedure

To attach service policy,

- a. Select the check box next to the name of the virtual network.
- b. Hover over to the end of the selected row and click the **Edit** icon.

The Edit Virtual Network page is displayed.

- c. Select the network policy from the Network Policies list.

3. Click **Save** to save the changes.

The Virtual Networks page is displayed.

## Launch Virtual Machine

### Step-by-Step Procedure

You can launch virtual machines from Contrail Command and test the traffic through the service chain by doing the following:

1. Launch the left virtual machine in left virtual network. See ["Create Virtual Machine" on page 661](#).
2. Launch the right virtual machine in right virtual network. See ["Create Virtual Machine" on page 661](#).
3. Ping the left virtual machine IP address from the right virtual machine.

Follow these steps to ping a virtual machine:

### Step-by-Step Procedure

- a. Click **Workloads>Instances**.

The Instances page is displayed.

- b. Click the open console icon next to **test-right-VM**.

The Console page is displayed.

- c. Log in using root user credentials.
- d. Ping the left virtual machine IP address (**190.0.2.3**) from the Console.

See [Figure 116 on page 666](#) for a sample output.

**Figure 116: Ping test-left-VM**

```
root@test-right-vm:~# ping -c 5 192.0.2.3
PING 192.0.2.3 (192.0.2.3) 56(84) bytes of data:
64 bytes from 192.0.2.3: icmp_seq=1 ttl=63 time=0.238 ms
64 bytes from 192.0.2.3: icmp_seq=2 ttl=63 time=0.208 ms
64 bytes from 192.0.2.3: icmp_seq=3 ttl=63 time=0.231 ms
64 bytes from 192.0.2.3: icmp_seq=4 ttl=63 time=0.210 ms
64 bytes from 192.0.2.3: icmp_seq=5 ttl=63 time=0.210 ms

--- 192.0.2.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 0.208/0.219/0.238/0.018 ms
root@test-right-vm:~#
```

## RELATED DOCUMENTATION

*Service Chaining*

*Example: Creating an In-Network-NAT Service Chain*

*Example: Creating a Transparent Service Chain by Using Contrail Command*

## Example: Creating an In-Network-NAT Service Chain by Using Contrail Command

### IN THIS SECTION

- [Prerequisites | 667](#)
- [Overview | 668](#)

This example provides instructions to create an in-network-nat service chain by using the Contrail Command user interface (UI).

## Prerequisites

- **Hardware and Software Requirements**

### Hardware

- Processor: 4 core x86
- Memory: 32GB RAM
- Storage: at least 128GB hard disk

### Software

- Contrail Release 3.2 or later



**NOTE:** For Contrail Networking Release 3.2 through Release 4.1, you use the Contrail Web UI. For more information, see [Example: Creating a In-Network-NAT Service Chain by Using Contrail Web UI](#).

- **Create Network IPAM (IP Address Management)**

1. Click **Overlay>IPAM**.

The IP Address Management page is displayed.

2. Click **Create** to create a new network IPAM.
3. Enter a name for the IPAM in the name field.
4. Select **Default** from the DNS list.
5. Enter valid IP address in the NTP Server IP field.
6. Enter domain name in the Domain Name field.
7. Click **Create**.

The IP Address Management page is displayed.

## 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 | 668](#)
- [Create Virtual Machine | 669](#)
- [Configure Service Template | 670](#)
- [Add Service Instance | 671](#)
- [Create Service Policy | 672](#)
- [Attach Service Policy | 673](#)
- [Launch Virtual Machine | 674](#)

These topics provide instructions to create an in-network-nat service chain.

### Create Virtual Network

#### Step-by-Step Procedure

Use the Contrail Command UI to create a left virtual network, right virtual network, and management virtual network.

To create a left virtual network:

1. Click **Overlay>Virtual Networks**.

The All Networks page is displayed.

2. Click **Create** to create a network.

The Create Virtual Network page is displayed.

3. In the **Name** field enter **test-left-VN** for the left virtual network.
4. Select **(Default) User defined subnet only** from the **Allocation Mode** list.
5. Click **+Add** in the Subnets section to add subnets.

### Step-by-Step Procedure

In the row that is displayed,

- a. Select an IPAM for the virtual network from the Network IPAM list.
  - b. Enter **192.0.2.0/24** in the **CIDR** field.
6. Click **Create**.

The All Networks page is displayed. All virtual networks that you created are displayed in this page.



**NOTE:** Management network is not used to route packets. This network is used to help debug issues with the virtual machine.

Repeat steps "2" on page 668 through "6" on page 669 to create the right virtual network (**test-right-VN**) and management virtual network (**test-mgmt-VN**).

## Create Virtual Machine

### Step-by-Step Procedure

Follow these steps to create a left virtual machine by using the Contrail Command UI.

1. Click **Workloads > Instances**.

The Instances page is displayed.

2. Click **Create**.

The Create Instance page is displayed.

3. Select **Virtual Machine** option button as the serve type.
4. Enter **test-left-VM** for the left virtual machine in the **Instance Name** field.
5. Select **Image** as the boot source from the **Select Boot Source** list.



**NOTE:** vSRX image with M1.large flavor is recommended for in-network-nat virtual machine.

6. Select **vSRX image** file from the **Select Image** list.
7. Select **M1.large** flavor from the **Select Flavor** list.
8. Select the network you want to associate with the left virtual machine by clicking > next to the name of the virtual machine listed in the Available Networks table.

For the left virtual machine, select **test-left-VN**. For the right virtual machine, select **test-right-VN**. For the management virtual machine, select **test-mgmt-VN**.

The network is added to the Allocated Networks table.

9. Select **nova** from the **Availability Zone** list.



**NOTE:** You can choose any other availability zone.

10. Select **5** from the **Count (1-10)** list.



**NOTE:** You can choose any value from 1 through 10.

11. Click **Create** to launch the left virtual machine instance.

The Instances page is displayed. The virtual machine instances that you created are listed on the Instances page.

Repeat steps "2" on page 669 through "11" on page 670 to create right virtual machine instance (**test-right-VM**) and management virtual machine instance (**test-mgmt-VM**).

## Configure Service Template

### Step-by-Step Procedure

Follow these steps to create a service template by using the Contrail Command UI:

1. Click **Services>Catalog**.

The VNF Service Templates page is displayed.

2. Click **Create**.

The Create VNF Service Template page is displayed.

3. Enter **test-service-template** in the **Name** field.
4. Select **v2** as the version type.



**NOTE:** Starting with Release 3.2, Contrail supports only *Service Chain Version 2 (v2)*.

5. Select **Virtual Machine** as the virtualization type.
6. Select **In-Network Nat** as the service mode.
7. Select **Firewall** as the service type.
8. From the Interface section,
  - Select **left** as the interface type from the **Interface Type** list.
  - Click **+ Add**.

The Interface Type list is added to the table.

Select **right** as the interface type.

- Click **+ Add** again.

Another Interface Type list is added to the table.

Select **management** as the interface type.



**NOTE:** The interfaces created on the virtual machine must follow the same sequence as that of the interfaces in the service template.

9. Click **Create** to create the service template.

The VNF Service Templates page is displayed. The service template that you created is displayed in the VNF Service Templates page.

## Add Service Instance

### Step-by-Step Procedure

Follow these steps to add a service instance by using the Contrail Command UI:

1. Click **Services>Deployments**.

The VNF Service Instances page is displayed.

2. Click **Create**.

The Create VNF Service Instance page is displayed.

3. Enter **test-service-instance** in the **Name** field.

4. Select **test-service-template - [in-network-nat, (left, right, management)] - v2** from the **Service Template** list.

The **Interface Type** and **Virtual Network** fields are displayed.

5. Select the virtual network for each interface type as given below.

- **left**—Select the left virtual network (**test-left-VN**) that you created.
- **right**—Select the right virtual network (**test-right-VN**) that you created.
- **management**—Select the management virtual network (**test-management-VN**) that you created.

6. Click the **Port Tuples** section and click **+Add**.

Select the virtual machine instance for each interface type as given below.

- **left**—Select the left virtual machine instance that you created.
- **right**—Select the right virtual machine instance that you created.
- **management**—Select the management virtual machine instance that you created.

7. Click **Create** to create the service instance.

The VNF Service Instances page is displayed. The service instance that you created is displayed in the VNF Service Instances page.

## Create Service Policy

### Step-by-Step Procedure

Follow these steps to create a service policy by using the Contrail Command UI.

1. Click **Overlay > Network Policies**.

The Network Policies page is displayed.

2. Click **Create**.

The Network Policy tab of the Create Network Policy page is displayed.



3. Enter **test-network-policy** in the **Policy Name** field.

4. In the **Policy Rule(s)** section,

- Select **pass** from the **Action** list.
- Select **ANY** from the **Protocol** list.
- Select **Network** from the **Source Type** list.
- Select the **test-left-VN** from the **Source** list.
- In the **Source Port** field, leave the default option, **Any**, as is.
- Select **< >** from the **Direction** list.
- Select **Network** from the **Destination Type** list.
- Select the **test-right-VN** from the **Destination** list.
- In the **Destination Ports** field, leave the default option, **Any**, as is.

5. Click **Create** to create the service policy.

The Network Policies page is displayed. All policies that you created are displayed in the Network Policies page.

## Attach Service Policy

### Step-by-Step Procedure

Follow these steps to attach a service policy:

1. Click **Overlay>Virtual Networks**.

The All networks page is displayed.

2. Attach service policy to the left virtual network (**test-left-VN**) and right virtual network (**test-right-VN**) that you created.

### Step-by-Step Procedure

To attach service policy,

- a. Select the check box next to the name of the virtual network.
- b. Hover over to the end of the selected row and click the **Edit** icon.

The Edit Virtual Network page is displayed.

- c. Select the network policy from the Network Policies list.

3. Click **Save** to save the changes.

The Virtual Networks page is displayed.

## Launch Virtual Machine

### Step-by-Step Procedure

You can launch virtual machines from Contrail Command and test the traffic through the service chain by doing the following:

1. Launch the left virtual machine in left virtual network. See ["Create Virtual Machine" on page 669](#).
2. Launch the right virtual machine in right virtual network. See ["Create Virtual Machine" on page 669](#).
3. Ping the left virtual machine IP address from the right virtual machine.

Follow these steps to ping a virtual machine:

### Step-by-Step Procedure

- a. Click **Workloads>Instances**.

The Instances page is displayed.

- b. Click the open console icon next to **test-right-VM**.

The Console page is displayed.

- c. Log in using root user credentials.

- d. Ping the left virtual machine IP address (**190.0.2.3**) from the Console.

## RELATED DOCUMENTATION

*Service Chaining*

---

*Example: Creating an In-Network Service Chain by Using Contrail Command*

---

*Example: Creating a Transparent Service Chain by Using Contrail Command*

## Example: Creating a Transparent Service Chain by Using Contrail Command

### IN THIS SECTION

- [Prerequisites | 675](#)
- [Overview | 676](#)
- [Configuration | 676](#)

This example provides step-by-step instructions to create a transparent service chain by using the Contrail Command user interface (UI).

### Prerequisites

- **Hardware and Software Requirements**

#### Hardware

- Processor: 4 core x86
- Memory: 32GB RAM
- Storage: at least 128GB hard disk

#### Software

- Contrail Release 3.2 or later



**NOTE:** For Contrail Networking Release 3.2 through Release 4.1, you use the Contrail Web UI. For more information, see [Example: Creating a Transparent Service Chain by Using Contrail Web UI](#).

- **Create Network IPAM (IP Address Management)**

1. Click **Overlay>IPAM**.

The IP Address Management page is displayed.

2. Click **Create** to create a new network IPAM.
3. Enter a name for the IPAM in the name field.

4. Select **Default** from the DNS list.
5. Enter valid IP address in the NTP Server IP field.
6. Enter domain name in the Domain Name field.
7. Click **Create**.

The IP Address Management page is displayed.

## 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 Networks | 677](#)
- [Create Secondary Virtual Network | 678](#)
- [Create Service Virtual Machine | 678](#)
- [Create Virtual Machine | 679](#)
- [Configure Service Template | 680](#)
- [Add Service Instance | 681](#)
- [Create Service Policy | 682](#)
- [Attach Service Policy | 683](#)
- [Launch Virtual Machine | 683](#)

These topics provide instructions to create a transparent service chain.

## Create Primary Virtual Networks

### Step-by-Step Procedure

Use the Contrail Command UI 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.

Follow these steps To create a left virtual network:

1. Click **Overlay>Virtual Networks**.

The All Networks page is displayed.

2. Click **Create** to create a network.

The Create Virtual Network page is displayed.

3. In the **Name** field enter **test-left-VN** for the left virtual network.

4. Select **(Default) User defined subnet only** from the **Allocation Mode** list.

5. Click **+Add** in the Subnets section to add subnets.

### Step-by-Step Procedure

In the row that is displayed,

- a. Select an IPAM for the virtual network from the Network IPAM list.

- b. Enter **192.0.2.0/24** in the **CIDR** field.

6. Click **Create**.

The All Networks page is displayed. All virtual networks that you created are displayed in this page.



**NOTE:** Management network is not used to route packets. This network is used to help debug issues with the virtual machine.

Repeat steps "2" on page 677 through "6" on page 677 to create the right virtual network (**test-right-VN**) and management virtual network (**test-mgmt-VN**).

## Create Secondary Virtual Network

### Step-by-Step Procedure

Use the Contrail Command UI to create three secondary virtual networks: left virtual network (**trans-left-VN**), right virtual network (**trans-right-VN**), and management virtual network (**trans-mgmt-VN**). 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 Networks" on page 677](#).

## Create Service Virtual Machine

### Step-by-Step Procedure

Follow these steps to create a service virtual machine (SVM) by using the Contrail Command UI.

1. Click **Workloads > Instances**.

The Instances page is displayed.

2. Click **Create**.

The Create Instance page is displayed.

3. Select **Virtual Machine** option button as the serve type.

4. Enter **test-SVM** in the **Instance Name** field.

5. Select **Image** as the boot source from the **Select Boot Source** list.



**NOTE:** vSRX image with M1.large flavor is recommended for in-network virtual machine.

6. Select **vSRX image** file from the **Select Image** list.

7. Select **M1.large** flavor from the **Select Flavor** list.

8. From the Available Networks table, select **trans-left-VN**, **trans-right-VN**, and **trans-mgmt-VN** networks that you want to associate with the SVM by clicking **>** next to the name of the virtual machine.

The network is added to the Allocated Networks table.

9. Select **nova** from the **Availability Zone** list.



**NOTE:** You can choose any other availability zone.

10. Select **5** from the **Count (1-10)** list.



**NOTE:** You can choose any value from 1 through 10.

11. Click **Create** to launch the left virtual machine instance.

The Instances page is displayed. The virtual machine instances that you created are listed on the Instances page.

## Create Virtual Machine

### Step-by-Step Procedure

Follow these steps to create a left virtual machine by using the Contrail Command UI.

1. Click **Workloads > Instances**.

The Instances page is displayed.

2. Click **Create**.

The Create Instance page is displayed.

3. Select **Virtual Machine** option button as the serve type.

4. Enter **test-left-VM** for the left virtual machine in the **Instance Name** field.

5. Select **Image** as the boot source from the **Select Boot Source** list.



**NOTE:** vSRX image with M1.large flavor is recommended for in-network virtual machine.

6. Select **vSRX image** file from the **Select Image** list.

7. Select **M1.large** flavor from the **Select Flavor** list.

8. From the Available Networks table, select **test-left-VN** network that you want to associate with the left virtual machine by clicking **>** next to the name of the virtual machine.

For the right virtual machine, select **test-right-VN**.

The network is added to the Allocated Networks table.

9. Select **nova** from the **Availability Zone** list.



**NOTE:** You can choose any other availability zone.

10. Select **5** from the **Count (1-10)** list.



**NOTE:** You can choose any value from 1 through 10.

11. Click **Create** to launch the left virtual machine instance.

The Instances page is displayed. The virtual machine instances that you created are listed on the Instances page.

Repeat steps "2" on page 679 through "11" on page 680 to create right virtual machine instance (**test-right-VM**).

## Configure Service Template

### Step-by-Step Procedure

Follow these steps to create a service template by using the Contrail Command UI:

1. Click **Services>Catalog**.

The VNF Service Templates page is displayed.

2. Click **Create**.

The Create VNF Service Template page is displayed.

3. Enter **test-service-template** in the **Name** field.

4. Select **v2** as the version type.



**NOTE:** Starting with Release 3.2, Contrail supports only *Service Chain Version 2 (v2)*.

5. Select **Virtual Machine** as the virtualization type.

6. Select **Transparent** as the service mode.

7. Select **Firewall** as the service type.



8. From the Interface section,

- Select **left** as the interface type from the **Interface Type** list.
- Click **+ Add**.

The Interface Type list is added to the table.

Select **right** as the interface type.

- Click **+ Add** again.

Another Interface Type list is added to the table.

Select **management** as the interface type.



**NOTE:** The interfaces created on the virtual machine must follow the same sequence as that of the interfaces in the service template.

9. Click **Create** to create the service template.

The VNF Service Templates page is displayed. The service template that you created is displayed in the VNF Service Templates page.

## Add Service Instance

### Step-by-Step Procedure

Follow these steps to add a service instance by using the Contrail Command UI:

1. Click **Services>Deployments**.

The VNF Service Instances page is displayed.

2. Click **Create**.

The Create VNF Service Instance page is displayed.

3. Enter **test-service-instance** in the **Name** field.

4. Select **test-service-template - [transparent, (left, right, management)] - v2** from the **Service Template** list.

The **Interface Type** and **Virtual Network** fields are displayed.

5. Select the virtual network for each interface type as given below.

- **left**—Select **trans-left-VN** virtual network that you created.

- **right**—Select the **trans-right-VN** virtual network that you created.
- **management**—Select the **trans-mgmt-VN** virtual network that you created.

6. Click the **Port Tuples** section and click **+Add**.

Select the virtual machine instance for each interface type as given below. The port tuples should match the interfaces of the SVM. See ["Create Service Virtual Machine" on page 678](#).

- **left**—Select the left virtual machine instance that you created.
- **right**—Select the right virtual machine instance that you created.
- **management**—Select the management virtual machine instance that you created.

7. Click **Create** to create the service instance.

The VNF Service Instances page is displayed. The service instance that you created is displayed in the VNF Service Instances page.

## Create Service Policy

### Step-by-Step Procedure

Follow these steps to create a service policy by using the Contrail Command UI.

1. Click **Overlay > Network Policies**.

The Network Policies page is displayed.

2. Click **Create**.

The Network Policy tab of the Create Network Policy page is displayed.

3. Enter **test-network-policy** in the **Policy Name** field.

4. In the **Policy Rule(s)** section,

- Select **pass** from the **Action** list.
- Select **ANY** from the **Protocol** list.
- Select **Network** from the **Source Type** list.
- Select the **test-left-VN** from the **Source** list.
- In the **Source Port** field, leave the default option, **Any**, as is.
- Select **< >** from the **Direction** list.

- Select **Network** from the **Destination Type** list.
- Select the **test-right-VN** from the **Destination** list.
- In the **Destination Ports** field, leave the default option, **Any**, as is.

5. Click **Create** to create the service policy.

The Network Policies page is displayed. All policies that you created are displayed in the Network Policies page.

## Attach Service Policy

### Step-by-Step Procedure

Follow these steps to attach a service policy:

1. Click **Overlay>Virtual Networks**.

The All networks page is displayed.

2. Select the **test-left-VN** network that you want to edit, and click the **Edit** icon.

The Edit Virtual Network page is displayed.



**NOTE:** For the right virtual network, edit **test-right-VN**.

3. Select **test-network-policy** from the Network Policies list.

4. Click **Save** to save the changes.

The Virtual Networks page is displayed.

Repeat steps "2" on page 683 through "4" on page 683 to attach the service policy to **test-right-VN**.

## Launch Virtual Machine

### RELATED DOCUMENTATION

*Service Chaining*

*Example: Creating an In-Network-NAT Service Chain by Using Contrail Command*

*Example: Creating an In-Network Service Chain by Using Contrail Command*

# Adding Physical Network Functions in Service Chains

## IN THIS CHAPTER

- [Using Physical Network Functions in Contrail Service Chains | 684](#)
- [Example: Adding a Physical Network Function Device to a Service Chain | 686](#)

## Using Physical Network Functions in Contrail Service Chains

### IN THIS SECTION

- [PNF Service Chaining Objects | 684](#)
- [Prerequisites and Assumptions | 685](#)

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](#) | 686

## Example: Adding a Physical Network Function Device to a Service Chain

### IN THIS SECTION

- [Prerequisites for Adding a PNF to a Service Chain | 686](#)

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

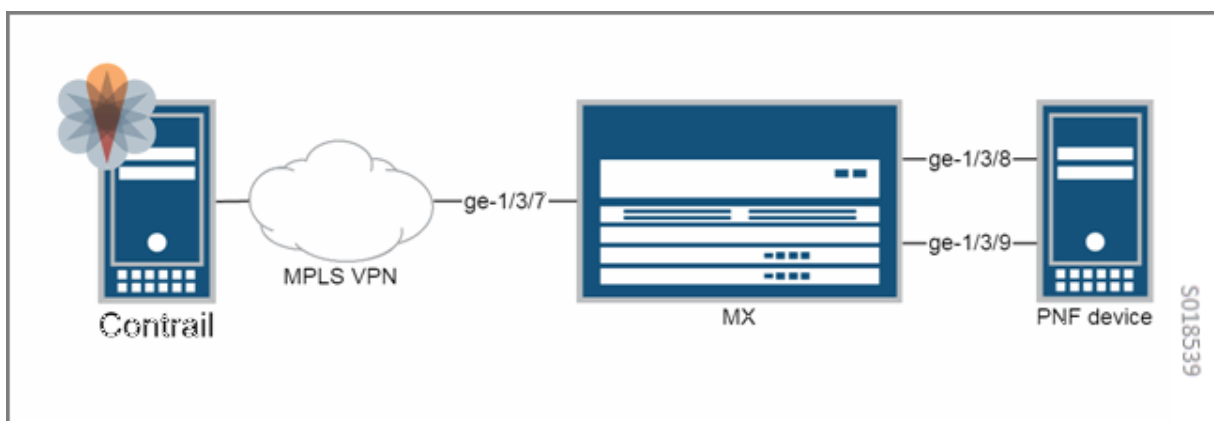
The following are the minimum requirements needed before you can add a PNF to a service chain using the procedure shown in the example included in this topic:

- 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;
        }
      }
    }
  }
}

```

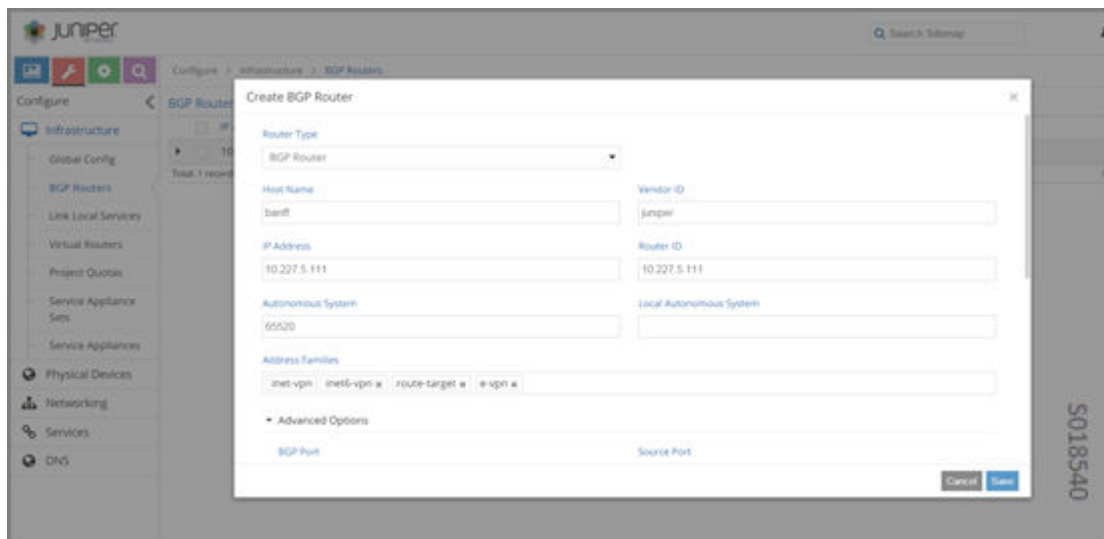
```

    }
  }
  routing-instances {
    <*-sc-entry-point> {
      forwarding-options {
        family inet {
          filter {
            input skip_tdf_service;
          }
        }
      }
    }
  }
}

```

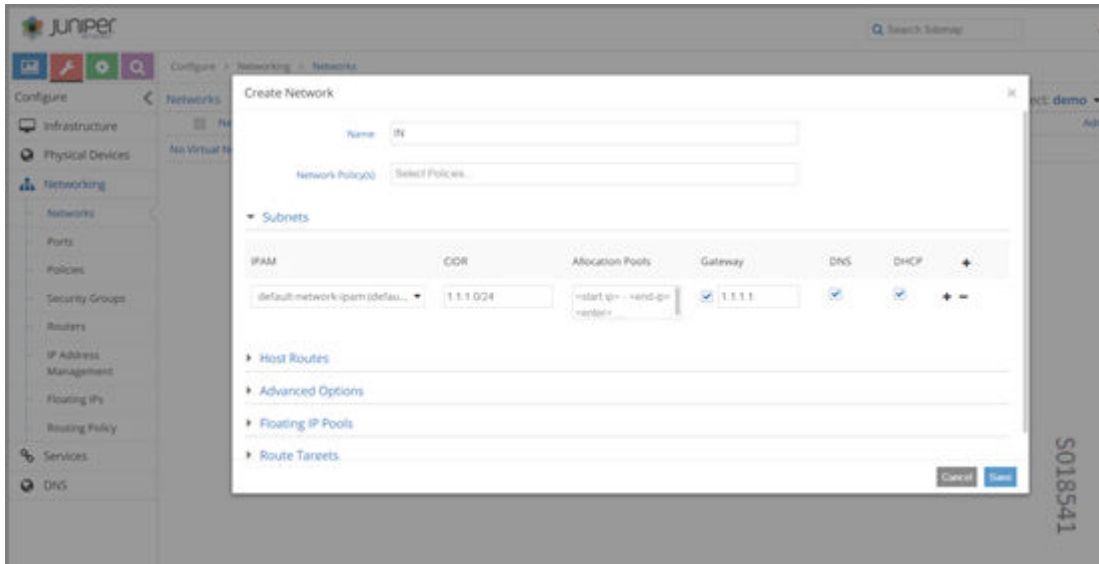
### Procedure: Adding a PNF to a Service Chain

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.

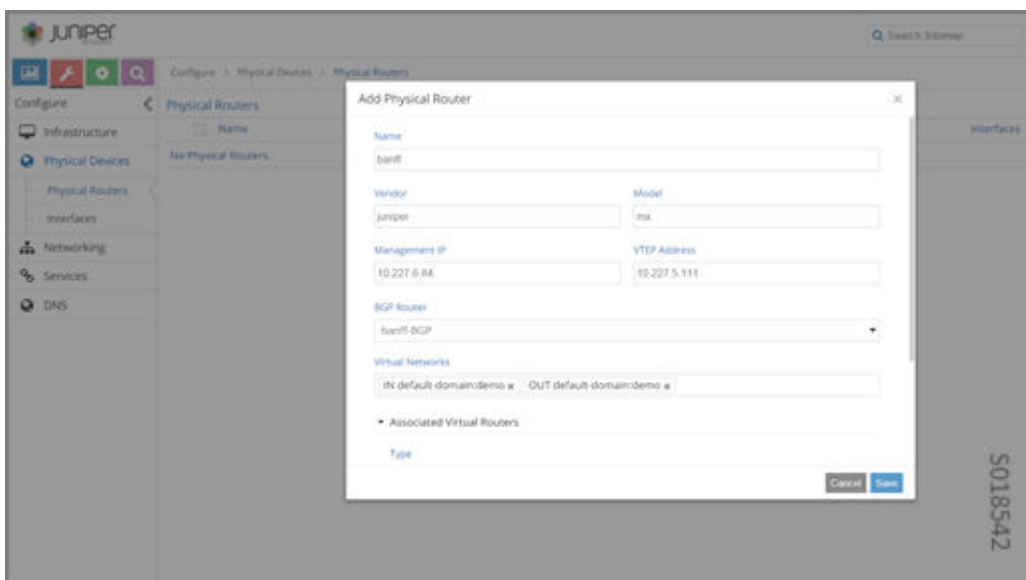


2. Create two virtual networks. Select **Configure > Networking > Networks** and create a network named **IN** and a network named **OUT**. The following figure provides an example.

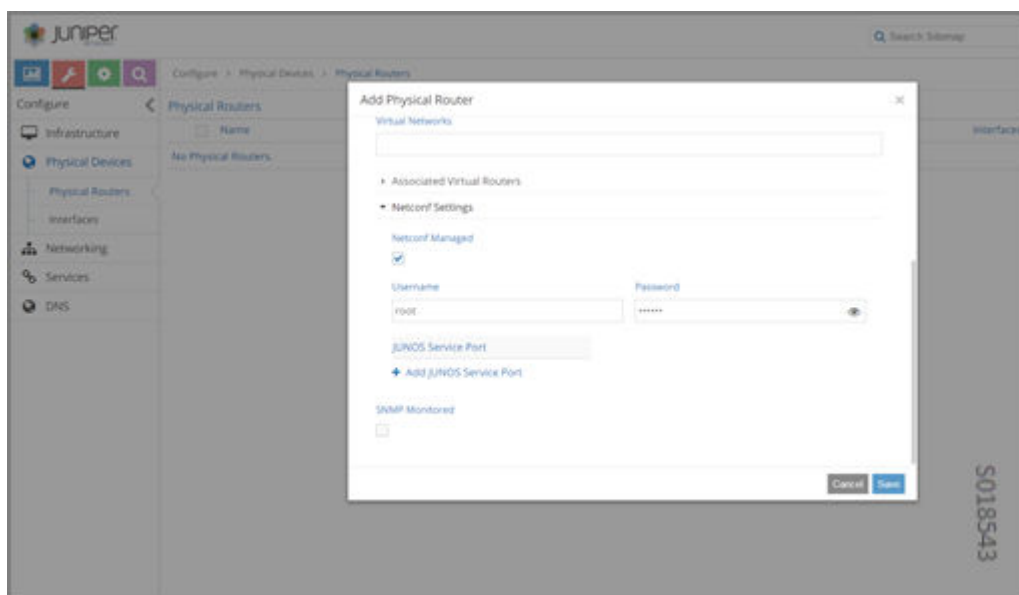




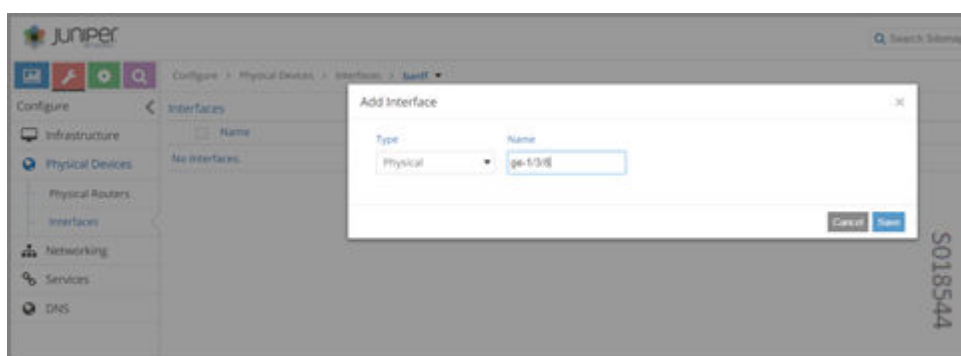
3. Create a physical router associated with the BGP router. Select **Configure > Physical Devices > Physical Routers** and create a physical router. The VTEP address of the physical router should be same as the BGP router's IP address. Associate the physical router with the BGP router created previously, and select for Virtual Networks the networks created for this example (**IN** and **OUT**). The following figure provides an example.



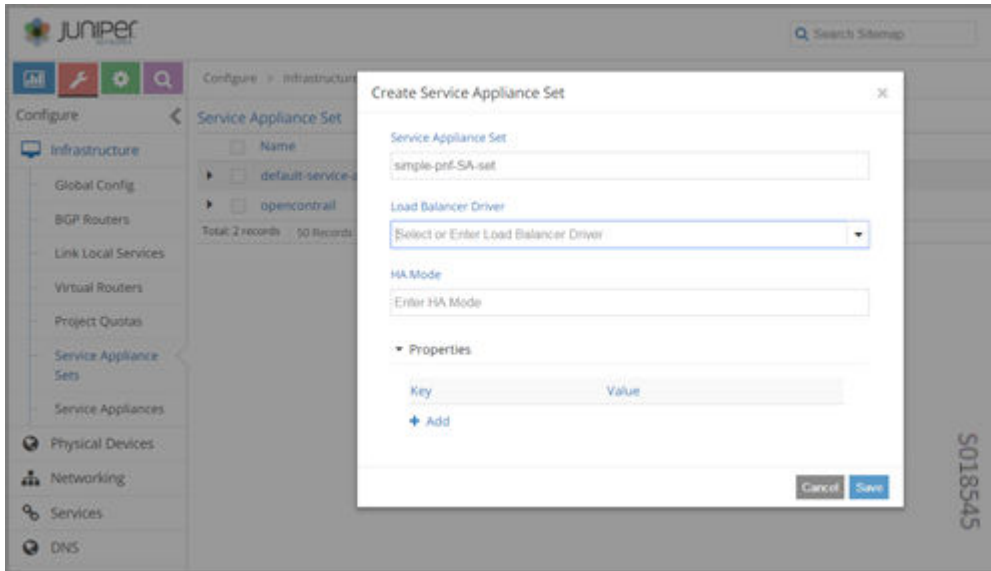
4. While still on the **Add Physical Router** window, use the slider to scroll down to the **Netconf Settings** section and add the appropriate NETCONF information for your system. The following figure provides an example.



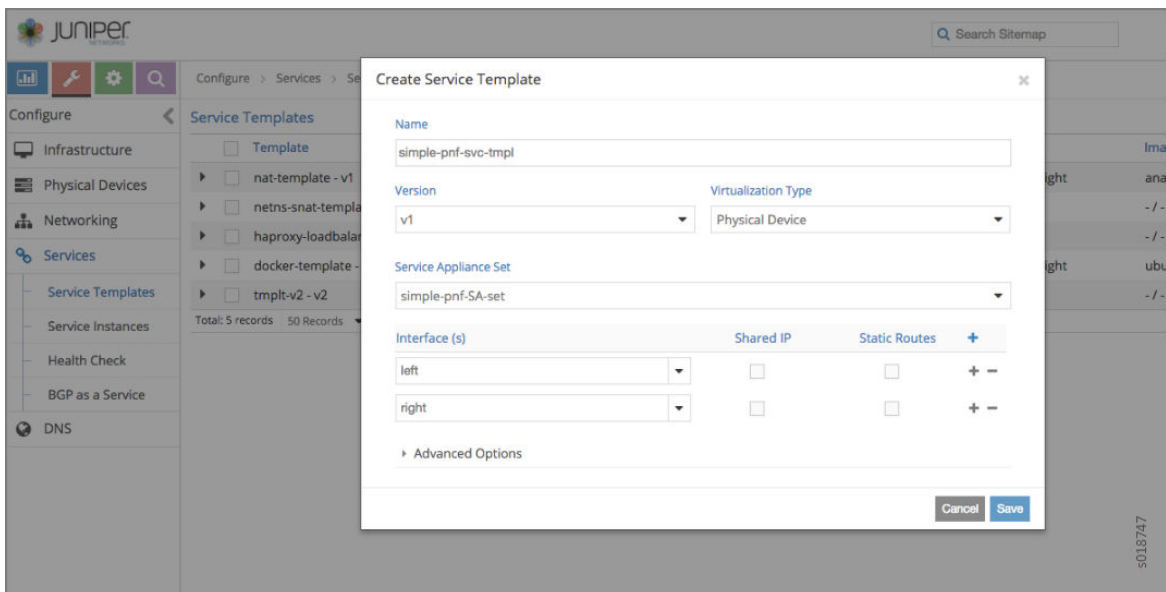
5. Add the physical interfaces that connect to the PNF device. Go to **Configure > Physical Devices > Interfaces** and select the PNF to get to the **Add Interfaces** window, where you enter the name and type for each interface. The following figure provides an example.



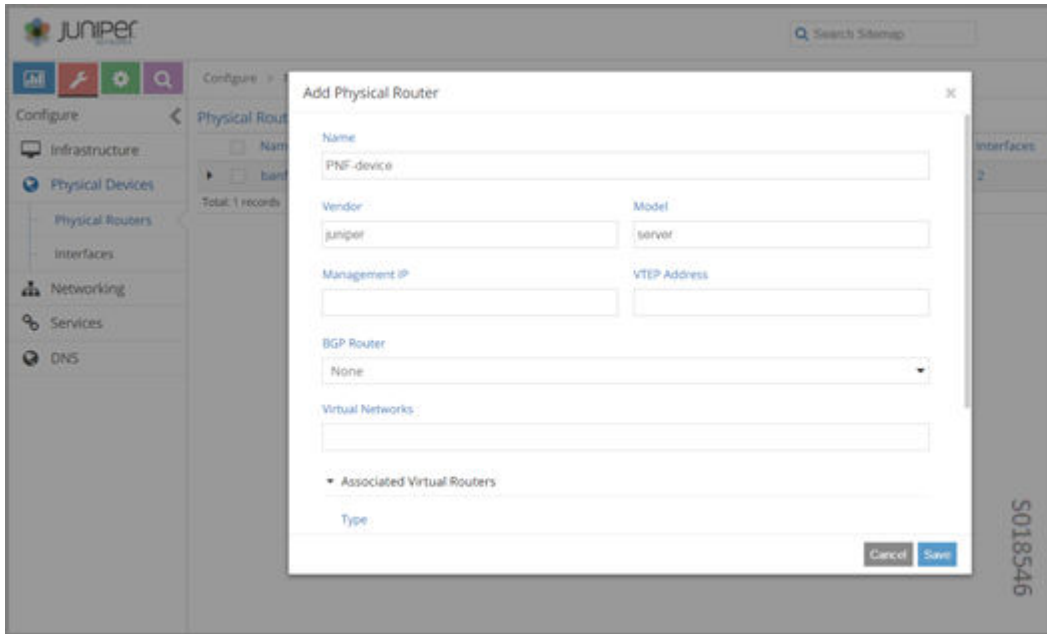
6. Add a service appliance set. Go to **Configure > Infrastructure > Service Appliance Sets** to get to the **Create Service Appliance Set** window, where you enter the name of the service appliance set. The following figure provides an example.



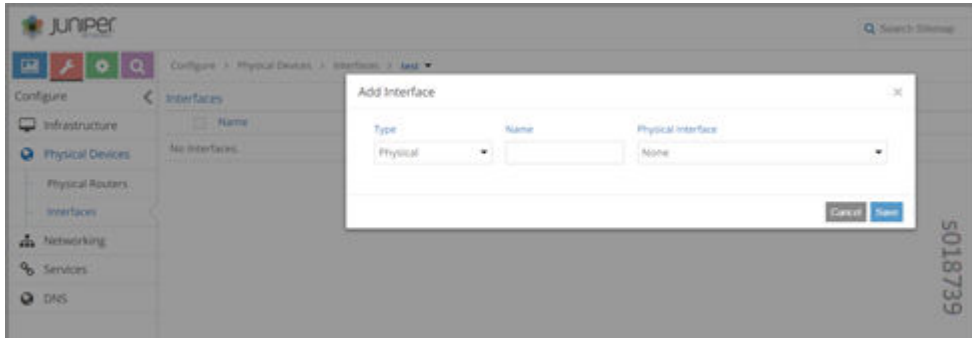
7. Configure a service template, **Configure > Services > Service Templates** and click the **Create** button on **Service Templates** to get to **Add Service Template**. Ensure that the **Virtualization Type** is set to **Physical Device**, and that the template is associated to the service appliance set previously created. The following figure provides an example.



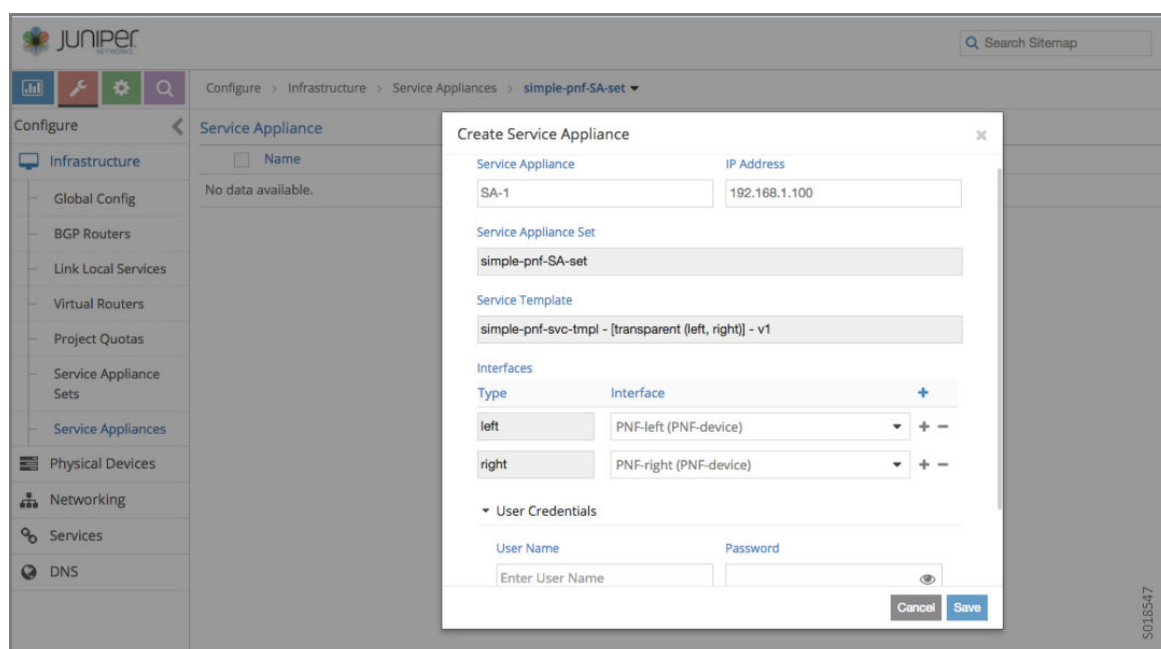
8. Add a physical router that represents the PNF device. Go to **Configure > Physical Devices > Physical Routers** to get to the **Add Physical Router** window, where you enter a name for the physical router. The following figure provides an example.



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*.

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*

[Using Physical Network Functions in Contrail Service Chains](#) | 684

# QoS Support in Contrail

## IN THIS CHAPTER

- [Quality of Service in Contrail | 694](#)
- [Configuring Network QoS Parameters | 703](#)

## Quality of Service in Contrail

### IN THIS SECTION

- [Overview: Quality of Service | 694](#)
- [Contrail QoS Model | 695](#)
- [QoS Configuration Parameters for Provisioning | 695](#)
- [Queuing Implementation | 697](#)
- [Contrail QoS Configuration Objects | 697](#)
- [Example: Mapping Traffic to Forwarding Classes | 699](#)
- [QoS Configuration Object Marking on the Packet | 700](#)
- [Queuing | 701](#)
- [Queue Selection in Datapath | 701](#)
- [Parameters for QoS Scheduling Configuration | 702](#)

### 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.

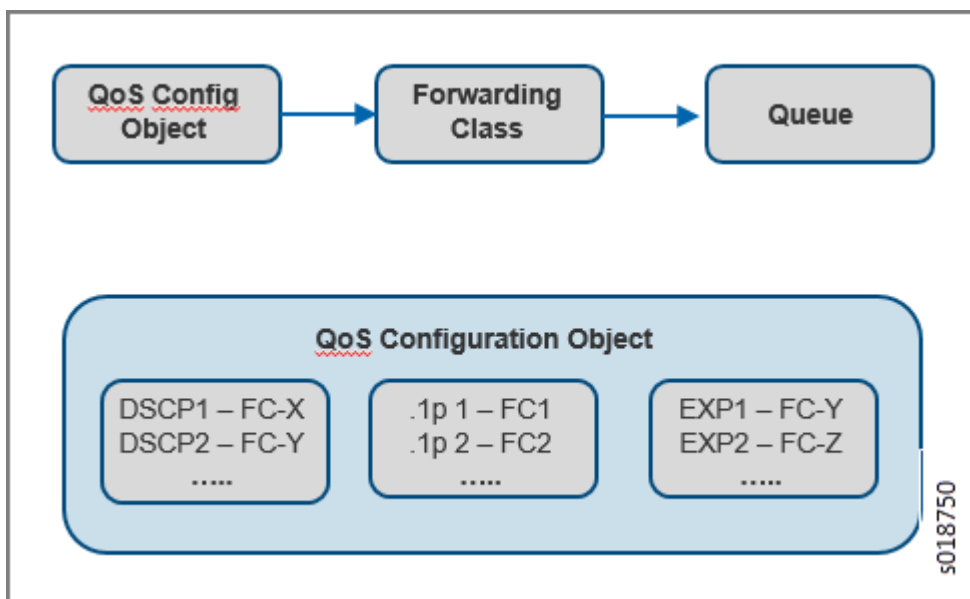
Processing of QoS marked packets to look up the corresponding forwarding class to be applied works as follows:

- 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.

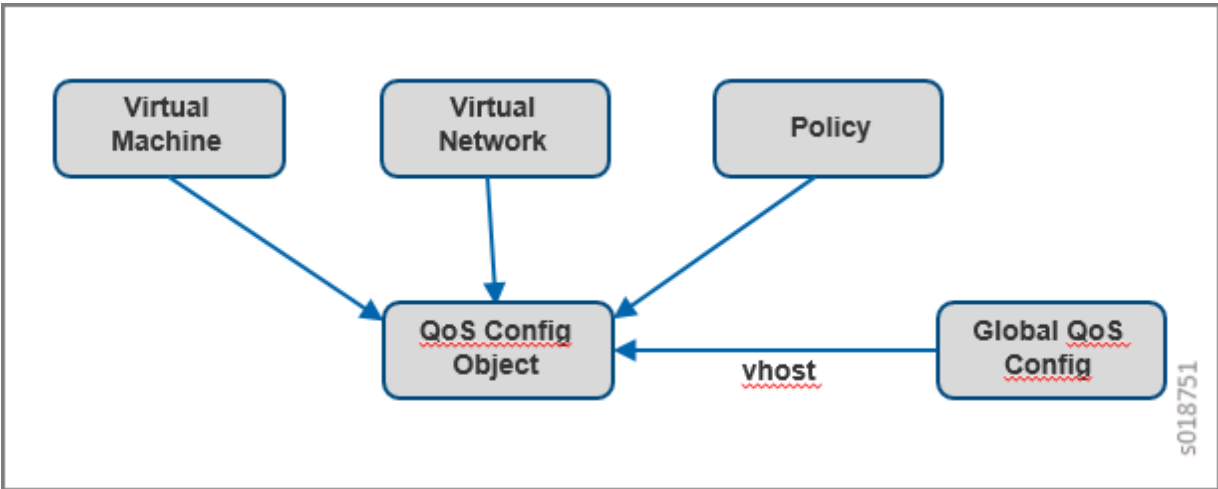
Figure 117 on page 698 shows the processing of QoS packets.

**Figure 117: 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 118 on page 699](#).

Figure 118: 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 43 on page 699 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 43: 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 44 on page 700, 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 44: 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 | 703](#)

<https://github.com/Juniper/contrail-controller/wiki/QoS> .

## Configuring Network QoS Parameters

### IN THIS SECTION

- [Overview | 703](#)
- [QoS Configuration Examples | 703](#)
- [Limitations | 705](#)

### 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 | 694

# BGP as a Service

## IN THIS CHAPTER

- [BGP as a Service | 706](#)
- [BGP as a Service in Contrail Release 3.1 | 710](#)

## BGP as a Service

## IN THIS SECTION

- [Contrail BGPaaS Features | 706](#)
- [BGPaaS Customer Use Cases | 708](#)
- [Configuring BGPaaS | 709](#)

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.

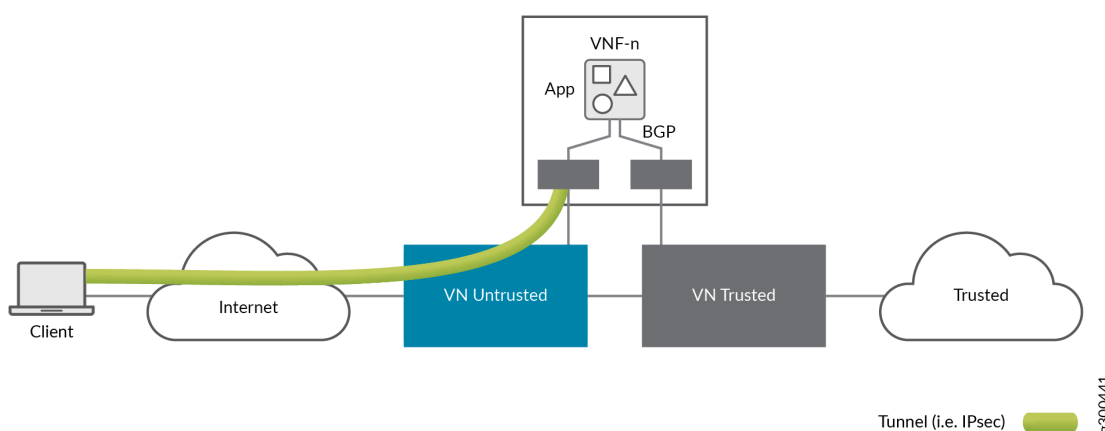
### Dynamic Tunnel Insertion Within a Tenant Overlay

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.

[Figure 119 on page 708](#) shows the need to insert a dynamic tunnel into a virtual network.

**Figure 119: Dynamic Tunnel Insertion**



### 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 Contrail user interface. On the left is a navigation menu with categories like Infrastructure, Physical Devices, Networking, Services, and DNS. The 'Services' section is expanded, showing 'Service Templates', 'Service Instances', 'Health Check', and 'BGP as a Service'. The main panel displays 'Configure > Services > BGP as a Service'. A modal dialog titled 'Create BGP as a Service' is open. It contains the following fields: 'Name' (bgpaas-1), 'IP Address' (11.95.197.1), 'Autonomous System' (50000), 'Address Family' (inet, inet6), 'Virtual Machine Interface(s)' (7d8fb01c-26af-4128-b030-ab446d3da0e5 (11.95.197.1)), and an 'Advanced Options' section with 'Hold Time' (90) and 'Admin State' (checked). 'Cancel' and 'Save' buttons are at the bottom right of the dialog.

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.

## BGP as a Service in Contrail Release 3.1

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. For more information on BGP as a Service, see ["BGP as a Service" on page 706](#).

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.

# Load Balancers

## IN THIS CHAPTER

- [Using Load Balancers in Contrail | 712](#)
- [Support for OpenStack LBaaS Version 2.0 APIs | 727](#)
- [Configuring Load Balancing as a Service in Contrail | 729](#)

## Using Load Balancers in Contrail

### IN THIS SECTION

- [Invoking LBaaS Drivers | 712](#)
- [Using a Service Appliance Set as the LBaaS Provider | 715](#)
- [Understanding the Load Balancer Agent | 716](#)
- [F5 Networks Load Balancer Integration in Contrail | 717](#)
- [Example: Creating a Load Balancer | 720](#)
- [Using the Avi Networks Load Balancer for Contrail | 721](#)

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

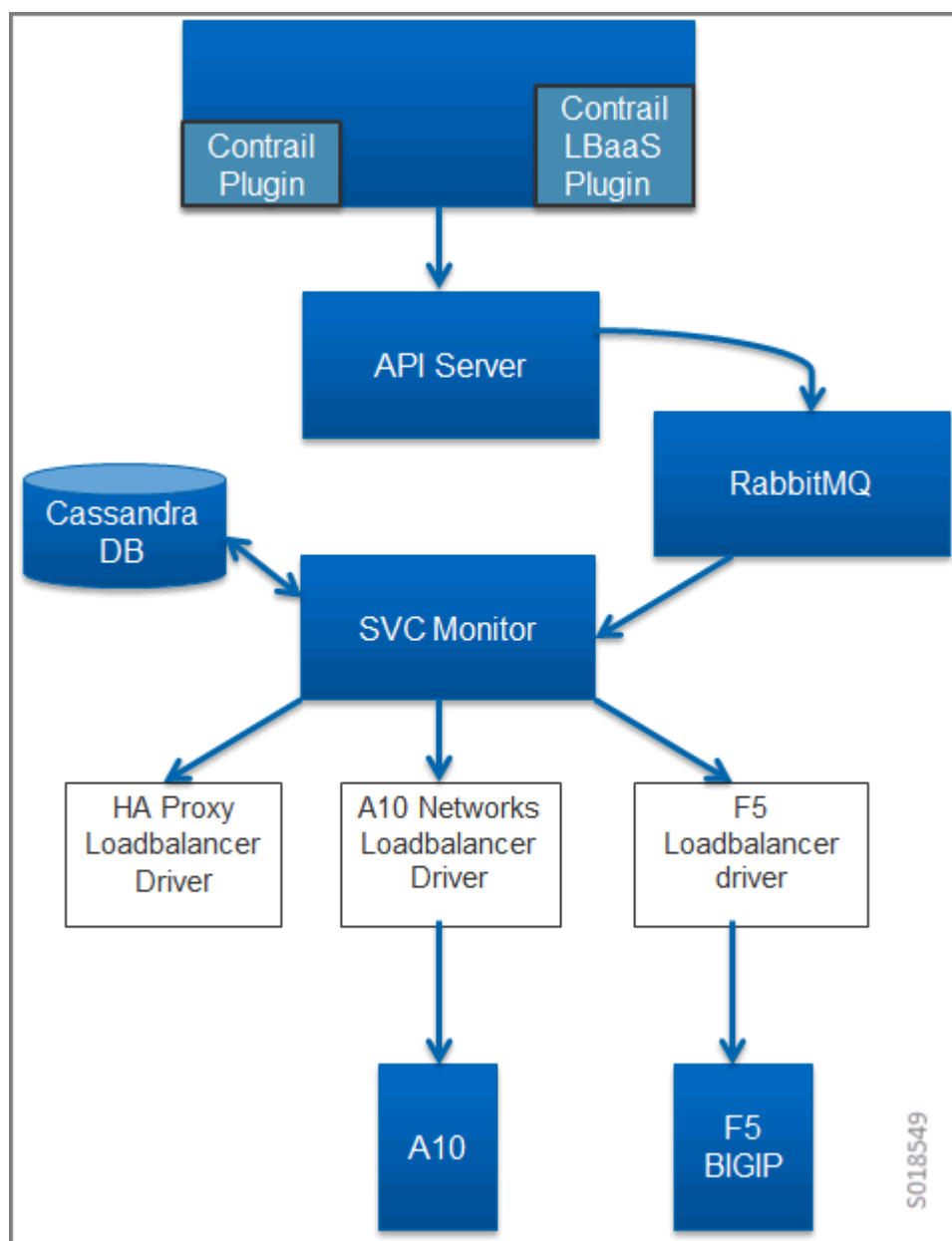
Starting with Contrail 3.0, the Neutron LBaaS plugin creates required configuration objects (such as pool, VIP, members, and monitor) in the Contrail API server, instead of within the Neutron plugin context, as in previous releases.

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 120 on page 714](#) provides an overview of the Contrail LBaaS components.

Figure 120: 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:

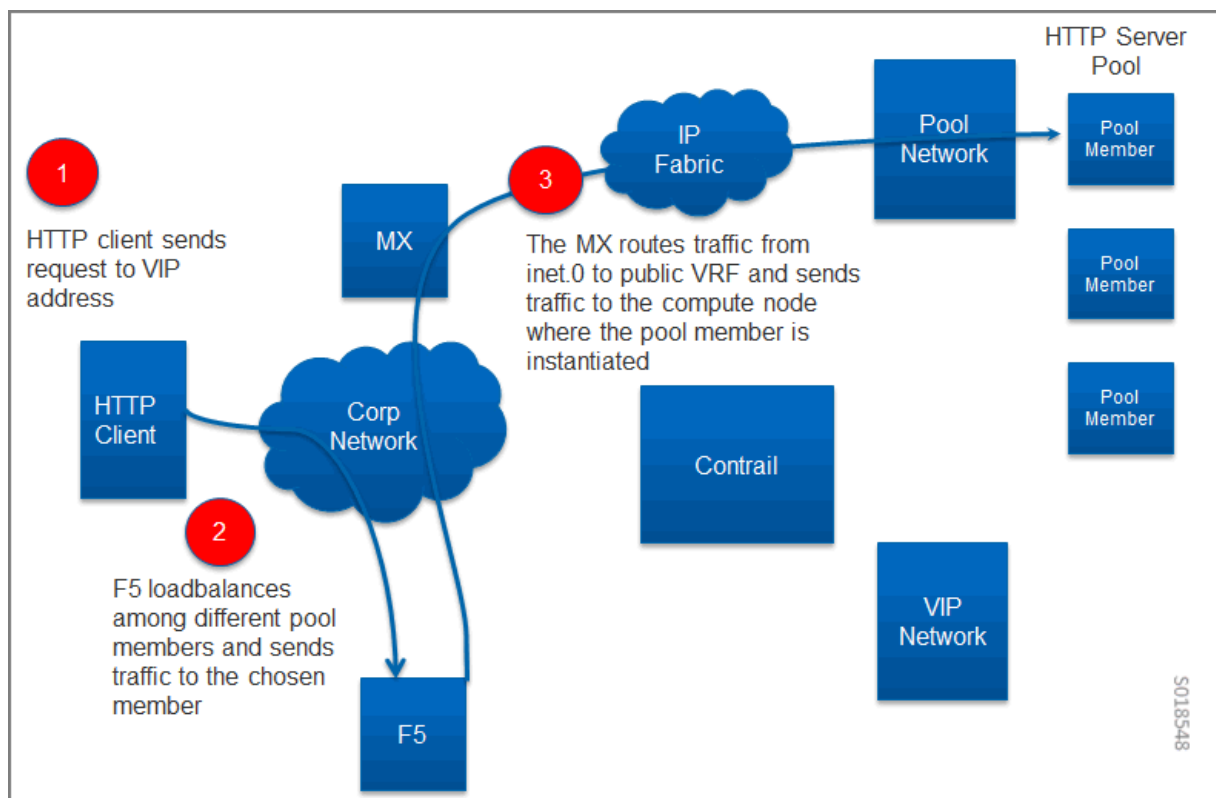
- publish route to attract VIP traffic
- pool network subnet route that points to the MX device

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 121 on page 719](#) shows the global routed traffic flow.

Figure 121: 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.AviLbaasDriver  
  
[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 ocavi_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

	<a href="#">Configuring Load Balancing as a Service in Contrail   729</a>	
	<a href="#">Support for OpenStack LBaaS Version 2.0 APIs   727</a>	

## Support for OpenStack LBaaS Version 2.0 APIs

IN THIS SECTION

- [Platform Support | 727](#)
- [Using OpenStack LBaaS Version 2.0 | 727](#)
- [Support for Multiple Certificates per Listener | 728](#)
- [Neutron Load-Balancer Creation | 728](#)

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 45 on page 727](#) shows which Contrail with OpenStack release combinations support which version of OpenStack LBaaS APIs.

**Table 45: 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 | 712](#)

[Configuring Load Balancing as a Service in Contrail | 729](#)

## Configuring Load Balancing as a Service in Contrail

### IN THIS SECTION

- [Overview: Load Balancing as a Service | 730](#)
- [Contrail LBaaS Implementation | 731](#)
- [Configuring LBaaS Using CLI | 732](#)
- [Configuring LBaaS using the Contrail Web UI | 734](#)

## 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.

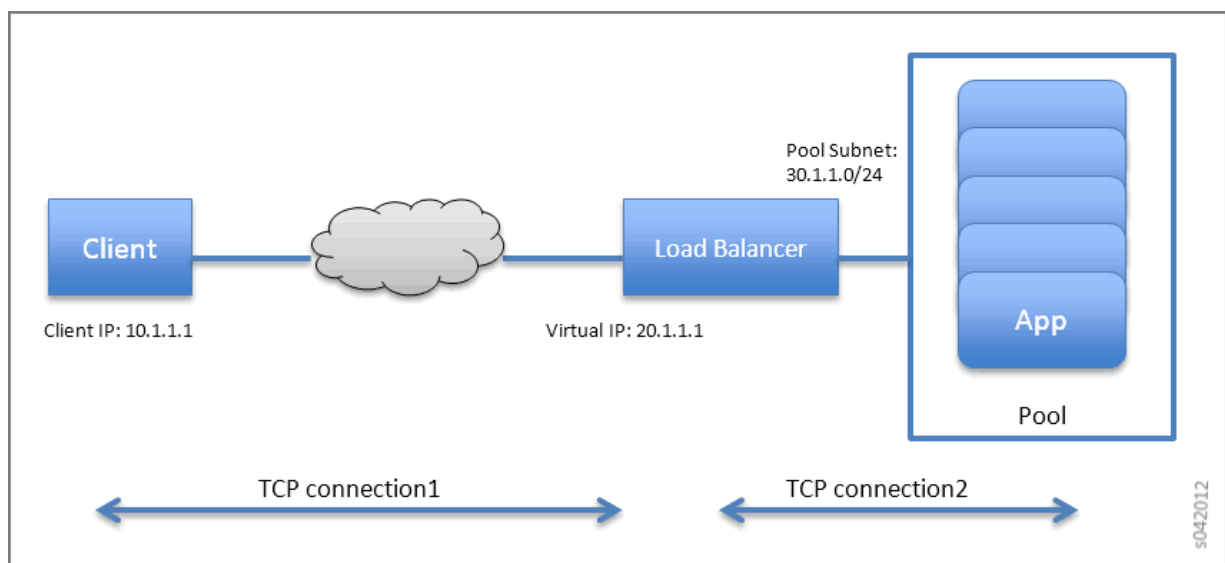
The LBaaS load balancer enables the creation of a pool of virtual machines serving applications, all front-ended by a virtual-ip. The LBaaS implementation has the following features:

- 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 vrouter.

### 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
```

### Configuring LBaaS using the Contrail Web UI

Create, edit, or delete load balancers using the Contrail Web UI. Use the following guidelines when creating load balancers:

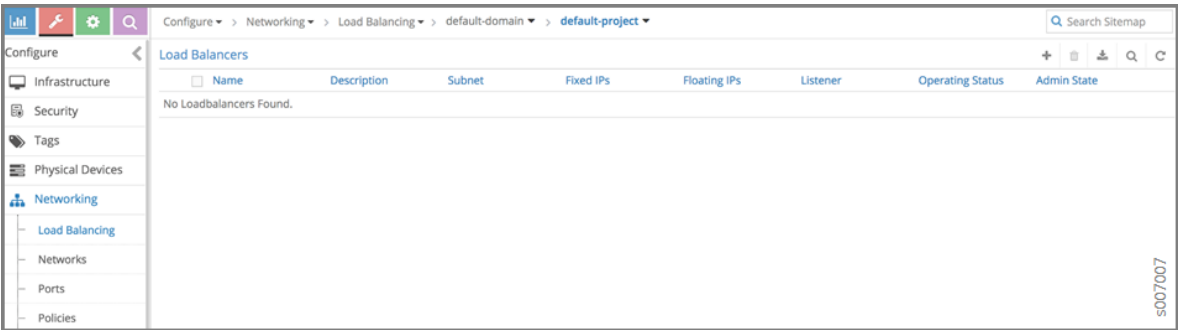
- Each load balancer consists of one or more listeners, pools, pool members, and health monitors.
- Listener: Port that listens for traffic from a particular load balancer. Multiple listeners can be associated with a single load balancer.
- Pool: Group of hosts that serves traffic from the load balancer.
- Pool Member: Server that is specified by the IP address and port for which it uses to serve the traffic it receives from the load balancer.
- Health Monitor: Health monitors are associated with pools and help divert traffic away from pool members that are temporarily offline.
- Each load balancer can have multiple pools with one or more listeners for each pool.
- The native load balancer has a single pool that is shared among multiple listeners.

### Creating a Load Balancer

Use the following steps to create a load balancer with the load balancer wizard.

1. Go to **Configure > Networking > Load Balancing**. A summary screen of the Load Balancers is displayed; see [Figure 122 on page 734](#).

Figure 122: Summary Screen of Load Balancers



2. To create a load balancer, click the



icon on the Load Balancers summary screen. The first window of the Create Load Balancer wizard is displayed.

Figure 123: Load Balancer Information

Add the load balancer information:

- **Name:** Name of the load balancer.
- **Description:** (Optional) Description of the load balancer.
- **Subnet:** Dropdown menu displays all subnets from list of all available networks. The subnet is the network on which to allocate the IP address of the load balancer.
- **Fixed IPs:** (Optional) IPv4 or IPv6 address.
- **Loadbalancer Provider:** Dropdown menu includes available options. Default is opencontrail.
- **Floating IP:** (Optional) IPv4 or IPv6 address.
- **Admin State:** Check the checkbox for UP or uncheck the checkbox for DOWN. Default is UP.

3. Click **Next**. The Listener fields are displayed.

Figure 124: Listener Information

The screenshot shows the 'Create Load Balancer' dialog box with the 'Listener' step selected. The progress bar indicates the following steps: Load Balancer (completed), Listener (current), Pool, Pool Member, and Monitor. The 'Listener' step contains the following fields:

- Name\***: Listner-http
- Description**: (empty)
- Protocol\***: HTTP
- Port\***: 80
- Connection Limit**: -1
- Admin State**: ☒ (checked)

At the bottom of the dialog, there are buttons for 'Cancel', 'Previous', and 'Next'. The 'Next' button is highlighted in blue.

Add the listener information:

- **Name:** Name of the listener.
- **Description:** (Optional) Description of the listener.
- **Protocol:** Dropdown menu includes HTTP, TCP, and TERMINATED\_HTTPS protocols. TERMINATED\_HTTPS is available only if the key-manager service is enabled and you have access to the lists of SSL certificates.
- **Port:** Must be an integer in the range of 1 to 65535.
- **Connection Limit:** (Optional) Default value is -1, indicating an infinite limit.
- **Admin State:** Check the checkbox for UP or uncheck the checkbox for DOWN. Default is UP.

4. Click **Next**. The Pool fields are displayed.



Figure 125: Pool Information

Create Load Balancer

Load Balancer Listener **Pool** Pool Member Monitor

Name\* Pool1

Description

Method\* LEAST\_CONNECTIONS

Protocol\* HTTP

Session Persistence Select Session Persistence

Admin State ☒

Global Custom Attribute

Default Custom Attribute

Cancel Previous Next

s007010

Add the pool information:


- **Name:** Name of the pool.
  - **Description:** (Optional) Description of the pool.
  - **Method:** Load balancing method used to distribute incoming requests. Dropdown menu includes LEAST\_CONNECTIONS, ROUND\_ROBIN, and SOURCE\_IP.
  - **Protocol:** The protocol used by the pool and its members for the load balancer traffic. Dropdown menu includes TCP, HTTP, and HTTPS.
  - **Session Persistence:** (Optional) Default value is an empty dictionary.
  - **Admin State:** Check the checkbox for UP or uncheck the checkbox for DOWN. Default is UP.
5. Click **Next**. The list of available pool member instances are displayed. To add an external member, click the  icon. Each pool member must have a unique IP address and port combination.

Figure 126: Pool Member Information

Create Load Balancer

Progress: Load Balancer (✓), Listener (✓), Pool (✓), **Pool Member (4)**, Monitor (5)

Name	Subnet	IP Address*	Port	Weight	Admin State	+

Buttons: Cancel, Previous, Next

s007011

The pool member information includes:

- **Name:** Name of the pool member.
- **Subnet:** The subnet in which to access the member.
- **IP Address:** The IP address of the member that is used to receive traffic from the load balancer.
- **Port:** The port to which the member listens to receive traffic from the load balancer.
- **Weight:** The default value is 1.
- **Admin State:** Check the checkbox for UP or uncheck the checkbox for DOWN. Default is UP.

6. Click **Next**. The Monitor fields are displayed.

Figure 127: Health Monitor Information

working > Load Balancing > default-domain > admin

Create Load Balancer

Load Balancer Listener Pool Pool Member Monitor

Monitor Type\* HTTP

Health check interval (sec)\* 5

Retry count before markdown\* 3

Timeout (sec)\* 5

HTTP Method GET

Expected HTTP Status Code 200

URL Path /

Cancel Previous Create Load Balancer

s007012

Add the health monitor information:

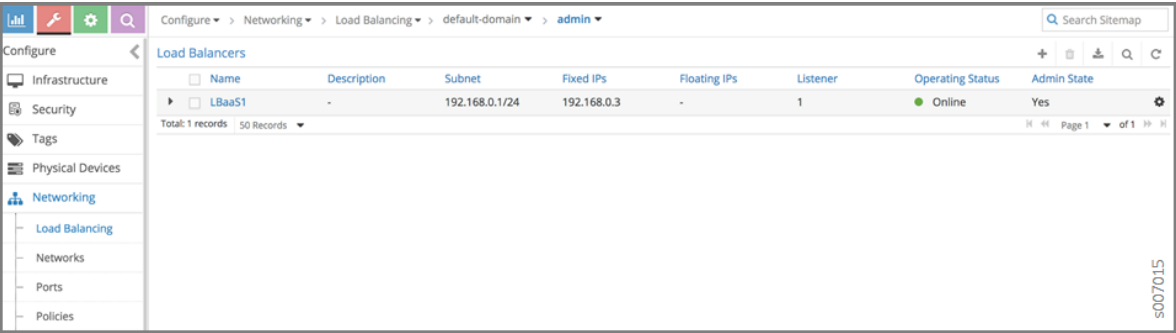
- **Monitor Type:** Dropdown menu includes HTTP, HTTPS, PING, and TCP.
- **Health check interval (sec):** The time interval, in seconds, between each health check.
- **Retry count before markdown:** The maximum number of failed health checks before the state of a member is changed to OFFLINE.
- **Timeout (sec):** The maximum number of seconds allowed for any given health check to complete. The timeout value should always be less than the health check interval.
- **HTTP Method:** Required if monitor type is HTTP. Dropdown menu includes GET and HEAD. The default value is GET.
- **Expected HTTP Status Code:** Required if monitor type is HTTP. The default value is 200.
- **URL Path:** Required if monitor type is HTTP. The default value is "/."
- **Admin State:** Check the checkbox for UP or uncheck the checkbox for DOWN. Default is UP.

## Viewing or Editing Load Balancers

Use the following steps to view or edit existing load balancers.

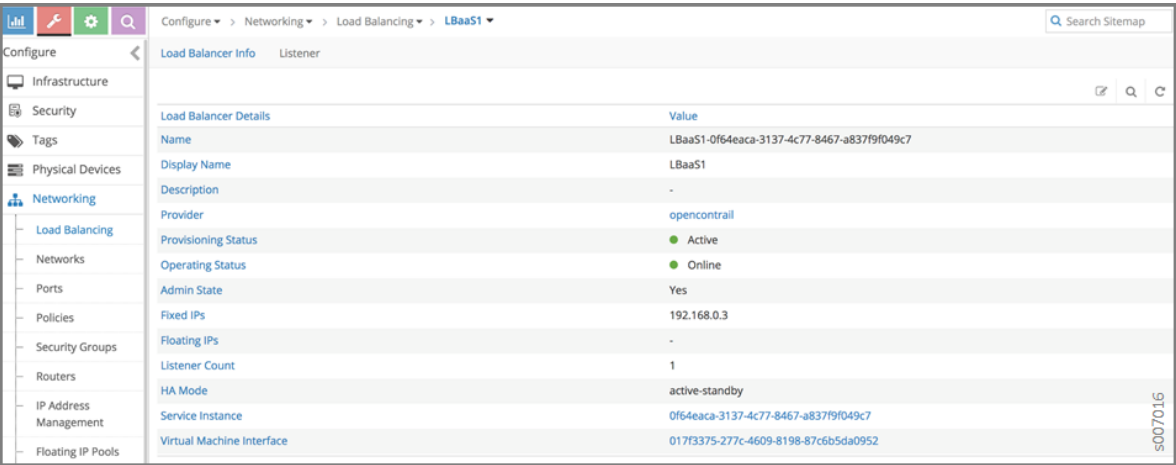
- 1. Go to **Configure > Networking > Load Balancing**. A summary screen of the Load Balancers is displayed.

Figure 128: Summary Screen of Load Balancers



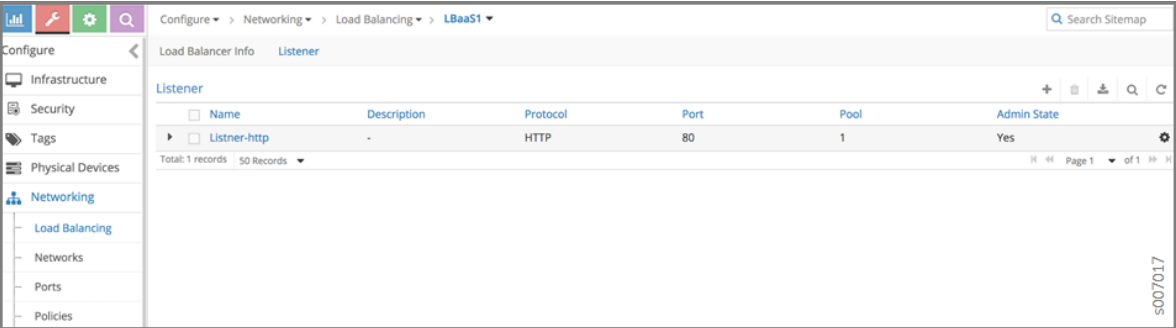
- 2. To view or edit a load balancer, click the name of a load balancer listed in the summary screen. The Load Balancer Info window is displayed.

Figure 129: Load Balancer Info Window



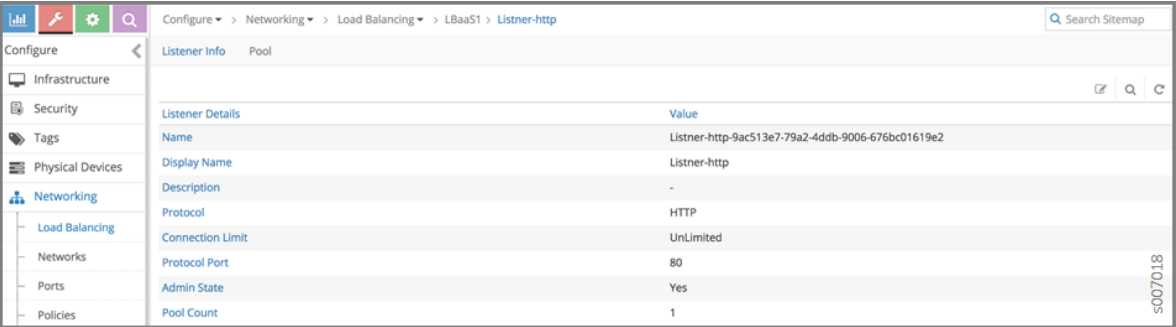
- 3. To view the list of listeners associated with the load balancer, click on the Listener tab. A summary screen of the listeners is displayed.

Figure 130: Summary Screen of Listeners



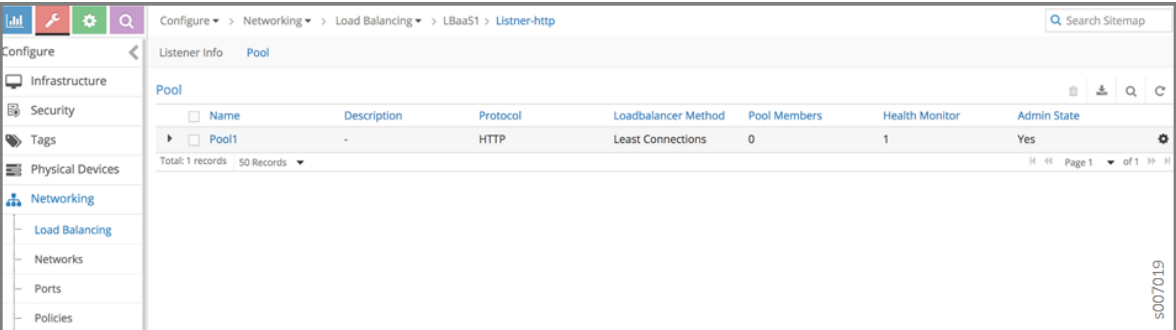
4. To view or edit a listener, click the name of a listener listed in the summary screen. The Listener Info window is displayed.

Figure 131: Listener Info Window



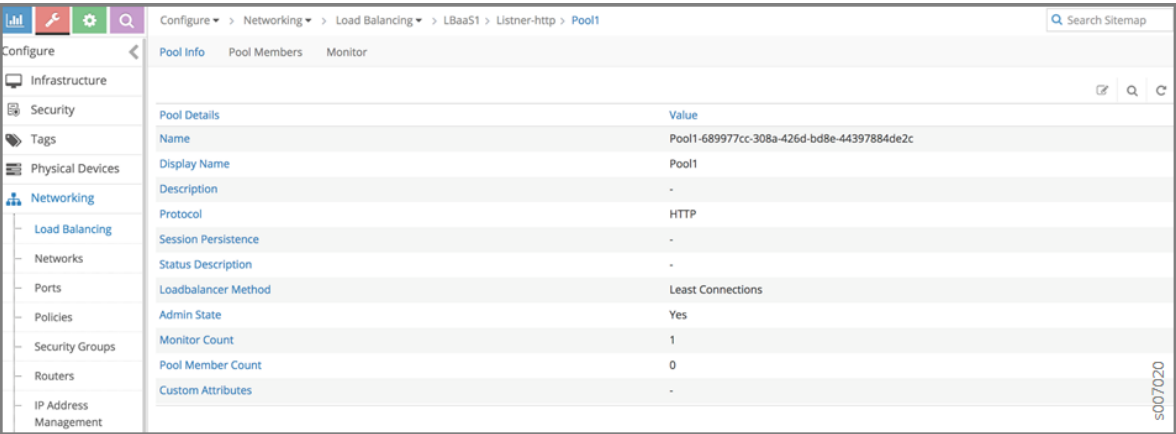
5. To view a list of pools associated with the listener, click on the Pool tab. A summary screen of the pools is displayed.

Figure 132: Summary Screen of Pools



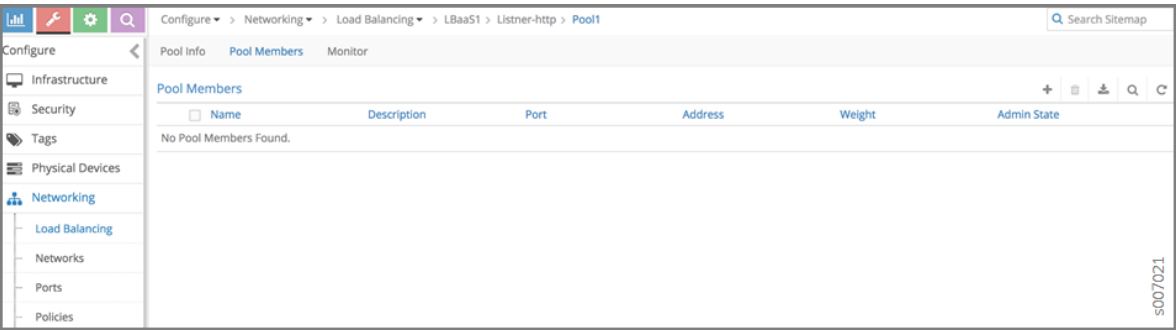
- 6. To view or edit a pool, click the name of a pool listed in the summary screen. The Pool Info window is displayed.

Figure 133: Pool Info Window



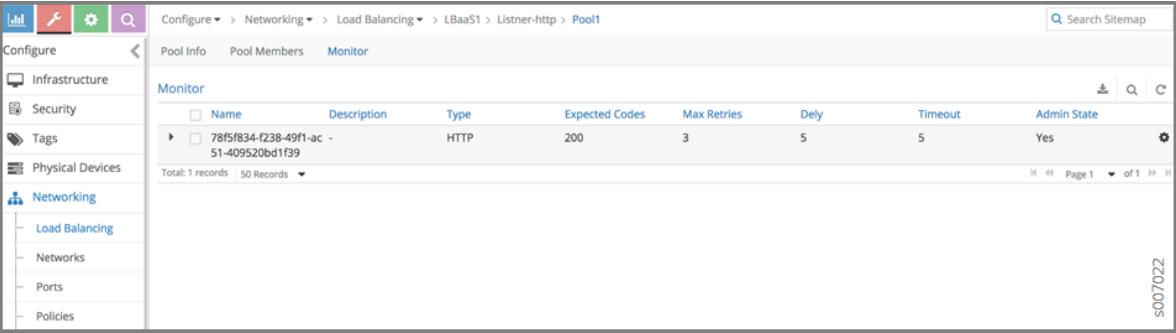
- 7. To view a list of members associated with the pool, click on the Pool Members tab. A summary screen of the pool members is displayed.

Figure 134: Pool Members Summary Screen



- 8. To view the health monitor details associated with the pool, click on the Monitor tab. The health monitor details are displayed.

Figure 135: Pool Members Summary Screen



Deleting a Load Balancer

Use the following steps to delete a load balancer.

- 1. Delete the members of the pools. Select the pool members you want to delete, then click the trashcan icon; see [Figure 134 on page 742](#).
- 2. Delete the pools. Select the pools you want to delete, then click the trashcan icon; see [Figure 132 on page 741](#).
- 3. Delete the listeners. Select the listeners you want to delete, then click the trashcan icon; see [Figure 130 on page 741](#).
- 4. Delete the load balancer. Select the load balancer you want to delete, then click the trashcan icon; see [Figure 128 on page 740](#).

RELATED DOCUMENTATION

- Using Load Balancers in Contrail | 712
- Support for OpenStack LBaaS Version 2.0 APIs | 727

# Optimizing Contrail

## IN THIS CHAPTER

- [Route Target Filtering | 744](#)
- [Source Network Address Translation \(SNAT\) | 747](#)
- [Multiqueue Virtio Interfaces in Virtual Machines | 752](#)
- [vRouter Command Line Utilities | 753](#)

## Route Target Filtering

### IN THIS SECTION

- [Introduction | 744](#)
- [Debugging and Troubleshooting Route Target Filtering | 745](#)
- [RTF Limitations in Contrail 1.10 | 746](#)

### 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 data center 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.

BgpNeighborListResp						
neighbors						
peer	peer_address	deleted	peer_asn	local_address	local_asn	encoding
nodec13	10.204.216.70	false	0	10.204.216.70	0	XMPP
more						

- 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

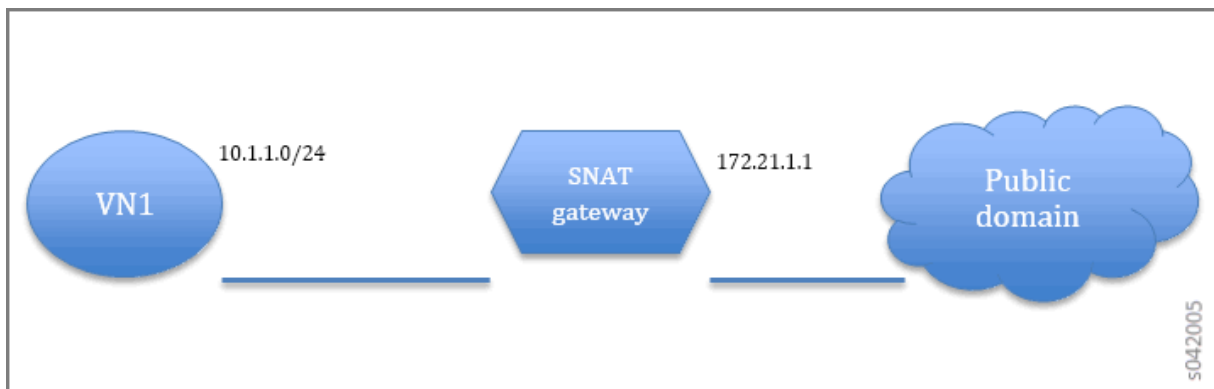
- [Overview | 747](#)
- [Neutron APIs for Routers | 748](#)
- [Network Namespace | 748](#)
- [Using the Web UI to Configure Routers with SNAT | 749](#)
- [Using the Web UI to Configure Distributed SNAT | 750](#)

### 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 136: 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
```

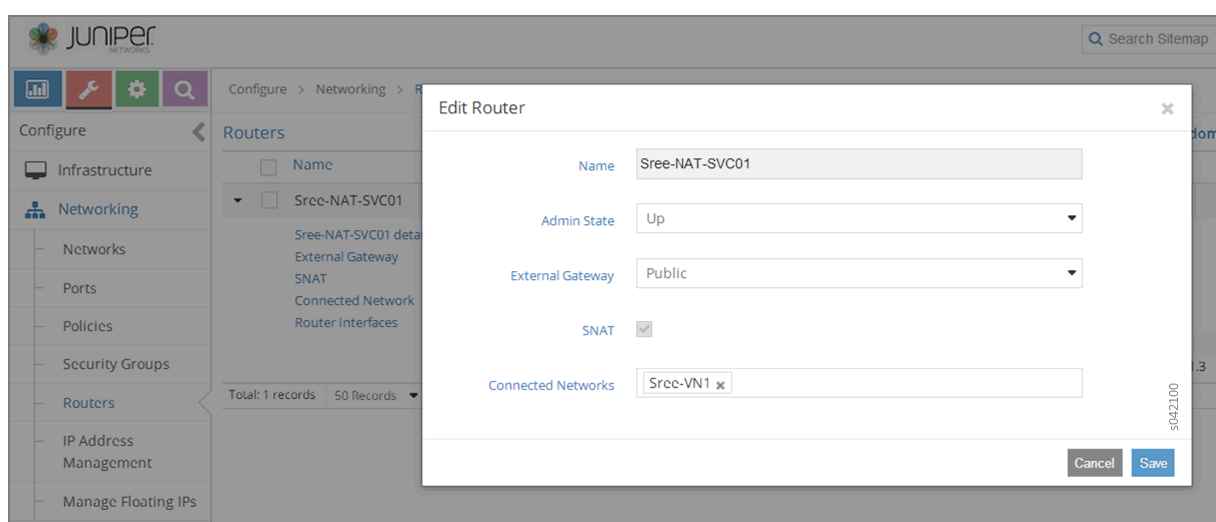
## Using the Web UI to Configure Routers with SNAT

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 137: 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 138: Router Status for SNAT**

Name	External Gateway	Connected Network	Admin State
Sree-NAT-SVC01	Public	Sree-VN1	Up

UUID	Network	IP
62a171f5-d323-4c70-942f-9cc02e973d53	Sree-VN1	100.1.1.3

You can view the real time status of a router with SNAT by viewing the instance console, as in the following.

**Figure 139: 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: OEMU (instance-00000057)
Inet addr: 100.1.1.2 Bcast: 100.1.1.255 Mask: 255.255.255.0
Inet6 addr: fe80::9c:9aff:febf:2e20/64 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:~$
  
```

## Using the Web UI to Configure Distributed SNAT

The distributed SNAT feature allows virtual machines to communicate with the IP fabric network using the existing forwarding infrastructure for compute node connectivity. This functionality is achieved through port address translation of virtual machine traffic using the IP address of the compute node as the public address.

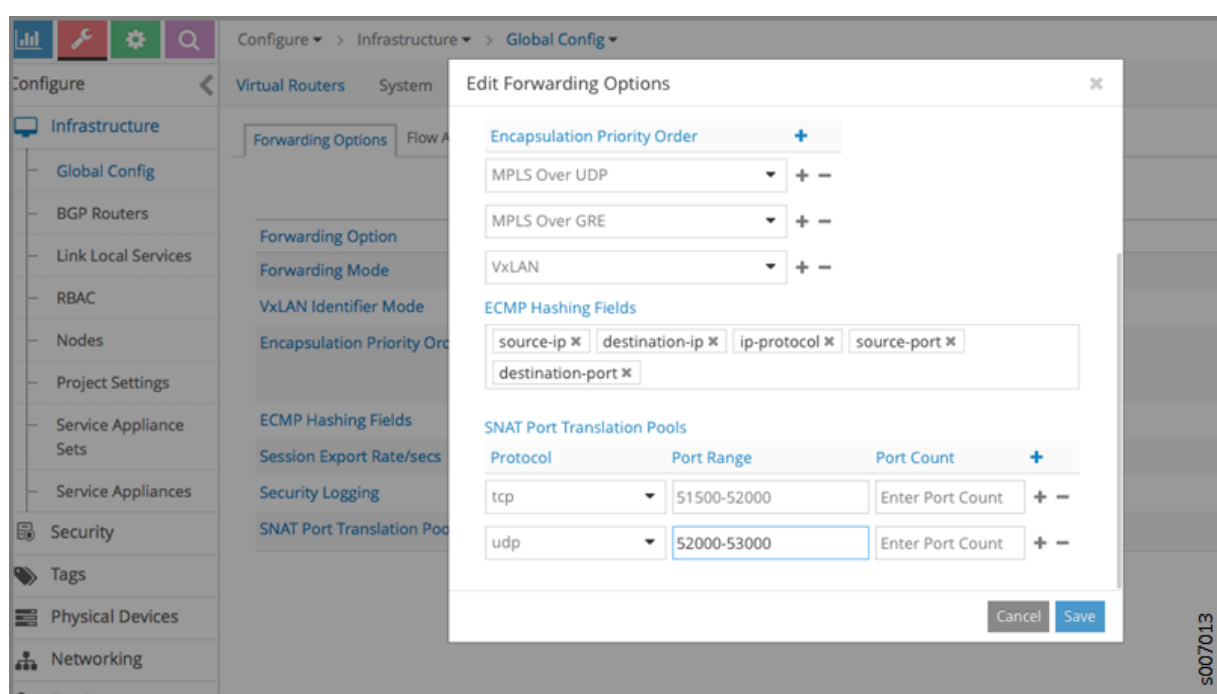
The following distributed SNAT use case is supported:

- Virtual networks with distributed SNAT enabled can communicate with the IP fabric network. The session must be initiated from a virtual machine. Sessions initiated from the external network are not supported.

Distributed SNAT is supported only for TCP and UDP, and you can configure discrete port ranges for both protocols.

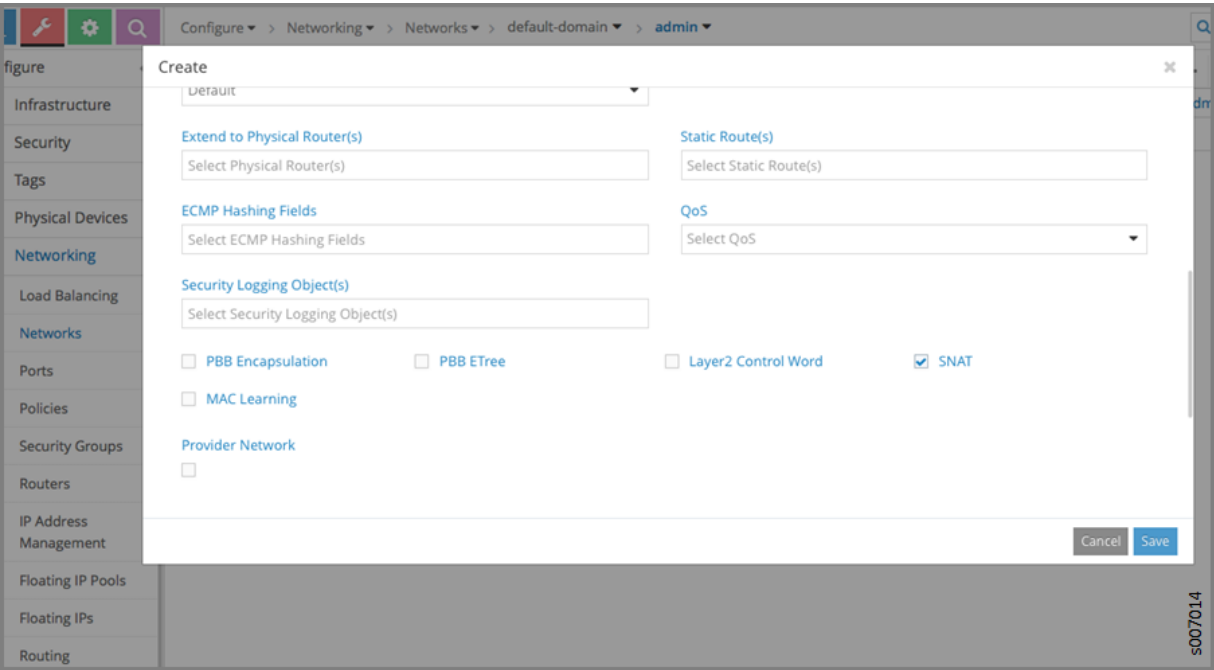
A pool of ports is used for distributed SNAT. To create a pool of ports, go to **Configure > Infrastructure > Global Config**. The following shows an example of a port range used for port address translation.

**Figure 140: Edit Forwarding Options Window**



To use distributed SNAT, you must enable SNAT on the virtual network. To enable SNAT on the virtual network, go to **Configure > Networking > Networks**. The following shows a virtual network for which SNAT has been enabled under Advanced Options.

Figure 141: Create Window



## Multiqueue Virtio Interfaces in Virtual Machines

### IN THIS SECTION

- [Multiqueue Virtio Overview | 752](#)
- [Requirements and Setup for Multiqueue Virtio Interfaces | 753](#)

Contrail 3.2 adds support for multiqueue for the DPDK-based router.

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 | 754](#)
- [vif Command | 755](#)

- [flow Command | 759](#)
- [vrfstats Command | 760](#)
- [rt Command | 761](#)
- [dropstats Command | 763](#)
- [mpls Command | 767](#)
- [mirror Command | 769](#)
- [vxlan Command | 771](#)
- [nh Command | 772](#)

### 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.

*(Continued)*

Command	Description
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 46: vif Fields**

vif Output Field	Description
vif0/X	The vrouter assigned name, where 0 is the router id and X is the index allocated to the interface within the vrouter.
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.
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>

Table 46: vif Fields (Continued)

vif Output Field	Description
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>
Rx	<p>RX packets:60 bytes:4873 errors:0</p> <p>Packets received by vrouter from this interface.</p>
Tx	<p>TX packets:21 bytes:2158 errors:0</p> <p>Packets transmitted out by vrouter on this interface.</p>

### 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.
--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.

*(Continued)*

Option	Description
--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]
           [-l]
  -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 vrouter.

**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
----	---	---

### 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 router 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.
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

## Contrail Security

---

- [Contrail Security | 777](#)
-



# Contrail Security

## IN THIS CHAPTER

- [Security Policy Enhancements | 777](#)

## Security Policy Enhancements

### IN THIS SECTION

- [Overview of Existing Network Policy and Security Groups in OpenStack and Contrail | 777](#)
- [Security Policy Enhancements | 778](#)
- [Using Tags and Configuration Objects to Enhance Security Policy | 779](#)
- [Configuration Objects | 780](#)
- [Using the Contrail Web User Interface to Manage Security Policies | 785](#)

## Overview of Existing Network Policy and Security Groups in OpenStack and Contrail

Contrail virtual networks are isolated by default. Workloads in a virtual network cannot communicate with workloads in other virtual networks, by default. A neutron router or a Contrail network policy may be used to connect two virtual networks. In addition, Contrail network policy also provides security between two virtual networks by allowing or denying specified traffic.

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.

In modern cloud environments, workloads are moving from one server to another, one rack to another and so on. Therefore, users must rely less on using IP addresses or other network coordinates to identify the endpoints to be protected. Instead users must leverage application attributes to author policies, so that the policies don't need to be updated on account of workload mobility.

A user might want to segregate traffic on different categories, such as the following examples:

- Application—The application being protected
- Tier—The tier (or component), within the application, being protected
- Deployment—The environment in which the instance of the application is deployed in
- Site—The geographical location in which the application is deployed in
- Many more possibilities for needing to segregate traffic.

Additionally, a user might need to group workloads based on combinations of tags. These intents are hard to express with existing network policy constructs or Security Group constructs. Besides, existing policy constructs leveraging the network coordinates, must continually be rewritten or updated each time workloads move and their associated network coordinates change.

## 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.

Starting with Contrail Release 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 Release 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

Starting with Contrail Release 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.

### 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
```

### 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.

- [ 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

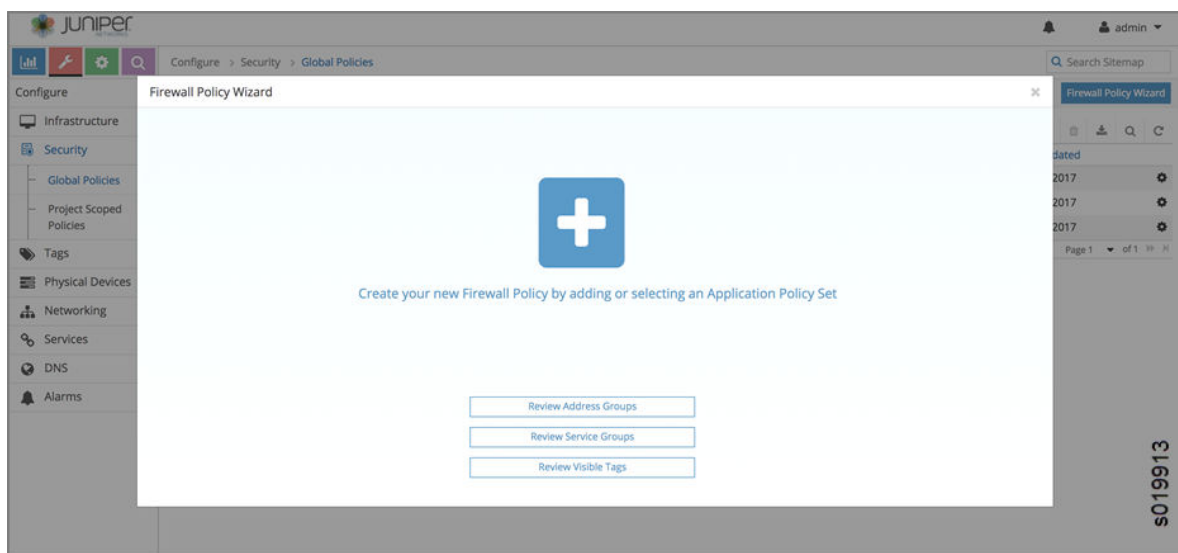
Whenever the API server receives a request to create/update a firewall policy rule object, it analyzes the object data to make sure that all virtual-networks, address-group, tag objects exist. If any of them do not exist, the request will be rejected. In addition, it will actually create a reference 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

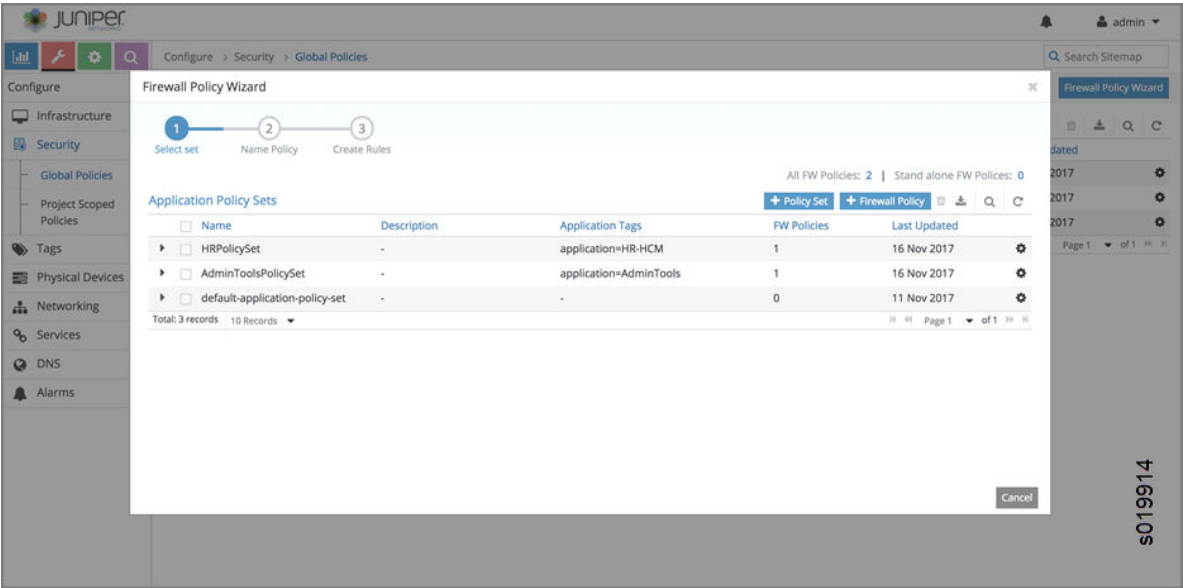
1. To add a security policy, go to **Configure > Security > Global Policies**. Near the upper right, click the button **Firewall Policy Wizard**. The **Firewall Policy Wizard** appears, where you can create your new firewall policy by adding or selecting an application policy set. See [Figure 142 on page 786](#).

Figure 142: Firewall Policy Wizard



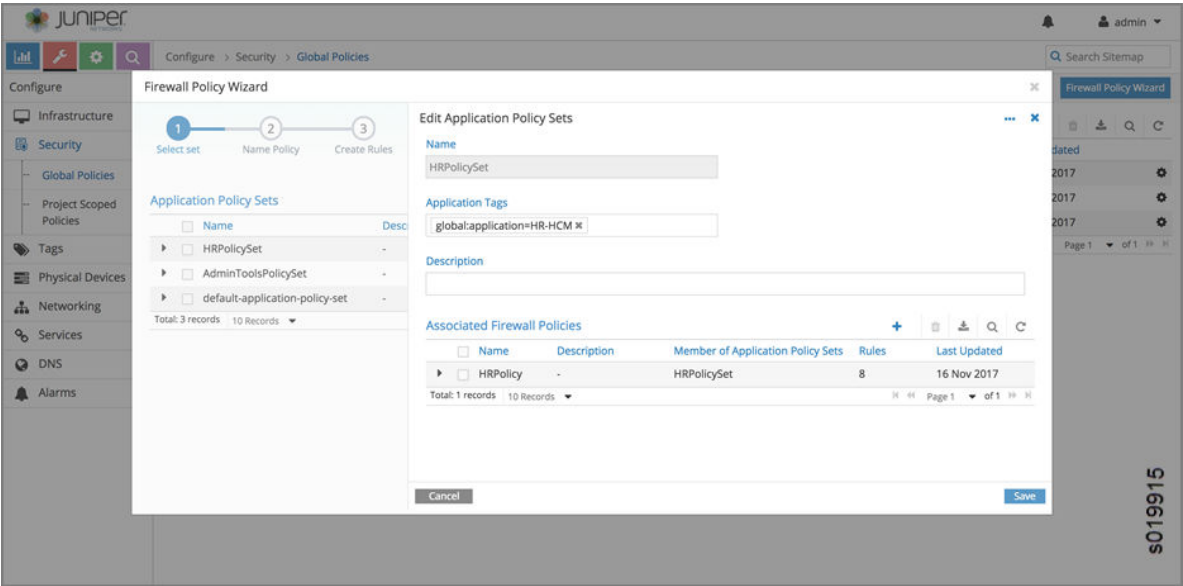
2. Click the large + on the Firewall Policy Wizard screen to view the **Application Policy Sets** window. The existing application policy sets are displayed. See [Figure 143 on page 787](#).

Figure 143: Application Policy Sets



- 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 144 on page 787](#), where the **HRPolicySet** has been selected.

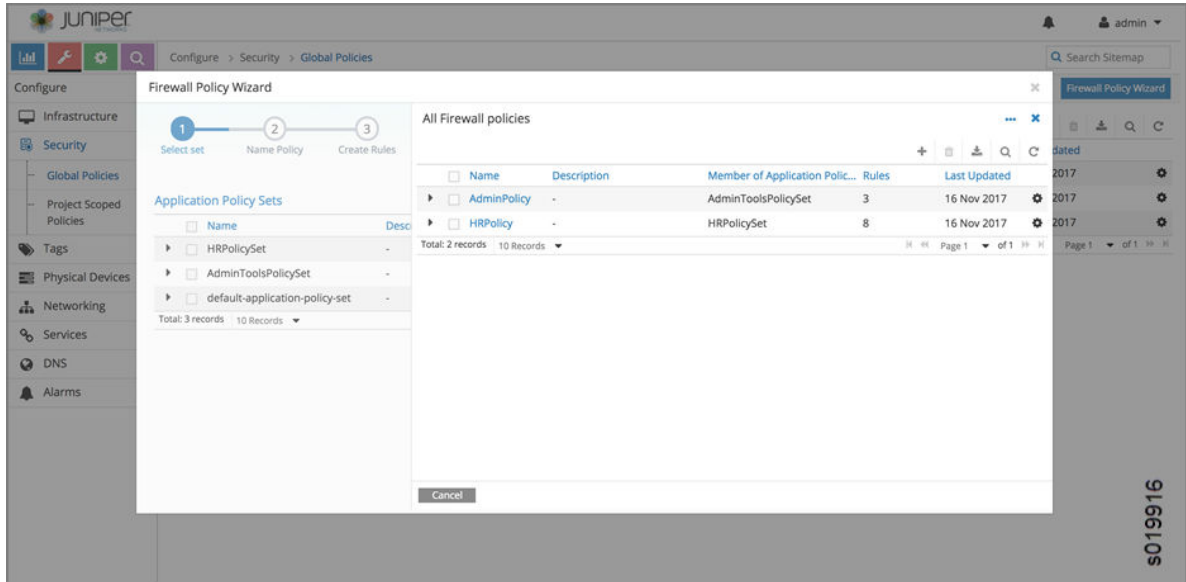
Figure 144: Edit Application Policy Sets



- To view all firewall policies, click the Application Policy Sets link in the left side.

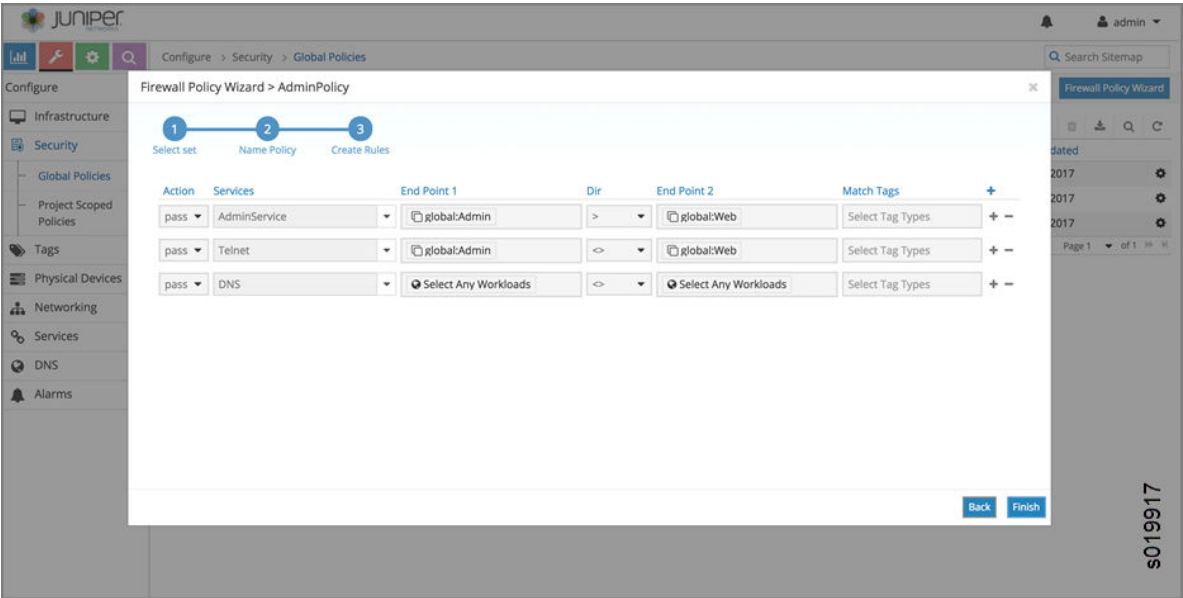
See [Figure 145 on page 788](#).

**Figure 145: All Firewall Policies**



5. Select any listed firewall policy to view or edit the rules associated with that policy. See [Figure 146 on page 789](#), 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 146: 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 147 on page 790](#).

Figure 147: Tags

Name	Associated Virtual Networks	Associated Ports	Associated Projects
application=AdminTools	-	ip-fabric (-)	-
application=HR-HCM	-	-	HRProd HRDev
deployment=Development	-	-	HRDev
deployment=Production	-	-	HRProd
site=France	-	-	HRProd
site=USA	-	-	HRDev
tier=Admin	-	ip-fabric (-)	-
tier=App	appvn (HRProd) appvn (HRDev)	-	-
tier=DB	dbvn (HRDev) dbvn (HRProd)	-	-
tier=Web	webvn (HRProd) webvn (HRDev)	-	-

2. You can click through any listed tag to see the rules to which the tag is applied. See [Figure 148 on page 790](#), 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 148: View Application Tags

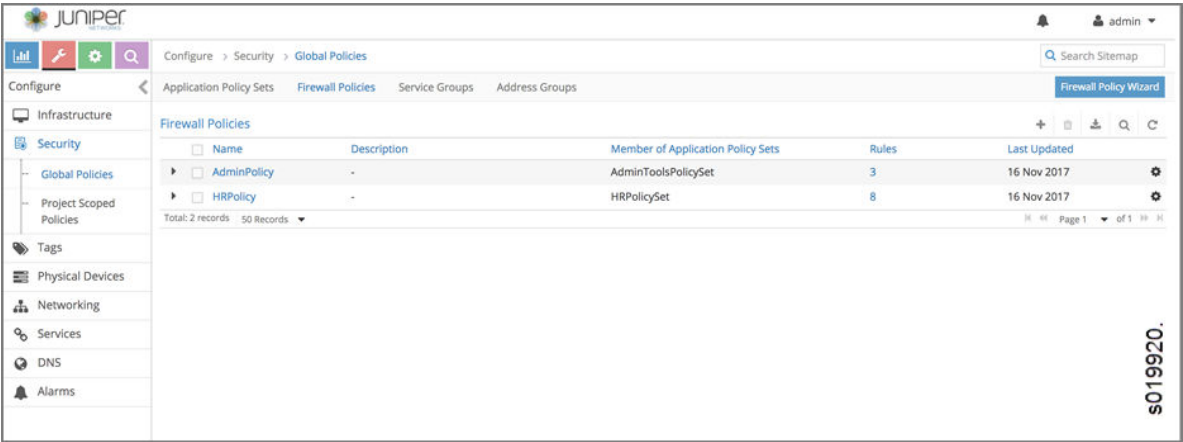
Name	Description	Application Tags	FW Policies	Last Updated
HRPolicySet	-	application=HR-HCM	1	16 Nov 2017
AdminToolsPolicySet	-	application=AdminTools	1	16 Nov 2017
default-application-policy-set	-	-	0	11 Nov 2017

### Viewing Global Policies

From **Configure > Security > Global Policies**, in addition to viewing the policies includes in application policy sets, you can also view all firewall policies, all service groups policies, and all address groups policies.

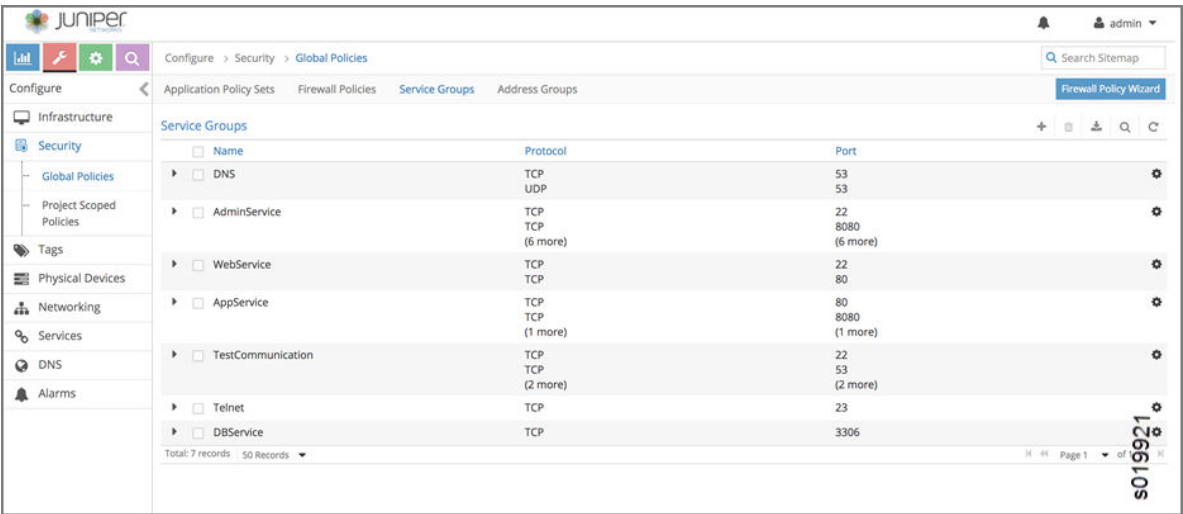
- 1. To view and manage the global firewall policies, from **Configure > Security > Global Policies**, click the Firewall Policies tab to view the details for system firewall policies, see [Figure 149 on page 791](#)

Figure 149: Firewall Policies



- 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 150 on page 791](#).

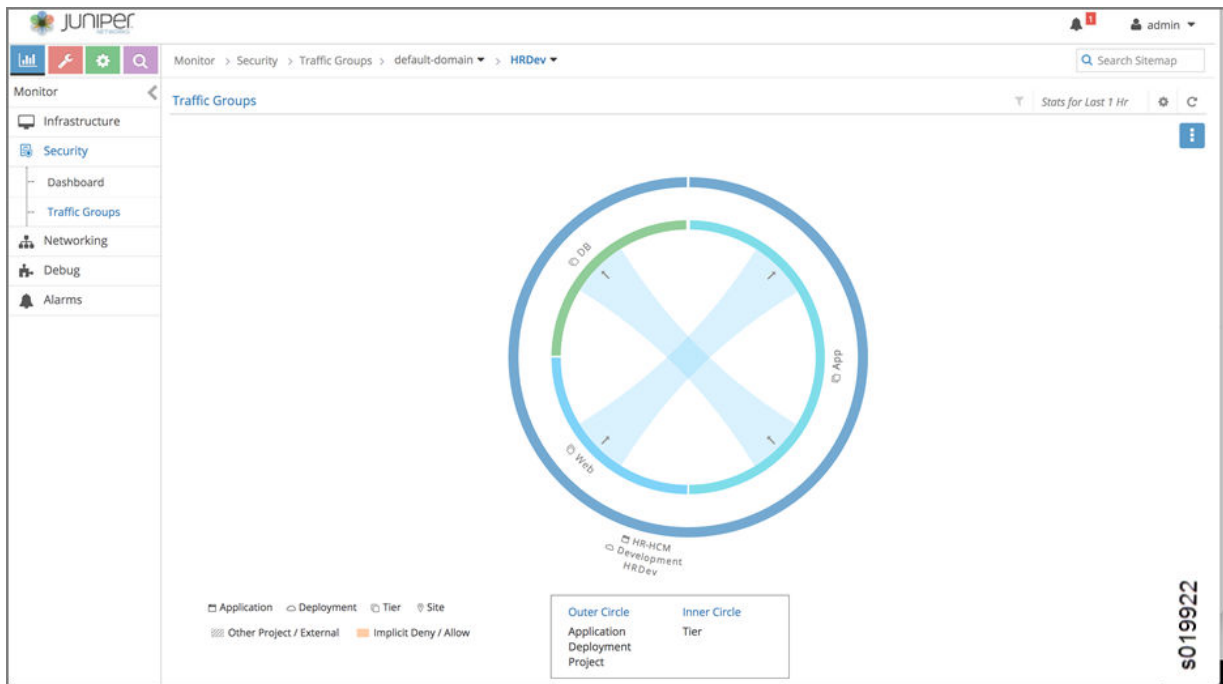
Figure 150: Service Groups



## Visualizing Traffic Groups

Use **Monitor > Security > Traffic Groups** to explore visual representations of how policies are applied to traffic groups. See [Figure 151 on page 792](#), 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.

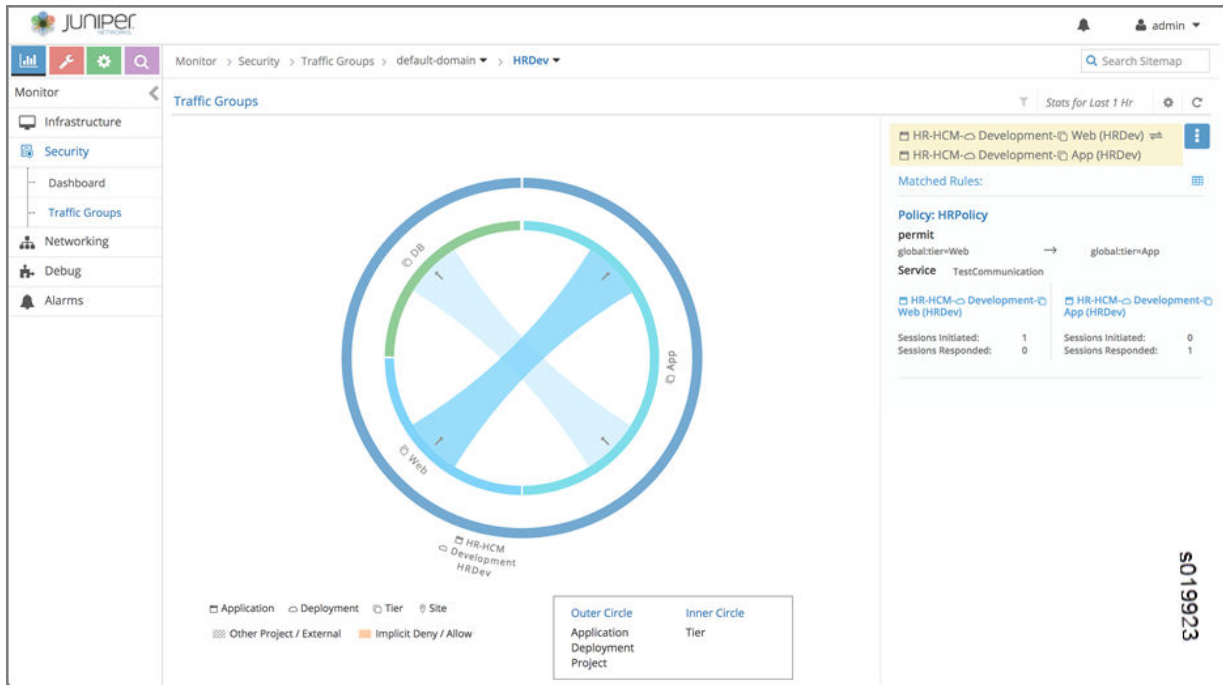
**Figure 151: Traffic Groups**



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 152 on page 793](#).

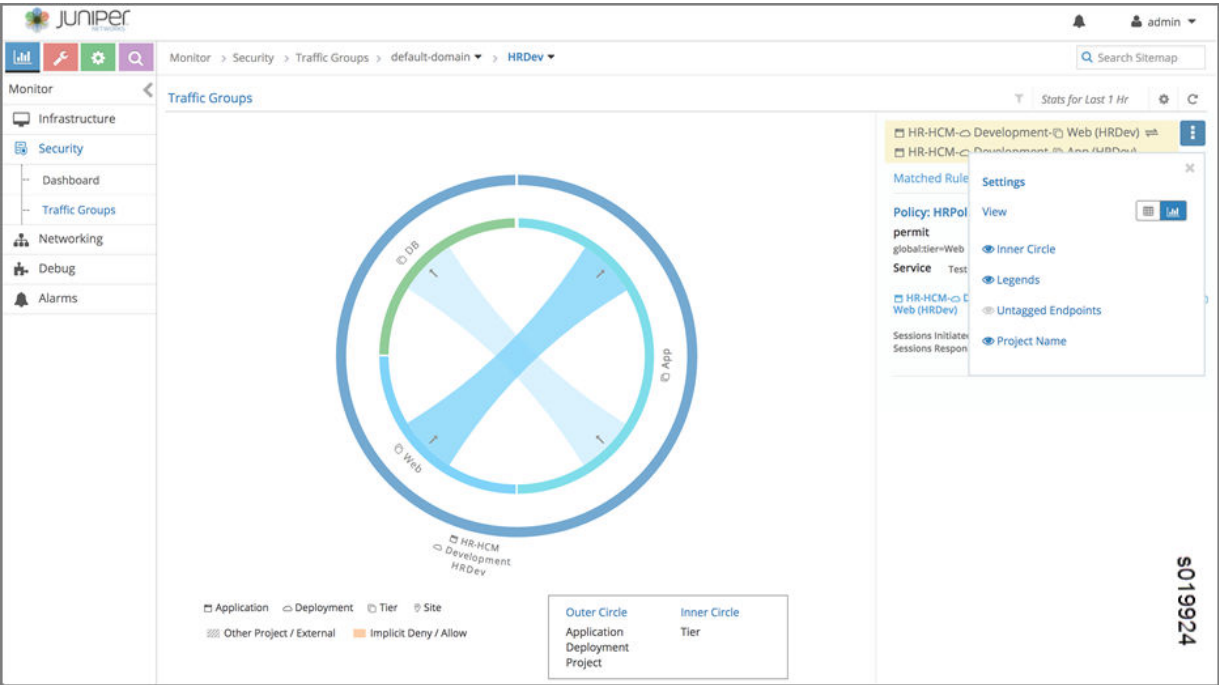


Figure 152: Traffic Groups, Policy Details



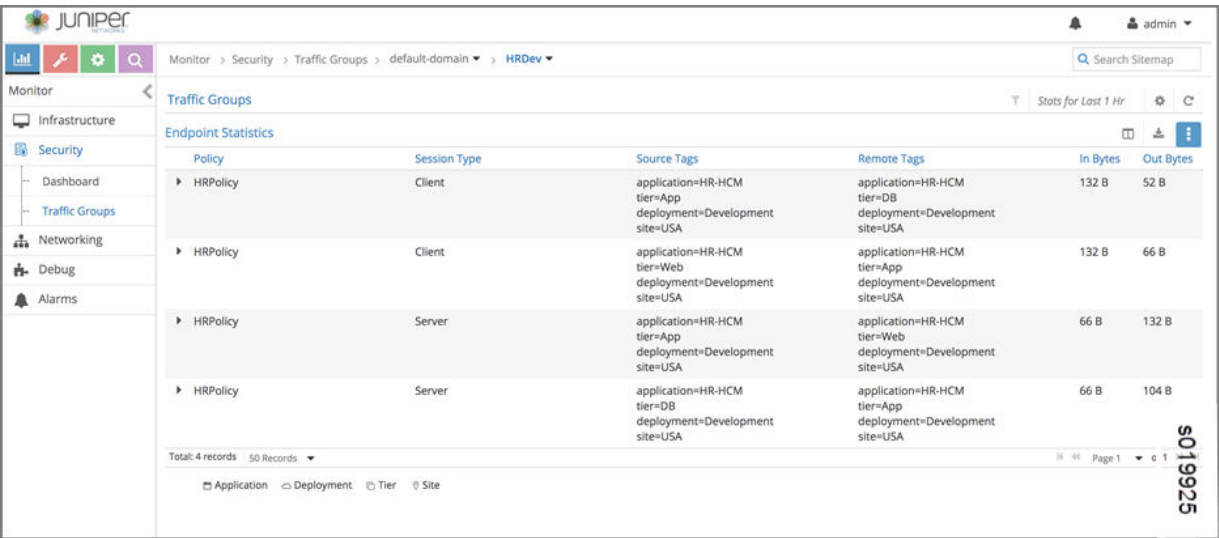
You can click in the right side of the screen to get to the **Settings** window, where you can change the type of view and change which items appear in the visual representation. See [Figure 153 on page 794](#).

Figure 153: 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 154 on page 794](#).

Figure 154: Traffic Groups, Endpoint Statistics



You can click deeper through any linked statistic to view more details about that statistic, see [Figure 156 on page 795](#) and [Figure 156 on page 795](#).

Figure 155: Traffic Groups, Details

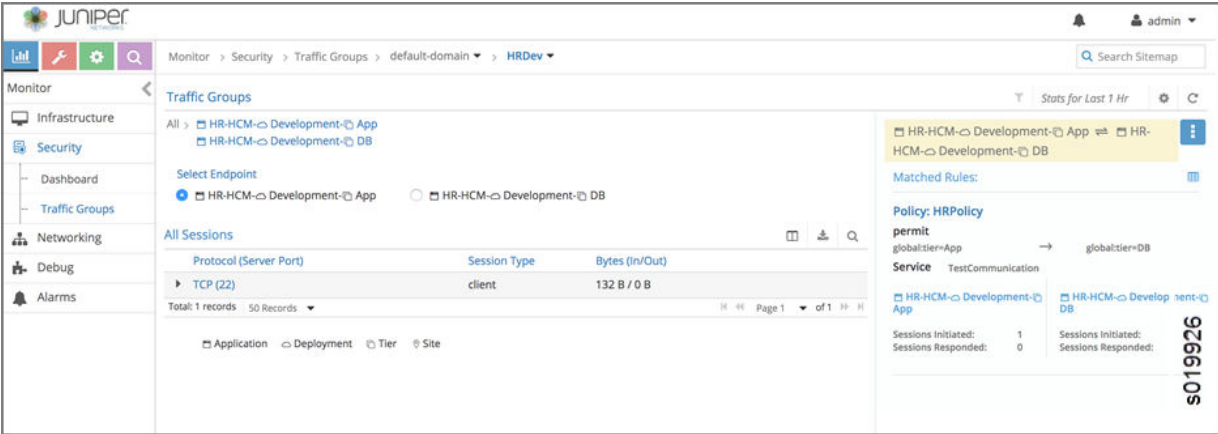
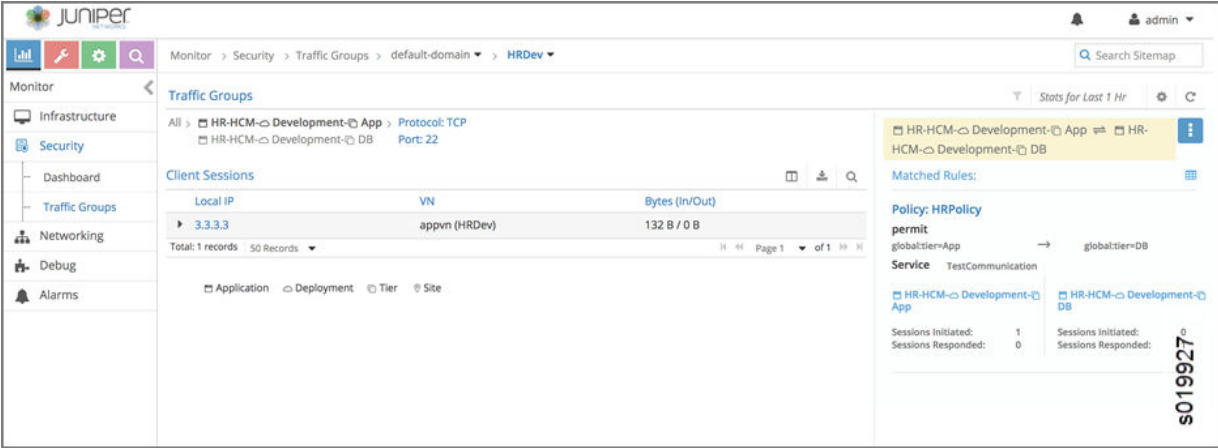
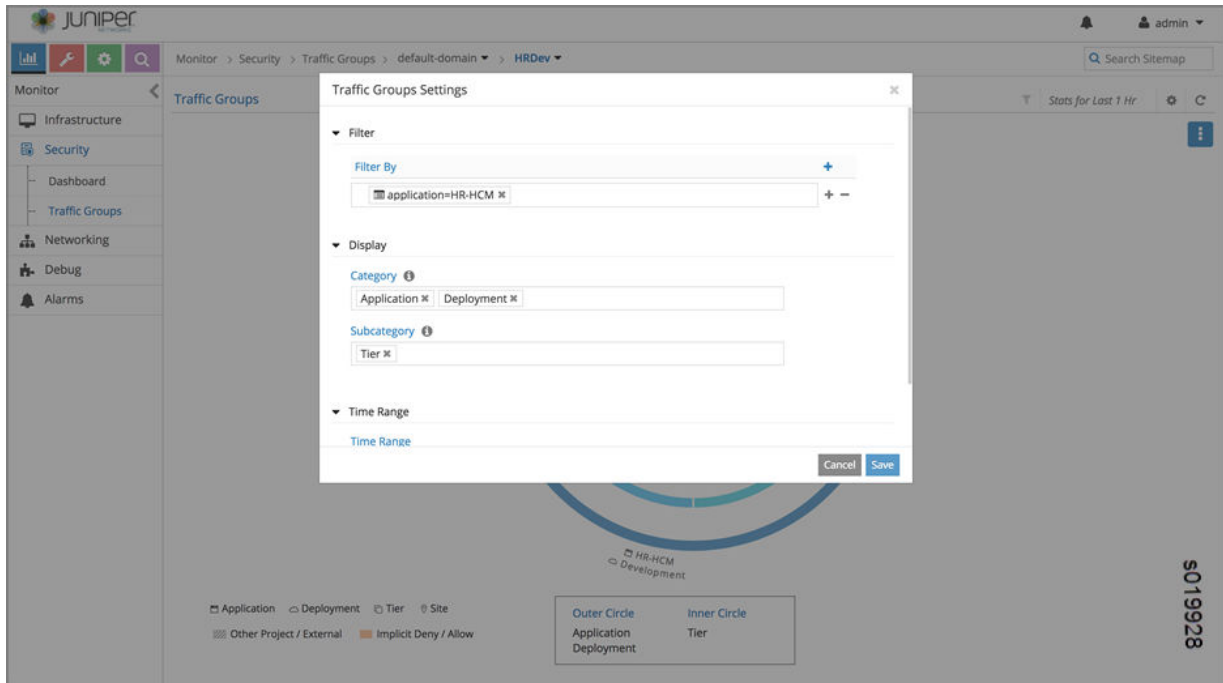


Figure 156: Traffic Groups, Details



You can change the settings of what statistics are displayed in each traffic group at the **Traffic Groups Settings** screen see [Figure 157 on page 796](#).

Figure 157: Traffic Groups Settings



# 5

PART

## Monitoring and Troubleshooting Contrail

- 
- [Configuring Traffic Mirroring to Monitor Network Traffic | 798](#)
  - [Understanding Contrail Analytics | 814](#)
  - [Configuring Contrail Analytics | 842](#)
  - [Using Contrail Analytics to Monitor and Troubleshoot the Network | 873](#)
  - [Common Support Answers | 964](#)
-

# Configuring Traffic Mirroring to Monitor Network Traffic

## IN THIS CHAPTER

- [Configuring Traffic Analyzers and Packet Capture for Mirroring | 798](#)
- [Configuring Interface Monitoring and Mirroring | 806](#)
- [Mirroring Enhancements | 807](#)
- [Analyzer Service Virtual Machine | 809](#)
- [Mapping VLAN Tags from a Physical NIC to a VMI \(NIC-Assisted Mirroring\) | 812](#)

## Configuring Traffic Analyzers and Packet Capture for Mirroring

### IN THIS SECTION

- [Traffic Analyzer Images | 798](#)
- [Configuring Traffic Analyzers | 799](#)
- [Setting Up Traffic Mirroring Using Configure > Networking > Services | 799](#)

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.

This section describes 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

Contrail Controller enables you to mirror captured packet traffic to a traffic analyzer. Follow these steps to mirror captured packet traffic:

1. Configure analyzer(s) on the host.
2. Set up rules for packet capture.

You can set up traffic mirroring using **Configure > Networking > Services**. For more information, see "[Setting Up Traffic Mirroring Using Configure > Networking > Services](#)" on page 799.

## Setting Up Traffic Mirroring Using Configure > Networking > Services

Follow these steps to set up traffic mirroring using **Configure > Networking > Services**.

1. Access **Configure > Services > Service Templates**.

The **Service Templates** screen appears; see [Figure 158 on page 800](#).

Figure 158: Service Templates

Template	Mode	Type / Version	Interface (s)	Image & Flavor
netns-nat-template	In-network-nat	Source-nat / v1	Right, Left	- / -
haproxy-loadbalancer-template	In-network-nat	Loadbalancer / v1	Right, Left	- / -
docker-template	Transparent	Firewall / v1	Management, Left, Right	ubuntu / -
nat-template	In-network-nat	Firewall / v1	Management, Left, Right	analyzer / m1.medium

- 2. To create a new service template, click the + icon.  
The **Create** window appears. Select the Service Template tab; see [Figure 159 on page 800](#).

Figure 159: Create Service Template

Service TemplatePermissions

Name

analyzer-service-template

Version

v2

Virtualization Type

Virtual Machine

Service Mode

Transparent

Service Type

Analyzer

Interface (s)

management

left

Cancel

Save

- 3. Complete the fields by using the guidelines in [Table 47 on page 801](#).



Table 47: 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.
<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>	<p>From the drop-down list, click the check boxes to indicate which interface types are used for this analyzer service template:</p> <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 160 on page 802](#).

Figure 160: Create Service Instances

Create

Service Instance

Permissions

Name

analyzer-service-instance

Service Template

analyzer-service-template - [transparent (m... ▼

Interface Type

management

left

Virtual Network

Select Virtual Network ▼

Select Virtual Network ▼

▶ Port Tuples

▶ Service Health Check

▶ Allowed Address Pair

▶ Static Route

Cancel

Save

s041858

5. Complete the fields by using the guidelines in [Table 48 on page 802](#).

Table 48: Create Service Instances Fields

Field	Description
Name	Enter a text name for this service instance.
Service Template	Select from a drop-down list of available service templates the template to use for this service instance, analyzer-service-template in this example.
Interface Type	Each interface configured in the service template for this instance appears in a list.
Virtual Network	Select from a drop-down list of available virtual networks the network for each interface that is configured for the instance.

Table 48: Create Service Instances Fields *(Continued)*

Field	Description
Save	Click <b>Save</b> to commit your changes.
Cancel	Click <b>Cancel</b> to clear your changes and start over.

6. 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 161 on page 803](#).

Figure 161: Create Policy

Create

Policy Permissions

Policy Name

Policy Name

Policy Rule(s)

Action	Protocol	Source	Ports	Direction	Destination	Ports	Log	Services	Mirror	QoS	+
--------	----------	--------	-------	-----------	-------------	-------	-----	----------	--------	-----	---

Cancel Save

s041859

- 7.
8. 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 162 on page 804](#).

Figure 162: Create Policy Rules

Create

Policy

Permissions

Policy Name

analyzer-policy

Policy Rule(s)

Action	Protocol	Source	Ports	Direction	Destination	Ports	Log	Services	Mirror	QoS	
PASS	ANY	ANY (All Networks i...)	ANY	<>	ANY (All Networks i...)	ANY	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	+ -

Cancel

Save

s041833

9. To add policy rules, complete the fields, using the guidelines in [Table 49 on page 804](#).

**NOTE:** When there is a network policy attached to the virtual network, any conflicting rules configured for the analyzer will not take effect.

Table 49: Add Rule Fields

Field	Description
Action	Select PASS or DENY as the rule action.
Protocol	Select the protocol for the policy rule, or select ANY.
Source	Select from multiple drop-down lists the source for this rule, including options under CIDR, Network, Policy, or Security Group.
Ports	Select from a drop-down list the source ports for the rule.
Direction	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>

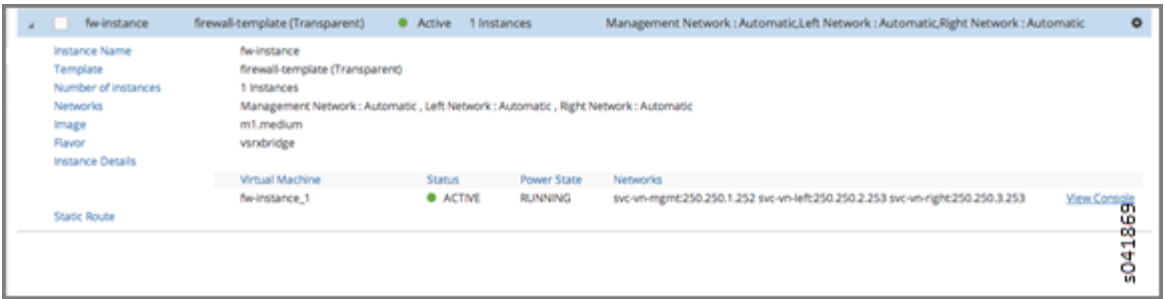
**Table 49: Add Rule Fields (Continued)**

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 163 on page 805](#). The analyzer service VM launches the Contrail-enhanced Wireshark as it starts and captures the mirrored packets destined to this service.

**Figure 163: Service Instances View Console**



## RELATED DOCUMENTATION

Configuring Interface Monitoring and Mirroring | 806

Mirroring Enhancements	807
Analyzer Service Virtual Machine	809
Mapping VLAN Tags from a Physical NIC to a VMI (NIC-Assisted Mirroring)	812

## Configuring Interface Monitoring and Mirroring

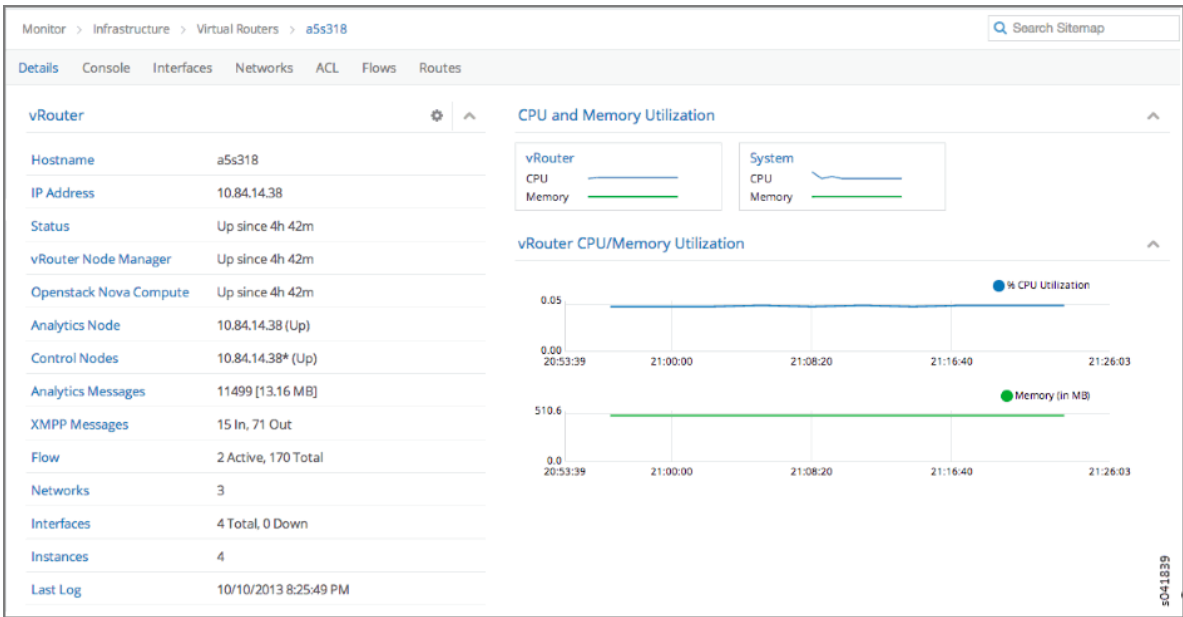
Contrail supports user monitoring of traffic on any guest virtual machine interface when using the Juniper Contrail user interface.

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 164 on page 806](#).

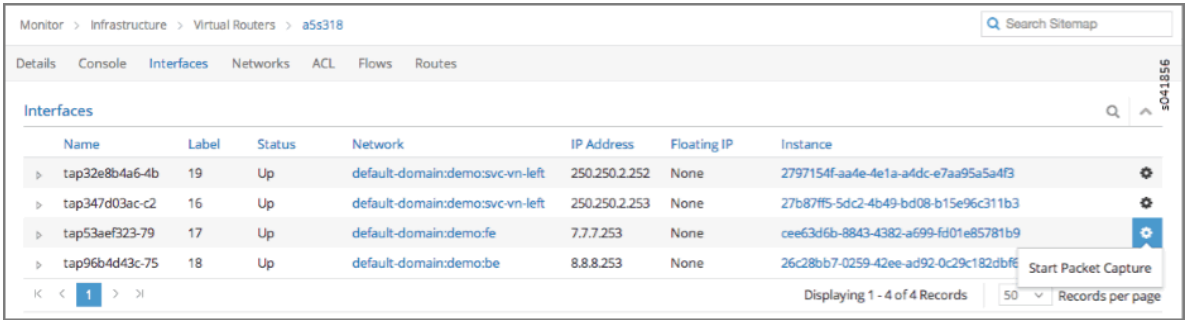
Figure 164: Individual vRouter



A list of interfaces for that vRouter appears.

- 3. For the interface to mirror, click the Action icon in the last column and select the option Packet Capture; see [Figure 165 on page 807](#).

Figure 165: Interfaces



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 | 807](#)
- [Analyzer Service Virtual Machine | 809](#)
- [Mapping VLAN Tags from a Physical NIC to a VMI \(NIC-Assisted Mirroring\) | 812](#)

Mirroring Enhancements

IN THIS SECTION

- Mirroring Specified Traffic | 808
- Configuring Headers and Next Hops | 808
- How Mirroring is Implemented | 808

## 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 | 806](#)

[Analyzer Service Virtual Machine | 809](#)

[Mapping VLAN Tags from a Physical NIC to a VMI \(NIC-Assisted Mirroring\) | 812](#)

## Analyzer Service Virtual Machine

### IN THIS SECTION

- [Packet Format for Analyzer | 809](#)
- [Metadata Format | 810](#)
- [Wireshark Changes | 811](#)
- [Troubleshooting Packet Display | 811](#)

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

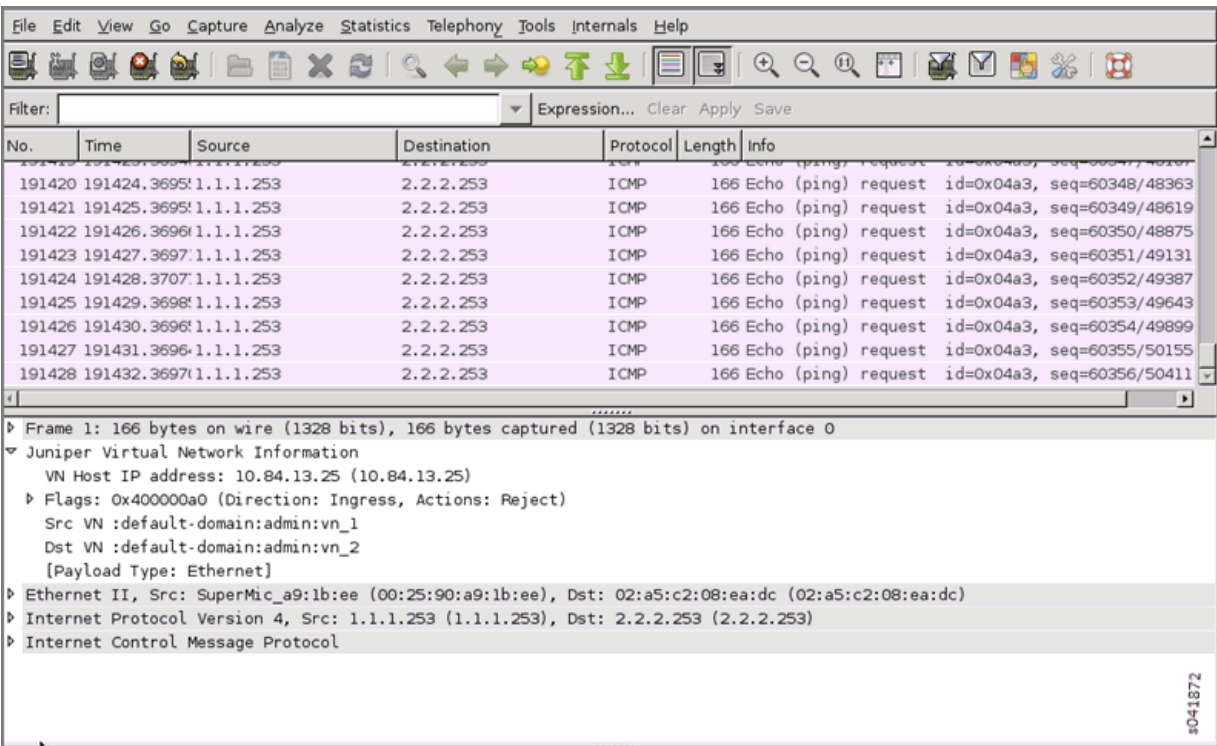
*TLV end*

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 166 on page 811](#).

**Figure 166: 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 | 806](#)

[Mirroring Enhancements | 807](#)

[Mapping VLAN Tags from a Physical NIC to a VMI \(NIC-Assisted Mirroring\) | 812](#)

## 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 | 806](#)

---

[Mirroring Enhancements | 807](#)

---

[Analyzer Service Virtual Machine | 809](#)

# Understanding Contrail Analytics

## IN THIS CHAPTER

- [Understanding Contrail Analytics | 814](#)
- [Contrail Alerts | 815](#)
- [Underlay Overlay Mapping in Contrail | 819](#)

## 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

<i>Contrail Alerts</i>
<i>Analytics Scalability</i>
<i>High Availability for Analytics</i>
<i>Ceilometer Support in Contrail</i>
<i>Underlay Overlay Mapping in Contrail</i>
<i>Monitoring the System</i>
<i>Debugging Processes Using the Contrail Introspect Feature</i>
<i>Monitor &gt; Infrastructure &gt; Dashboard</i>
<i>Monitor &gt; Infrastructure &gt; Control Nodes</i>
<i>Monitor &gt; Infrastructure &gt; Virtual Routers</i>
<i>Monitor &gt; Infrastructure &gt; Analytics Nodes</i>
<i>Monitor &gt; Infrastructure &gt; Config Nodes</i>
<i>Monitor &gt; Networking</i>
<a href="#">Understanding Flow Sampling</a>
<i>Query &gt; Flows</i>
<i>Query &gt; Logs</i>
<i>System Log Receiver in Contrail Analytics</i>
<i>Example: Debugging Connectivity Using Monitoring for Troubleshooting</i>

## Contrail Alerts

### IN THIS SECTION

- [Alert API Format | 816](#)
- [Analytics APIs for Alerts | 817](#)
- [Analytics APIs for SSE Streaming | 818](#)

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: "eyJ0aW1lc3RhbnRhbXAiOiAxNDQyOTk1MzQ5MTUzMTc4LCAiaHR0cF9wb3J0Ijog
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 | 820](#)
- [Underlay Overlay Analytics Available in Contrail | 820](#)
- [Architecture and Data Collection | 821](#)
- [New Processes/Services for Underlay Overlay Mapping | 821](#)
- [External Interfaces Configuration for Underlay Overlay Mapping | 822](#)
- [Physical Topology | 822](#)
- [SNMP Configuration | 823](#)

- [Link Layer Discovery Protocol \(LLDP\) Configuration | 823](#)
- [IPFIX and sFlow Configuration | 823](#)
- [Sending pRouter Information to the SNMP Collector in Contrail | 826](#)
- [pRouter UVEs | 826](#)
- [Contrail User Interface for Underlay Overlay Analytics | 828](#)
- [Enabling Physical Topology on the Web UI | 829](#)
- [Viewing Topology to the Virtual Machine Level | 829](#)
- [Viewing the Traffic of any Link | 829](#)
- [Trace Flows | 830](#)
- [Search Flows and Map Flows | 831](#)
- [Overlay to Underlay Flow Map Schemas | 832](#)
- [Module Operations for Overlay Underlay Mapping | 835](#)
- [SNMP Collector Operation | 835](#)
- [Topology Module Operation | 837](#)
- [IPFIX and sFlow Collector Operation | 838](#)
- [Troubleshooting Underlay Overlay Mapping | 839](#)
- [Script to add pRouter Objects | 839](#)

## 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.

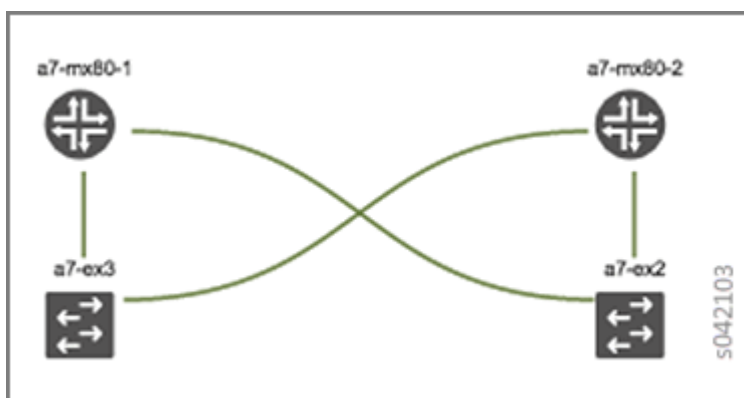
### Physical Topology

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 167: 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 168: 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 169: 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"
  }
]
```

s042104

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 170: 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: [...]
  }
}

```

s042435

## 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.

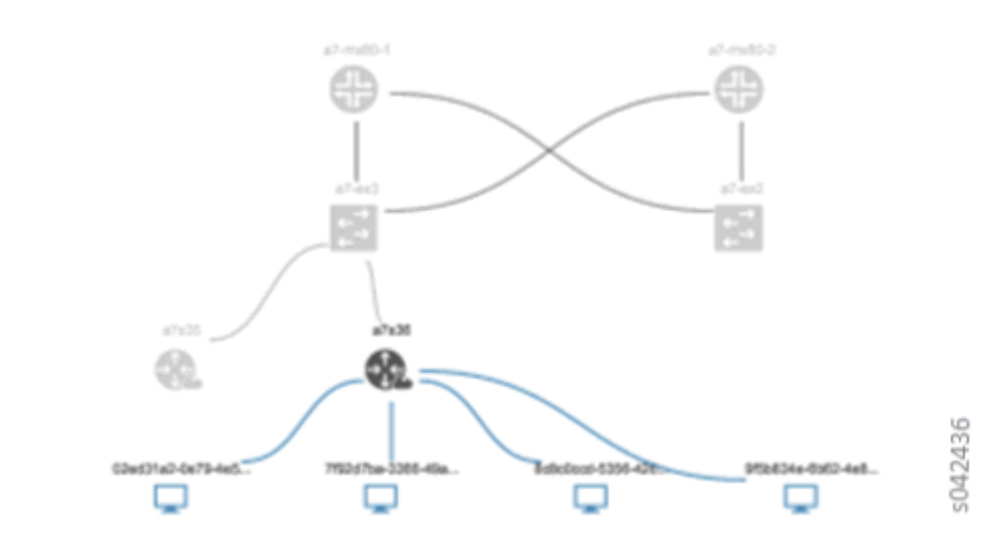
```
service supervisor-webui restart
```

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.

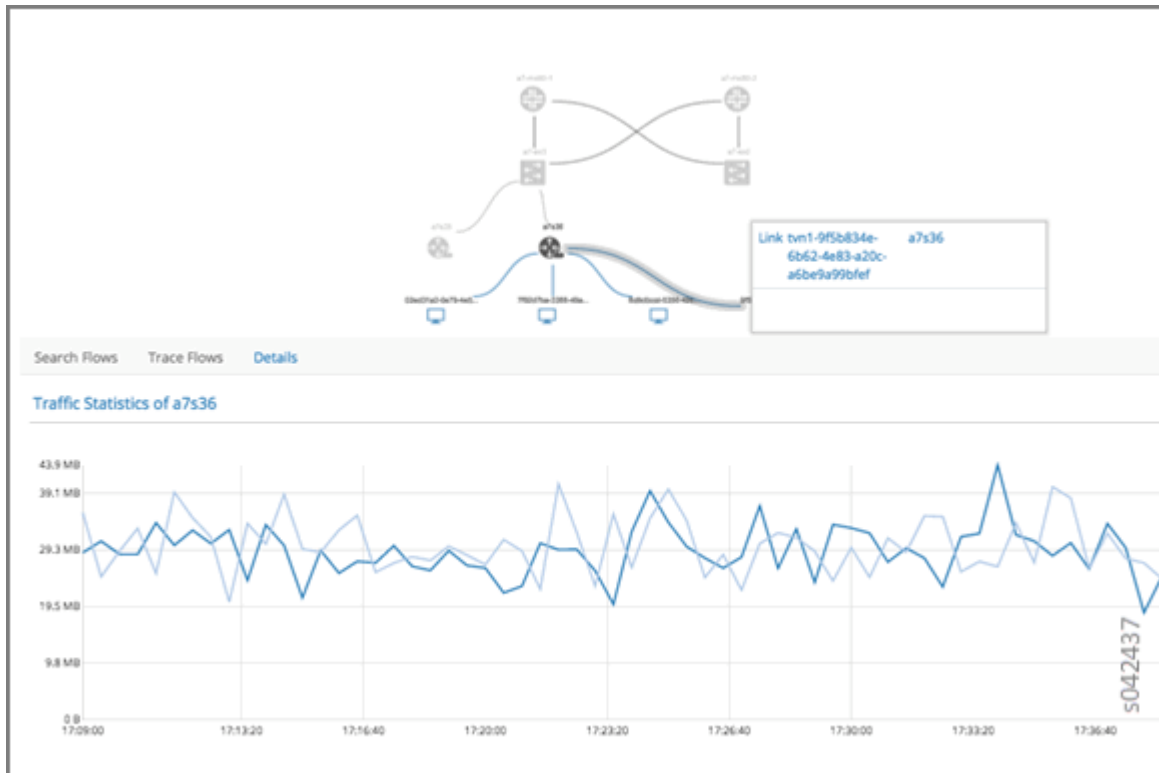
Figure 171: Physical Topology Related to a vRouter



## 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.

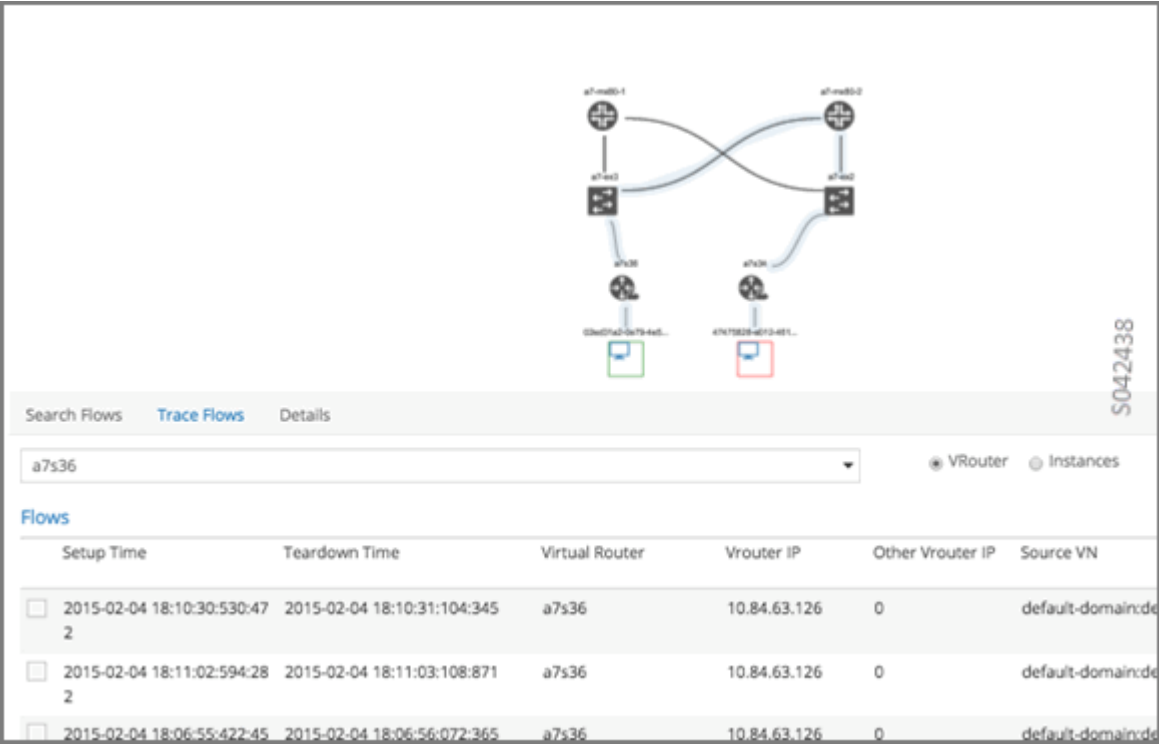
Figure 172: Traffic Statistics Graph



## 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 173: List of Active Flows



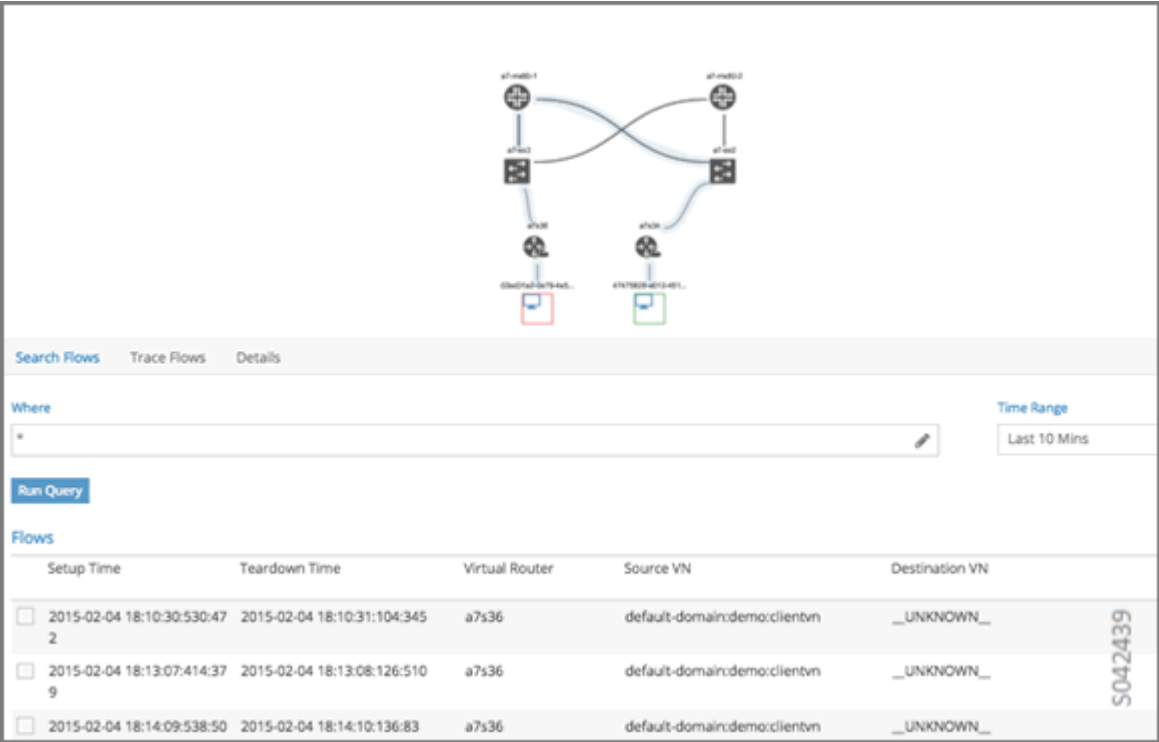
*Limitations of Trace Flow Feature*

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 174: 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 LldpTable 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 | 842](#)
- [High Availability for Analytics | 844](#)
- [System Log Receiver in Contrail Analytics | 844](#)
- [Sending Flow Messages to the Contrail System Log | 845](#)
- [Ceilometer Support in a Contrail Cloud | 846](#)
- [User Configuration for Analytics Alarms and Log Statistics | 854](#)
- [Alarms History | 863](#)
- [Node Memory and CPU Information | 865](#)
- [Role- and Resource-Based Access Control for the Contrail Analytics API | 866](#)
- [Configuring Analytics as a Standalone Solution | 867](#)
- [Configuring Secure Sandesh and Introspect for Contrail Analytics | 870](#)

## 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.

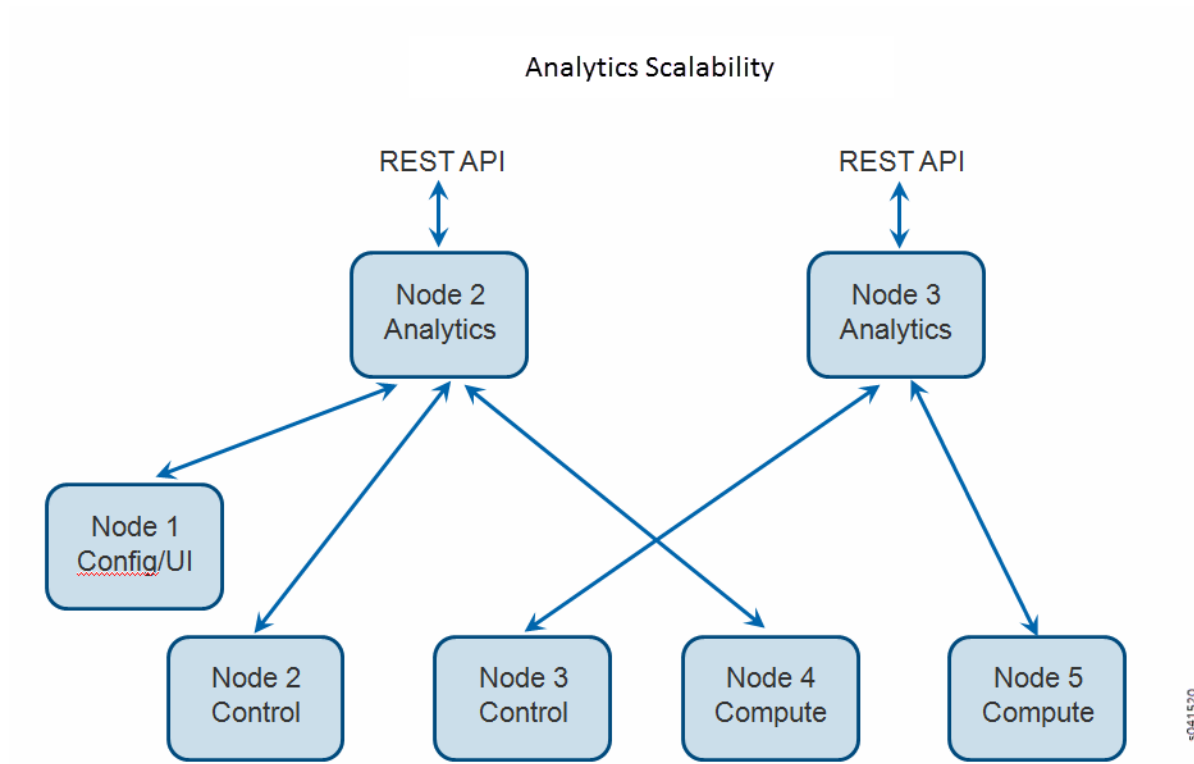
The following is the recommended locations for the various component roles of the Contrail system for a 5-node configuration.

- 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 175 on page 843 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 175: 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.

## System Log Receiver in Contrail Analytics

### IN THIS SECTION

- [Overview | 845](#)
- [Redirecting System Logs to Contrail Collector | 845](#)
- [Exporting Logs from Contrail Analytics | 845](#)

## 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-7622a0c8eeea 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.

## Ceilometer Support in a Contrail Cloud

### IN THIS SECTION

[Overview](#) | 847

- [Ceilometer Details | 847](#)
- [Verification of Ceilometer Operation | 848](#)
- [Contrail Ceilometer Plugin | 850](#)
- [Ceilometer Installation and Provisioning | 853](#)

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 |
```

```

15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create | delta | network | b3b72b98-f61e-4e1f-9a9b-84f4f3ddec0b |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create | delta | network | cb829abd-e6a3-42e9-a82f-0742db55d329 |
15c0240142084d16b3127d6f844adbd9 | 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 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet | gauge | subnet | 09105ed1-1654-4b5f-8c12-f0f2666fa304 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet | gauge | subnet | 4bf00aac-407c-4266-a048-6ff52721ad82 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet.create | delta | subnet | 09105ed1-1654-4b5f-8c12-f0f2666fa304 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet.create | delta | subnet | 4bf00aac-407c-4266-a048-6ff52721ad82 |
15c0240142084d16b3127d6f844adbd9 | 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      |
```

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	
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'ceed483222f9453ab1d7bcdd353971bc', 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 | 854](#)
- [Examples: Detecting Anomalies | 856](#)
- [Configuring the User-Defined Log Statistic | 857](#)
- [Implementing the User-Defined Log Statistic | 860](#)

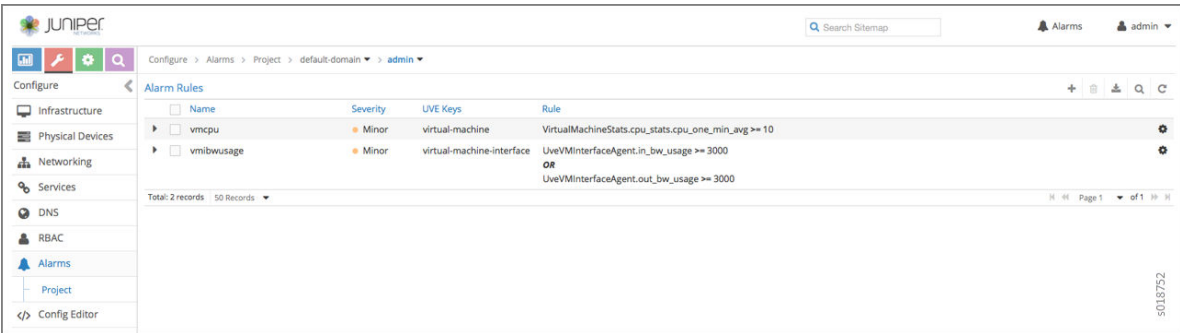
## Configuring Alarms Based on User-Visible Entities Data

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 50 on page 855](#) for field descriptions.

The screenshot shows the 'Edit' dialog for an alarm rule. The 'Alarm Rule' tab is selected. The fields are as follows:

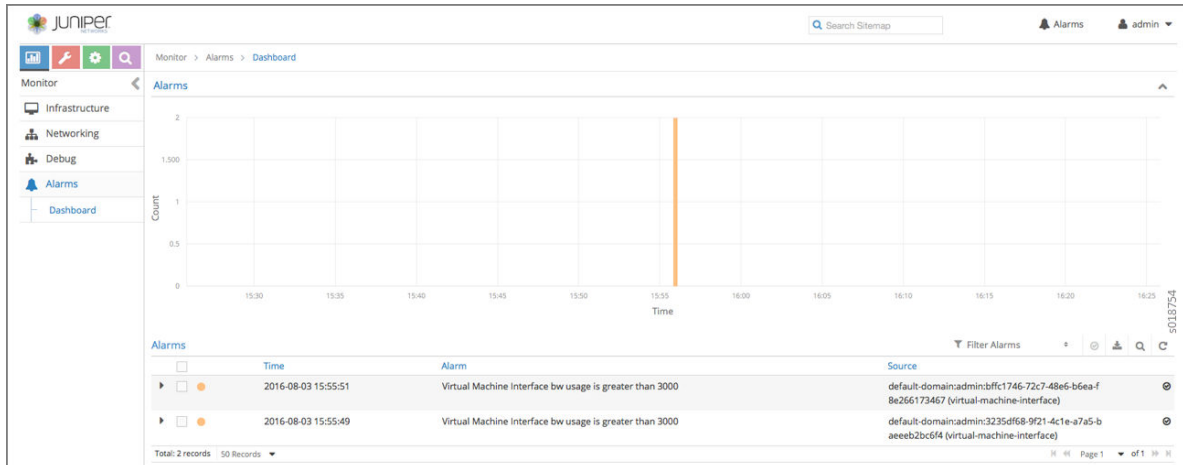
- Name:** vmibwusage
- Severity:** Minor
- UVE Keys:** virtual-machine-interface
- Description:** Virtual Machine Interface bw usage is greater than 3000
- Rule:**
  - UveVMInterfaceAgent.in\_bw\_usage >= 3000
  - OR
  - UveVMInterfaceAgent.out\_bw\_usage >= 3000

At the bottom right of the dialog are 'Cancel' and 'Save' buttons. The background shows a sidebar with 'Configure > Alarms' and a list of alarm rules.

Table 50: 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 contrail-alarm-gen 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_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.

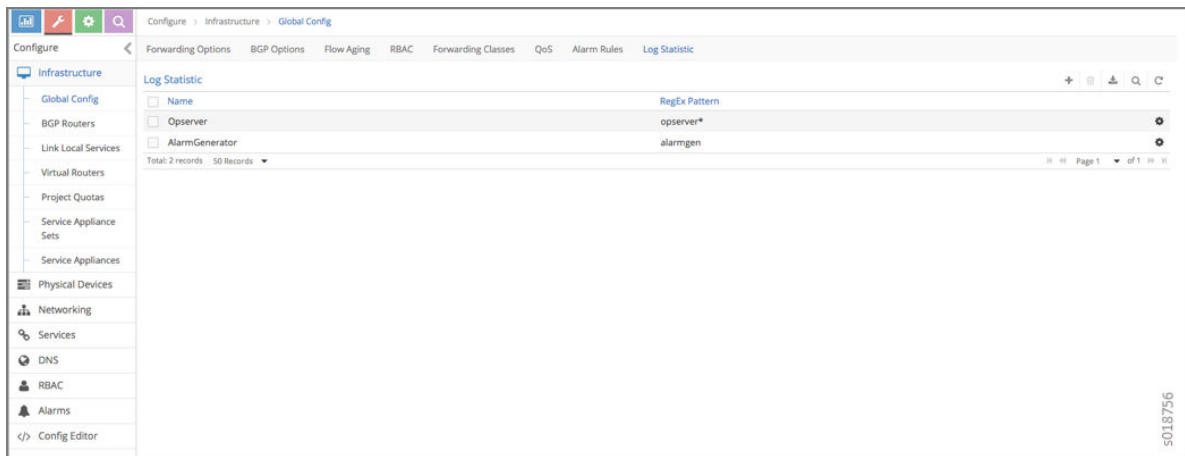
Starting with Contrail Release 3.1, the user-defined log statistic feature provides additional abilities for monitoring and troubleshooting by enabling the user to set a counter on any regular Perl-type

expression. Each time the pattern is found in any system logs, UVEs, or object logs, the counter is incremented.

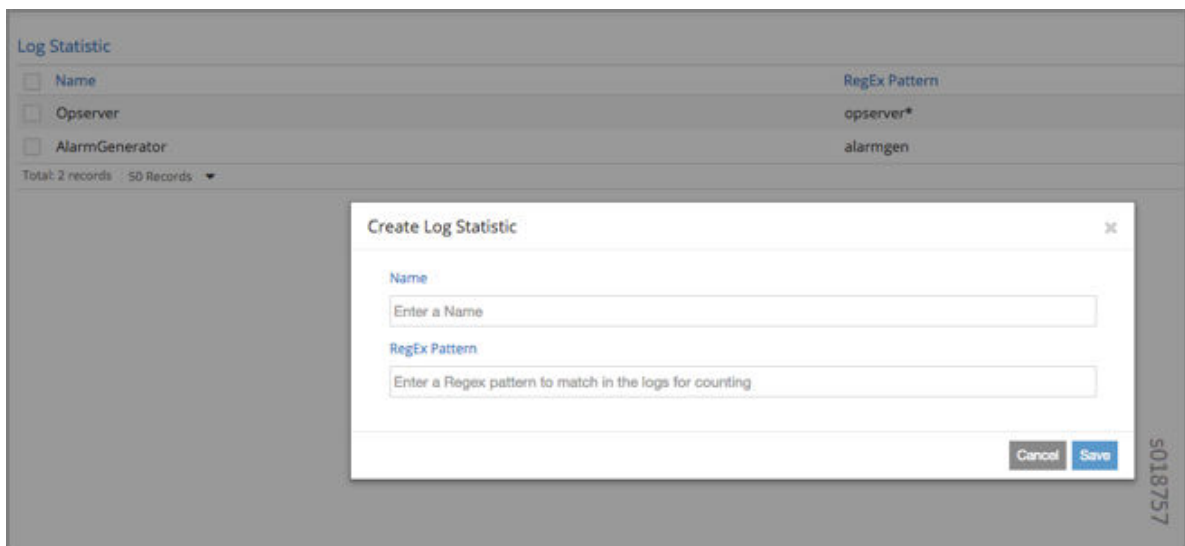
The user-defined log statistic can be configured from the Contrail UI or from the command line, using `vnc_api`.

To configure the user-defined log statistic from the Contrail UI:

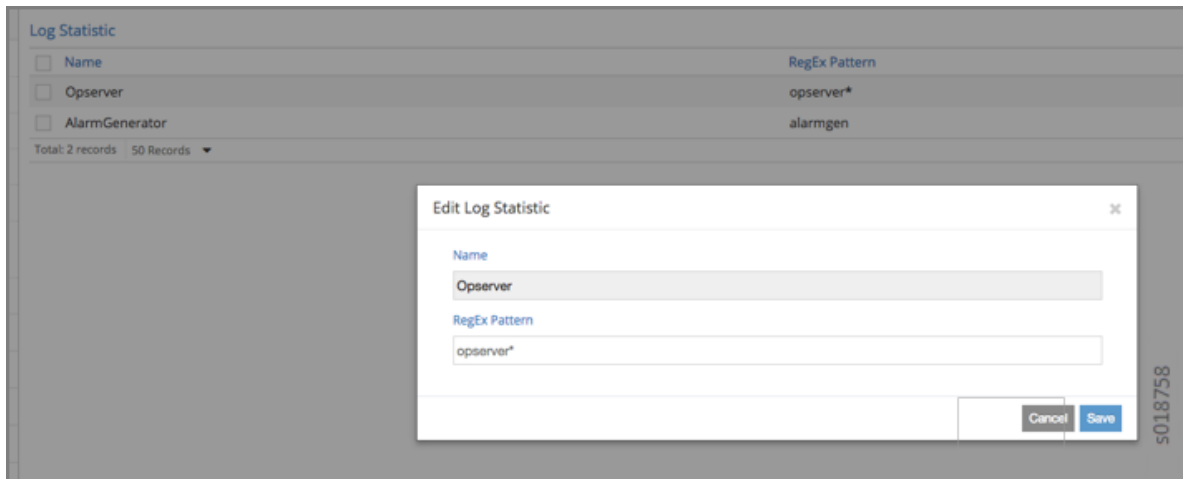
1. Navigate to **Configure > Infrastructure > Global Config** and select **Log Statistic**.



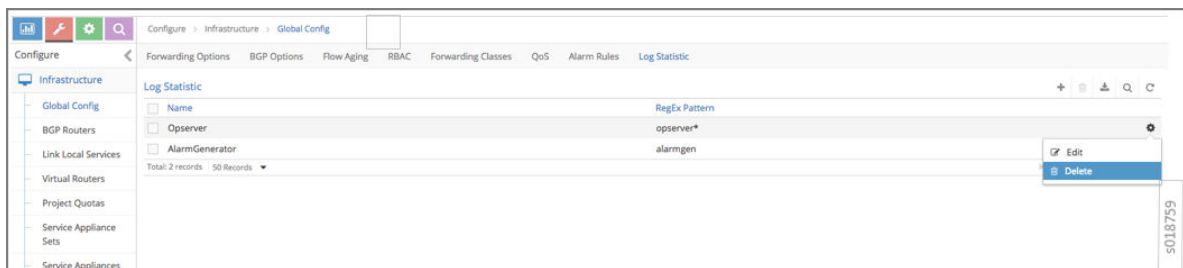
2. To create a log statistic, click the plus (+) icon to access the **Create Log Statistic** screen. Enter a name for the user-defined log statistic, and in the **RegEx Pattern** field, enter the Perl-type expression to look for and count.



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 | 863](#)

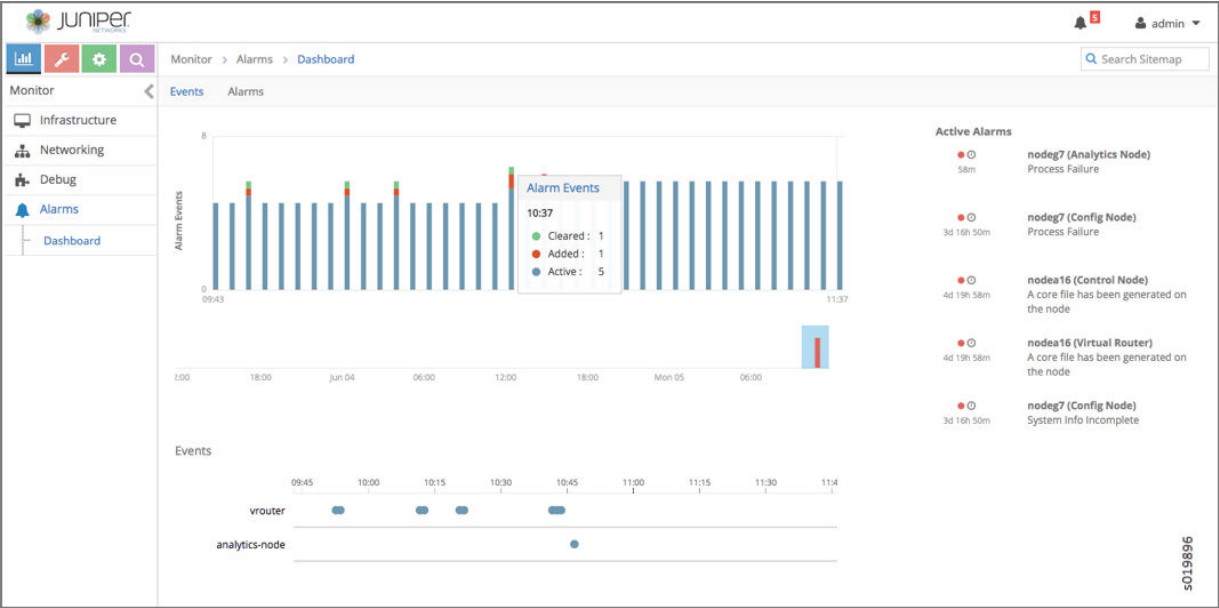
Starting with Contrail Release 4.0, you can view a history of alarms that were raised or reset. You can also view a history of user-visible entities (UVEs) that have been changed.

### 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 176 on page 864](#) 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.

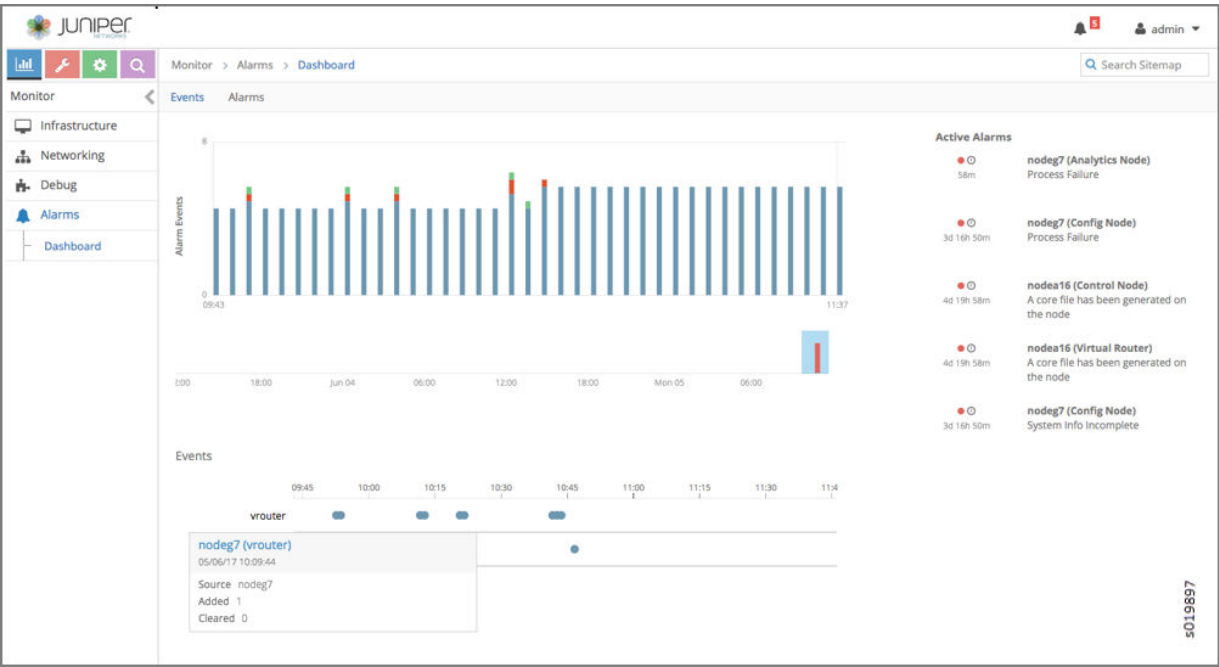
You can also use a `contrail-status` query to view the alarms history. Additionally, the `contrail-status` displays a history of added, updated, and removed information for UVEs in Contrail.

Figure 176: 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 177 on page 864](#).

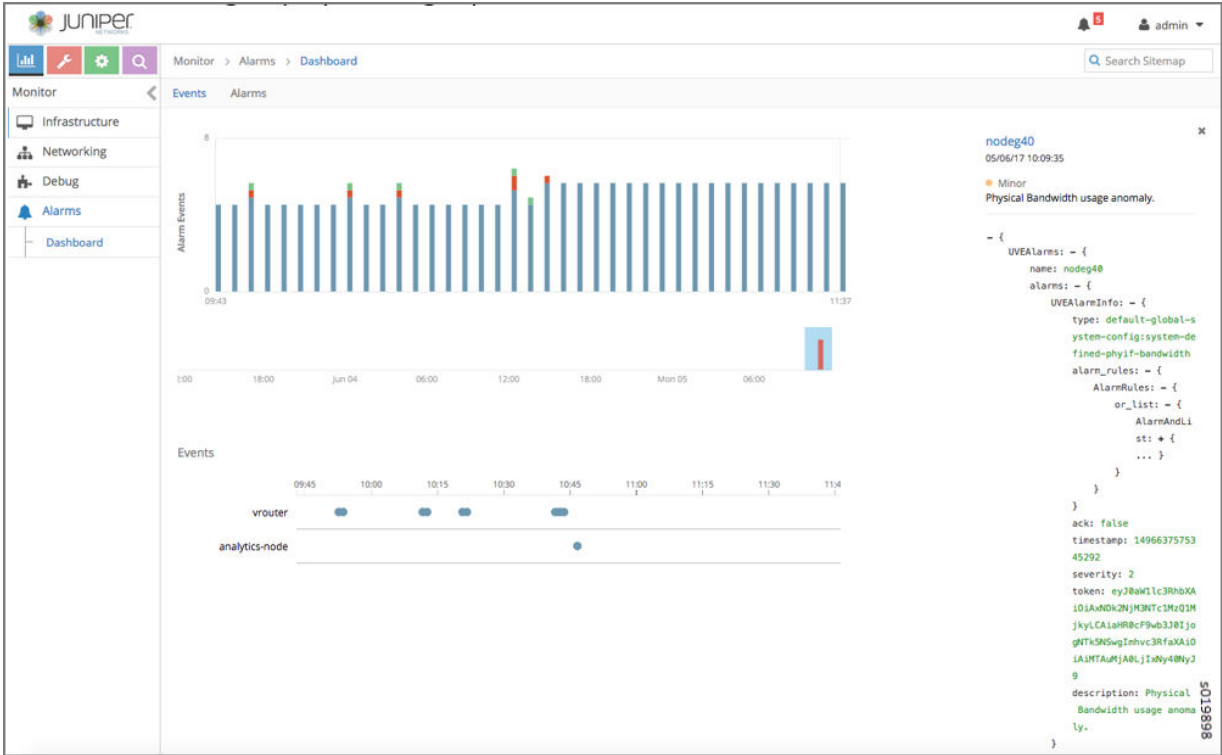
Figure 177: Events Log Tooltip





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 178 on page 865](#).

**Figure 178: Detailed Event Log**



## RELATED DOCUMENTATION

[User Configuration for Analytics Alarms and Log Statistics](#) | 854

## 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](#) | 854

## 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](#) | 854

## Configuring Analytics as a Standalone Solution

### IN THIS SECTION

- [Overview: Contrail Analytics as a Standalone Solution](#) | 867
- [Configuration Examples for Standalone](#) | 868

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 | 870](#)

*Understanding Contrail Analytics*

## Configuring Secure Sandesh and Introspect for Contrail Analytics

### IN THIS SECTION

- [Configuring Secure Sandesh Connection | 871](#)
- [Configuring Secure Introspect Connection | 871](#)

## 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 51 on page 872](#).

**Table 51: 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 | 873](#)
- [Debugging Processes Using the Contrail Introspect Feature | 877](#)
- [Monitor > Infrastructure > Dashboard | 882](#)
- [Monitor > Infrastructure > Control Nodes | 886](#)
- [Monitor > Infrastructure > Virtual Routers | 897](#)
- [Monitor > Infrastructure > Analytics Nodes | 911](#)
- [Monitor > Infrastructure > Config Nodes | 919](#)
- [Monitor > Networking | 923](#)
- [Query > Flows | 935](#)
- [Query > Logs | 945](#)
- [Understanding Flow Sampling | 952](#)
- [Example: Debugging Connectivity Using Monitoring for Troubleshooting | 957](#)

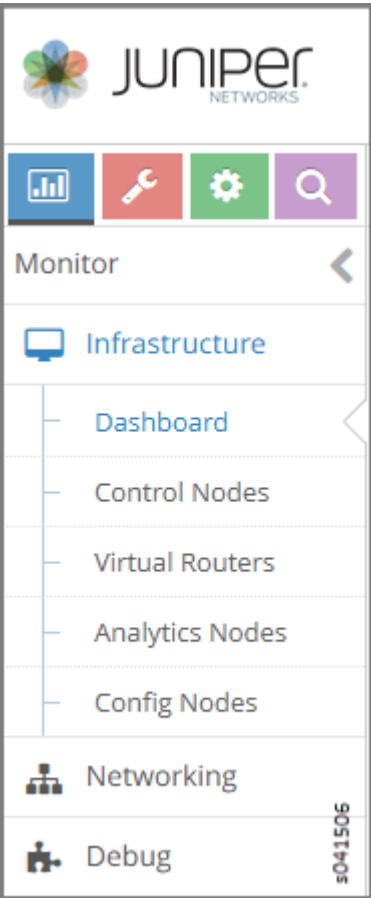
## Monitoring the System

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 179 on page 874](#)

Figure 179: Monitor Menu



See [Table 52 on page 874](#) for descriptions of the items available under each of the menu options from the **Monitor** icon.

Table 52: 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 52: 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>See <i>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>See <i>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. See <i>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. See <i>Monitor &gt; Infrastructure &gt; Config Nodes</i>.</p>

Table 52: 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 52: 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.

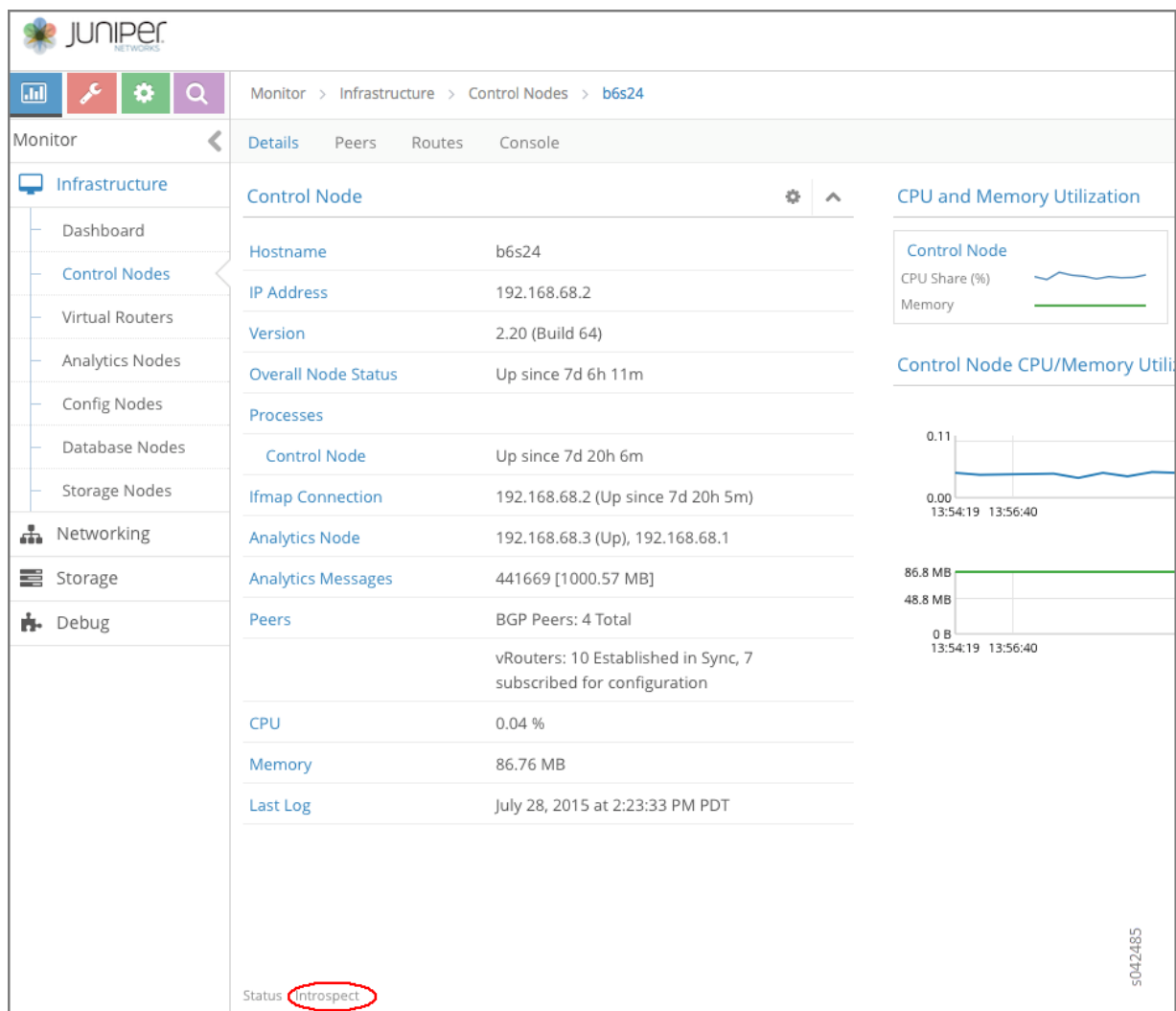
You can display the HTTP introspect of a Contrail daemon directly by accessing the following Introspect ports:

- *<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 180 on page 879](#) shows the contrail-control infrastructure page. Notice the Introspect link at the bottom of the Control Nodes Details tab window.

Figure 180: Control Nodes Details Tab Window



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 181 on page 880 shows the Controller Introspect window.

Figure 181: Controller Introspect Window

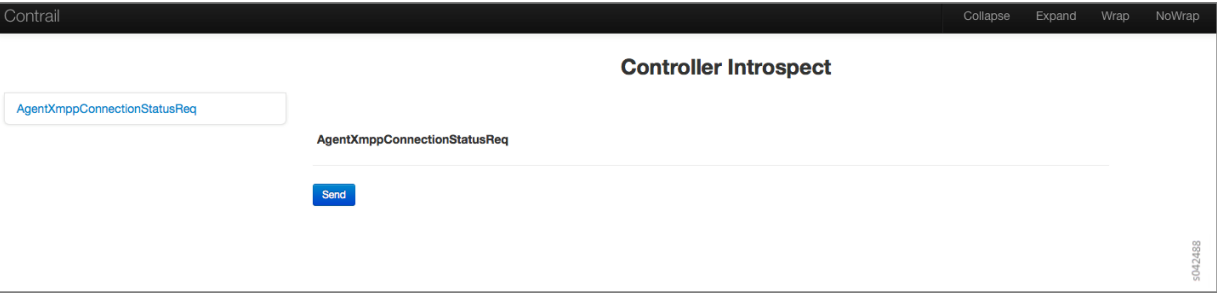


Figure 182 on page 880 shows an example of the BGP Peer (bgp\_peer.xml) Introspect page.

Figure 182: BGP Peer Introspect Page

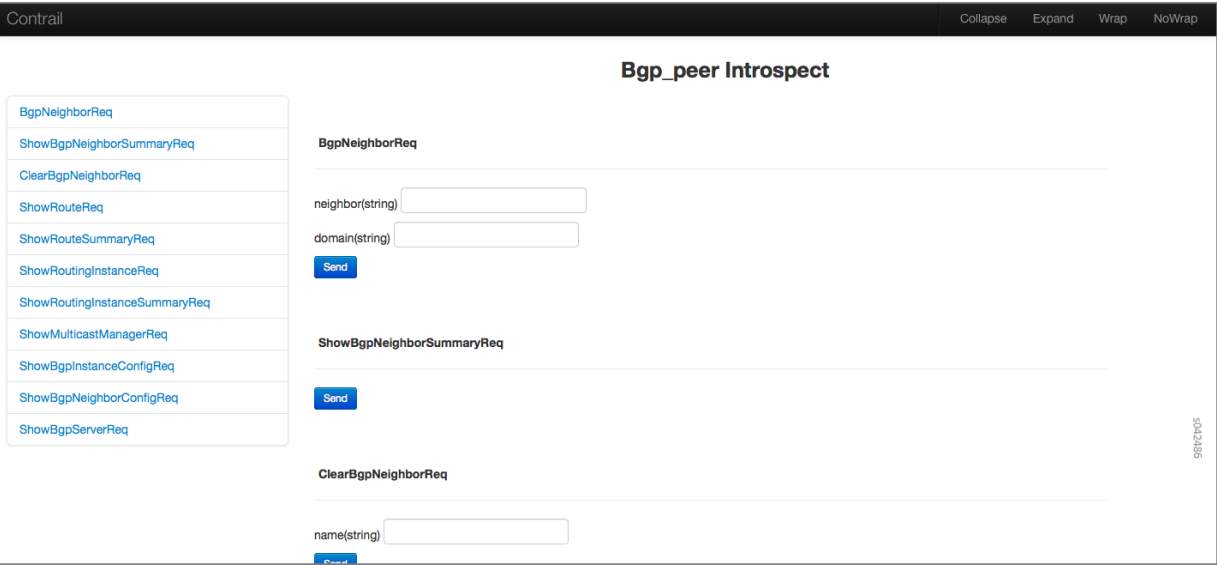




Figure 183 on page 881 shows an example of the BGP Neighbor Summary Introspect page.

Figure 183: 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	XMPP	internal	Established	192.168.68.2	-
b6s18	false	-	192.168.69.5	-	0	XMPP	internal	Established	192.168.68.2	-
b6s13	false	-	192.168.69.8	-	0	XMPP	internal	Established	192.168.68.2	-
b6s7	false	-	192.168.69.11	-	0	XMPP	internal	Established	192.168.68.2	-
b6s33	false	-	192.168.68.11	-	0	XMPP	internal	Established	192.168.68.2	-
b6s9	false	-	192.168.69.10	-	0	XMPP	internal	Established	192.168.68.2	-
b6s26	false	-	192.168.68.4	-	0	XMPP	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 184 on page 882 shows an example of the Agent (agent.xml) Introspect page.

Figure 184: Agent Introspect Page

Contrail

CollapseExpandWrapNoWrap

AgentXmppConnectionStatus

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 | 883](#)
- [Monitor Individual Details from the Dashboard | 883](#)
- [Using Bubble Charts | 884](#)
- [Color-Coding of Bubble Charts | 885](#)

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 185 on page 883](#).

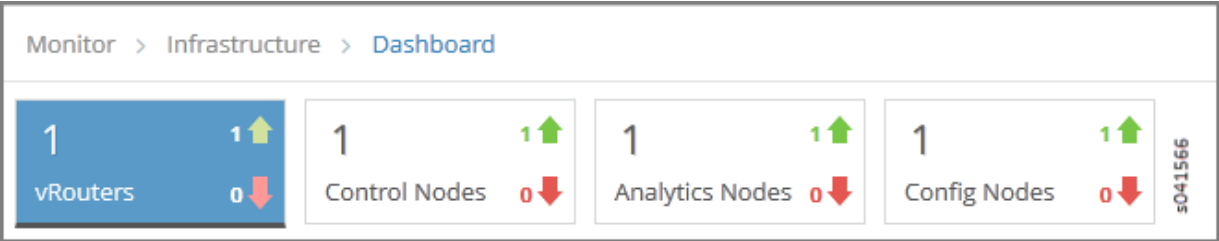
**Figure 185: Monitor > Infrastructure > Dashboard**



## 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 186 on page 884](#). 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 186: Dashboard Summary Boxes



Detailed information about monitoring each of the areas represented by the boxes is provided in the links in [Table 53 on page 884](#).

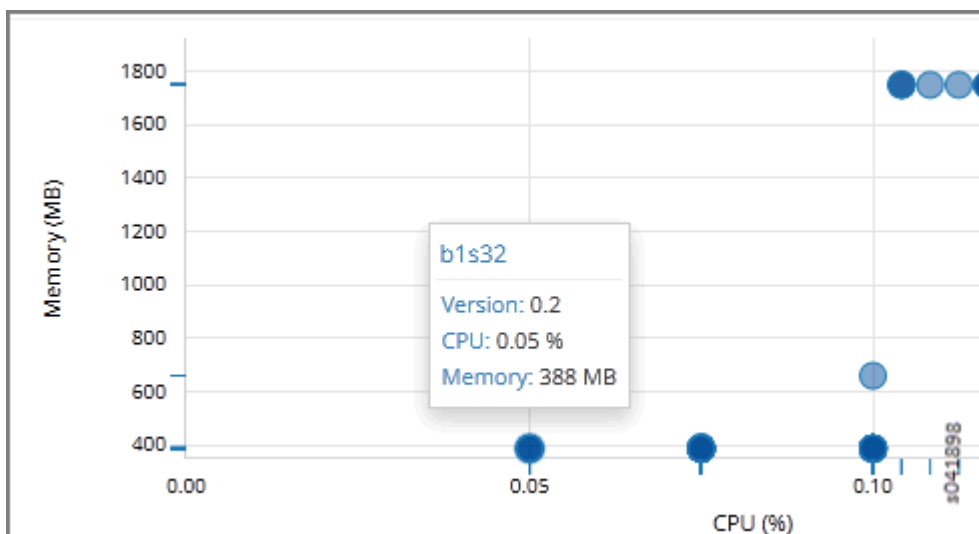
Table 53: Dashboard Summary Boxes

Box	For More Information
vRouters	<i>Monitor &gt; Infrastructure &gt; Virtual Routers</i>
Control Nodes	<i>Monitor &gt; Infrastructure &gt; Control Nodes</i>
Analytics Nodes	<i>Monitor &gt; Infrastructure &gt; Analytics Nodes</i>
Config Nodes	<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 187 on page 885](#). You can click through the summary information to get more details about the component.

Figure 187: 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 | 886](#)
- [Monitor Individual Control Node Details | 887](#)
- [Monitor Individual Control Node Console | 889](#)
- [Monitor Individual Control Node Peers | 892](#)
- [Monitor Individual Control Node Routes | 894](#)

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 188 on page 886](#). See [Table 54 on page 887](#) for descriptions of the fields on this screen.

**Figure 188: Control Nodes Summary**



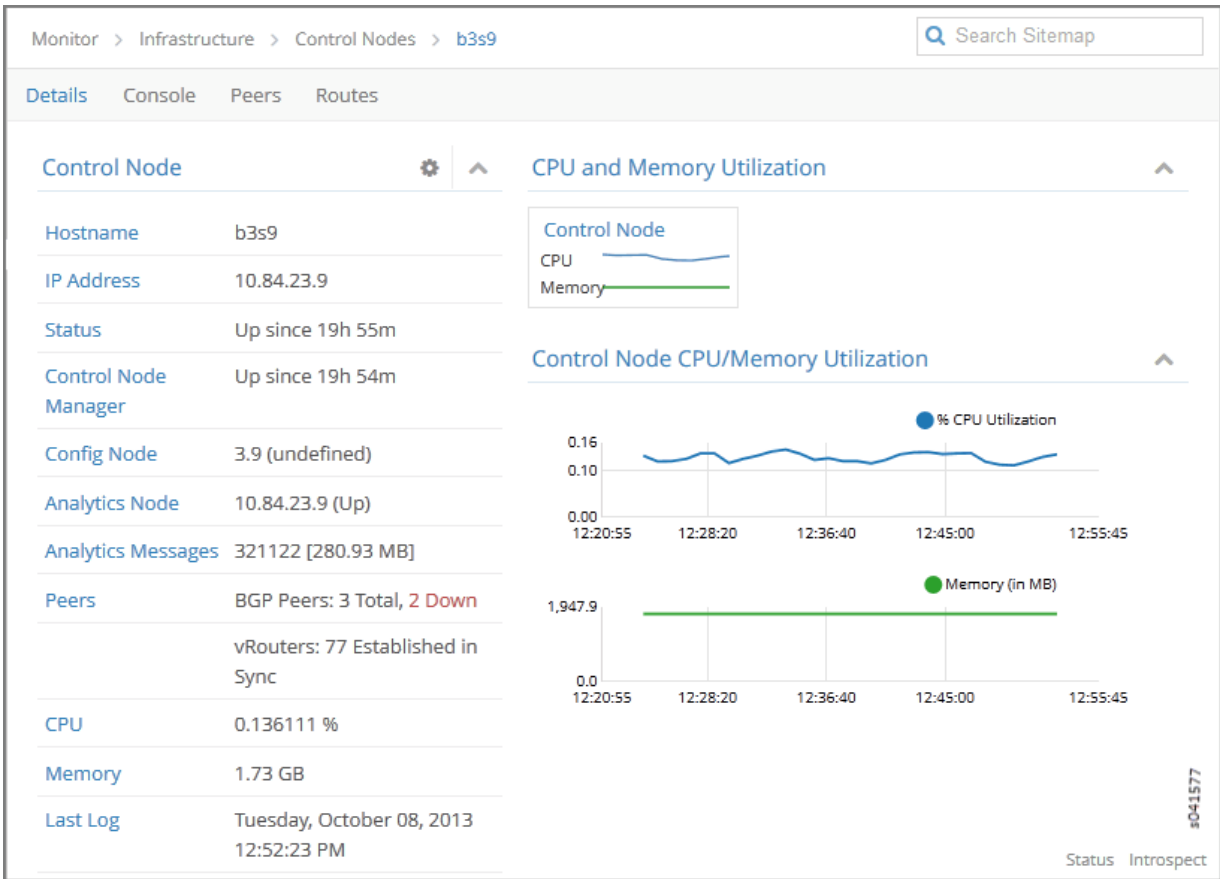
Table 54: Control Nodes Summary Fields

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 189 on page 888](#).

Figure 189: 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 55 on page 888](#) for descriptions of the fields on this tab.

Table 55: Individual Control Node—Details Tab Fields

Field	Description
Hostname	The host name defined for this control node.
IP Address	The IP address of the selected node.
Status	The operational status of the control node.
Control Node Manager	The operational status of the control node manager.



Table 55: 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/ 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 190 on page 890](#).

Figure 190: Individual Control Node—Console Tab

Monitor > Infrastructure > Control Nodes > b3s9

Search Sitemap

Details
Console
Peers
Routes

### Console Logs

Time Range

Custom

From Time

Oct 08, 2013 02:26:33 PM

To Time

Oct 08, 2013 02:31:33 PM

Log Category

All

Log Type

any

Log Level

SYS\_DEBUG

Limit

Limit 10 mess

Auto Refresh

☒

Display Logs

Reset

Time	Category	Log Type	Log
2013-10-08 14:31:30:351:353	BGP	BgpStateMachineSessionMessageLog	Bgp Peer 10.84.23.252 : P fsm::EvConnectTimerExp
2013-10-08 14:31:27:971:482	BGP	BgpStateMachineSessionMessageLog	Bgp Peer 10.84.23.253 : P state Connect
2013-10-08 14:31:24:970:157	BGP	BgpStateMachineSessionMessageLog	Bgp Peer 10.84.23.253 : P fsm::EvConnectTimerExp
2013-10-08 14:30:58:220:866	BGP	BgpStateMachineSessionMessageLog	Bgp Peer 10.84.23.252 : P state Connect

See [Table 56 on page 890](#) for descriptions of the fields on the **Console** tab screen.

Table 56: 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>All</li> <li>_default_</li> <li>XMPP</li> <li>TCP</li> </ol>

Table 56: Control Node: Console Tab Fields *(Continued)*

Field	Description
<b>Log Type</b>	Select a log type to display.
<b>Log Level</b>	<p>Select a log severity level to display:</p> <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>	<p>Select from a list an amount to limit the number of messages displayed:</p> <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 56: 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 191 on page 893](#).

Figure 191: Individual Control Node—Peers Tab

Monitor > Infrastructure > Control Nodes > b3s9

Search Sitemap

Details
Console
Peers
Routes

Peers

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: - {

```

See Table 57 on page 893 for descriptions of the fields on the **Peers** tab screen.

Table 57: 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 57: 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 192 on page 894](#).

**Figure 192: Individual Control Node—Routes Tab**

Details

Console

Peers

Routes

Routing Instance

All

Address Family

All

Limit 50 Routes

Peer Source

All

Prefix

Prefix

Protocol

All

Display Routes

Reset

Routes

Routing Table

Prefix

Protocol

Source

Next hop

Label

Secur...

Origin VN

bgp.l3vpn.0

10.84.21.1:13:192.168.30.240/32

XMPP

b1s1

10.84.21.1

28

3

default-domain:demo.v n30

BGP

10.84.23.9

10.84.21.1

28

3

default-domain:demo.v n30

10.84.21.1:14:192.168.31.242/32

XMPP

b1s1

10.84.21.1

29

3

default-domain:demo.v n31

BGP

10.84.23.9

10.84.21.1

29

3

default-domain:demo.v n31

10.84.21.1:1:192.168.2.231/32

XMPP

b1s1

10.84.21.1

16

3

default-domain:demo.v n2

See [Table 58 on page 895](#) for descriptions of the fields on the **Routes** tab screen.

Table 58: 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>	<p>Select an address family for which to display the active routes:</p> <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)	<p>Select to limit the display of active routes:</p> <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>	<p>Select a protocol for which to display the active routes:</p> <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 58: 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

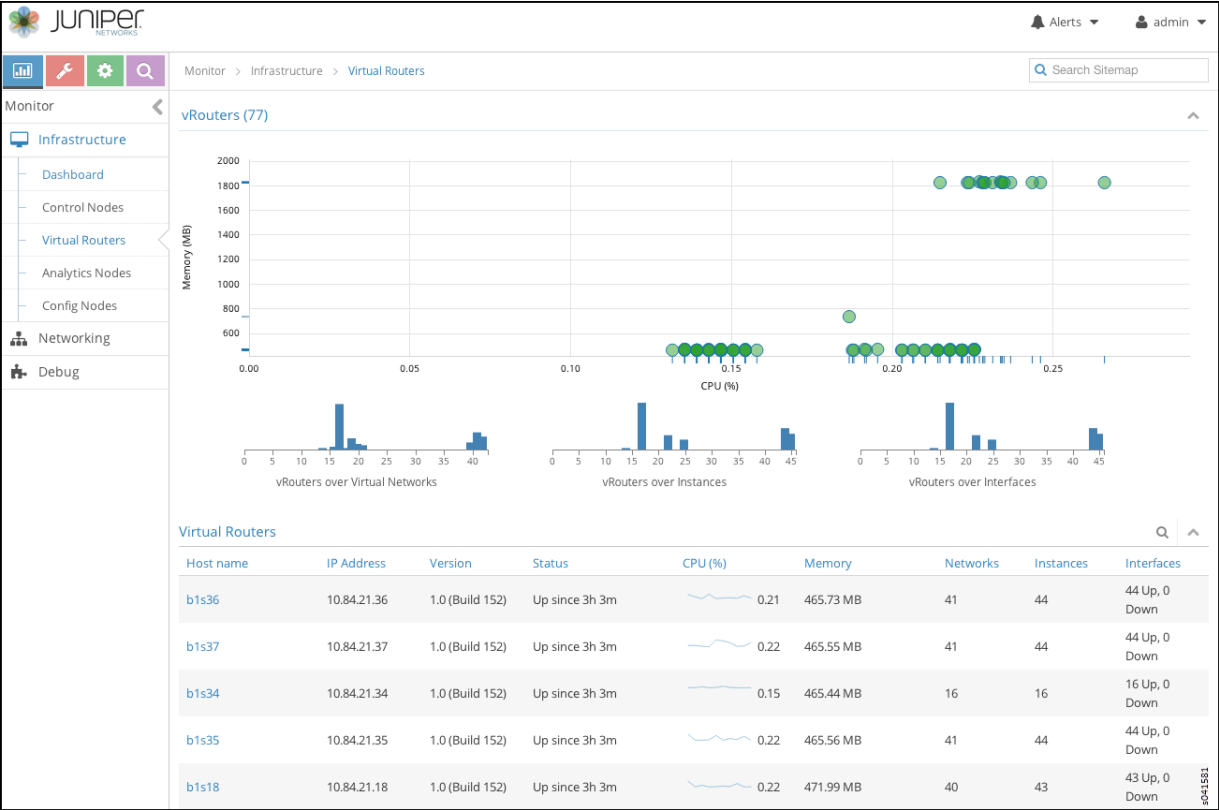
### IN THIS SECTION

- [Monitor vRouters Summary | 897](#)
- [Monitor Individual vRouters Tabs | 899](#)
- [Monitor Individual vRouter Details Tab | 899](#)
- [Monitor Individual vRouters Interfaces Tab | 901](#)
- [Monitor Individual vRouters Networks Tab | 903](#)
- [Monitor Individual vRouters ACL Tab | 904](#)
- [Monitor Individual vRouters Flows Tab | 906](#)
- [Monitor Individual vRouters Routes Tab | 907](#)
- [Monitor Individual vRouter Console Tab | 908](#)

### Monitor vRouters Summary

Click **Monitor > Infrastructure > Virtual Routers** to view the **vRouters** summary screen. See [Figure 193 on page 898](#).

Figure 193: vRouters Summary



See [Table 59 on page 898](#) for descriptions of the fields on the **vRouters Summary** screen.

Table 59: vRouters Summary Fields

Field	Description
Host name	The name of the vRouter. Click the name of any vRouter to reveal more details.
IP Address	The IP address of the vRouter.
Version	The version of software installed on the system.
Status	The current operational status of the vRouter – Up or Down.
CPU (%)	The CPU percentage currently in use by the selected vRouter.

Table 59: 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 194 on page 900](#).
- **Console**—similar display of information as on individual control nodes **Console** tab. See [Figure 200 on page 909](#).
- **Interfaces**—details about associated interfaces. See [Figure 195 on page 902](#).
- **Networks**—details about associated networks. See [Figure 196 on page 903](#).
- **ACL**—details about access control lists. See [Figure 197 on page 905](#).
- **Flows**—details about associated traffic flows. See [Figure 198 on page 906](#).
- **Routes**—details about associated routes. See [Figure 199 on page 908](#).

## 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 194 on page 900](#). See [Table 60 on page 900](#) for descriptions of the fields on this tab.

Figure 194: Individual vRouters—Details Tab

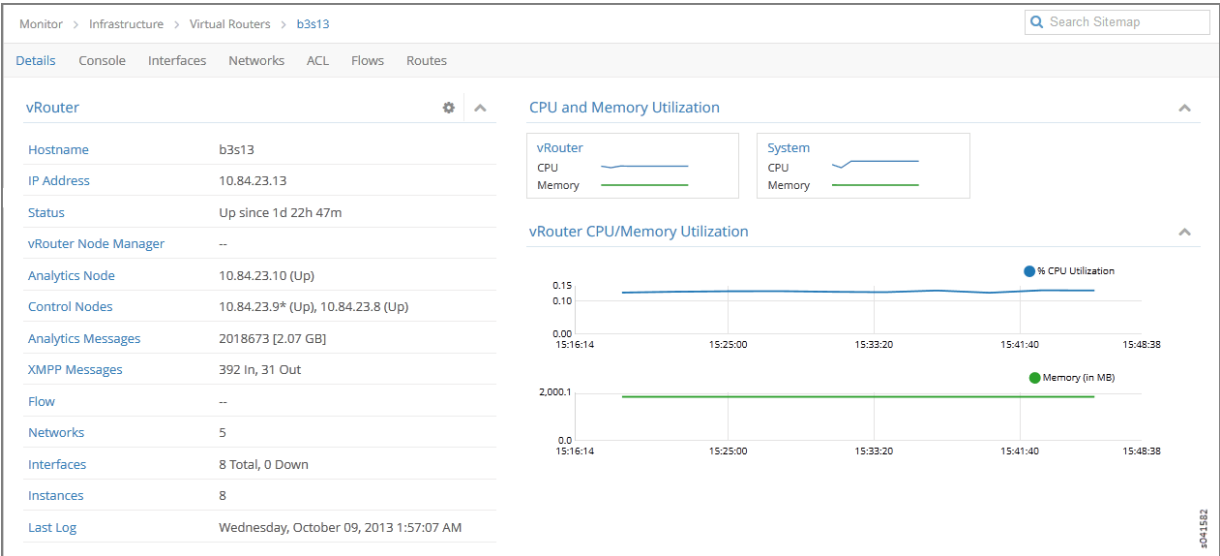


Table 60: vRouters Details Tab Fields

Field	Description
Hostname	The hostname of the vRouter.
IP Address	The IP address of the selected vRouter.
Status	The operational status of the vRouter.
vRouter Node Manager	The operational status of the vRouter node manager.
Analytics Node	The IP address of the node from which analytics (monitor) information is derived.
Control Nodes	The IP address of the configuration node associated with this vRouter.
Analytics Messages	The total number of analytics messages in and out from this node.
XMPP Messages	The total number of XMPP messages that have gone in and out of this vRouter.
Flow	The number of active flows and the total flows for this vRouter.

Table 60: 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 195 on page 902](#). See [Table 61 on page 902](#) for descriptions of the fields on the **Interfaces** tab screen.

Figure 195: Individual vRouters—Interfaces Tab

Monitor > Infrastructure > Virtual Routers > b1s36

Q Search Sitemap

Details Console Interfaces Networks ACL Flows Routes

Interfaces

Q ^

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",

#041583

Table 61: vRouters: Interfaces Tab Fields

Field	Description
Name	The name of the interface.
Label	The label for the interface.
Status	The current status of the interface.
Network	The network associated with the interface.
IP Address	The IP address of the interface.
Floating IP	Displays any floating IP addresses associated with the interface.

Table 61: 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 196 on page 903](#). See [Table 62 on page 904](#) for descriptions of the fields on the **Networks** tab screen.

Figure 196: Individual vRouters—Networks Tab

Monitor > Infrastructure > Virtual Routers > b1s36

Search Sitemap

Details Console Interfaces **Networks** ACL Flows Routes

### Networks

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: - {

```

s041584

Table 62: vRouters: Networks Tab Fields

Field	Description
Name	The name of each network associated with this vRouter.
ACLs	The name of the access control list associated with the listed network.
VRF	The identifier of the VRF associated with the listed network.
Action	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 197 on page 905](#). See [Table 63 on page 905](#) for descriptions of the fields on the **ACL** tab screen.



Figure 197: 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 63: vRouters: ACL Tab Fields

Field	Description
UUID	The universal unique identifier (UUID) associated with the listed ACL.
Flows	The flows associated with the listed ACL.
Action	The traffic action defined by the listed ACL.
Protocol	The protocol associated with the listed ACL.
Source Network or Prefix	The name or prefix of the source network associated with the listed ACL.
Source Port	The source port associated with the listed ACL.

**Table 63: 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 198 on page 906](#). See [Table 64 on page 907](#) for descriptions of the fields on the **Flows** tab screen.

**Figure 198: Individual vRouters—Flows Tab**

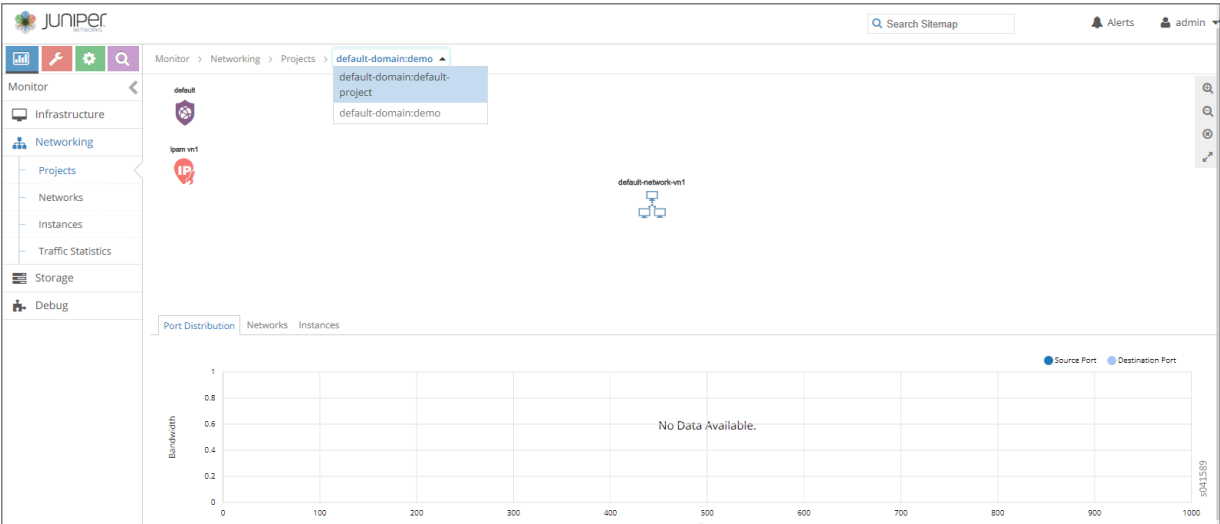


Table 64: vRouters: Flows Tab Fields

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 199 on page 908](#). See [Table 65 on page 908](#) for descriptions of the fields on the **Routes** tab screen.

Figure 199: Individual vRouters—Routes Tab

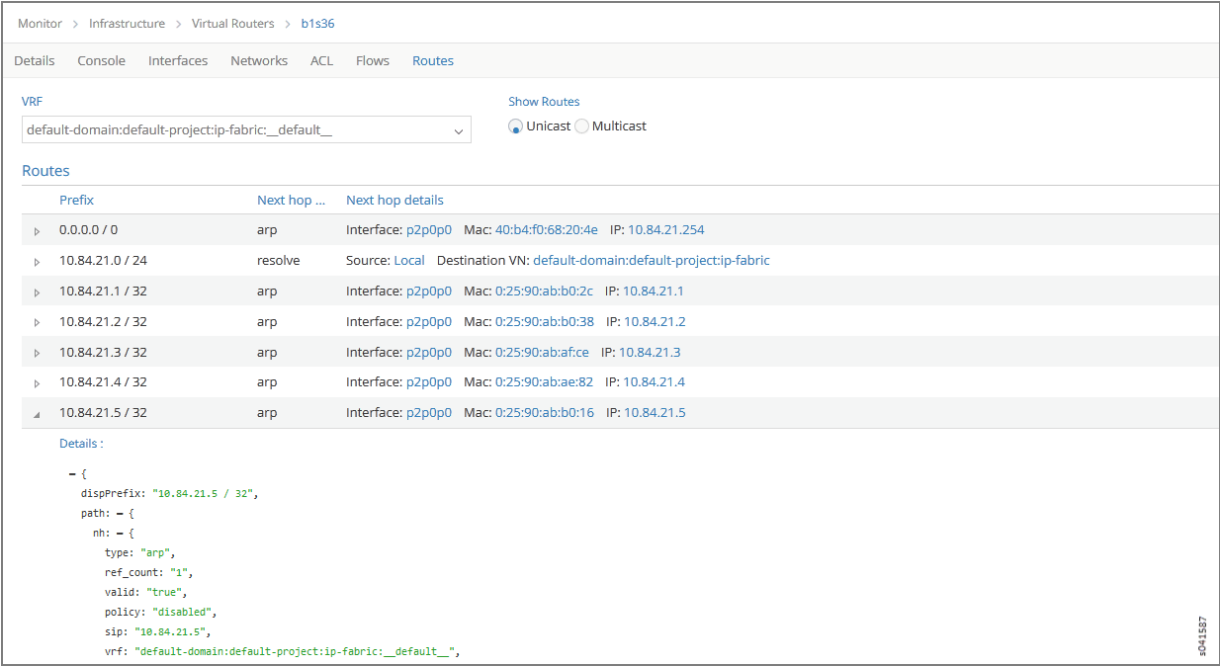


Table 65: vRouters: Routes Tab Fields

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 200 on page 909](#). See [Table 66 on page 909](#) for descriptions of the fields on the **Console** tab screen.

Figure 200: Individual vRouter—Console Tab

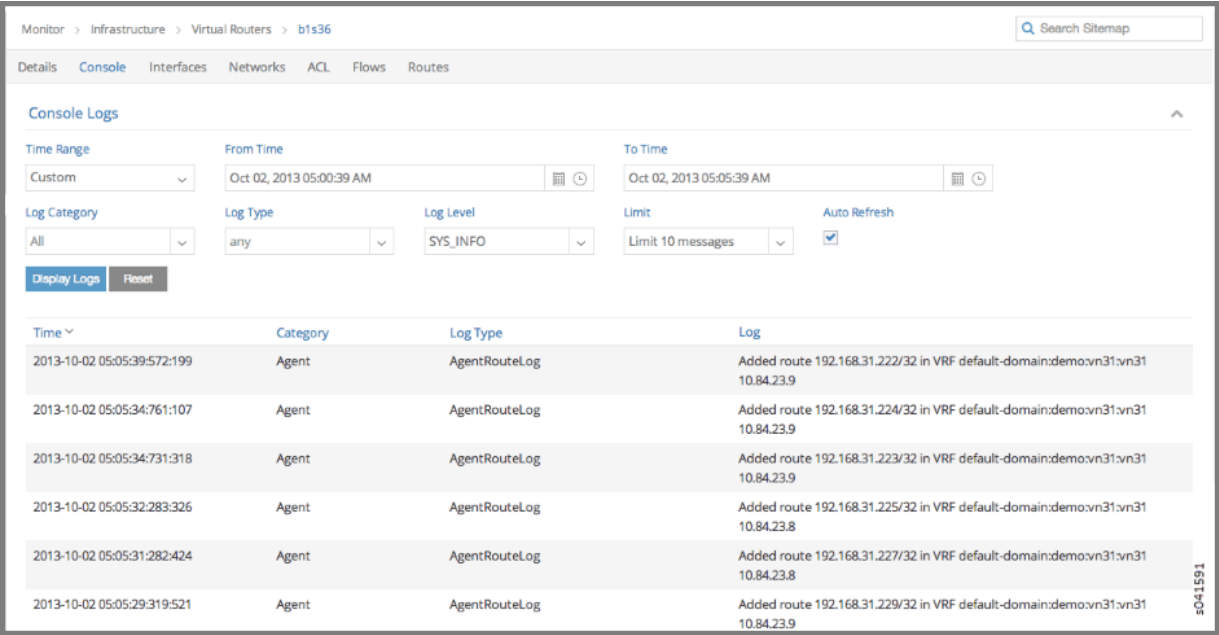


Table 66: Control Node: Console Tab Fields

Field	Description
Time Range	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.
From Time	If you select <b>Custom</b> in <b>Time Range</b> , enter the start time.
To Time	If you select <b>Custom</b> in <b>Time Range</b> , enter the end time.
Log Category	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>
Log Type	Select a log type to display.

Table 66: 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.

*Columns*

**Table 66: 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 | 911](#)
- [Monitor Analytics Individual Node Details Tab | 913](#)
- [Monitor Analytics Individual Node Generators Tab | 914](#)
- [Monitor Analytics Individual Node QE Queries Tab | 915](#)
- [Monitor Analytics Individual Node Console Tab | 916](#)

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 201 on page 912](#). See [Table 67 on page 912](#) for descriptions of the fields on the analytics summary.

Figure 201: Analytics Nodes Summary



Table 67: Fields on Analytics Nodes Summary

Field	Description
Host name	The name of this node.
IP address	The IP address of this node.
Version	The version of software installed on the system.
Status	The current operational status of the node — Up or Down — and the length of time it is in that state.
CPU (%)	The average CPU percentage usage for this node.
Memory	The average memory usage for this node.
Generators	The total number of generators for this node.

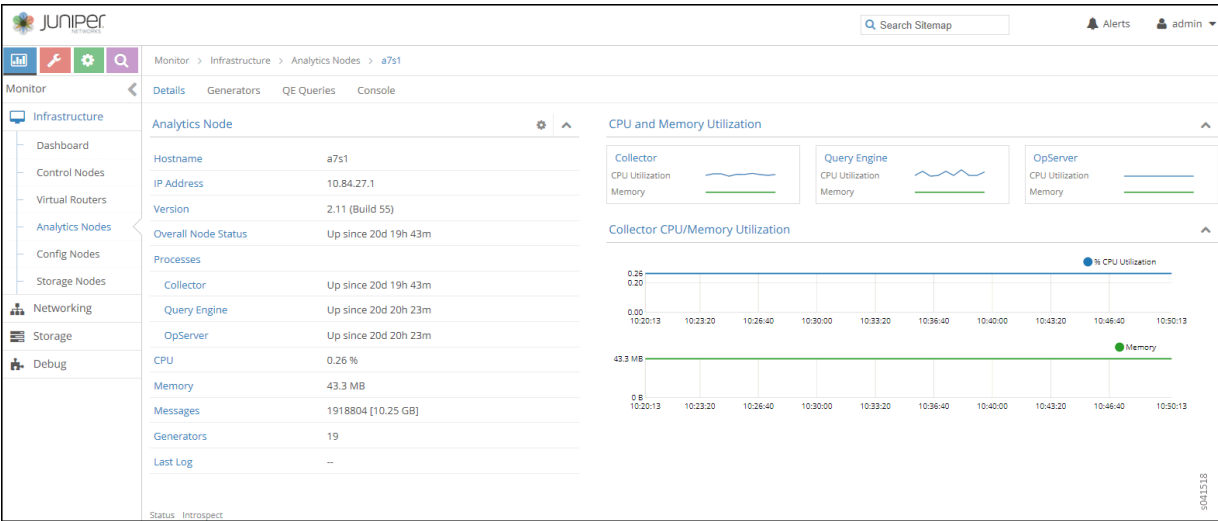


## Monitor Analytics Individual Node Details Tab

Click the name of any analytics node displayed on the analytics summary to view the **Details** tab for that node. See [Figure 202 on page 913](#).

See [Table 68 on page 913](#) for descriptions of the fields on this screen.

**Figure 202: Monitor Analytics Individual Node Details Tab**



**Table 68: 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 68: 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 203 on page 914](#). Click the expansion arrow next to any generator name to reveal more details. See [Table 69 on page 915](#) for descriptions of the fields on the **Peers** tab screen.

Figure 203: 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-collector: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 69: Monitor Analytics Individual Node Generators Tab Fields

Field	Description
Name	The host name of the generator.
Status	The current status of the peer— Up or Down — and the length of time in that state.
Messages	The number of messages sent and received from this peer.
Bytes	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 204 on page 915](#).  
See [Table 70 on page 915](#) for descriptions of the fields on the **QE Queries** tab screen.

Figure 204: Individual Analytics Node—QE QueriesTab



Table 70: Analytics Node QE Queries Tab Fields

Field	Description
Enqueue Time	The length of time this message has been in the queue waiting to be delivered.
Query	The query message.

Table 70: Analytics Node QE Queries Tab Fields *(Continued)*

Field	Description
Progress (%)	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 205 on page 916](#). See [Table 71 on page 916](#) for descriptions of the fields on the **Console** tab screen.

Figure 205: Analytics Individual Node—Console Tab

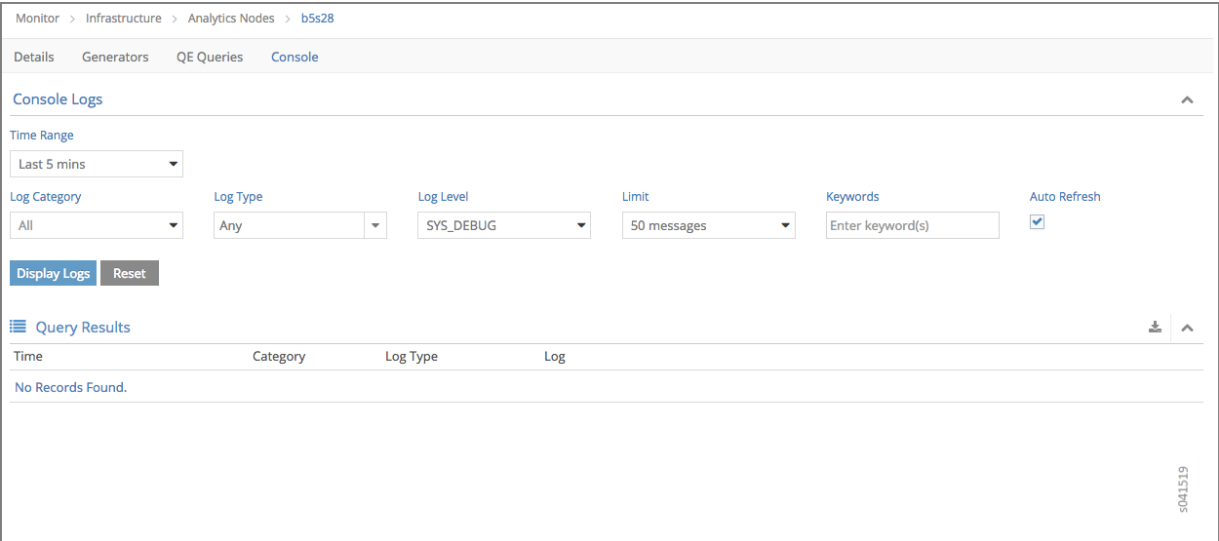


Table 71: Monitor Analytics Individual Node Console Tab Fields

Field	Description
Time Range	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 71: Monitor Analytics Individual Node Console Tab Fields** *(Continued)*

Field	Description
<b>Log Category</b>	<p>Select a log category to display:</p> <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>	<p>Select a log severity level to display:</p> <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 71: Monitor Analytics Individual Node Console Tab Fields *(Continued)*

Field	Description
(Limit field)	<p>Select the number of messages to display:</p> <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

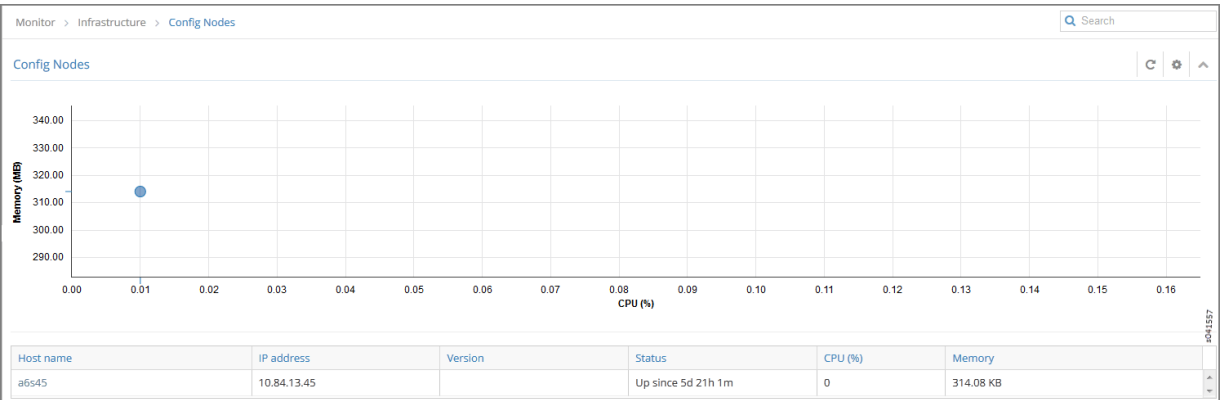
- [Monitor Config Nodes | 919](#)
- [Monitor Individual Config Node Details | 920](#)
- [Monitor Individual Config Node Console | 921](#)

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 206 on page 919](#).

Figure 206: Config Nodes Summary



[Table 72 on page 919](#) describes the fields in the Config Nodes summary.

Table 72: Config Nodes Summary Fields

Field	Description
Host name	The name of this node.

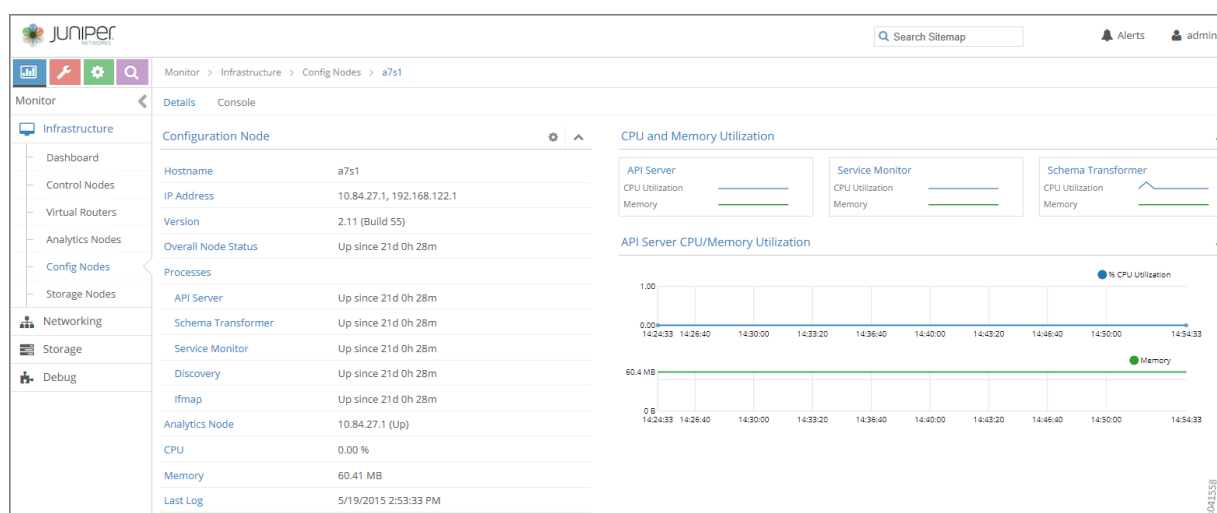
Table 72: Config Nodes Summary Fields *(Continued)*

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 207 on page 920](#).

Figure 207: Individual Config Nodes— Details Tab



[Table 73 on page 921](#) describes the fields on the Details screen.



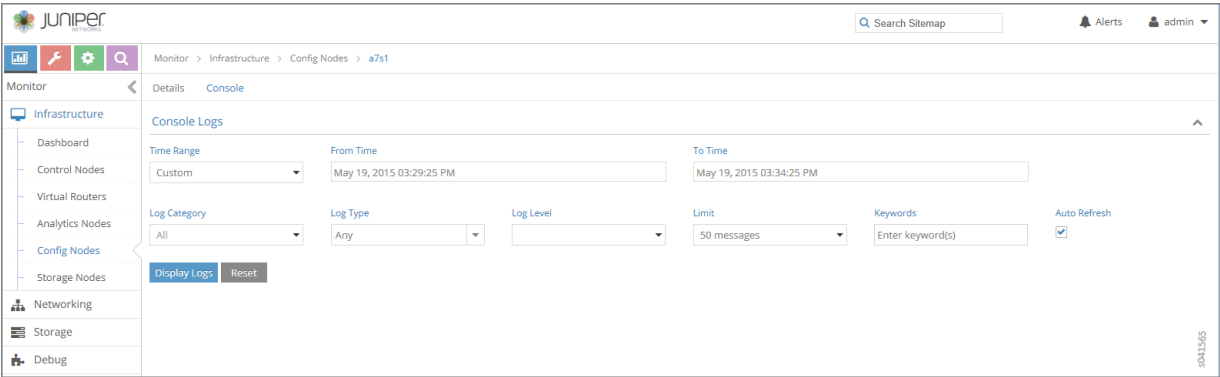
**Table 73: Individual Config Nodes— Details Tab Fields**

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 208 on page 922](#).

Figure 208: Individual Config Node—Console Tab



See [Table 74 on page 922](#) for descriptions of the fields on the **Console** tab screen.

Table 74: 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 74: Individual Config Node-Console Tab Fields *(Continued)*

Field	Description
Keywords	Enter any key words by which to filter the log messages displayed.
Auto Refresh	Click the check box to automatically refresh the display if more messages occur.
Display Logs	Click this button to refresh the display if you change the display criteria.
Reset	Click this button to clear any selected display criteria and reset all criteria to their default settings.

## Monitor > Networking

IN THIS SECTION

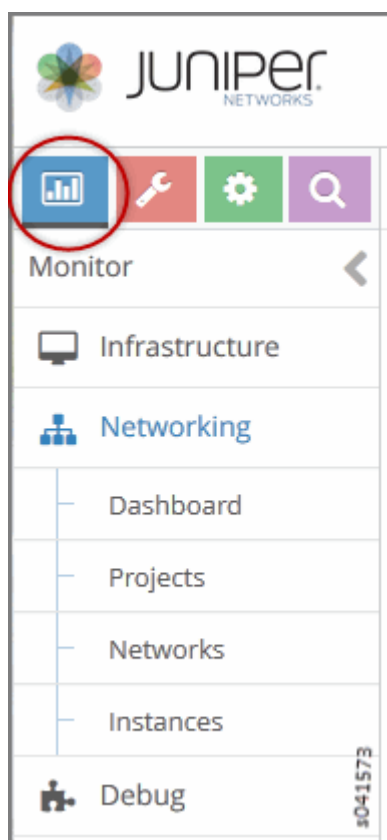
- [Monitor > Networking Menu Options | 923](#)
- [Monitor -> Networking -> Dashboard | 924](#)
- [Monitor > Networking > Projects | 926](#)
- [Monitor Projects Detail | 927](#)
- [Monitor > Networking > Networks | 930](#)

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 209 on page 924](#) shows the menu options available under **Monitor > Networking**.

Figure 209: 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 210 on page 925](#).

Figure 210: Traffic Statistics for Domain Window

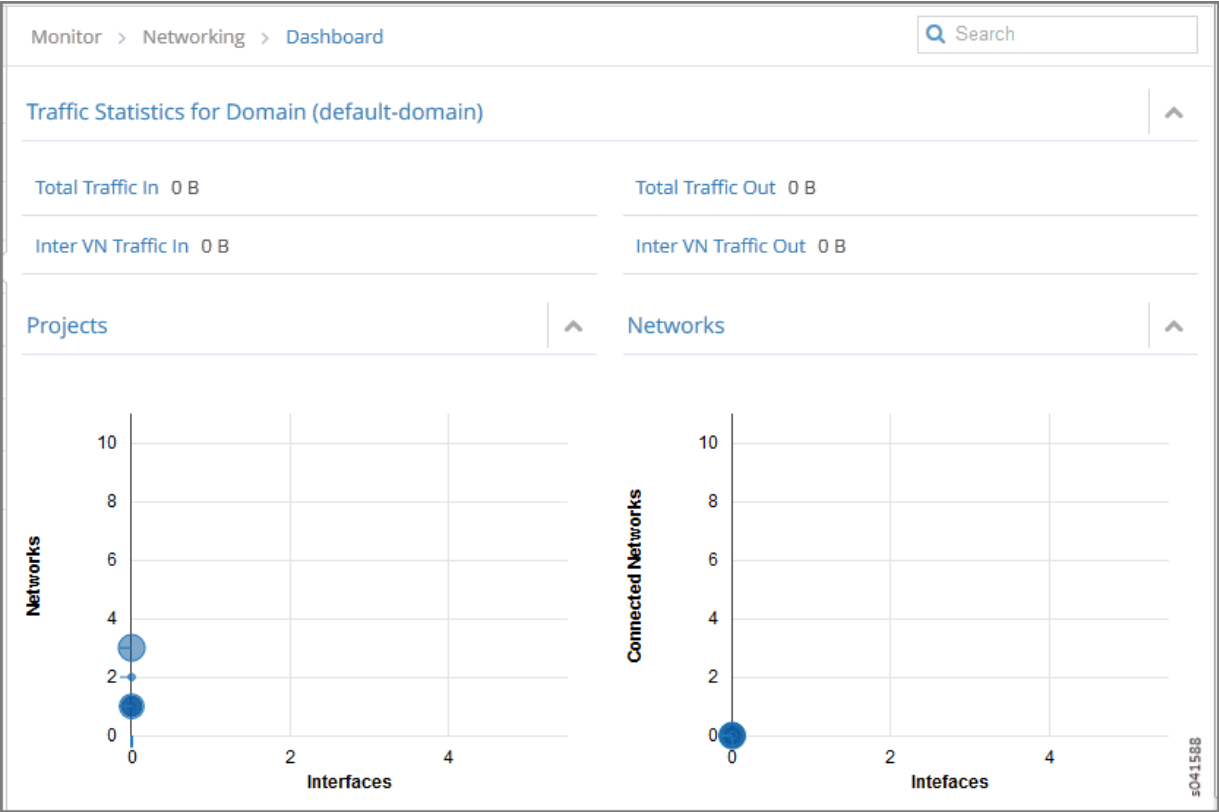


Table 75 on page 925 describes the fields in the Traffic Statistics for Domain window.

Table 75: Projects Summary Fields

Field	Description
Total Traffic In	The volume of traffic into this domain
Total Traffic Out	The volume of traffic out of this domain.
Inter VN Traffic In	The volume of inter-virtual network traffic into this domain.
Inter VN Traffic Out	The volume of inter-virtual network traffic out of this domain.

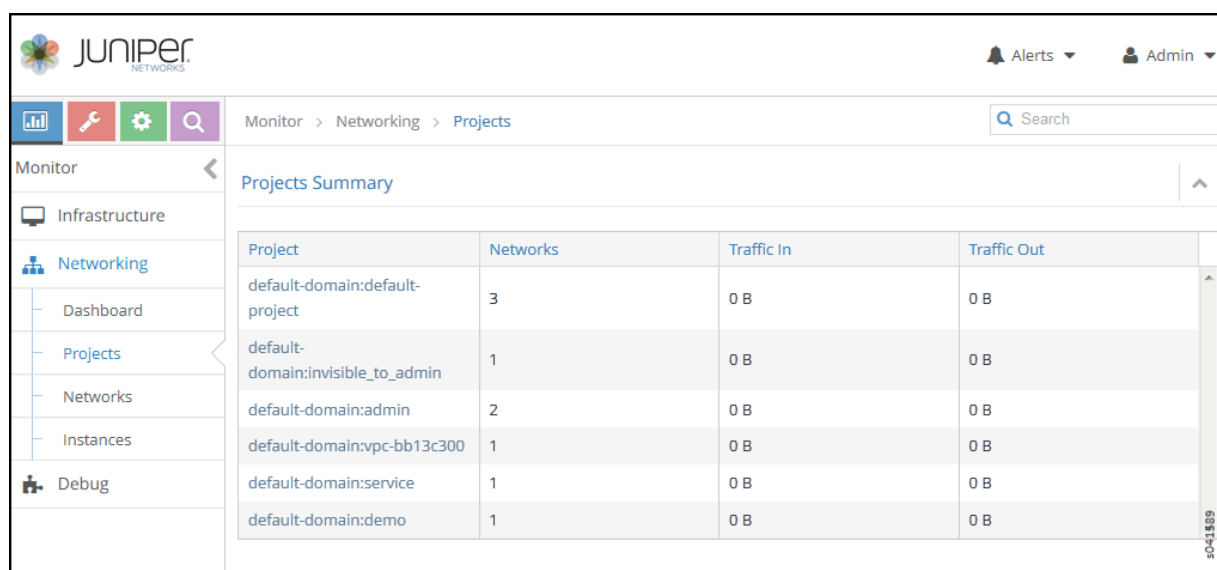
Table 75: Projects Summary Fields *(Continued)*

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 211 on page 926](#).

Figure 211: 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

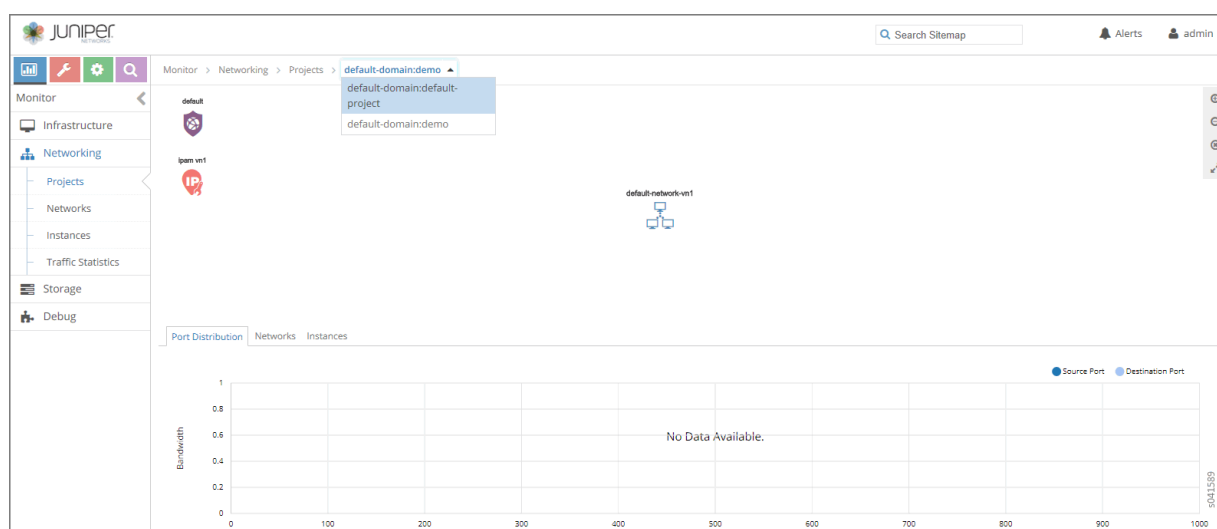
See [Table 76 on page 927](#) for descriptions of the fields on this screen.

**Table 76: 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

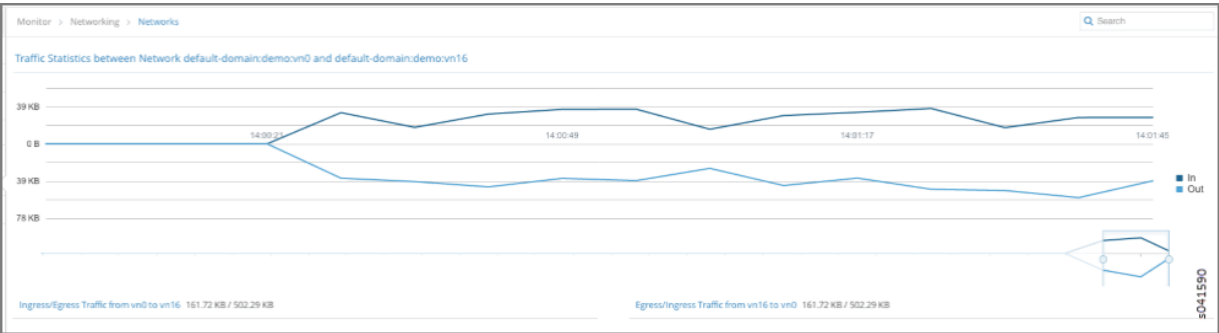
You can click any of the projects listed on the Projects Summary to get details about connectivity, source and destination port distribution, and instances. When you click an individual project, the Summary tab for Connectivity Details is displayed as shown in [Figure 212 on page 927](#). Hover over any of the connections to get more details.

**Figure 212: 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 213 on page 928](#).

**Figure 213: 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 214: Projects Instances Summary**

Monitor > Networking > Projects > default-domain:admin

Summary Instances

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	2.2.2.253 250.250.1.253 (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.

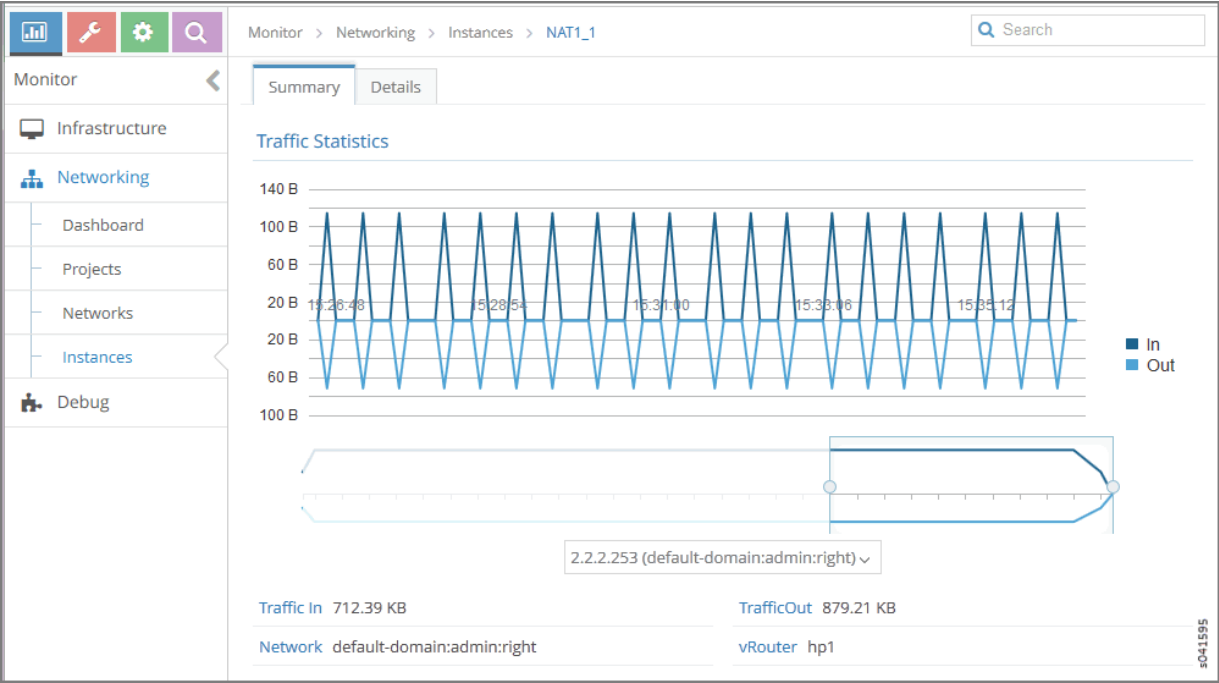


Table 77: Projects Instances Summary Fields

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 215 on page 930](#).

Figure 215: Instance Traffic Statistics



## Monitor > Networking > Networks

Select **Monitor > Networking > Networks** to view a summary of the virtual networks in your system. See [Figure 216 on page 930](#).

Figure 216: Network Summary

Monitor > Networking > Networks

Networks Summary

Network	Instances	Traffic (In/Out) (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

Table 78: 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 217 on page 931 shows the **Summary** tab for an individual network, which displays connectivity details and traffic statistics for the selected network.

Figure 217: Individual Network Connectivity Details—Summary Tab

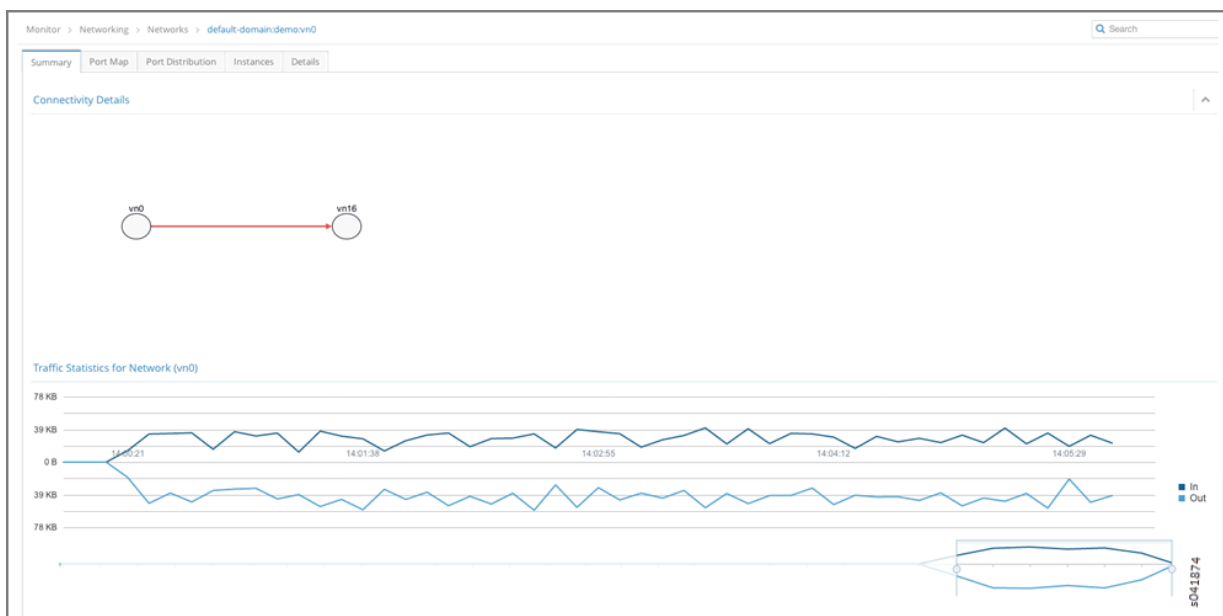


Figure 218 on page 932 shows the **Port Map** tab for an individual network, which displays the relative distribution of traffic for this network by protocol, by port.

**Figure 218: Individual Network-- Port Map Tab**

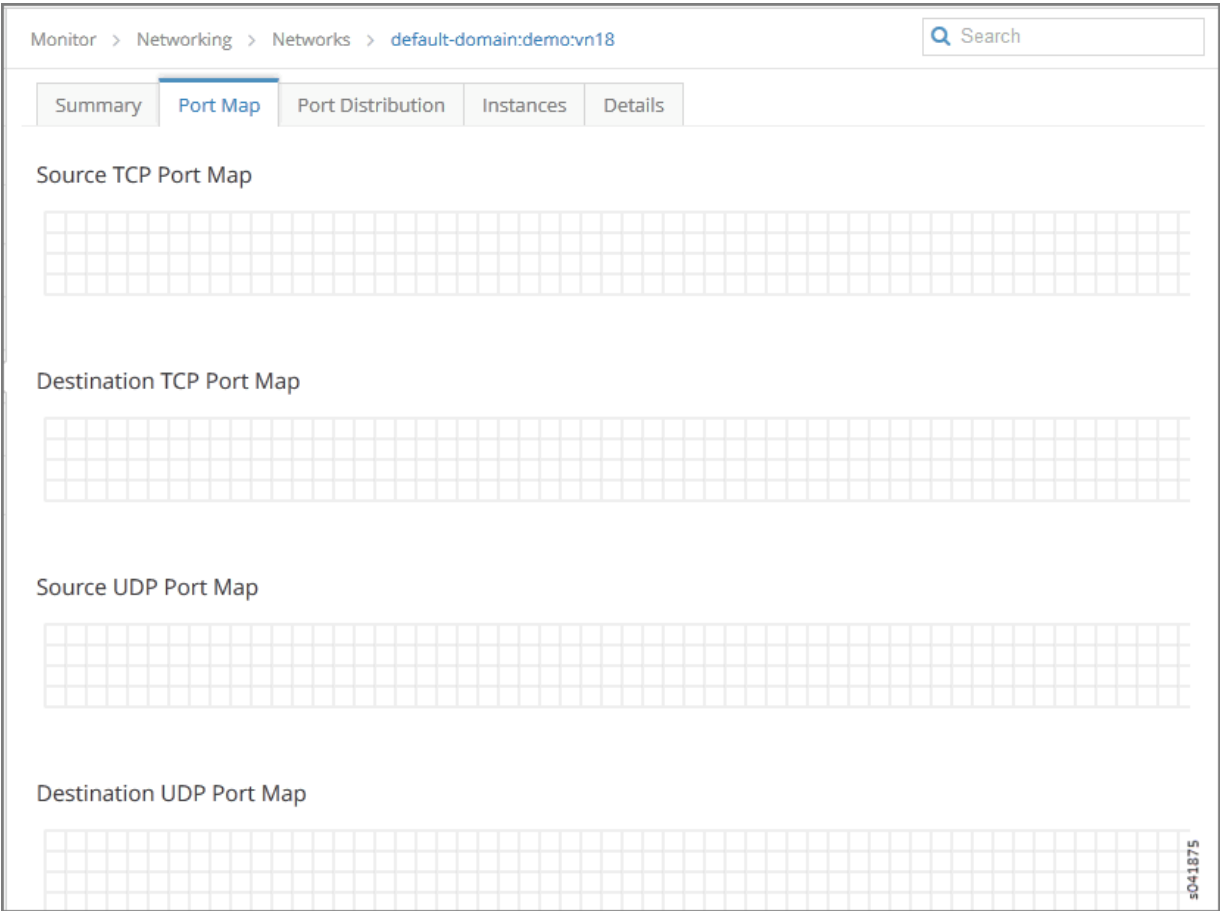


Figure 219 on page 933 shows the **Port Distribution** tab for an individual network, which displays the relative distribution of traffic in and out by source port and destination port.

Figure 219: Individual Network-- Port Distribution Tab

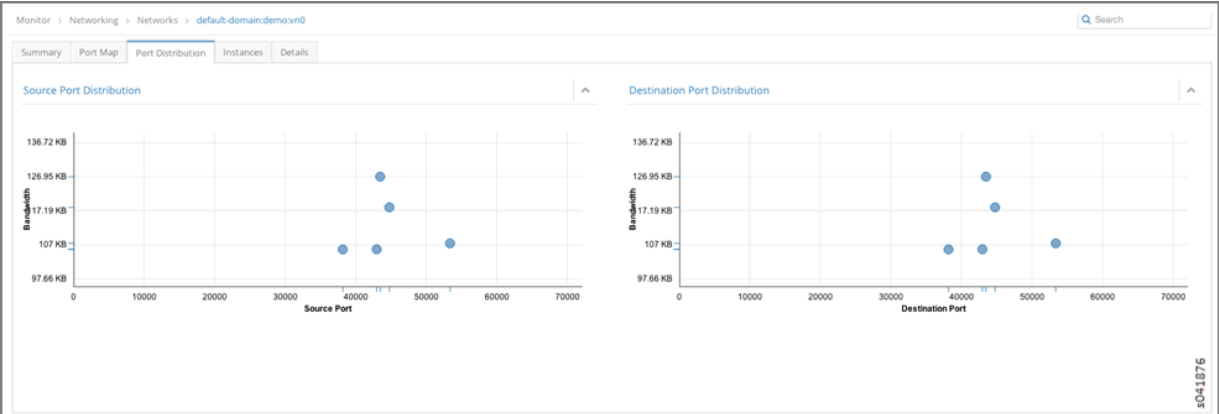


Figure 220 on page 934 shows the **Instances** tab for an individual network, which displays details for each instance associated with this network, including the number of interfaces, the associated vRouter, the instance IP address, and the volume of traffic in and out.

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 220: Individual Network Instances Tab

Monitor > Networking > Networks > default-domain:demo:vn18

Search

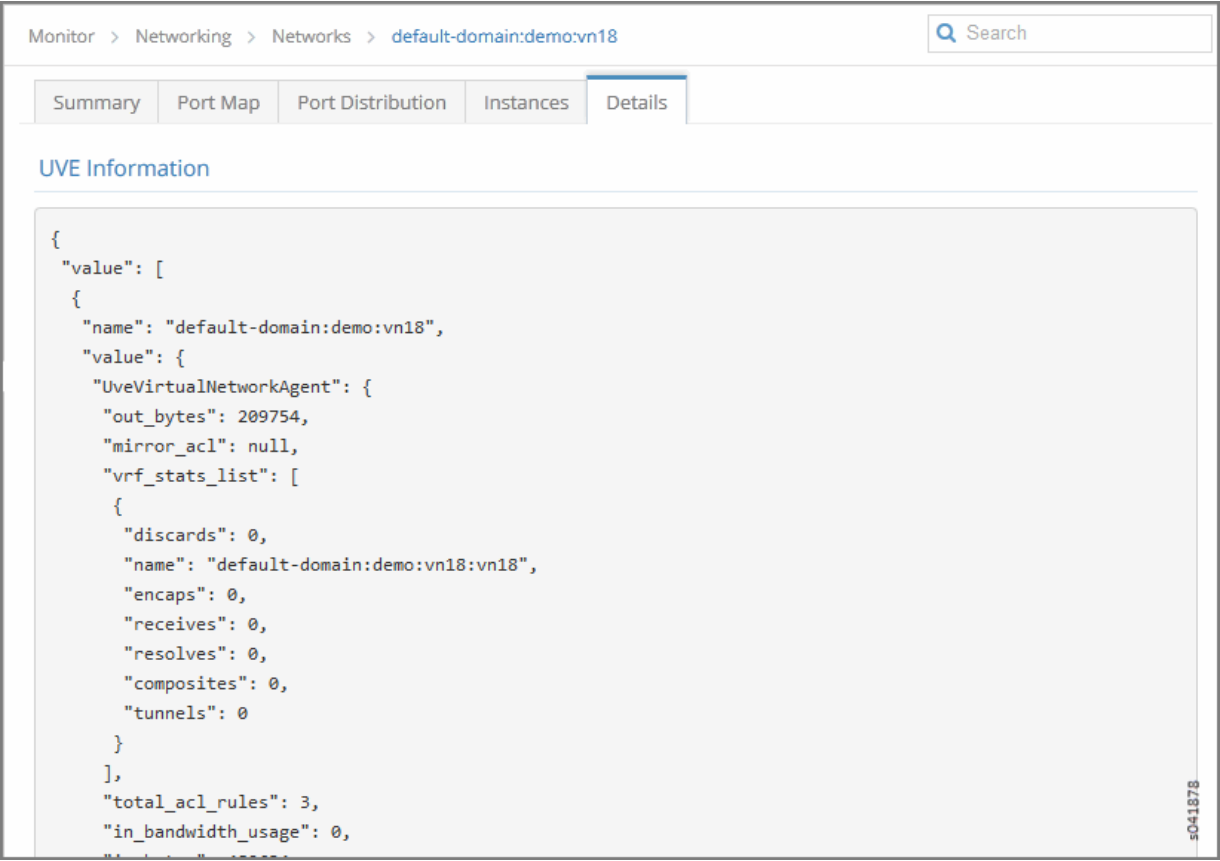
SummaryPort MapPort DistributionInstancesDetails

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	
	<div>InterfacesIP Address: 192.168.18.236Label: 17Mac Address: 02:e9:94:e7:0e:56Network: default-domain:demo:vn18Traffic (In/Out): 1.13 KB/712.00 B</div> <div>UUID22a42bf6-fccc-4db3-b5ac-80082bbebfef</div> <div>CPU0.01</div> <div>Memory1.23 GB / 15.63 GB</div> <div>(Used/Total)</div>						
▸	vn18_vm-f676567a-826f-4e9d-9a81-b4649b7fcde2	1	b1s15	192.168.18.235		1.13 KB / 712.00 B	5041877

Figure 221 on page 935 shows the **Details** tab for an individual network, which displays the code used to define this network --the User Virtual Environment (UVE) code.

Figure 221: Individual Network Details Tab



## Query > Flows

### IN THIS SECTION

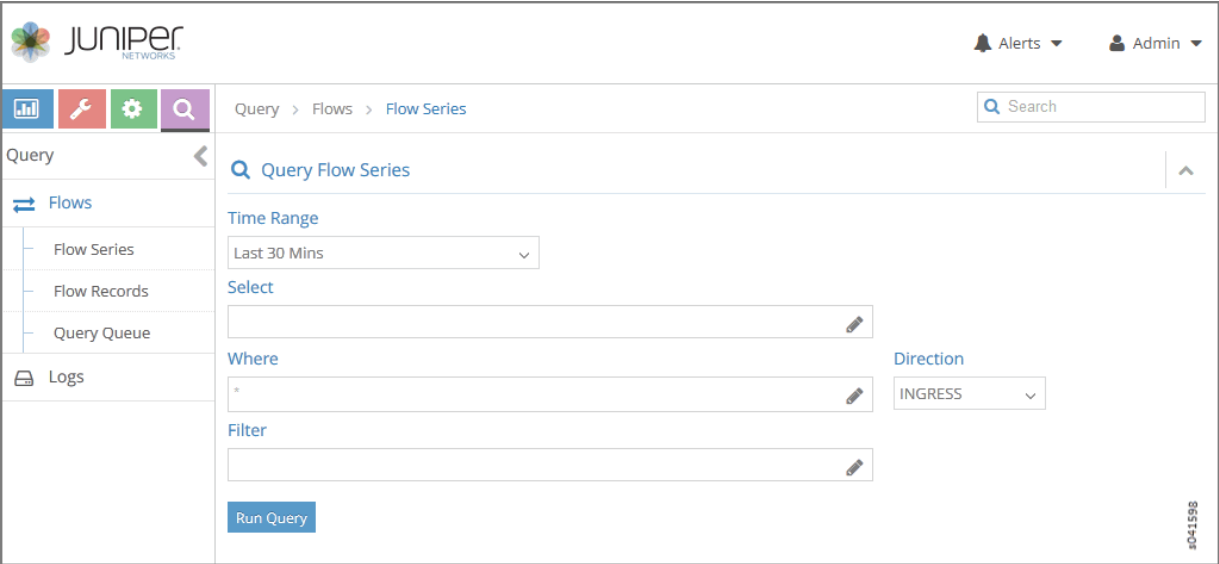
- [Query > Flows > Flow Series | 936](#)
- [Example: Query Flow Series | 939](#)
- [Query > Flow Records | 941](#)
- [Query > Flows > Query Queue | 944](#)

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 222 on page 936](#)

**Figure 222: Query Flow Series Window**



The query fields available on the screen for the **Flow Series** tab are described in [Table 79 on page 937](#). Enter query data into the fields to create a SQL-like query to display and analyze flows.



Table 79: 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>	Click the edit button (pencil icon) to open a <b>Select</b> window (Figure 223 on page 938), where you can click one or more boxes to select the fields to display from the flow series, such as <b>Source VN</b> , <b>Dest VN</b> , <b>Bytes</b> , <b>Packets</b> , and more.
<b>Where</b>	Click the edit button (pencil icon) to open a query-writing window, where you can specify query values for variables such as <b>sourcevn</b> , <b>sourceip</b> , <b>destvn</b> , <b>destip</b> , <b>protocol</b> , <b>sport</b> , <b>dport</b> .
<b>Direction</b>	Select the desired flow direction: <b>INGRESS</b> or <b>EGRESS</b> .
<b>Filter</b>	Click the edit button (pencil icon) to open a <b>Filter</b> window (Figure 224 on page 939), where you can select filter items to sort by, the sort order, and limits to the number of results returned.
<b>Run Query</b>	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.
(graph buttons)	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.

Table 79: Query Flow Series Fields *(Continued)*

Field	Description
Export	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 223 on page 938](#). 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 223: 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 arranged in three columns, each with an unchecked checkbox to its left:

- 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". A vertical scroll bar is visible on the right side of the attribute list.

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 224 on page 939](#).

Figure 224: 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

Limit By

ASC

Cancel

Apply

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 225 on page 939](#).

Figure 225: Example: Query Flow Series

Query Flow Series

Time Range

Last 10 Mins

Select

sourcevn, destvn, time-granularity, sum(bytes)

Where

\*

Filter

Run Query

Time Granularity

10

secs

Direction

INGRESS

The query returns tabular time series data, see [Figure 226 on page 940](#), for the following combinations of Source VN and Dest VN:

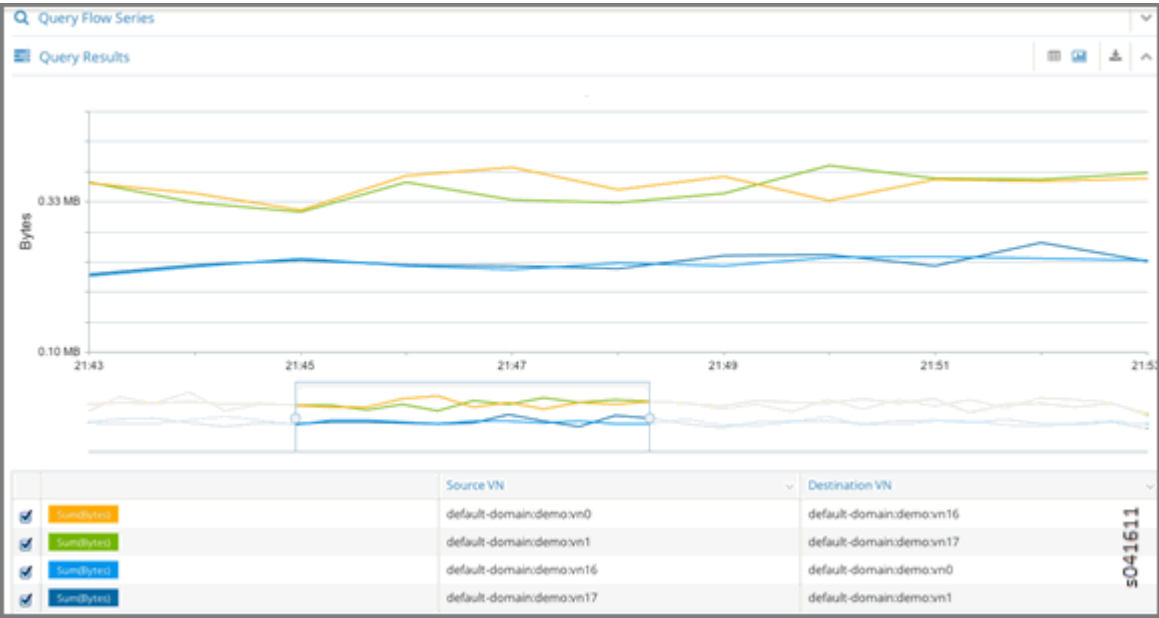
1. Flow Class 1: Source VN = default-domain:demo:front-end, Dest VN=\_\_UNKNOWN\_\_
2. Flow Class 2: Source VN = default-domain:demo:front-end, Dest VN=default-domain:demo:back-end

**Figure 226: Query Flow Series Tabular Results**

Query Flow Series					
Query 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 227 on page 941](#).

Figure 227: 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 228: Flow Records

The figure shows the 'Query > Flows > Flow Records' interface. It includes a search bar, a 'Query Flow Records' title, and several input fields for creating a query. The 'Time Range' is set to 'Last 10 Mins'. The 'Select' field is empty. The 'Where' field contains an asterisk (\*). The 'Direction' is set to 'INGRESS'. A 'Run Query' button is at the bottom.

Query > Flows > Flow Records

Search

Query Flow Records

Time Range  
Last 10 Mins

Select  
[Empty field]

Where  
\*

Direction  
INGRESS

Run Query

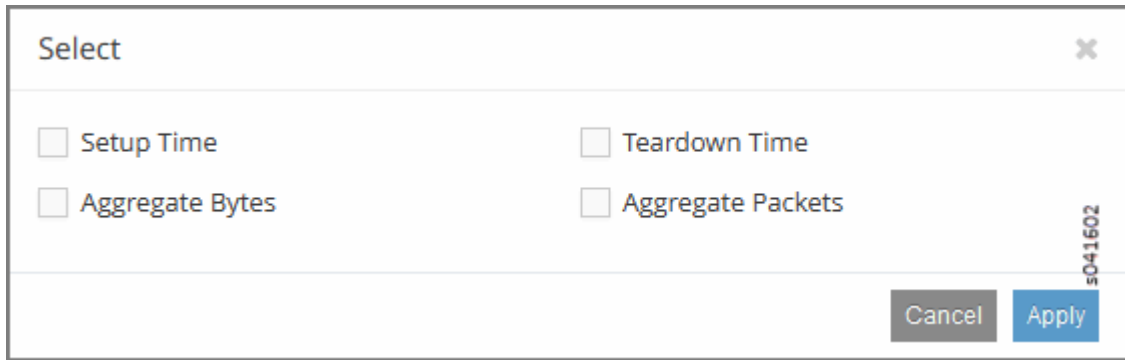
The query fields available on the screen for the **Flow Records** tab are described in [Table 80 on page 942](#). Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 80: 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>	Click the edit button (pencil icon) to open a <b>Select</b> window ( <a href="#">Figure 229 on page 943</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> .
<b>Where</b>	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> .
<b>Direction</b>	Select the desired flow direction: <b>INGRESS</b> or <b>EGRESS</b> .
<b>Run Query</b>	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.
<b>Export</b>	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.

The **Select** window allows you to select one or more attributes to display for the flow records selected, see [Figure 229 on page 943](#).

Figure 229: Flow Records Select Window



The image shows a 'Select' dialog window with a title bar containing a close button (X). The window has a light gray background and a thin border. Inside, there are four checkboxes arranged in a 2x2 grid. The first row contains 'Setup Time' and 'Teardown Time'. The second row contains 'Aggregate Bytes' and 'Aggregate Packets'. All checkboxes are currently unchecked. At the bottom right of the window, there are two buttons: 'Cancel' (gray) and 'Apply' (blue). A small, vertical, rotated text '5041602' is visible on the right side of the window, near the bottom right corner.

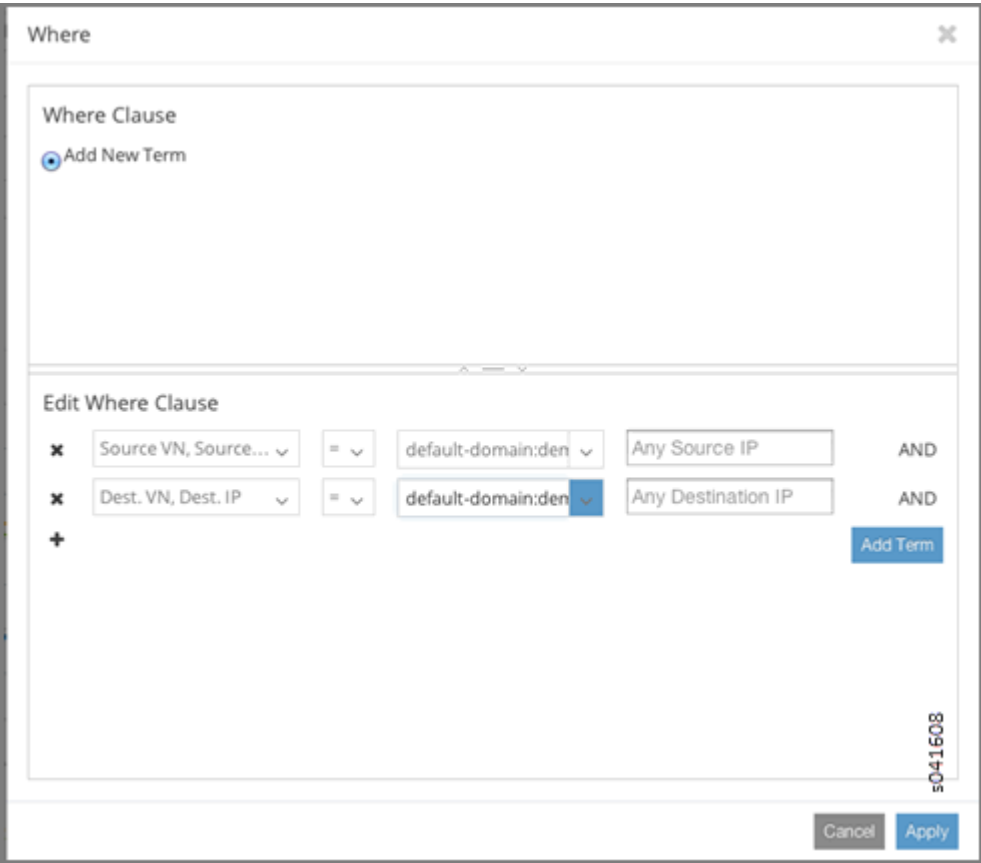
Select	
<input type="checkbox"/> Setup Time	<input type="checkbox"/> Teardown Time
<input type="checkbox"/> Aggregate Bytes	<input type="checkbox"/> Aggregate Packets
<div>Cancel Apply</div>	

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 230: 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 231 on page 944](#).

Figure 231: Flows Query Queue

Query > Flows > Query Queue							Search Sitemap
Flow 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		
1 2 3 4 5 6 >							Displaying 1 - 6 of 31 Records



The query fields available on the screen for the **Flow Records** tab are described in [Table 81 on page 945](#). Enter query data into the fields to create an SQL-like query to display and analyze flows.

**Table 81: 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](#)

**Query > Logs**

IN THIS SECTION

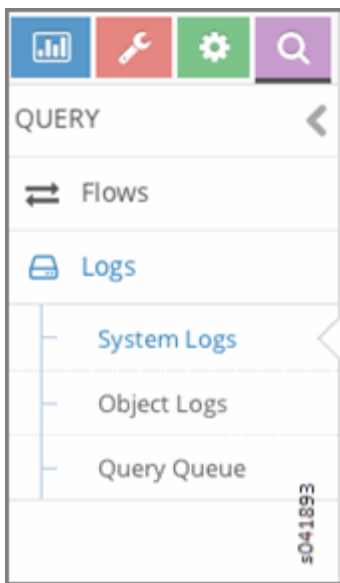
- [Query > Logs Menu Options | 946](#)
- [Query > Logs > System Logs | 946](#)
- [Sample Query for System Logs | 948](#)

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 232 on page 946](#).

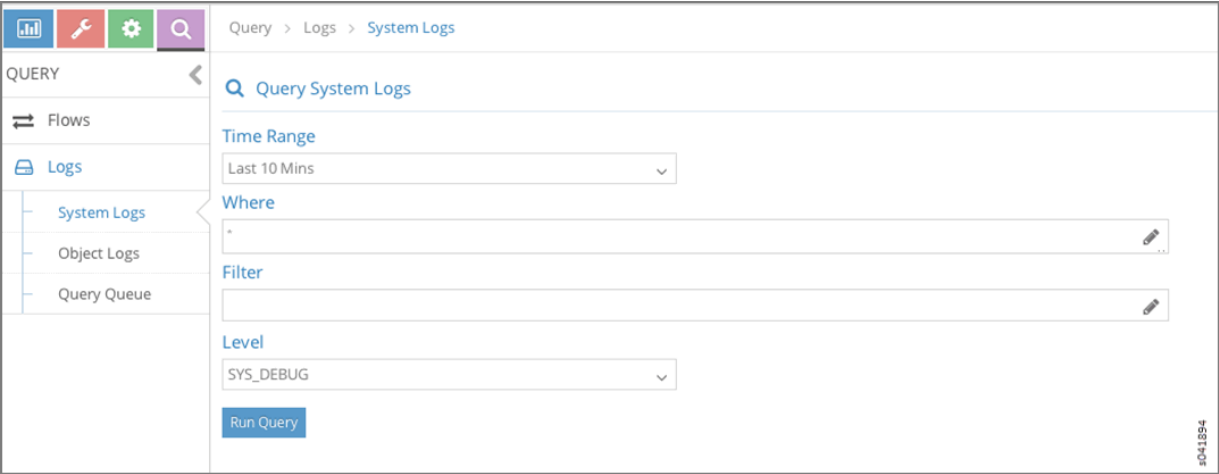
**Figure 232: Query > Logs**



### 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 233 on page 947](#).

Figure 233: Query > Logs > System Logs



The query fields available on the **Query System Logs** screen are described in [Table 82 on page 947](#).

Table 82: Query System Logs Fields

Field	Description
<b>Time Range</b>	Select a range of time for which to see the system logs: <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> If you click Custom, enter a desired time range in two new fields: <b>From Time</b> and <b>To Time</b> .
<b>Where</b>	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.

Table 82: Query System Logs Fields *(Continued)*

Field	Description
<b>Level</b>	<p>Select the message severity level to view:</p> <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 234 on page 949](#).

Figure 234: Edit Where Clause

The screenshot shows a window titled "Where" with a close button (X) in the top right corner. Inside the window, there is a section titled "Where Clause" with a radio button labeled "Add New Term". Below this, there is a section titled "Edit Where Clause". This section contains two rows of input fields. The first row has a dropdown menu with "ModuleId", an equals sign, a dropdown menu with "=", a text input field with "VRouterAgent", and a dropdown menu with "AND". The second row has a dropdown menu with "Source", an equals sign, a dropdown menu with "=", a text input field with "b1s16", and a dropdown menu with "AND". To the left of each row is a small "x" icon, and to the right of each row is a small "+" icon. At the bottom right of the "Edit Where Clause" section is a blue button labeled "Add Term". At the bottom of the window are two buttons: "OK" and "Cancel".

2. The information you defined at the Where screen displays on the **Query System Logs**. Enter any more defining information needed; see [Figure 235 on page 950](#). When finished, click **Run Query** to display the results.

Figure 235: 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 236 on page 950](#).

Figure 236: Query > Logs > 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 83 on page 951](#).

Table 83: 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>	Select from a list of available identifiers the name of the object you wish to use.
<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 83: 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 | 952](#)
- [Flow Handling | 953](#)
- [Flow Aging | 954](#)
- [TCP State-Based Flow Handling and Aging | 954](#)

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 (bytes during the interval) / (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. Additionally, starting with Contrail Release 5.0, the hash tuple can be reduced further, to three-tuple or two-tuple.

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.

Starting with Contrail Release 5.0, fat flow can also be configured at the virtual network (VN) level. When configured at the VN level, the fat flow configuration is applied to all VMIs under the configured VN.

Also starting with Contrail Release 5.0, fat flow is enhanced with the option to support aggregation of multiple flows into a single flow by ignoring source and destination ports or IP addresses, with the following possible options:

- ignore source and/or destination ports
- ignore source and/or destination IP addresses
- ignore a combination of source and/or destination ports and IP addresses

## Configuring Fat Flows

The user creates a `fat-flow-protocol` object, with source and destination options configured, and associates it at the VMI or VN level.

In schema, the ProtocolType object fat-flow-protocol is used to configure if IP addresses or ports are to be ignored, where the value 0 indicates both source and destination ports are to be ignored. Use the virtual-network-fat-flow-protocols for the fat flow object to configure the fat flow at the VN level.

In the Contrail Web UI, to configure fat flow source and destination options at the port, go to **Configure > Networking > Ports**, and edit a port. Select the **Fat-Flows** section and edit the fat flow configuration in the **Ignore Address** field, where the options are **None**, **Source**, or **Destination**.

To configure fat flow at the VN level, go to **Configure > Networking > Networks**, and select **Edit > Fat Flows**. where you can edit ports and IP addresses to be ignored by the selected network.

## Use Case Example

Consider a simple use case in which traffic from different sources of VN1 goes to the Internet via a service instance (SI). Because each source in VN1 can visit different sites in the Internet, the user might want to ignore all the Internet addresses in the flows.

The following fat flow configurations can achieve this:

Case 1—Source, SI, and destination are in different computes

- Left interface of SI—Ignore source
- Right interface of SI—Ignore destination

Case 2—Source and SI are in same compute, but destination is in different compute

- Source interface—Ignore destination
- Left interface of SI—Ignore source
- Right Interface of SI—Ignore destination

Case 3—Source is in different compute, but SI and destination are in same compute

- LeftInterface of SI—Ignore source
- Right interface of SI—Ignore destination
- Destination interface—Ignore source

## Fat Flow Limitations

Fat flow cannot be used when a VMI is ECMP with multiple instances of an ECMP group running on the same compute node.

Fat flow configuration to ignore source or destination address is not supported for NAT flows. This will result in short flows.

RELATED DOCUMENTATION

| *Query > Flows*

Example: Debugging Connectivity Using Monitoring for Troubleshooting

IN THIS SECTION

- [Using Monitoring to Debug Connectivity | 957](#)

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 237 on page 957](#)

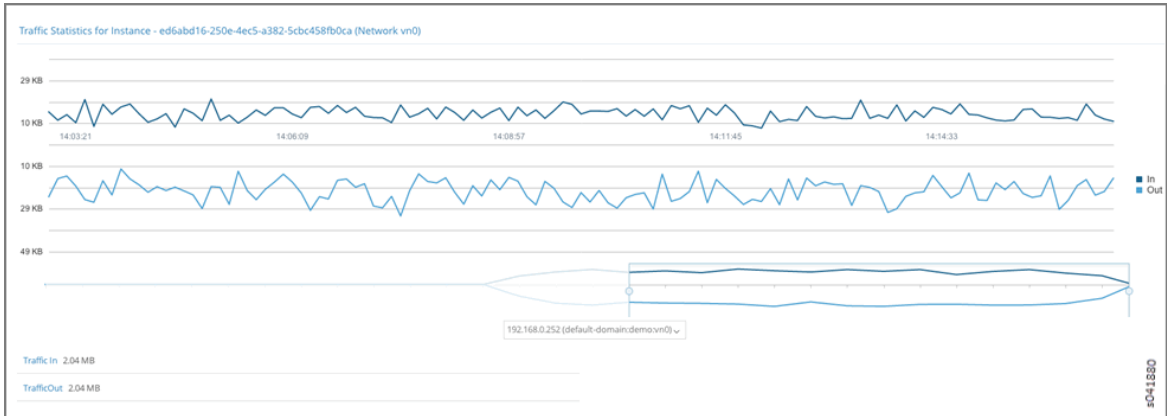
Figure 237: Navigate to Instance



Monitor > Networking > Networks > default-domain:demo:vn0				Search
Summary   Port Map   Port Distribution   <b>Instances</b>   Details				
Instance	Traffic In	Traffic Out		
ed6abd16-250e-4ec5-a382-5cbc458fb0ca	1.73 MB	1.74 MB		
682b7414-c4ba-45ee-91bc-9c22cd6c09d	1.72 MB	1.72 MB		

2. Click the instance to view **Traffic Statistics for Instance**. see [Figure 238 on page 958](#).

Figure 238: Traffic Statistics for Instance



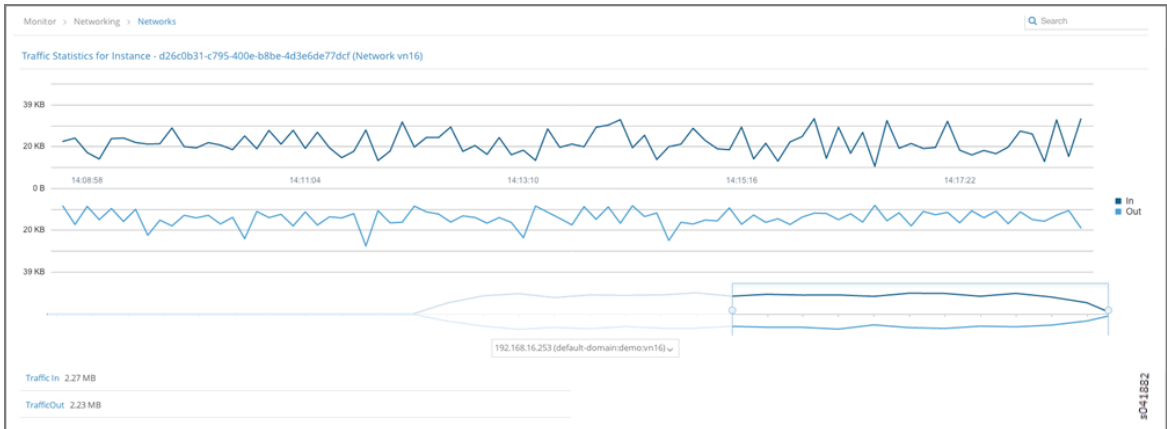
3. Instance d26c0b31-c795-400e-b8be-4d3e6de77dcf with IP address 192.168.0.253 in the virtual network vn16. see [Figure 239 on page 958](#) and [Figure 240 on page 958](#).

Figure 239: Navigate to Instance

Monitor > Networking > Networks > default-domain:demo:vn16

Instance	Traffic In	Traffic Out
d26c0b31-c795-400e-b8be-4d3e6de77dcf	2.18 MB	2.13 MB
23045415-b679-4d3a-8f9d-96c162de28be	2.11 MB	2.16 MB

Figure 240: Traffic Statistics for Instance



4. 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 241 on page 959](#).

Figure 241: Navigate to a3s18 Interfaces

Monitor > Infrastructure > Virtual Routers > a3s18							Search
Details	Console	Interfaces	Networks	ACL	Flows	Routes	
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-a382-5cbc458b0ca	a3s18
tap249de2e1-97	18	Up	default-domain:demo:vn16	192.168.16.252	None	2304515-b679-4d9a-8f9d-96c162de28be	
tap5b3b3d63-74	19	Up	default-domain:demo:vn17	192.168.17.252	None	99311eda-261e-47e8-b4a7-8d126d7499ef	
tapc740843c-6b	17	Up	default-domain:demo:vn1	192.168.1.252	None	20244e9f-a4ed-4a32-803f-15c5323572e	

5. 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 242 on page 959](#).

Figure 242: Navigate to a3s19 Interfaces

Monitor > Infrastructure > Virtual Routers > a3s19							Search
Details	Console	Interfaces	Networks	ACL	Flows	Routes	
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-b8be-4d3e6de77dcf	a3s19
tapb257d21d-d3	18	Up	default-domain:demo:vn1	192.168.1.253	None	eebce321-7536-46e7-a454-cef1f13ac695	
tapc83e3d87-66	17	Up	default-domain:demo:vn17	192.168.17.253	None	b2425f5-6f7e-4060-9478-81a4a825541	
tapc5ea97e3-55	16	Up	default-domain:demo:vn0	192.168.0.253	None	682b7414-c4ba-45ee-91bc-9c22cd86c69d	

6. **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 243 on page 959](#) and [Figure 244 on page 960](#).

Figure 243: ACL Connectivity a3s18

Monitor > Infrastructure > Virtual Routers > a3s18										Search
Details	Console	Interfaces	Networks	ACL	Flows	Routes				
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	a3s18
		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	
b1cd9810-ef9c-41f8-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 244: ACL Connectivity a3s19

Monitor > Infrastructure > Virtual Routers > a3s19									
Details Console Interfaces Networks <b>ACL</b> Flows Routes									
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
b32143a3-0e90-4ae2-9c...	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
b8cf9810-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

7. Next, verify the routes on the control node for routing instances default-domain:demo:vn0:vn0 and default-domain:demo:vn16:vn16; see [Figure 245 on page 960](#) and [Figure 246 on page 961](#).

Figure 245: Routes default-domain:demo:vn0:vn0

Monitor > Infrastructure > Control Nodes > a3s15							
Details Console Peers <b>Routes</b>							
Routing Instance		Address Family		Limit 50 Routes			
Peer Source		Prefix		Display Routes Reset			
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	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	inetmcast	XMPP	a3s19	10.84.17.5	0	100	-



Figure 246: Routes default-domain:demo:vn16:vn16

Monitor > Infrastructure > Control Nodes > a3s15							
<div> <div>Details</div> <div>Console</div> <div>Peers</div> <div>Routes</div> </div>							
<div> <div>Routing Instance</div> <div>default-domain:demo:vn16:vn16</div> </div>		<div> <div>Address Family</div> <div>All</div> </div>		<div> <div>Limit 50 Routes</div> <div>Display Routes</div> <div>Reset</div> </div>			
<div> <div>Peer Source</div> <div>All</div> </div>		<div> <div>Prefix</div> <div>Prefix</div> </div>					
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	-
192.168.0.253/32	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	-
192.168.16.252/32	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	-
192.168.16.253/32	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	-
192.168.16.253/32	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	-
10.84.17.4:2:255.255.255.255,0.0.0.0	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	-
10.84.17.5:2:255.255.255.255,0.0.0.0	inetmcast	XMPP	a3s19	10.84.17.5	0	100	-

8. 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 247 on page 961](#).

Figure 247: 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

504.1839

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 248 on page 962](#).

Figure 248: Verify Route and Next Hop a3s19

Monitor > Infrastructure > Virtual Routers > a3s19		
Details	Console	Interfaces
Networks	ACL	Flows
Routes		
VRF	default-domain:demo:vn16:vn16	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	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

10. 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 249 on page 962](#) and [Figure 250 on page 963](#).

Figure 249: Flows for a3s18

Monitor > Infrastructure > Virtual Routers > a3s18

Details

Console

Interfaces

Networks

ACL

Flows

Routes

Search

Active Flows: 64

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

s041891

Figure 250: Flows for a3s19

Monitor > Infrastructure > Virtual Routers > a3s19

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	1394273/1604	21:00:23.514251 2013-Aug-06

## Common Support Answers

### IN THIS CHAPTER

- [Debugging Ping Failures for Policy-Connected Networks | 964](#)
- [Debugging BGP Peering and Route Exchange in Contrail | 972](#)
- [Troubleshooting the Floating IP Address Pool in Contrail | 990](#)
- [Removing Stale Virtual Machines and Virtual Machine Interfaces | 1019](#)
- [Troubleshooting Link-Local Services in Contrail | 1024](#)

## 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 251: 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=tapb80d9c6a-67](http://nodef1.englab.juniper.net:8085/Snh_ItfReq?name=tapb80d9c6a-67)

Figure 252: 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 253: 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

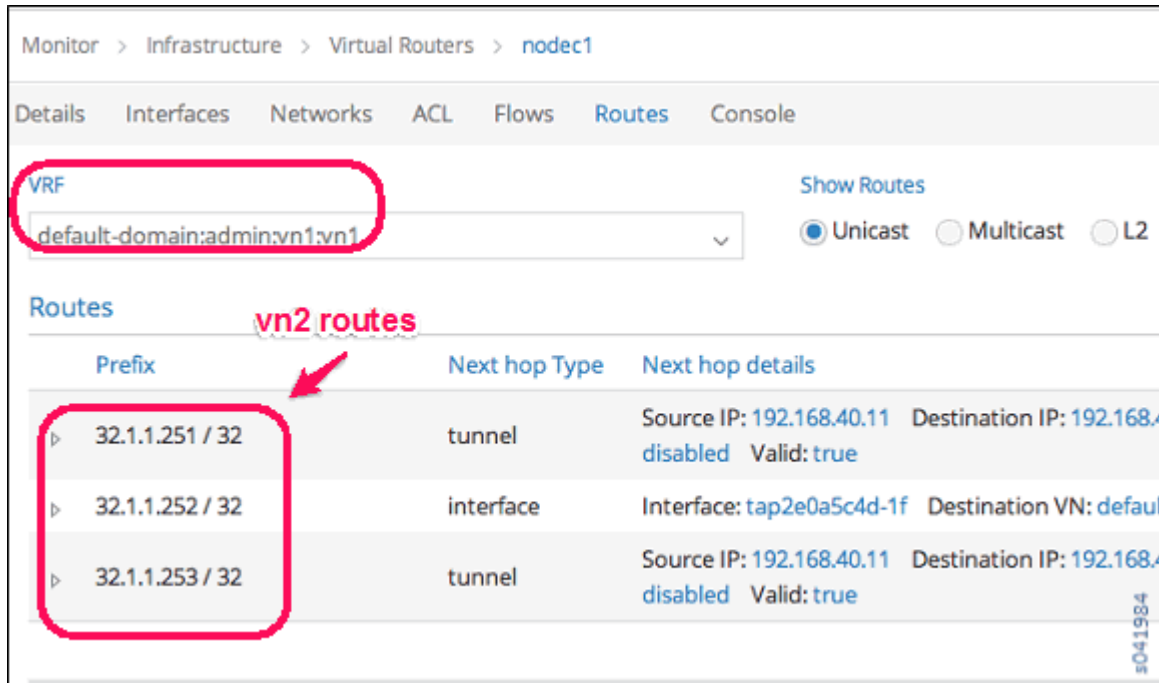
Check the virtual network policy configuration with route information:

Figure 254: Virtual Network Policy Configuration Window

Network	Attached Policies	IP Blocks
vn1	default-analyzer-analyzer-policy vn1-vn2	31.1.1.0/24
vn2	allow_all	32.1.1.0/24

Check the VN1 route information for VN2 routes:

Figure 255: Virtual Network Route Information Window

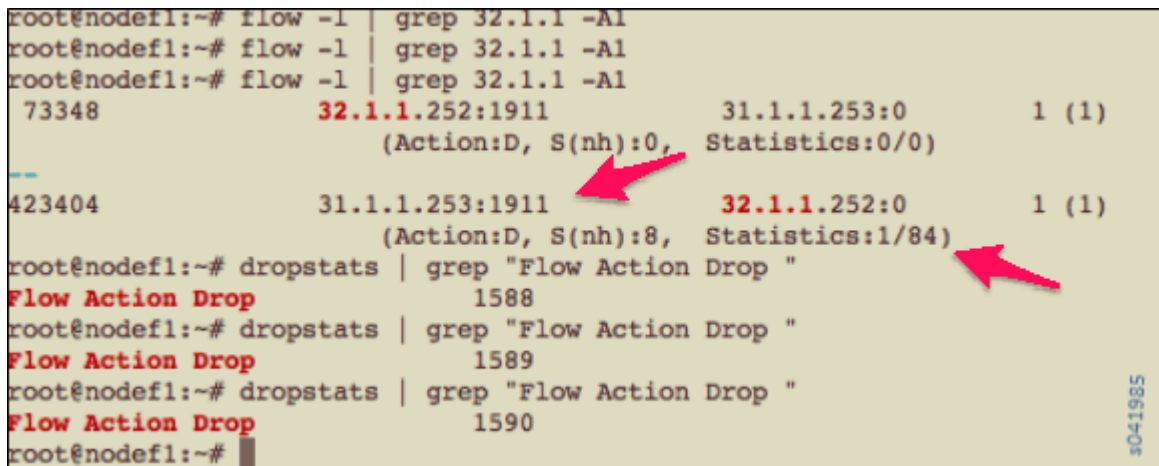


If a route is missing, ping fails. Flow inspection in the compute node displays **Action: D(rop)**.

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 256: Flow and Dropstats Command List



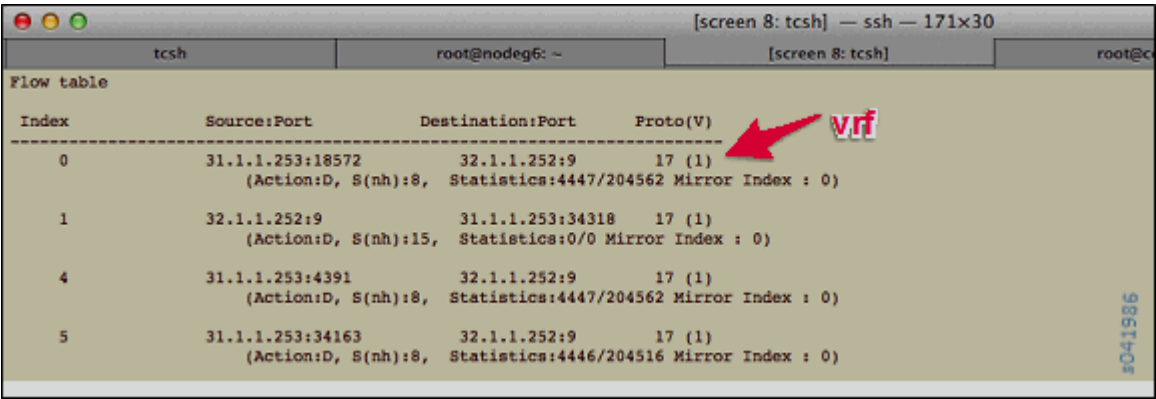
To help in debugging flows, you can use the detailed flow query from the agent introspect page for the compute node.

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 257: Flow Command Output Window



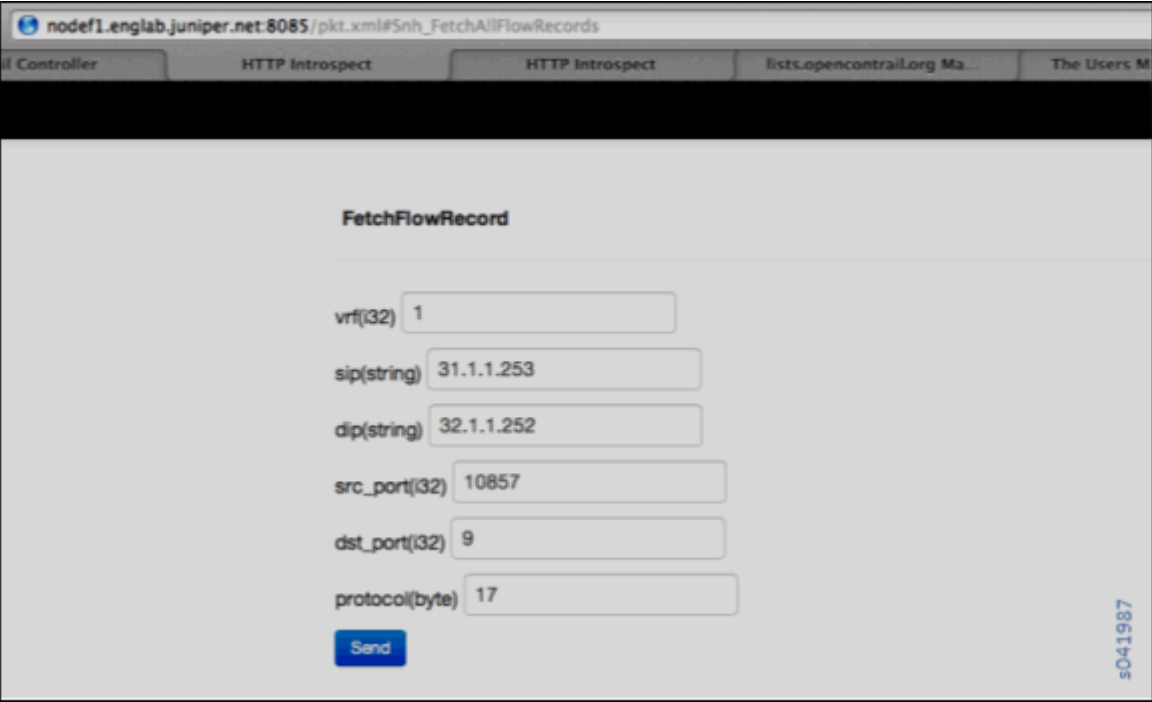
Flow table

Index	Source:Port	Destination:Port	Proto(V)
0	31.1.1.253:18572 (Action:D, S(nh):8,	32.1.1.252:9 Statistics:4447/204562 Mirror Index : 0)	17 (1) <b>vrf</b>
1	32.1.1.252:9 (Action:D, S(nh):15,	31.1.1.253:34318 Statistics:0/0 Mirror Index : 0)	17 (1)
4	31.1.1.253:4391 (Action:D, S(nh):8,	32.1.1.252:9 Statistics:4447/204562 Mirror Index : 0)	17 (1)
5	31.1.1.253:34163 (Action:D, S(nh):8,	32.1.1.252:9 Statistics:4446/204516 Mirror Index : 0)	17 (1)

Fetching details of a single flow:

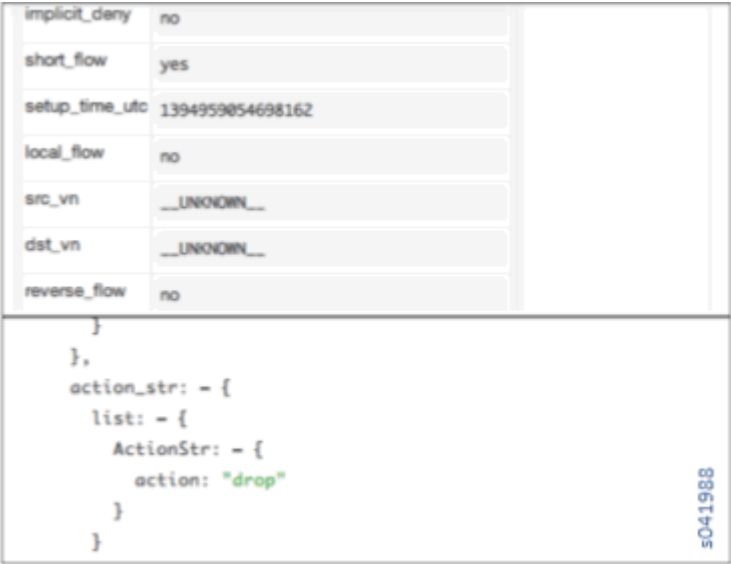


Figure 258: Fetch Flow Record Window



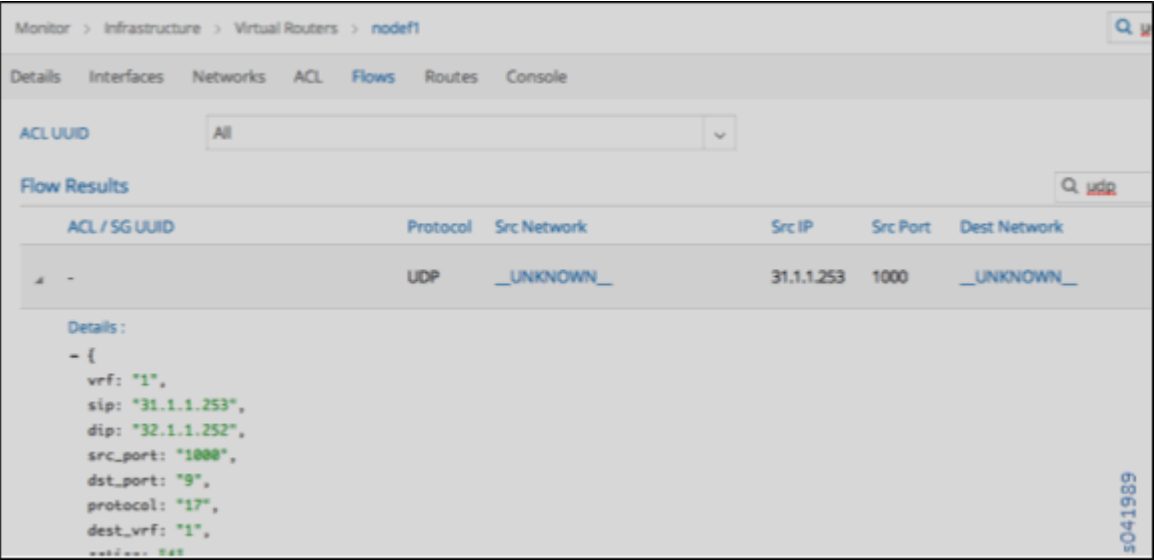
Output from **FetchFlowRecord** shows unresolved IP addresses:

Figure 259: Unresolved IP Address Window



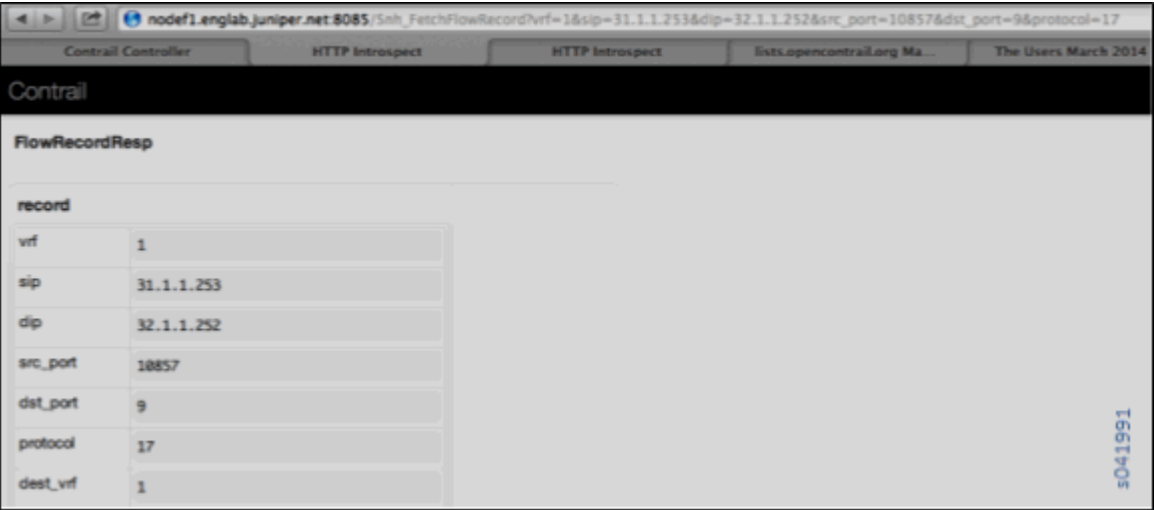
You can also retrieve information about unresolved flows from the Contrail UI, as shown in the following:

Figure 260: 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 261: 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 262: 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	<div><div>policy</div><div><div>action</div><div>g</div></div><div><div>acl</div><div>acl</div><div><div>uuid</div><div>8b0329d7-ad9e-41ac-af2e-30f4dbc2b5ae</div></div></div><div><div>action_str</div><div>action_str</div><div><div>action</div><div>deny</div></div></div></div>

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 | 972](#)
- [Verifying the BGP Routers | 972](#)
- [Verifying the Route Exchange | 976](#)
- [Debugging Route Exchange with Policies | 978](#)
- [Debugging Peering with an MX Series Router | 980](#)
- [Debugging a BGP Peer Down Error with Incorrect Family | 982](#)
- [Configuring MX Peering \(iBGP\) | 984](#)
- [Checking Route Exchange with an MX Series Peer | 986](#)
- [Checking the Route in the MX Series Router | 988](#)

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 263: 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 264: 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.englab.juniper.net:8082/bgp-router/9702853f-5e48-417f-bd72-c00a12cc0200",
    - attr: {
      - session: [
        - {
          - attributes: [
            - {
              bgp_router: null,
              - address_families: {
                - family: {
                  "inet-vpn",
                  "e-vpn"
                }
              }
            },
            ],
          },
          uuid: null
        }
      ],
      },
      uuid: "9702853f-5e48-417f-bd72-c00a12cc0200"
    },
  ],
],

```

### 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?`

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:___default___:nodea20	controll	64512	10.204.216.16	10.204.216.16
					<a href="#">address_families</a> <a href="#">inet-vpn</a> <a href="#">e-vpn</a>



**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 268: Sample Output, Show Routing Instance:

default-domain:demo:vn1	default-domain:demo:vn1	4	import_target target:64512:1	export_target target:64512:1	tables	peers
					name	peers
					default-domain:demo:vn1:vn1_inet.0	nodea20
					default-domain:demo:vn1:vn1_inet.0	nodea20
					default-domain:demo:vn1:vn1_inetcast.0	nodea20
default-domain:demo:vn2	default-domain:demo:vn2	5	import_target target:64512:2	export_target target:64512:2	tables	peers
					name	peers
					default-domain:demo:vn2:vn2_inet.0	nodea22
					default-domain:demo:vn2:vn2_inet.0	nodea22
					default-domain:demo:vn2: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





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 271: 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 gre udp	16	true	default-domain:demo:vn1:inet.0	communities security group: 5 originvn:64512:4 target:64512:1	default-domain:demo:vn1	0
10.204.216.16	-	10.204.216.16	tunnel_encap gre udp	16	false		communities security group: 5 originvn:64512:4 target:64512:1	default-domain:demo:vn1	0
source	as_path	next_hop	tunnel_encap	label	replicated	primary_table	communities	origin_vn	flags
nodea22	-	10.204.216.18	tunnel_encap gre udp	16	true	default-domain:demo:vn2:inet.0	communities security group: 5 originvn:64512:5 target:64512:2	default-domain:demo:vn2	0
10.204.216.16	-	10.204.216.18	tunnel_encap gre udp	16	false		communities security group: 5 originvn:64512:5 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 272: Create Policy Window

Create Policy

Policy Name

any\_any

Policy Rules

Action	Protocol	Source Network	Source Ports	Direction	Destination Network	Destination Ports	Apply Service	Mirror to
PAS	ANY	default-domain:	Source	<>	default-domain:	Destinat	<input type="checkbox"/>	<input type="checkbox"/>

CancelSave

2. Validate that the routing instances have the correct import\_target configuration.

http: //<host ip address>:8083/Snh\_ShowRoutingInstanceReq?name=

Figure 273: Sample Output, Validate Import Target:

default-domain:demo:vn1:vn1	default-domain:demo:vn1	4	import_target target:64512:1 target:64512:2	export_target target:64512:1
default-domain:demo:vn2:vn2	default-domain:demo:vn2	5	import_target target:64512:1 target:64512:2	export_target 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 274: Sample Output, Route Import:

ShowRouteResp

tables																
routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	feasible_paths	routes									
default-domain:demo:vn2:vn2	default-domain:demo:vn2:vn2:inet.0	2	4	1	3	0	<div><div>routes</div><table><tr><th>prefix</th><th>last_modified</th><th>paths</th></tr><tr><td>1.1.1.253/32</td><td>2014-Feb-10 12:02:47.261344</td><td><div><div>paths</div><div>protocol</div><div>XMPF</div><div>BGP</div></div></td></tr><tr><td>2.2.2.253/32</td><td>2014-Feb-10 11:34:35.469899</td><td><div><div>paths</div><div>protocol</div><div>XMPF</div><div>BGP</div></div></td></tr></table></div>	prefix	last_modified	paths	1.1.1.253/32	2014-Feb-10 12:02:47.261344	<div><div>paths</div><div>protocol</div><div>XMPF</div><div>BGP</div></div>	2.2.2.253/32	2014-Feb-10 11:34:35.469899	<div><div>paths</div><div>protocol</div><div>XMPF</div><div>BGP</div></div>
prefix	last_modified	paths														
1.1.1.253/32	2014-Feb-10 12:02:47.261344	<div><div>paths</div><div>protocol</div><div>XMPF</div><div>BGP</div></div>														
2.2.2.253/32	2014-Feb-10 11:34:35.469899	<div><div>paths</div><div>protocol</div><div>XMPF</div><div>BGP</div></div>														

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 275: Edit Global ASN Window

Edit Global ASN

Global ASN

54321

Cancel Save

- 2. Configure the eBGP peer for the MX Series router. Use the Contrail Web UI or Python provisioning.

Figure 276: 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>:8081/analytics/uves/bgp-peers`. 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 277: 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 278: 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
  }
}

```

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 279: Edit Global ASN Window

2. Configure the MX Series IBGP peer, using Contrail WebUI or Python provisioning.

Figure 280: 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 281: 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 282: Sample Established IBGP Peer Introspect Window

neighbors										
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 283: 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.13vpn.0	2	2	2	0	0	routes
							prefix
							10.204.216.253:5:0.0.0.0/0
							10.204.216.253:5:10.204.218.0/24

2. Configure a public virtual network.

Figure 284: 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.13vpn.0	2	2	2	0	0	routes
							prefix
							10.204.216.253:5:0.0.0.0/0
							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 285: Routing Instance Public IPv4 Route Table

tables	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
							prefix
							0.0.0.0/0
							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](http://<host ip address>:8083/Snh_ShowRouteReq?x=default-domain:admin:public:public.inet.0)

**Figure 286: 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><tr><th>prefix</th><th>last_modified</th><th>paths</th></tr><tr><td>0.0.0.0/0</td><td>2014-Mar-14 10:05:05.719026</td><td><table><tr><th>protocol</th></tr><tr><td>BGP</td></tr></table></td></tr><tr><td>10.204.218.0/24</td><td>2014-Mar-14 10:05:05.720017</td><td><table><tr><th>protocol</th></tr><tr><td>BGP</td></tr></table></td></tr><tr><td>11.2.3.253/32</td><td>2014-Mar-14 10:18:48.797958</td><td><table><tr><th>protocol</th></tr><tr><td>BGP</td></tr></table></td></tr></table>	prefix	last_modified	paths	0.0.0.0/0	2014-Mar-14 10:05:05.719026	<table><tr><th>protocol</th></tr><tr><td>BGP</td></tr></table>	protocol	BGP	10.204.218.0/24	2014-Mar-14 10:05:05.720017	<table><tr><th>protocol</th></tr><tr><td>BGP</td></tr></table>	protocol	BGP	11.2.3.253/32	2014-Mar-14 10:18:48.797958	<table><tr><th>protocol</th></tr><tr><td>BGP</td></tr></table>	protocol	BGP
prefix	last_modified	paths																							
0.0.0.0/0	2014-Mar-14 10:05:05.719026	<table><tr><th>protocol</th></tr><tr><td>BGP</td></tr></table>	protocol	BGP																					
protocol																									
BGP																									
10.204.218.0/24	2014-Mar-14 10:05:05.720017	<table><tr><th>protocol</th></tr><tr><td>BGP</td></tr></table>	protocol	BGP																					
protocol																									
BGP																									
11.2.3.253/32	2014-Mar-14 10:18:48.797958	<table><tr><th>protocol</th></tr><tr><td>BGP</td></tr></table>	protocol	BGP																					
protocol																									
BGP																									

5. Verify the route in the bgp.l3vpn.0 table.

[http://<host ip address>:8083/Snh\\_ShowRouteReq?x=bgp.l3vpn.0](http://<host ip address>:8083/Snh_ShowRouteReq?x=bgp.l3vpn.0)

**Figure 287: BGP Routing Instance Route Table**

tables							
routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	infeasible_paths	routes
default-domain:default-project:(p-fabric:___default...	bgp.l3vpn.0	3	3	2	1	0	<div>routes</div> <div>prefix</div> <div>10.204.216.253:5:0.0.0.0/0</div> <div>10.204.216.253:5:10.204.218.0/24</div> <div>10.204.216.70:1:11.2.3.253/32</div>

## 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 | 991](#)
- [Example | 992](#)
- [Example: MX80 Configuration for the Gateway | 993](#)
- [Ping the Floating IP from the Public Network | 996](#)
- [Troubleshooting Details | 997](#)
- [Get the UUID of the Virtual Network | 997](#)
- [View the Floating IP Object in the API Server | 998](#)
- [View floating-ips in floating-ip-pools in the API Server | 1002](#)
- [Check Floating IP Objects in the Virtual Machine Interface | 1005](#)
- [View the BGP Peer Status on the Control Node | 1009](#)
- [Querying Routes in the Public Virtual Network | 1010](#)
- [Verification from the MX80 Gateway | 1012](#)
- [Viewing the Compute Node Vnsw Agent | 1014](#)
- [Advanced Troubleshooting | 1016](#)

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:

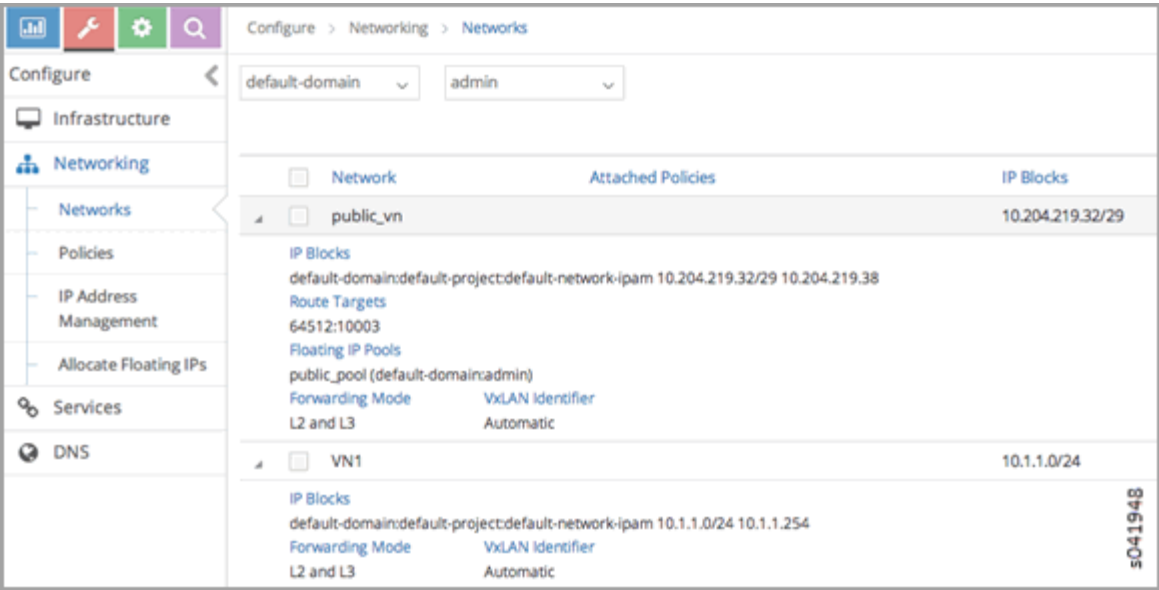
Public virtual network:

- 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

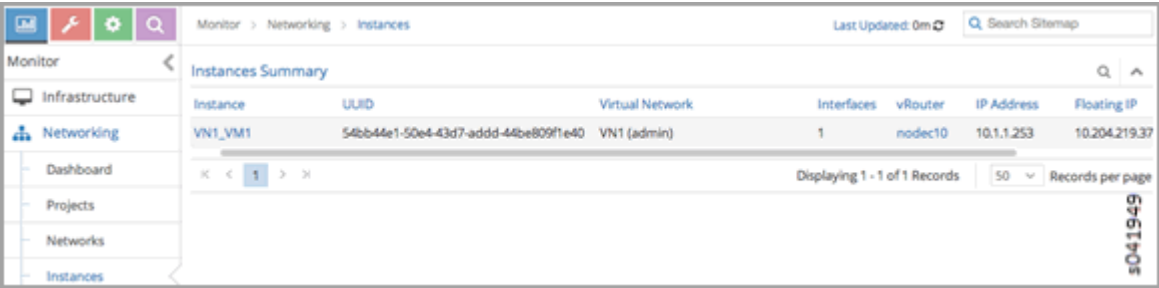
Private virtual network:

- 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.



Configure > Infrastructure > BGP Peer				
<div> <div> <div>Configure</div> <div>Infrastructure</div> <div>BGP Peer</div> <div>Forwarding Options</div> <div>Link Local Services</div> <div>Networking</div> <div>Services</div> <div>DNS</div> </div> <div> <div>Global ASN</div> <div>Create</div> </div> </div>				
<div> <div> <div>IP Address</div> <div>Type</div> <div>Vendor</div> <div>HostName</div> </div> <div> <div>10.204.216.253</div> <div>BGP Peer</div> <div>mx</div> <div>mx1</div> </div> <div> <div>Peers</div> <div>Vendor</div> <div>BGP ASN</div> <div>Router ID</div> <div>BGP Port</div> <div>Address family</div> </div> <div> <div>nodec7,nodec8</div> <div>mx</div> <div>64512</div> <div>10.204.216.253</div> <div>179</div> <div>inet-vpn</div> </div> </div>				
<div> <div> <div>10.204.216.64</div> <div>Control Node</div> <div>contrail</div> <div>nodec7</div> </div> <div> <div>Peers</div> <div>Vendor</div> <div>BGP ASN</div> <div>Router ID</div> <div>BGP Port</div> <div>Address family</div> </div> <div> <div>mx1,nodec8</div> <div>contrail</div> <div>64512</div> <div>10.204.216.64</div> <div>179</div> <div>inet-vpn,e-vpn</div> </div> </div>				
<div> <div> <div>10.204.216.65</div> <div>Control Node</div> <div>contrail</div> <div>nodec8</div> </div> <div> <div>Peers</div> <div>Vendor</div> <div>BGP ASN</div> <div>Router ID</div> <div>BGP Port</div> <div>Address family</div> </div> <div> <div>mx1,nodec7</div> <div>contrail</div> <div>64512</div> <div>10.204.216.65</div> <div>179</div> <div>inet-vpn,e-vpn</div> </div> </div>				

### 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 {
```

```
        source-address 10.204.216.253;

        gre;

        destination-networks {

            10.204.216.0/24;

            10.204.217.0/24;

        }

    }

}

protocols {

    bgp {

        group control-nodes {

            type internal;

            local-address 10.204.216.253;

            keep all;

            family inet-vpn {

                unicast;

            }

            neighbor 10.204.216.64;

            neighbor 10.204.216.65;

        }

    }

}
```

```

    }

}

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          |

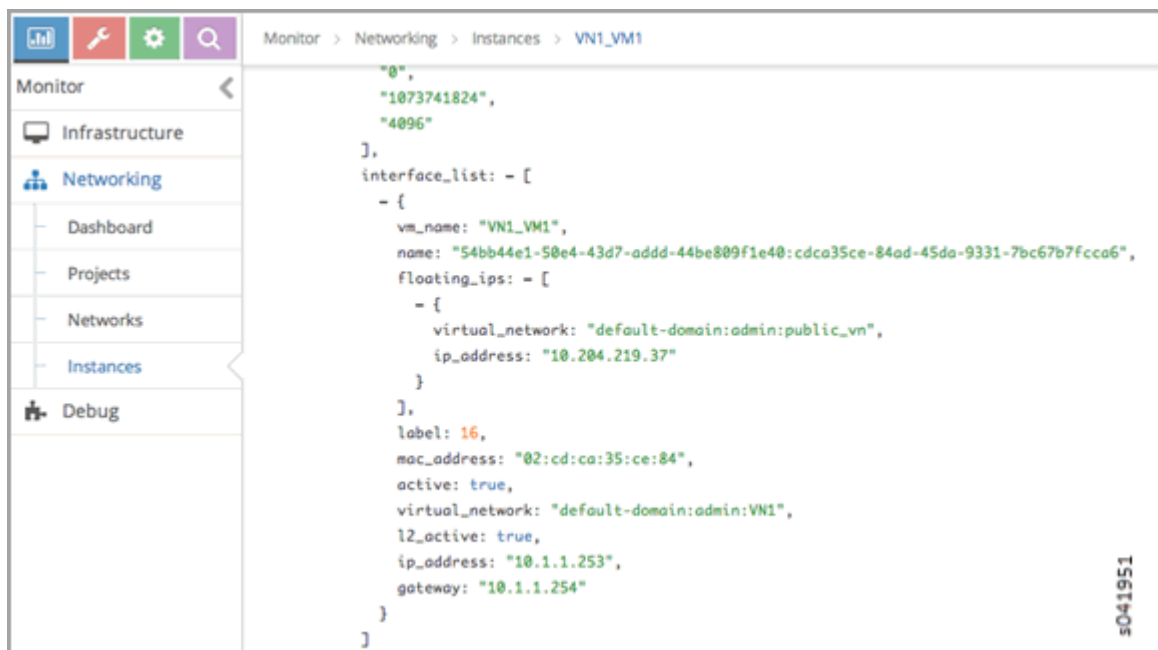
```

```
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:



### 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	1595/ 1597

Displaying 1 - 2 of 2 Records | 50 Records per page

- Using the control-node Introspect:

`http://nodec7:8083/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_vnpublic_vn.inet.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
	10.204.219.37/32	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`

### Example



[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

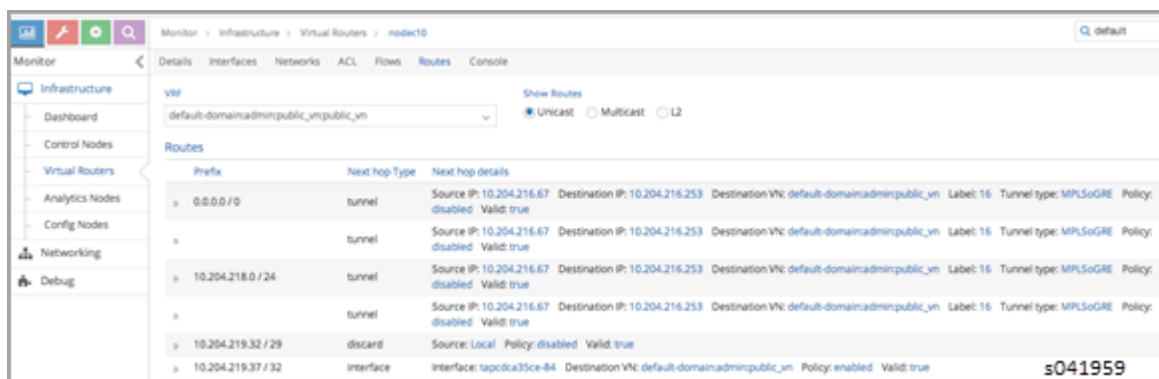
The compute node introspect can be accessed from port 8085. In the following examples, the compute nodes are nodec9 and nodec10.

### 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 VNI: 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 VNI: 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 VNI: 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.



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		
			if	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		
			if	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 | 1019](#)
- [Show Virtual Machines | 1021](#)
- [Show Virtual Machines Using Python API | 1022](#)
- [Delete Methods | 1024](#)

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 | 1024](#)
- [Troubleshooting Procedure for Link-Local Services | 1025](#)
- [Metadata Service | 1026](#)
- [Troubleshooting Procedure for Link-Local Metadata Service | 1026](#)

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

169.254.169.100

123

IP

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							
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 m2	

Checking the tap interface status in the http agent introspect:

`http://<compute-node-ip>:8085/Snh_ItfReq?name=`

itf_list				
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 172.17.28.5	123

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	slp	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.

metadata_server_port	45094
metadata_requests	2
metadata_responses	0
metadata_proxy_sessions	2
metadata_internal_errors	0

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
```

# 6

PART

## Conrail Commands and APIs

---

- [Conrail Commands | 1030](#)
  - [Conrail Application Programming Interfaces \(APIs\) | 1060](#)
-

# Contrail Commands

## IN THIS CHAPTER

- [Getting Contrail Node Status | 1030](#)
- [contrail-logs \(Accessing Log File Messages\) | 1042](#)
- [contrail-status \(Viewing Node Status\) | 1045](#)
- [contrail-version \(Viewing Version Information\) | 1047](#)
- [service \(Managing Services\) | 1050](#)
- [Backing Up Contrail Databases Using JSON Format | 1052](#)

## Getting Contrail Node Status

### IN THIS SECTION

- [Overview | 1030](#)
- [UVE for NodeStatus | 1031](#)
- [Node Status Features | 1031](#)
- [Using Introspect to Get Process Status | 1038](#)
- [contrail-status script | 1040](#)

### 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",
```

```

    type: "Discovery",

    name: "IfmapServer",

    description: "SubscribeResponse"

  },

{

  server_addrs:
[

    "10.84.13.45:5998"

  ],

  status: "Up",

  type: "Discovery",

  name: "xmpp-server",

  description: "Publish Response - HeartBeat"

}

]

}

]

}

}

```

## Using Introspect to Get Process Status

The user can also view the state of a specific process by using the introspect mechanism.

### Example: Introspect of NodeStatus

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" aggttype="union"><list type="struct" size="1"><ProcessStatus><module_id
type="string" identifier="1">ControlNode</module_id><instance_id type="string" identifier="2">0</
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_addrs
type="list" identifier="3"><list type="string" size="1"><element>10.84.13.45:8443</element></
list></server_addrs><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_addrs type="list" identifier="3"><list type="string"
size="1"><element>10.84.13.45:8086</element></list></server_addrs><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_addrs type="list" identifier="3"><list
type="string" size="1"><element>10.84.13.45:5998</element></list></server_addrs><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_addrs type="list" identifier="3"><list type="string"
size="1"><element>10.84.13.45:5998</element></list></server_addrs><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_addrs type="list"
identifier="3"><list type="string" size="1"><element>10.84.13.45:5998</element></list></
server_addrs><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 | 1042](#)
- [Option Descriptions | 1043](#)
- [Example Uses | 1044](#)

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 {ObjectVNTTable, ObjectVMTable, ObjectSITable, ObjectVRouter, ObjectBgpPeer,
ObjectRoutingInstance, ObjectBgpRouter, ObjectXmppConnection, ObjectCollectorInfo,
ObjectGeneratorInfo, ObjectConfigNode}
--object-id OBJECT_ID        Show all logs (default: False)
--object-select-field {ObjectLog, SystemLog}
                             Logs of object type (default: None)
--trace TRACE                Logs of object name (default: None)
                             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 | 1046](#)
- [Description | 1046](#)
- [Required Privilege Level | 1046](#)
- [Sample Output | 1046](#)
- [Release Information | 1047](#)

## 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 | 1048](#)
- [Description | 1048](#)
- [Required Privilege Level | 1048](#)
- [Sample Output | 1048](#)
- [Sample Output | 1049](#)
- [Release Information | 1049](#)

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> <b>contrail-version</b>			
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 | 1050
- Description | 1050
- Options | 1050
- Required Privilege Level | 1051
- Sample Output | 1051
- Release Information | 1051

### 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 `supervisord`, which is open source software written in Python. Each Contrail node type, such as `compute`, `control`, and so on, has an instance of `supervisord` that, when running, launches Contrail services as child processes. All `supervisord` instances display in `contrail-status` output with the prefix `supervisor`. If the `supervisord` 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 `supervisord` 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 | 1052](#)
- [Simple Backup Using JSON Format | 1052](#)
- [Restore Simple Database Backup | 1053](#)
- [Example Backup and Restore With JSON | 1054](#)

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/site-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/site-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	systemd/syst...	7 weeks ago	Up 2 weeks		"/lib/ analytics
f5aed0a2efc3	172.30.109.59:5100/contrail410-contrail-analyticsdb:mainline	systemd/syst...	7 weeks ago	Up 2 weeks		"/lib/ analyticsdb
0ff200b12112	172.30.109.59:5100/contrail410-contrail-controller:mainline	systemd/syst...	7 weeks ago	Up 2 weeks		"/lib/ controller
6fec888f8145	registry:2	entrypoint.sh /e...	7 weeks ago	Up 2 weeks		"/ registry

```

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 \(UVEs\) | 1060](#)
- [Log and Flow Information APIs | 1074](#)
- [Working with Neutron | 1082](#)
- [Support for Amazon VPC APIs on Contrail OpenStack | 1086](#)

## Contrail Analytics Application Programming Interfaces (APIs) and User-Visible Entities (UVEs)

## IN THIS SECTION

- [User-Visible Entities | 1061](#)
- [Common UVEs in Contrail | 1062](#)
- [Virtual Network UVE | 1062](#)
- [Virtual Machine UVE | 1063](#)
- [vRouter UVE | 1063](#)
- [UVEs for Contrail Nodes | 1064](#)
- [Wild Card Query of UVEs | 1064](#)
- [Filtering UVE Information | 1064](#)

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 | 1074](#)

## Log and Flow Information APIs

### IN THIS SECTION

- [HTTP GET APIs | 1074](#)
- [HTTP POST API | 1075](#)
- [POST Data Format Example | 1075](#)
- [Query Types | 1077](#)
- [Examining Query Status | 1077](#)
- [Examining Query Chunks | 1078](#)
- [Example Queries for Log and Flow Data | 1078](#)

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 `idl` 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, ObjectVNTable,
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
           %             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/ObjectVNTTable",
    "name": "ObjectVNTTable"
  },
  {
    "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",
"Message", "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": "Message", "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,
      "Message": "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 | 1082](#)
- [Network Sharing in Neutron | 1083](#)
- [Commands for Neutron Network Sharing | 1084](#)
- [Support for Neutron APIs | 1084](#)
- [Contrail Neutron Plugin | 1085](#)
- [DHCP Options | 1085](#)
- [Incompatibilities | 1085](#)

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 | 1086](#)
- [Mapping Amazon VPC Features to OpenStack Contrail Features | 1087](#)
- [VPC and Subnets Example | 1088](#)
- [Euca2ools CLI for VPC and Subnets | 1089](#)
- [Security in VPC: Network ACLs Example | 1089](#)
- [Euca2ools CLI for Network ACLs | 1091](#)
- [Security in VPC: Security Groups Example | 1091](#)
- [Euca2ools CLI for Security Groups | 1092](#)
- [Elastic IPs in VPC | 1093](#)
- [Euca2ools CLI for Elastic IPs | 1093](#)
- [Euca2ools CLI for Route Tables | 1094](#)
- [Supported Next Hops | 1094](#)
- [Internet Gateway Next Hop Euca2ools CLI | 1095](#)
- [NAT Instance Next Hop Euca2ools CLI | 1095](#)
- [Example: Creating a NAT Instance with Euca2ools CLI | 1095](#)

## 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 84: 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
Security Groups	Security Groups

**Table 84: Amazon VPC and OpenStack Contrail Feature Comparison (Continued)**

Amazon VPC Feature	OpenStack Contrail Feature
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	