



Contrail™

Contrail Feature Guide

Release

3.0



Modified: 2016-07-28

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

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. and/or its affiliates in the United States and other countries. All other trademarks may be 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

3.0

Copyright © 2018 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://www.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 the Documentation	xxxi
	Documentation and Release Notes	xxxi
	Documentation Conventions	xxxi
	Documentation Feedback	xxxiii
	Requesting Technical Support	xxxiv
	Self-Help Online Tools and Resources	xxxiv
	Opening a Case with JTAC	xxxiv
Part 1	Overview	
Chapter 1	Understanding Contrail Controller	3
	Contrail Overview	3
	Contrail Description	4
	Contrail Major Components	4
	Contrail Control Nodes	4
	Contrail Compute Nodes – XMPP Agent and vRouter	4
	Contrail Solution	4
Part 2	Installing and Upgrading Contrail	
Chapter 2	Supported Platforms and Server Requirements	9
	Supported Platforms	9
	Server Requirements	10
Chapter 3	Installing Contrail and Provisioning Roles	11
	Installation Overview	11
	Downloading Installation Software	12
	Installing the Operating System and Contrail Packages	13
	Configuring System Settings	14
	Installing the Contrail Packages, Part One (CentOS or Ubuntu)	15
	Setting Up the Testbed Definitions File	17
	Testbed Definitions File Settings for Deploying Contrail with an Existing OpenStack Node	20
	Supporting Multiple Interfaces on Servers and Nodes	22
	Support for Multiple Interfaces	22
	Number of cfgm Nodes Supported	23
	Uneven Number of Database Nodes Required	23
	Support for VLAN Interfaces	23
	Support for Bonding Options	23
	Support for Static Route Options	23
	Server Interface Examples	24
	Interface Naming and Configuration Management	24

Chapter 4

Setting Up Interfaces and Installing	25
Sample testbed.py File With Exclusive Interfaces	25
Installing the Contrail Packages, Part Two (CentOS or Ubuntu) — Installing on the Remaining Machines	27
Configuring the Control Node	30
Adding or Removing a Compute Node in an Existing Contrail Cluster	35
Installation and Configuration Scenarios	37
Setting Up and Using a Simple Virtual Gateway with Contrail	37
Introduction to the Simple Gateway	37
How the Simple Gateway Works	38
Setup Without Simple Gateway	38
Setup With a Simple Gateway	39
Simple Gateway Configuration Features	40
Packet Flows with the Simple Gateway	41
Packet Flow Process From the Virtual Network to the Public Network	41
Packet Flow Process From the Public Network to the Virtual Network	42
Four Methods for Configuring the Simple Gateway	43
Using Fab Provisioning to Configure the Simple Gateway	43
Using the vRouter Configuration File to Configure the Simple Gateway	44
Using Thrift Messages to Dynamically Configure the Simple Gateway	45
How to Dynamically Create a Virtual Gateway	45
How to Dynamically Delete a Virtual Gateway	46
Using Devstack to Configure the Simple Gateway	47
Common Issues with Simple Gateway Configuration	48
Configuring MD5 Authentication for BGP Sessions	48
Installing Contrail with Red Hat OpenStack	50
Overview: Contrail with Red Hat OpenStack	50
Procedure for Installing RHOSP5	50
Install and Configure Contrail	51
Download and Install Contrail-Install-Packages to the First Config Node	52
Update the Testbed.py File	52
Complete the Installation on Remaining Nodes	54
Appendix: Installing with RDO	55
Install All-in-One OpenStack	56
Configuring OpenStack Nova Docker with Contrail	56
Nova Docker Support in Contrail Overview	56
Contrail and OpenStack Nova Docker Platform Support	56
Deploying a Contrail Compute Node to Work With OpenStack Nova Docker	57
Saving the Docker Image to Glance	57
Launching a Docker Container	58
Limitations	58
Configuring the Data Plane Development Kit (DPDK) Integrated with Contrail vRouter	58
DPDK Support in Contrail	58
Preparing the testbed.py File for Provisioning a Contrail Cluster Node with DPDK	59

Provisioning a Contrail Cluster with DPDK	60
Creating a Flavor for DPDK in OpenStack Kilo	60
Configuring Single Root I/O Virtualization (SR-IOV)	61
Overview: Configuring SR-IOV	61
Preparing the testbed.py File for Provisioning a Contrail Cluster with SR-IOV	61
Enabling ASPM in BIOS	62
Configuring SR-IOV Features Without env.sriov in testbed.py	62
Launching SR-IOV Virtual Machines	63
Using the Contrail UI to Enable and Launch an SR-IOV Virtual Machine	63
Using the CLI to Enable and Launch SR-IOV Virtual Machines	64
Configuring Virtual Networks for Hub-and-Spoke Topology	65
Route Targets for Virtual Networks in Hub-and-Spoke Topology	65
Example: Configuring Hub-and-Spoke Virtual Networks	66
Troubleshooting Hub-and-Spoke Topology	66
Configuring Transport Layer Security-Based XMPP in Contrail	70
Overview: TLS-Based XMPP	70
TLS XMPP in Contrail	70
Configuring XMPP Client and Server in Contrail	70
Configuring Control Node for XMPP Server	71
Configuring DNS Server for XMPP Server	71
Configuring Control Node for XMPP Client	71
Chapter 5 Using Contrail with VMware vCenter	73
Installing and Provisioning VMware vCenter with Contrail	73
Overview: Integrating Contrail with vCenter Server	73
Different Modes of vCenter Integration with Contrail	74
vCenter-Only Mode	74
vCenter-as-Compute Mode	75
Preparing the Installation Environment	76
Installation for vCenter-Only Mode	76
Installing the vCenter-Only Components	77
Installation for vCenter-as-Compute Mode	78
Installing the vCenter-as-Compute Components	80
Verification	81
Adding Hosts or Nodes	81
Adding an ESXi Host to an Existing vCenter Cluster	81
Adding a vCenter Compute Node	82
Underlay Network Configuration for ContrailVM	82
Standard Switch Setup	82
Distributed Switch Setup	83
PCI Pass-Through Setup	85
SR-IOV Setup	87

	Sample Testbed.py Files for Contrail vCenter	90
	Sample Testbed.py File for vCenter-Only Mode	90
	Sample Testbed.py File for vCenter-as-Compute Mode	93
	Using the Contrail and VMWare vCenter User Interfaces to Manage the	
	Network	95
	Overview: User Interfaces for Contrail Integration with VMware vCenter . . .	95
	Contrail User Interface	96
	Contrail vCenter User Interface	96
	Feature Configuration for Contrail vCenter	97
	Creating a Virtual Network	97
	Creating a Virtual Machine	104
	Create a Virtual Machine – vCenter UI	104
	Configuring the vCenter Network in Contrail UI	110
Chapter 6	Using Server Manager to Automate Provisioning	111
	Installing Server Manager	111
	Installation Requirements for Server Manager	111
	Platform Support	111
	Installation Prerequisites	112
	Installing Server Manager	112
	Finishing the Provisioning	113
	Starting the Server Manager Service	113
	Upgrading Server Manager Software	114
	Prerequisite to Upgrading	114
	Steps for a New Installation	114
	Server Manager Installation Completion Checks	114
	Server Manager Checks	114
	Server Manager Client Checks	115
	Server Manager Webui Checks	115
	Sample Configurations for Server Manager Templates	115
	Sample Settings	115
	The dhcp.template File	115
	The named.conf.options File	116
	The named.template File	116
	The sendmail.cf File	116
	Using Server Manager to Automate Provisioning	116
	Overview of Server Manager	117
	Server Manager Requirements and Assumptions	117
	Server Manager Component Interactions	118
	Configuring Server Manager	119
	Configuring the Cobbler DHCP Template	120
	User-Defined Tags for Server Manager	121
	Server Manager Client Configuration File	121
	Restart Services	123
	Accessing Server Manager	123
	Communicating with the Server Manager Client	123
	Server Manager Commands for Configuring Servers	124
	Create New Servers or Update Existing Servers	124
	Delete Servers	126

Show Server Configuration	127
Server Manager Commands for Managing Clusters	127
Server Manager Commands for Managing Tags	130
Server Manager Commands for Managing Images	132
Server Manager Operational Commands for Managing Servers	136
Reimaging Server(s)	136
Provisioning and Configuring Roles on Servers	137
Adding and Deleting Roles	139
Restarting Server(s)	140
Show Status of Server(s)	141
Server Manager REST API Calls	142
REST APIs for Server Manager Configuration Database Entries	143
API: Add a Server	143
API: Delete Servers	144
API: Retrieve Server Configuration	144
API: Add an Image	144
API: Upload an Image	145
API: Get Image Information	145
API: Delete an Image	145
API: Add or Modify a Cluster	146
API: Delete a Cluster	146
API: Get Cluster Configuration	147
API: Get All Server Manager Configurations	147
API: Reimage Servers	147
API: Provision Servers	147
API: Restart Servers	148
Example: Reimaging and Provisioning a Server	148
Using the Server Manager Web User Interface	151
Log In to Server Manager	151
Create a Cluster for Server Manager	152
Working with Servers in the Server Manager User Interface	159
Add a Server	159
Edit Tags for Servers	161
Using the Edit Config Option for Multiple Servers	161
Filter Servers by Tag	162
Viewing Server Details	162
Configuring Images and Packages	163
Add New Image or Package	163
Selecting Server Manager Actions for Clusters	164
Reimage a Cluster	164
Provision a Cluster	164
Installing and Using Server Manager Lite	165
Server Manager Lite Overview	165
Installing Server Manager Lite	165
Provisioning Using SM-Lite	166
Displaying the Cluster Status	167
Displaying the SM-Lite Installation and Provisioning Log Files	167
Contrail Provisioning Log Files	167

Chapter 7

Extending Contrail to Physical Routers, Bare Metal Servers, Switches, and Interfaces	169
Using ToR Switches and OVSDb to Extend the Contrail Cluster to Other	
Instances	169
Support for ToR Switch and OVSDb Overview	169
ToR Services Node (TSN)	170
Contrail ToR Agent	170
Configuration Model	170
Control Plane	171
Data Plane	172
Using the Web Interface to Configure ToR Switch and Interfaces	172
Provisioning with Fab Commands	174
Prerequisite Configuration for QFX5100 Series Switch	176
Debug QFX5100 Configuration	177
Changes to Agent Configuration File	178
REST APIs	179
Configuring High Availability for the Contrail OVSDb ToR Agent	179
Overview: High Availability for a ToR Switch	179
High Availability Solution for Contrail ToR Agent	179
Failover Methodology Description	181
Failure Scenarios	181
Redundancy for HAProxy	183
Configuration for ToR Agent High Availability	183
Testbed.py and Provisioning for High Availability	184
Using Device Manager to Manage Physical Routers	184
Support for Physical Routers Overview	185
Configuration Model	185
Configuring a Physical Router	186
Alternate Ways to Configure a Physical Router	188
Device Manager Configurations	188
Prerequisite Configuration Required on MX Series Device	189
Debugging Device Manager Configuration	189
Configuration Scenarios	189
Configuring Physical Routers Using REST APIs	189
Sample Python Script Using Rest API for Configuring an MX Device	189
Device Manager Functionality	189
Dynamic Tunnels	190
Web UI Configuration	190
BGP Groups	192
Extending the Private Network	194
Extending the Public Network	196
Ethernet VPN Configuration	197
Floating IP Addresses and Source Network Address Translation for Guest	
Virtual Machines and Bare Metal Servers	198
Samples of Generated Configurations for an MX Series Device	206
Scenario 1: Physical Router With No External Networks	206
Scenario 2: Physical Router With External Network, Public VRF	208
Scenario 3: Physical Router With External Network, Public VRF, and	
EVPN	209

	Scenario 4: Physical Router With External Network, Public VRF, and Floating IP Addresses for a Bare Metal Server	210
	REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces	210
	Introduction: REST APIs for Extending Contrail Cluster	210
	REST API for Physical Routers	210
	REST API for Physical Interfaces	212
	REST API for Logical Interfaces	213
Chapter 8	Installing and Using Contrail Storage	215
	Installing and Using Contrail Storage	215
	Overview of the Contrail Storage Solution	215
	Basic Storage Functionality with Contrail	216
	Ceph Block and Object Storage Functionality	216
	Using the Contrail Storage User Interface	217
	Hardware Specifications	218
	Software Files for Compute Storage Nodes	218
	Contrail OpenStack Nova Modifications	218
	Installing the Contrail Storage Solution	219
	Using Fabric Commands to Install and Configure Storage	219
	Fabric Installation Procedure	220
	Using Server Manager to Install and Configure Storage	222
	Server Manager Installation Procedure for Storage	222
	Example: Configurations for Storage for Reimaging and Provisioning a Server	223
	Storage Installation Limits	228
Chapter 9	Upgrading Contrail Software	231
	Upgrading Contrail Software	231
	DKMS for vRouter Kernel Module	234
Part 3	Configuring Contrail	
Chapter 10	Configuring Virtual Networks	237
	Creating Projects in OpenStack for Configuring Tenants in Contrail	238
	Creating Virtual Networks and Policies in Juniper Networks Contrail	239
	Creating a Virtual Network—Juniper Networks Contrail	240
	Deleting a Virtual Network with Juniper Networks Contrail	242
	Creating a Network Policy with Juniper Networks Contrail	244
	Associating a Network to a Policy—Juniper Networks Contrail	246
	Associating Network Policies Overview	246
	Associating a Network Policy to a Network	246
	Creating Virtual Networks and Policies in OpenStack Contrail	247
	Creating a Virtual Network—OpenStack Contrail	248
	Deleting a Virtual Network with OpenStack Contrail	250
	Creating a Network Policy with OpenStack Contrail	252
	Associating a Network to a Policy—OpenStack Contrail	255
	Associating Network Policies Overview	256
	Associating a Network Policy to a Network	256

	Creating an Image and Launching a Virtual Machine	257
	Creating an Image	257
	Launching a Virtual Machine (Instance)	260
	Creating a Floating IP Address Pool and Allocating it to a Virtual Machine	262
	Creating a Floating IP Address Pool	263
	Allocating a Floating IP Address to a Virtual Machine	264
	Using Security Groups with Virtual Machines (Instances)	266
	Security Groups Overview	266
	Creating Security Groups and Adding Rules	266
	Support for IPv6 Networks in Contrail	269
	Overview: IPv6 Networks in Contrail	270
	Creating IPv6 Virtual Networks in Contrail	270
	Address Assignments	270
	Adding IPv6 Peers	271
	Configuring EVPN and VXLAN	272
	Configuring the VXLAN Identifier Mode	274
	Configuring Forwarding	276
	Configuring the VXLAN Identifier	277
	Configuring Encapsulation Methods	278
Chapter 11	Example of Deploying a Multi-Tier Web Application Using Contrail	281
	Example: Deploying a Multi-Tier Web Application	281
	Multi-Tier Web Application Overview	281
	Example: Setting Up Virtual Networks for a Simple Tiered Web Application	282
	Verifying the Multi-Tier Web Application	284
	Sample Addressing Scheme for Simple Tiered Web Application	285
	Sample Physical Topology for Simple Tiered Web Application	286
	Sample Physical Topology Addressing	286
	Sample Network Configuration for Devices for Simple Tiered Web Application	287
Chapter 12	Configuring Services	293
	Configuring DNS Servers	293
	DNS Overview	293
	Defining Multiple Virtual Domain Name Servers	294
	IPAM and Virtual DNS	294
	DNS Record Types	295
	Configuring DNS Using the Interface	296
	Configuring DNS Using Scripts	301
	Configuring Discovery Service	302
	Contrail Discovery Service Introduction	302
	Discovery Service Registration and Publishing	303
	Discovery Service Subscription	303
	Discovery Service REST API	304
	Discovery Service Heartbeats	306
	Discovery Service Internal Databases	306
	Discovery Service Client Library	306
	Discovery Service Debugging	306

	Support for Multicast	306
	Subnet Broadcast	307
	All-Broadcast/Limited-Broadcast and Link-Local Multicast	307
	Host Broadcast	308
	Using Static Routes with Services	308
	Static Routes for Service Instances	308
	Configuring Static Routes on a Service Instance	309
	Configuring Static Routes on Service Instance Interfaces	310
	Configuring Static Routes as Host Routes	312
	Configuring Metadata Service	312
	BGP as a Service	313
	Contrail BGPaaS Features	313
	BGPaaS Customer Use Cases	314
	Dynamic Tunnel Insertion Within a Tenant Overlay	314
	Dynamic Network Reachability of Applications	315
	Liveness Detection for High Availability	315
	Configuring BGPaaS	315
	Configuring BGPaaS Using VNC API	316
	Using the Contrail User Interface to Configure BGPaaS	316
Chapter 13	Configuring Service Chaining	319
	Service Chaining	319
	Service Chaining Basics	319
	Service Chaining Configuration Elements	321
	Service Chaining MX Series Configuration	323
	Example: Creating an In-Network or In-Network-NAT Service Chain	324
	Creating an In-Network or In-Network-NAT Service Chain	325
	Example: Creating a Transparent Service Chain	333
	Creating a Transparent Mode Service Chain	333
	Example: Creating a Service Chain With the CLI	337
	CLI for Creating a Service Chain	337
	CLI for Creating a Service Template	338
	CLI for Creating a Service Instance	338
	CLI for Creating a Service Policy	338
	Example: Creating a Service Chain with VSRX and In-Network or Routed Mode	339
	ECMP Load Balancing in the Service Chain	340
	Customized Hash Field Selection for ECMP Load Balancing	341
	Overview: Custom Hash Feature	341
	Using ECMP Hash Fields Selection	343
	Configuring ECMP Hash Fields Over Service Chains	343
	Sample Flows	344
	Sample Traffic Flow Path Without Custom ECMP Hash Fields	344
	Sample Traffic Flow Path With Custom ECMP Hash Fields	344
	Using the Contrail Heat Template	345
	Introduction to Heat	345
	Heat Architecture	346
	Heat Version 2 Resources	346
	Deprecation of Heat Version 1 Resources	346

	Heat Version 2 with Service Chaining and Port Tuple Sample Workflow . . .	347
	Example: Creating a Service Template Using Heat	347
	Service Chain Route Reorigination	349
	Overview: Service Chains in Contrail	349
	Route Aggregation	350
	Schema for Route Aggregation	351
	Configuring and Troubleshooting Route Aggregation	352
	Routing Policy	357
	Applying Routing Policy	358
	Routing Policy Configuration	360
	Configuring and Troubleshooting Routing Policy	361
	Using a VNC Script to Create Routing Policy	362
	Verify Routing Policy in API Server	363
	Verify Routing Policy in the Control Node	364
	Verify Routing Policy Configuration in the Control Node	364
	Verify Routing Policy Configuration on the Routing Instance	364
	Control for Route Reorigination	365
	Configuring and Troubleshooting Reorigination Control	366
	Service Instance Health Check	367
	Health Check Overview	367
	Health Check Object Configuration	367
	Health Check Modes	368
	Using the Health Check	368
	Health Check Process	369
Chapter 14	Adding Physical Network Functions in Service Chains	371
	Using Physical Network Functions in Contrail Service Chains	371
	PNF Service Chaining Objects	371
	Prerequisites and Assumptions	372
	Example: Adding a Physical Network Function Device to a Service Chain	372
	Prerequisites for Adding a PNF to a Service Chain	373
Chapter 15	Load Balancers	381
	Using Load Balancers in Contrail 3.0 and Greater	381
	Invoking LBaaS Drivers	381
	Using a Service Appliance Set as the LBaaS Provider	383
	Understanding the Load Balancer Agent	384
	F5 Networks Load Balancer Integration in Contrail	384
	F5 Load Balancer Global Routed Mode	385
	Traffic Flow in Global Routed Mode	385
	Routing Traffic to Pool Members	386
	Routing Reverse Traffic from Pool Members to the F5 Device	386
	Initial Configuration on an F5 Device	387
	Initial Configuration on an MX Series Device Used as DC Gateway	387
	Configuration on MX Device for Each Pool Member	387
	Example: Creating a Load Balancer	387
	Configuring Load Balancing as a Service in Contrail	388
	Overview: Load Balancing as a Service	388
	Contrail LBaaS Implementation	389

Chapter 16	Configuring High Availability	395
	Juniper OpenStack High Availability	395
	Introduction	395
	Contrail High Availability	396
	OpenStack High Availability	396
	Supported Platforms	396
	Juniper OpenStack High Availability Architecture	396
	Juniper OpenStack Objectives	397
	Limitations	397
	Solution Components	398
	Virtual IP with Load Balancing	398
	Failure Handling	398
	Deployment	399
	Minimum Hardware Requirement	399
	Compute	399
	Network	400
	Installation	400
	Testbed File for Fab	401
	High Availability Support	404
	Contrail High Availability Features	404
	Configuration Options for Enabling Contrail High Availability	405
	Supported Cluster Topologies for High Availability	405
	Deploying OpenStack and Contrail on the Same High Available Nodes	405
	Deploying OpenStack and Contrail on Different High Available Nodes	406
	Deploying Contrail Only on High Available Nodes	406
	Example: Adding New OpenStack or Contrail Roles to an Existing High Availability Cluster	407
	Cluster	407
	Adding New OpenStack or Contrail Roles to an Existing High Availability Cluster	407
	Purging a Controller From an Existing Cluster	408
	Replacing a Node With a Node That Has the Same IP Address	409
	Known Limitations and Configuration Guidelines	409
	Understanding How the System Adds a New Node to an Existing Cluster	410
Chapter 17	Configuring Multitenancy Support	413
	Configuring Multitenancy Support	413
	Multitenancy Permissions	413
	API Server	414
	API Library Keystone Integration	414
	Supporting Utilities	414
	Configuring Network QoS Parameters	415
	Overview	416
	QoS Configuration Examples	416
	Limitations	417

Chapter 18	Optimizing Contrail	419
	vRouter Command Line Utilities	419
	Overview	419
	vif Command	420
	flow Command	422
	vrfstats Command	424
	rt Command	424
	dropstats Command	425
	mpls Command	429
	mirror Command	431
	vxlan Command	432
	nh Command	433
	Route Target Filtering	435
	Introduction	435
	Debugging and Troubleshooting Route Target Filtering	436
	RTF Limitations in Contrail 1.10	437
	Source Network Address Translation (SNAT)	437
	Overview	437
	Neutron APIs for Routers	438
	Network Namespace	439
	Using Web UI to Configure Routers with SNAT	439
Part 4	Monitoring and Troubleshooting Contrail	
Chapter 19	Configuring Traffic Mirroring to Monitor Network Traffic	443
	Configuring Traffic Analyzers and Packet Capture for Mirroring	443
	Traffic Analyzer Images	443
	Configuring Traffic Analyzers	444
	Setting Up Traffic Mirroring Using Monitor > Debug > Packet Capture	444
	Setting Up Traffic Mirroring Using Configure > Networking > Services	448
	Configuring Interface Monitoring and Mirroring	453
	Analyzer Service Virtual Machine	454
	Packet Format for Analyzer	455
	Metadata Format	455
	Wireshark Changes	456
	Troubleshooting Packet Display	456
Chapter 20	Understanding Contrail Analytics	459
	Contrail Analytics Overview	459
	Contrail Alerts	460
	Alert API Format	460
	Analytics APIs for Alerts	461
	Analytics APIs for SSE Streaming	462

Built-in Node Alerts	462
Underlay Overlay Mapping in Contrail	463
Overview: Underlay Overlay Mapping using Contrail Analytics	464
Underlay Overlay Analytics Available in Contrail	464
Architecture and Data Collection	465
New Processes/Services for Underlay Overlay Mapping	465
External Interfaces Configuration for Underlay Overlay Mapping	466
Physical Topology	466
SNMP Configuration	467
Link Layer Discovery Protocol (LLDP) Configuration	467
IPFIX and sFlow Configuration	467
Sending pRouter Information to the SNMP Collector in Contrail	468
pRouter UVEs	469
Contrail User Interface for Underlay Overlay Analytics	470
Viewing Topology to the Virtual Machine Level	470
Viewing the Traffic of any Link	471
Trace Flows	471
Search Flows and Map Flows	472
Overlay to Underlay Flow Map Schemas	473
Module Operations for Overlay Underlay Mapping	476
SNMP Collector Operation	476
Topology Module Operation	477
IPFIX and sFlow Collector Operation	477
Troubleshooting Underlay Overlay Mapping	478
Script to add pRouter Objects	479
Chapter 21	
Configuring Contrail Analytics	481
Analytics Scalability	481
High Availability for Analytics	482
System Log Receiver in Contrail Analytics	483
Overview	483
Redirecting System Logs to Contrail Collector	483
Exporting Logs from Contrail Analytics	483
Ceilometer Support in a Contrail Cloud	484
Overview	484
Ceilometer Details	484
Verification of Ceilometer Operation	485
Contrail Ceilometer Plugin	487
Ceilometer Installation and Provisioning	489

Chapter 22	Using Contrail Analytics to Monitor and Troubleshoot the Network	491
	Monitoring the System	492
	Debugging Processes Using the Contrail Introspect Feature	494
	Monitor > Infrastructure > Dashboard	498
	Monitor Dashboard	498
	Monitor Individual Details from the Dashboard	499
	Using Bubble Charts	499
	Color-Coding of Bubble Charts	500
	Monitor > Infrastructure > Control Nodes	500
	Monitor Control Nodes Summary	501
	Monitor Individual Control Node Details	501
	Monitor Individual Control Node Console	503
	Monitor Individual Control Node Peers	505
	Monitor Individual Control Node Routes	506
	Monitor > Infrastructure > Virtual Routers	507
	Monitor vRouters Summary	508
	Monitor Individual vRouters Tabs	509
	Monitor Individual vRouter Details Tab	509
	Monitor Individual vRouters Interfaces Tab	510
	Configuring Interface Monitoring and Mirroring	511
	Monitor Individual vRouters Networks Tab	512
	Monitor Individual vRouters ACL Tab	513
	Monitor Individual vRouters Flows Tab	514
	Monitor Individual vRouters Routes Tab	515
	Monitor Individual vRouter Console Tab	516
	Monitor > Infrastructure > Analytics Nodes	518
	Monitor Analytics Nodes	518
	Monitor Analytics Individual Node Details Tab	519
	Monitor Analytics Individual Node Generators Tab	520
	Monitor Analytics Individual Node QE Queries Tab	521
	Monitor Analytics Individual Node Console Tab	522
	Monitor > Infrastructure > Config Nodes	523
	Monitor Config Nodes	523
	Monitor Individual Config Node Details	524
	Monitor Individual Config Node Console	525
	Monitor > Networking	526
	Monitor > Networking Menu Options	526
	Monitor -> Networking -> Dashboard	527
	Monitor > Networking > Projects	528
	Monitor Projects Detail	529
	Monitor > Networking > Networks	531
	Query > Flows	534
	Query > Flows > Flow Series	534
	Example: Query Flow Series	537
	Query > Flow Records	538
	Query > Flows > Query Queue	540

	Query > Logs	541
	Query > Logs Menu Options	542
	Query > Logs > System Logs	542
	Sample Query for System Logs	543
	Query > Logs > Object Logs	544
	Understanding Flow Sampling	546
	Flow Sampling	546
	Flow Handling	547
	Flow Aging	547
	TCP State-Based Flow Handling and Aging	548
	TCP State-Based Flow Handling	548
	Protocol Based Flow Aging	548
	Fat Flow	548
	Example: Debugging Connectivity Using Monitoring for Troubleshooting	549
	Using Monitoring to Debug Connectivity	549
Chapter 23	Common Support Answers	555
	Debugging Ping Failures for Policy-Connected Networks	555
	Debugging BGP Peering and Route Exchange in Contrail	561
	Example Cluster	562
	Verifying the BGP Routers	562
	Verifying the Route Exchange	565
	Debugging Route Exchange with Policies	567
	Debugging Peering with an MX Series Router	569
	Debugging a BGP Peer Down Error with Incorrect Family	571
	Configuring MX Peering (iBGP)	573
	Checking Route Exchange with an MX Series Peer	575
	Checking the Route in the MX Series Router	576
	Troubleshooting the Floating IP Address Pool in Contrail	577
	Example Cluster	578
	Example	579
	Example: MX80 Configuration for the Gateway	580
	Ping the Floating IP from the Public Network	582
	Troubleshooting Details	582
	Get the UUID of the Virtual Network	582
	View the Floating IP Object in the API Server	583
	View floating-ips in floating-ip-pools in the API Server	586
	Check Floating IP Objects in the Virtual Machine Interface	588
	View Floating IP Objects in the IFMAP Server View	591
	View the BGP Peer Status on the Control Node	596
	Querying Routes in the Public Virtual Network	596
	Verification from the MX80 Gateway	597
	Viewing the Compute Node Vnsw Agent	599
	Advanced Troubleshooting	602
	Removing Stale Virtual Machines and Virtual Machine Interfaces	604
	Problem Example	604
	Show Virtual Machines	605
	Show Virtual Machines Using Python API	607
	Delete Methods	608

	Troubleshooting Link-Local Services in Contrail	608
	Overview of Link-Local Services	608
	Troubleshooting Procedure for Link-Local Services	608
	Metadata Service	611
	Troubleshooting Procedure for Link-Local Metadata Service	611
Part 5	Contrail Commands and APIs	
Chapter 24	Contrail Commands	615
	contrail-logs (Accessing Log File Messages)	615
	Command-Line Options for Contrail-Logs	615
	Option Descriptions	616
	Example Uses	616
	contrail-status (Viewing Node Status)	618
	contrail-version (Viewing Version Information	619
	service (Managing Services)	621
	Backing Up and Restoring Configurations	622
	Back up Procedure	622
	Restore Procedure	623
	Restore Steps Continued	632
	Finishing	633
Chapter 25	Contrail Application Programming Interfaces (APIs)	635
	Contrail Analytics Application Programming Interfaces (APIs) and User-Visible	
	Entities (UVEs)	635
	User-Visible Entities	635
	Common UVEs in Contrail	637
	Virtual Network UVE	637
	Virtual Machine UVE	637
	vRouter UVE	637
	UVEs for Contrail Nodes	638
	Wild Card Query of UVEs	638
	Filtering UVE Information	639
	Contrail Node Status	645
	Overview	645
	UVE for NodeStatus	645
	Node Status Features	646
	Using Introspect to Get Process Status	651
	contrail-status script	652
	Log and Flow Information APIs	654
	HTTP GET APIs	654
	HTTP POST API	654
	POST Data Format Example	655
	Query Types	656
	Examining Query Status	657
	Examining Query Chunks	657
	Example Queries for Log and Flow Data	657

Working with Neutron	660
Data Structure	660
Network Sharing in Neutron	661
Commands for Neutron Network Sharing	661
Support for Neutron APIs	662
Contrail Neutron Plugin	662
DHCP Options	663
Incompatibilities	663
Support for Amazon VPC APIs on Contrail OpenStack	663
Overview of Amazon Virtual Private Cloud	664
Mapping Amazon VPC Features to OpenStack Contrail Features	664
VPC and Subnets Example	665
Euca2ools CLI for VPC and Subnets	666
Security in VPC: Network ACLs Example	666
Euca2ools CLI for Network ACLs	667
Security in VPC: Security Groups Example	667
Euca2ools CLI for Security Groups	668
Elastic IPs in VPC	669
Euca2ools CLI for Elastic IPs	669
Euca2ools CLI for Route Tables	669
Supported Next Hops	670
Internet Gateway Next Hop Euca2ools CLI	670
NAT Instance Next Hop Euca2ools CLI	671
Example: Creating a NAT Instance with Euca2ools CLI	671

List of Figures

Part 2	Installing and Upgrading Contrail	
Chapter 3	Installing Contrail and Provisioning Roles	11
	Figure 1: Configure > Infrastructure > BGP Routers	31
	Figure 2: BGP Routers Summary	31
	Figure 3: Create BGP Router	32
	Figure 4: Control Nodes	33
	Figure 5: Control Node Details	34
	Figure 6: Control Node Peers Tab	34
Chapter 4	Installation and Configuration Scenarios	37
	Figure 7: Virtual Network Setup With a Simple Gateway	40
	Figure 8: Edit BGP Router Window	49
	Figure 9: Contrail Introspect	70
Chapter 5	Using Contrail with VMware vCenter	73
	Figure 10: Contrail vCenter-Only Solution	75
	Figure 11: Contrail vCenter-as-Compute Solution	75
	Figure 12: Standard Switch Setup	83
	Figure 13: Distributed Switch Setup	84
	Figure 14: PCI Pass-Through with Single Control Data Interface	85
	Figure 15: PCI Pass-Through Setup with Bond Control Interface	86
	Figure 16: SR-IOV Setup	87
	Figure 17: SR-IOV With Single Control Data Interface	88
	Figure 18: SR-IOV With Bond Control Data Interface	89
Chapter 6	Using Server Manager to Automate Provisioning	111
	Figure 19: Server Manager Component Interactions	118
Chapter 7	Extending Contrail to Physical Routers, Bare Metal Servers, Switches, and Interfaces	169
	Figure 20: Configuration Model	171
	Figure 21: Add Physical Router Window	173
	Figure 22: Add Interface Window	174
	Figure 23: High Availability Solution for Contrail ToR Agent	180
	Figure 24: Failure Scenarios	182
	Figure 25: Redundancy for HAProxy	183
	Figure 26: Add OVSDb Managed ToR Window	184
	Figure 27: Contrail Configuration Model	186
	Figure 28: Add Physical Router Window	187
	Figure 29: Add Interface Window	188
	Figure 30: Edit Global Config Window	191

Figure 31: Edit BGP Router Window	192
Figure 32: Edit Physical Router Window for BGP Groups	193
Figure 33: Edit Physical Router Window for Extending Private Networks	194
Figure 34: Edit Network Gateway Window	196
Figure 35: Logical Topology for Floating IP and SNAT	199

Part 3

Chapter 10

Configuring Contrail

Configuring Virtual Networks	237
Figure 36: OpenStack Projects	238
Figure 37: Add Project	238
Figure 38: Add IP Address Management	240
Figure 39: Configure Networks	241
Figure 40: Create Network	241
Figure 41: Configure Networks	243
Figure 42: Policies Window	244
Figure 43: Create Policy Window	244
Figure 44: Configure > Networking > Networks	246
Figure 45: Edit Network	247
Figure 46: Networks Window	248
Figure 47: Create Network Window	248
Figure 48: Create Network Window Subnet Tab	249
Figure 49: OpenStack Networks	250
Figure 50: OpenStack Network Detail, Associated Instances Tab	251
Figure 51: Instances	251
Figure 52: Network Policy	252
Figure 53: Create Network Policy	252
Figure 54: Network Policy	253
Figure 55: Edit Policy Rules	254
Figure 56: Networks Screen	256
Figure 57: Edit Network Policy	256
Figure 58: OpenStack Images Window	257
Figure 59: OpenStack Create An Image Window	258
Figure 60: OpenStack Instances	260
Figure 61: Launch Instance, Details Tab	261
Figure 62: Launch Instance, Networking Tab	262
Figure 63: Configure > Networking > Networks	263
Figure 64: Edit Network	263
Figure 65: Manage Floating IPs	264
Figure 66: Allocate Floating IP Window	265
Figure 67: Associate Floating IP	266
Figure 68: Security Groups	267
Figure 69: Edit Security Group Rules	267
Figure 70: Add Rule	268
Figure 71: Create Security Group	269
Figure 72: Associate Security Group at Launch Instance	269
Figure 73: Global Config Window for VXLAN ID	274
Figure 74: Edit Global Config Window for VXLAN Identifier Mode	275
Figure 75: Edit Network Window	276

	Figure 76: Edit Network Window for VXLAN Identifier	278
	Figure 77: Edit Global Config Window for Encapsulation Priority Order	279
Chapter 11	Example of Deploying a Multi-Tier Web Application Using Contrail	281
	Figure 78: Simple Tiered Web Use Case	282
	Figure 79: Create Floating IP Pool	283
	Figure 80: Allocate Floating IP	283
	Figure 81: Sample Physical Topology for Simple Tiered Web Application	286
	Figure 82: Sample Physical Topology Addressing	287
Chapter 12	Configuring Services	293
	Figure 83: DNS Servers Examples	294
	Figure 84: IPAM and Virtual DNS	295
	Figure 85: Example Usage for NS Record Type	296
	Figure 86: Configure DNS Records	296
	Figure 87: Add DNS	297
	Figure 88: Add DNS Record	298
	Figure 89: Associate IPAMs to DNS	299
	Figure 90: Configure IP Address Management	300
	Figure 91: DNS Server	300
Chapter 13	Configuring Service Chaining	319
	Figure 92: Service Chaining	320
	Figure 93: Contrail Service Chain	320
	Figure 94: Create Networks	325
	Figure 95: Add Service Template	326
	Figure 96: Add Service Template Shared IP	328
	Figure 97: Service Templates	328
	Figure 98: Create Service Instances	329
	Figure 99: Create Service Instances	330
	Figure 100: Service Instance Details	330
	Figure 101: Service Instance Console	331
	Figure 102: Create Policy	331
	Figure 103: Edit Network	332
	Figure 104: Launch Instances	332
	Figure 105: Create Networks	333
	Figure 106: Add Service Template	334
	Figure 107: Create Service Instances	335
	Figure 108: Service Instance Details	336
	Figure 109: Create Policy	336
	Figure 110: Launch Instances	337
	Figure 111: Load Balancing a Service Chain	340
Chapter 18	Optimizing Contrail	419
	Figure 112: Virtual Network With a Private Subnet	438
	Figure 113: Edit Router Window to Enable SNAT	439
	Figure 114: Router Status for SNAT	440
	Figure 115: Instance Details Window	440

Part 4	Monitoring and Troubleshooting Contrail	
Chapter 19	Configuring Traffic Mirroring to Monitor Network Traffic	443
	Figure 116: Packet Capture	444
	Figure 117: Create Analyzer	445
	Figure 118: Analyzer Rules	445
	Figure 119: Create Analyzer Associate Networks	447
	Figure 120: Launch Analyzer VM	447
	Figure 121: Packet Capture Display	448
	Figure 122: Service Templates	448
	Figure 123: Add Service Template	449
	Figure 124: Create Service Instances	450
	Figure 125: Create Policy	451
	Figure 126: Policy Rules	451
	Figure 127: Service Instances View Console	453
	Figure 128: Individual vRouter	454
	Figure 129: Interfaces	454
	Figure 130: Wireshark Packet Display	456
Chapter 20	Understanding Contrail Analytics	459
	Figure 131: Analytics Topology	466
	Figure 132: Add Physical Router Window	469
	Figure 133: Sample Output From a pRouter REST API	469
	Figure 134: Sample Output From a pRouter UVE	470
	Figure 135: Physical Topology Related to a vRouter	471
	Figure 136: Traffic Statistics Graph	471
	Figure 137: List of Active Flows	472
	Figure 138: Underlay Path	473
Chapter 21	Configuring Contrail Analytics	481
	Figure 139: Analytics Scalability	482
Chapter 22	Using Contrail Analytics to Monitor and Troubleshoot the Network	491
	Figure 140: Monitor Menu	492
	Figure 141: Control Nodes Details Tab Window	495
	Figure 142: Controller Introspect Window	496
	Figure 143: BGP Peer Introspect Page	496
	Figure 144: BGP Neighbor Summary Introspect Page	497
	Figure 145: Agent Introspect Page	498
	Figure 146: Monitor > Infrastructure > Dashboard	498
	Figure 147: Dashboard Summary Boxes	499
	Figure 148: Bubble Summary Information	500
	Figure 149: Control Nodes Summary	501
	Figure 150: Individual Control Node—Details Tab	502
	Figure 151: Individual Control Node—Console Tab	503
	Figure 152: Individual Control Node—Peers Tab	505
	Figure 153: Individual Control Node—Routes Tab	506
	Figure 154: vRouters Summary	508
	Figure 155: Individual vRouters—Details Tab	509
	Figure 156: Individual vRouters—Interfaces Tab	511

Figure 157: Individual vRouter	512
Figure 158: Interfaces	512
Figure 159: Individual vRouters—Networks Tab	513
Figure 160: Individual vRouters—ACL Tab	514
Figure 161: Individual vRouters—Flows Tab	515
Figure 162: Individual vRouters—Routes Tab	516
Figure 163: Individual vRouter—Console Tab	517
Figure 164: Analytics Nodes Summary	519
Figure 165: Monitor Analytics Individual Node Details Tab	520
Figure 166: Individual Analytics Node—Generators Tab	521
Figure 167: Individual Analytics Node—QE QueriesTab	521
Figure 168: Analytics Individual Node—Console Tab	522
Figure 169: Config Nodes Summary	524
Figure 170: Individual Config Nodes— Details Tab	524
Figure 171: Individual Config Node—Console Tab	525
Figure 172: Monitor Networking Menu Options	527
Figure 173: Traffic Statistics for Domain Window	527
Figure 174: Monitor > Networking > Projects	528
Figure 175: Monitor Projects Connectivity Details	529
Figure 176: Traffic Statistics Between Networks	529
Figure 177: Projects Instances Summary	530
Figure 178: Instance Traffic Statistics	531
Figure 179: Network Summary	531
Figure 180: Individual Network Connectivity Details—Summary Tab	532
Figure 181: Individual Network— Port Map Tab	532
Figure 182: Individual Network— Port Distribution Tab	533
Figure 183: Individual Network Instances Tab	533
Figure 184: Individual Network Details Tab	534
Figure 185: Query Flow Series Window	535
Figure 186: Flow Series Select	536
Figure 187: Flow Series Filter	537
Figure 188: Example: Query Flow Series	537
Figure 189: Query Flow Series Tabular Results	538
Figure 190: Query Flow Series Graphical Results	538
Figure 191: Flow Records	539
Figure 192: Flow Records Select Window	540
Figure 193: Where Clause Window	540
Figure 194: Flows Query Queue	541
Figure 195: Query > Logs	542
Figure 196: Query > Logs > System Logs	542
Figure 197: Edit Where Clause	544
Figure 198: Sample Query System Logs	544
Figure 199: Query > Logs > Object Logs	545
Figure 200: Navigate to Instance	549
Figure 201: Traffic Statistics for Instance	549
Figure 202: Navigate to Instance	550
Figure 203: Traffic Statistics for Instance	550
Figure 204: Navigate to a3s18 Interfaces	550
Figure 205: Navigate to a3s19 Interfaces	550

	Figure 206: ACL Connectivity a3s18	551
	Figure 207: ACL Connectivity a3s19	551
	Figure 208: Routes default-domain:demo:vn0:vn0	551
	Figure 209: Routes default-domain:demo:vn16:vn16	552
	Figure 210: Verify Route and Next Hop a3s18	552
	Figure 211: Verify Route and Next Hop a3s19	553
	Figure 212: Flows for a3s18	553
	Figure 213: Flows for a3s19	553
Chapter 23	Common Support Answers	555
	Figure 214: Virtual Machine Status Window	556
	Figure 215: Tap Interface Status Window	556
	Figure 216: Policies, Attachments, and Traffic Rule Status Window	557
	Figure 217: Virtual Network Policy Configuration Window	557
	Figure 218: Virtual Network Route Information Window	557
	Figure 219: Flow and Dropstats Command List	558
	Figure 220: Flow Command Output Window	558
	Figure 221: Fetch Flow Record Window	559
	Figure 222: Unresolved IP Address Window	559
	Figure 223: Unresolved Flow Details Window	560
	Figure 224: Protocol-Specific Flow Sample	560
	Figure 225: Protocol-Specific Flow Sample With Deny Action	561
	Figure 226: Sample Output, BGP Routers:	563
	Figure 227: Sample Output, BGP Router References:	564
	Figure 228: Sample Output, BGP Neighbor Config:	564
	Figure 229: Sample Output, BGP Peering Config:	565
	Figure 230: Sample Output, BGP Neighbor States:	565
	Figure 231: Sample Output, Show Routing Instance:	566
	Figure 232: Sample Output, Validate Route:	566
	Figure 233: Sample Output, Validate L3vpn Table:	567
	Figure 234: Sample Output, Validate L3vpn Table, Scrolled:	567
	Figure 235: Create Policy Window	568
	Figure 236: Sample Output, Validate Import Target:	568
	Figure 237: Sample Output, Route Import:	569
	Figure 238: Edit Global ASN Window	569
	Figure 239: Create BGP Peer Window	570
	Figure 240: Sample BGP Peer UVE	571
	Figure 241: Sample Established BGP Peer UVE	572
	Figure 242: Edit Global ASN Window	573
	Figure 243: Create BGP Peer Window	573
	Figure 244: Sample Established IBGP Peer UVE	574
	Figure 245: Sample Established IBGP Peer Introspect Window	574
	Figure 246: Routing Instance Route Table	575
	Figure 247: Routing Instance Route Table	575
	Figure 248: Routing Instance Public IPv4 Route Table	575
	Figure 249: Virtual Machine Routing Instance Public IPv4 Route Table	576
	Figure 250: BGP Routing Instance Route Table	576

List of Tables

	About the Documentation	xxxi
	Table 1: Notice Icons	xxxii
	Table 2: Text and Syntax Conventions	xxxii
Part 2	Installing and Upgrading Contrail	
Chapter 3	Installing Contrail and Provisioning Roles	11
	Table 3: Create BGP Router Fields	32
Chapter 4	Installation and Configuration Scenarios	37
	Table 4: Contrail and OpenStack Nova Docker Support	57
Chapter 6	Using Server Manager to Automate Provisioning	111
	Table 5: Server Manager Parameters	120
	Table 6: Server Manager Add Server Command Options	124
	Table 7: Server Manager Delete Server Command Options	126
	Table 8: Server Manager Show Server Command Options	127
	Table 9: Server Manager Add Cluster Command Options	128
	Table 10: Server Manager Delete Cluster Command Options	129
	Table 11: Server Manager Show Cluster Command Options	130
Chapter 7	Extending Contrail to Physical Routers, Bare Metal Servers, Switches, and Interfaces	169
	Table 12: Contrail Objects in the OVSDb	171
Part 3	Configuring Contrail	
Chapter 10	Configuring Virtual Networks	237
	Table 13: Add IP Address Management Fields	240
	Table 14: Create Network Fields	241
	Table 15: Create Policy Fields	245
	Table 16: Create Network Fields	249
	Table 17: Edit Policy Rules Fields	254
	Table 18: Create An Image Fields	258
	Table 19: Launch Instance Details Tab Fields	261
	Table 20: Add Rule Fields	268
Chapter 11	Example of Deploying a Multi-Tier Web Application Using Contrail	281
	Table 21: Sample Addressing Scheme for Example	285
Chapter 12	Configuring Services	293
	Table 22: DNS Record Types Supported	295
	Table 23: Add DNS Fields	297

	Table 24: Add DNS Record Fields	298
	Table 25: Associate IPAMs to DNS Fields	299
	Table 26: DNS Modes	300
	Table 27: DNS Scripts	301
Chapter 13	Configuring Service Chaining	319
	Table 28: Add Service Template Fields	326
	Table 29: Create Service Instances Fields	329
	Table 30: Add Service Template Fields	334
	Table 31: Create Service Instances Fields	335
	Table 32: Health Check Configurable Fields	368
Chapter 18	Optimizing Contrail	419
	Table 33: vif Fields	421
Part 4	Monitoring and Troubleshooting Contrail	
Chapter 19	Configuring Traffic Mirroring to Monitor Network Traffic	443
	Table 34: Analyzer Rule Fields	446
	Table 35: Add Service Template Fields	449
	Table 36: Create Service Instances Fields	450
	Table 37: Add Rule Fields	451
Chapter 22	Using Contrail Analytics to Monitor and Troubleshoot the Network	491
	Table 38: Monitor Menu Options	492
	Table 39: Dashboard Summary Boxes	499
	Table 40: Control Nodes Summary Fields	501
	Table 41: Individual Control Node—Details Tab Fields	502
	Table 42: Control Node: Console Tab Fields	503
	Table 43: Control Node: Peers Tab Fields	505
	Table 44: Control Node: Routes Tab Fields	506
	Table 45: vRouters Summary Fields	508
	Table 46: vRouters Details Tab Fields	510
	Table 47: vRouters: Interfaces Tab Fields	511
	Table 48: vRouters: Networks Tab Fields	513
	Table 49: vRouters: ACL Tab Fields	514
	Table 50: vRouters: Flows Tab Fields	515
	Table 51: vRouters: Routes Tab Fields	516
	Table 52: Control Node: Console Tab Fields	517
	Table 53: Fields on Analytics Nodes Summary	519
	Table 54: Monitor Analytics Individual Node Details Tab Fields	520
	Table 55: Monitor Analytics Individual Node Generators Tab Fields	521
	Table 56: Analytics Node QE Queries Tab Fields	521
	Table 57: Monitor Analytics Individual Node Console Tab Fields	522
	Table 58: Config Nodes Summary Fields	524
	Table 59: Individual Config Nodes— Details Tab Fields	525
	Table 60: Individual Config Node—Console Tab Fields	525
	Table 61: Projects Summary Fields	528
	Table 62: Projects Summary Fields	528
	Table 63: Projects Instances Summary Fields	530

Table 64: Network Summary Fields	531
Table 65: Query Flow Series Fields	535
Table 66: Query Flow Records Fields	539
Table 67: Query Flow Records Fields	541
Table 68: Query System Logs Fields	543
Table 69: Object Logs Query Fields	545

Part 5

Contrail Commands and APIs

Chapter 25

Contrail Application Programming Interfaces (APIs)	635
Table 70: Amazon VPC and OpenStack Contrail Feature Comparison	664

About the Documentation

- Documentation and Release Notes on page xxxi
- Documentation Conventions on page xxxi
- Documentation Feedback on page xxxiii
- Requesting Technical Support on page xxxiv

Documentation and Release Notes

To obtain the most current version of all Juniper Networks® technical documentation, see the product documentation page on the Juniper Networks website at <https://www.juniper.net/documentation/>.

If the information in the latest release notes differs from the information in the documentation, follow the product Release Notes.

Juniper Networks Books publishes books by Juniper Networks engineers and subject matter experts. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration. The current list can be viewed at <https://www.juniper.net/books>.

Documentation Conventions

Table 1 on page xxxii defines notice icons used in this guide.

Table 1: Notice Icons







Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.
	Tip	Indicates helpful information.
	Best practice	Alerts you to a recommended use or implementation.

Table 2 on page xxxii defines the text and syntax conventions used in this guide.

Table 2: Text and Syntax Conventions

Convention	Description	Examples
Bold text like this	Represents text that you type.	To enter configuration mode, type the configure command: user@host> configure
Fixed-width text like this	Represents output that appears on the terminal screen.	user@host> show chassis alarms No alarms currently active
<i>Italic text like this</i>	<ul style="list-style-type: none"> Introduces or emphasizes important new terms. Identifies guide names. Identifies RFC and Internet draft titles. 	<ul style="list-style-type: none"> A policy <i>term</i> is a named structure that defines match conditions and actions. <i>Junos OS CLI User Guide</i> RFC 1997, <i>BGP Communities Attribute</i>
<i>Italic text like this</i>	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name: [edit] root@# set system domain-name <i>domain-name</i>

Table 2: Text and Syntax Conventions (continued)

Convention	Description	Examples
Text like this	Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components.	<ul style="list-style-type: none">To configure a stub area, include the stub statement at the [edit protocols ospf area area-id] hierarchy level.The console port is labeled CONSOLE.
< > (angle brackets)	Encloses optional keywords or variables.	stub <default-metric <i>metric</i>>;
(pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	broadcast multicast (<i>string1</i> <i>string2</i> <i>string3</i>)
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	rsvp { # Required for dynamic MPLS only
[] (square brackets)	Encloses a variable for which you can substitute one or more values.	community name members [<i>community-ids</i>]
Indentation and braces ({ })	Identifies a level in the configuration hierarchy.	[edit] routing-options { static { route default { nexthop <i>address</i>; retain; } } }
;(semicolon)	Identifies a leaf statement at a configuration hierarchy level.	
GUI Conventions		
Bold text like this	Represents graphical user interface (GUI) items you click or select.	<ul style="list-style-type: none">In the Logical Interfaces box, select All Interfaces.To cancel the configuration, click Cancel.
> (bold right angle bracket)	Separates levels in a hierarchy of menu selections.	In the configuration editor hierarchy, select Protocols>Ospf .

Documentation Feedback

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation. You can provide feedback by using either of the following methods:

- Online feedback rating system—On any page of the Juniper Networks TechLibrary site at <https://www.juniper.net/documentation/index.html>, simply click the stars to rate the content, and use the pop-up form to provide us with information about your experience. Alternately, you can use the online feedback form at <https://www.juniper.net/documentation/feedback/>.

- E-mail—Send your comments to techpubs-comments@juniper.net. Include the document or topic name, URL or page number, and software version (if applicable).

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or Partner Support Service support contract, or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at <https://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf>.
- Product warranties—For product warranty information, visit <https://www.juniper.net/support/warranty/>.
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: <https://www.juniper.net/customers/support/>
- Search for known bugs: <https://prsearch.juniper.net/>
- Find product documentation: <https://www.juniper.net/documentation/>
- Find solutions and answer questions using our Knowledge Base: <https://kb.juniper.net/>
- Download the latest versions of software and review release notes: <https://www.juniper.net/customers/csc/software/>
- Search technical bulletins for relevant hardware and software notifications: <https://kb.juniper.net/InfoCenter/>
- Join and participate in the Juniper Networks Community Forum: <https://www.juniper.net/company/communities/>
- Open a case online in the CSC Case Management tool: <https://www.juniper.net/cm/>

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: <https://entitlementsearch.juniper.net/entitlementsearch/>

Opening a Case with JTAC

You can open a case with JTAC on the Web or by telephone.

- Use the Case Management tool in the CSC at <https://www.juniper.net/cm/>.
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see <https://www.juniper.net/support/requesting-support.html>.

PART 1

Overview

- [Understanding Contrail Controller on page 3](#)

CHAPTER 1

Understanding Contrail Controller

- [Contrail Overview on page 3](#)
- [Contrail Description on page 4](#)

Contrail Overview

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

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

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

Using Contrail, a tenant can define, manage, and control the connectivity, services, and security policies of the virtual network. The tenant or other users can use the self-service graphical user interface to easily create virtual network nodes, add and remove IP services (such as firewall, load balancing, DNS, and the like) to their virtual networks, then connect the networks using traffic policies that are simple to create and apply. Once created, policies can be applied across multiple network nodes, changed, added, and deleted, all from a simple browser-based interface.

Contrail can be used with open cloud orchestration systems such as OpenStack or CloudStack. 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 on page 4](#)

Contrail Description

- [Contrail Major Components on page 4](#)
- [Contrail Solution on page 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 on page 3](#)
 - [Installation Overview on page 11](#)

PART 2

Installing and Upgrading Contrail

- [Supported Platforms and Server Requirements on page 9](#)
- [Installing Contrail and Provisioning Roles on page 11](#)
- [Installation and Configuration Scenarios on page 37](#)
- [Using Contrail with VMware vCenter on page 73](#)
- [Using Server Manager to Automate Provisioning on page 111](#)
- [Extending Contrail to Physical Routers, Bare Metal Servers, Switches, and Interfaces on page 169](#)
- [Installing and Using Contrail Storage on page 215](#)
- [Upgrading Contrail Software on page 231](#)

CHAPTER 2

Supported Platforms and Server Requirements

- [Supported Platforms on page 9](#)
- [Server Requirements on page 10](#)

Supported Platforms

Contrail Networking Release 3.0 is supported on the OpenStack Juno and Kilo releases, on the following operating system versions:

- Ubuntu 14.04.2
- CentOS 7.2
- RHOSP7
- vCenter 5.5
 - vCenter is limited to Ubuntu 14.04.2 (Linux kernel version: 3.13.0-40-generic).
 - vCenter 6.0 is also supported as Beta.

Contrail Cloud Release 3.0 is only supported on Ubuntu 14.04.2, and is no longer supported on Ubuntu 12.04 or CentOS 6.x.

Additionally, OpenStack Icehouse is no longer supported.

Contrail Release 3.0.1 provides support for OpenStack Liberty on Ubuntu 14.04.2. Liberty is the 12th release of the OpenStack open source software for building public, private, and hybrid clouds.

Following is the supported Linux kernel version for each distribution supported on Contrail Release 3.0.

- CentOS 7.1—Linux kernel version 3.10.0-229.el7
- Ubuntu 14.04.2— kernel version 3.13.0-40-generic
- Red Hat 7.1—Linux kernel version 3.10.0-229.el7
- vCenter 5.5—vRouter VM on Ubuntu 14.04 kernel version 3.13.0-40-generic

Server Requirements

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

Each server must have a minimum of:

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

For a production environment, each server must have a minimum of:

- 256 GB memory
- 500 GB hard drive
- 16 CPU cores



NOTE: If you are using Contrail Storage, additional hardware requirements can be found in [“Installing and Using Contrail Storage” on page 215](#), Hardware Specifications.

Related Documentation

- [Installation Overview on page 11](#)
- [Downloading Installation Software on page 12](#)

CHAPTER 3

Installing Contrail and Provisioning Roles

- [Installation Overview on page 11](#)
- [Downloading Installation Software on page 12](#)
- [Installing the Operating System and Contrail Packages on page 13](#)
- [Configuring System Settings on page 14](#)
- [Installing the Contrail Packages, Part One \(CentOS or Ubuntu\) on page 15](#)
- [Setting Up the Testbed Definitions File on page 17](#)
- [Testbed Definitions File Settings for Deploying Contrail with an Existing OpenStack Node on page 20](#)
- [Supporting Multiple Interfaces on Servers and Nodes on page 22](#)
- [Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines on page 27](#)
- [Configuring the Control Node on page 30](#)
- [Adding or Removing a Compute Node in an Existing Contrail Cluster on page 35](#)

Installation Overview

The Contrail Controller is typically installed on multiple servers. The base software image is installed on all servers to be used, then provisioning scripts are run that launch role-based components of the software.

The roles used for the installed system include:

- *cfgm*—Runs Contrail configuration manager (config-node)
- *openstack*—Runs OpenStack services such as Nova, Quantum, and the like
- *collector*—Runs monitoring and analytics services
- *compute*—Runs vRouter service and launches tenant virtual machines (VMs)
- *control*—Runs the control plane service
- *database*—Runs analytics and configuration database services
- *webui*—Runs the administrator web-based user interface service

The roles are run on multiple servers in an operating installation. A single node can have multiple roles. The roles can also run on a single server for testing or demonstration purposes.

[“Installing the Operating System and Contrail Packages” on page 13](#) describes installing the Contrail Controller software onto multiple servers.

Your account team can help you determine the number of servers needed for your specific implementation.

**Related
Documentation**

- [Server Requirements on page 10](#)
- [Downloading Installation Software on page 12](#)
- [Installing the Operating System and Contrail Packages on page 13](#)
- [Installation Overview on page 11](#)
- [Downloading Installation Software on page 12](#)
- [Installing the Operating System and Contrail Packages on page 13](#)
- [Configuring System Settings on page 14](#)
- [Installing the Contrail Packages, Part One \(CentOS or Ubuntu\) on page 15](#)
- [Setting Up the Testbed Definitions File on page 17](#)
- [Testbed Definitions File Settings for Deploying Contrail with an Existing OpenStack Node on page 20](#)
- [Supporting Multiple Interfaces on Servers and Nodes on page 22](#)
- [Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines on page 27](#)
- [Configuring the Control Node on page 30](#)
- [Adding or Removing a Compute Node in an Existing Contrail Cluster on page 35](#)

Downloading Installation Software

All components necessary for installing the Contrail Controller are available as:

- an **RPM** file (**contrail-install-packages-1.xx-xxx.el6.noarch.rpm**) that can be used to install the Contrail system on an appropriate CentOS operating system.
- a **Debian** file (**contrail-install-packages-1.xx-xxx-xxxxxx_all.deb**) that can be used to install the Contrail system on an appropriate Ubuntu operating system.

Versions 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
<http://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

Related Documentation

- [Installing the Operating System and Contrail Packages on page 13](#)
- [Configuring System Settings on page 14](#)
- [Setting Up the Testbed Definitions File on page 17](#)
- [Installing the Contrail Packages, Part One \(CentOS or Ubuntu\) on page 15](#)
- [Download Software](#)

Installing the Operating System and Contrail Packages

Install the stock CentOS or Ubuntu operating system image appropriate for your version of Contrail onto the server. See “[Supported Platforms](#)” on [page 9](#). Then install Contrail packages separately.

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

1. Install a CentOS or Ubuntu minimal distribution as desired on all servers. Follow the published operating system installation procedure for the selected operating system; refer to the website for the operating system.
2. After rebooting all of the servers after installation, verify that you can log in to each of them using the root password defined during installation.
3. After the initial installations on all servers, configure some items specific to your systems, (see “[Configuring System Settings](#)” on [page 14](#)), then begin the first part of the installation (see “[Installing the Contrail Packages, Part One \(CentOS or Ubuntu\)](#)” on [page 15](#)).

Related Documentation

- [Supported Platforms on page 9](#)
- [Installation Overview on page 11](#)
- [Downloading Installation Software on page 12](#)
- [Installing the Operating System and Contrail Packages on page 13](#)
- [Configuring System Settings on page 14](#)
- [Installing the Contrail Packages, Part One \(CentOS or Ubuntu\) on page 15](#)
- [Setting Up the Testbed Definitions File on page 17](#)
- [Supporting Multiple Interfaces on Servers and Nodes on page 22](#)
- [Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines on page 27](#)
- [Configuring the Control Node on page 30](#)
- [Adding or Removing a Compute Node in an Existing Contrail Cluster on page 35](#)
- [Download Software](#)

Configuring System Settings

After installing the base image on all servers being used in the installation, and before running role provisioning scripts, perform the following steps to configure items specific to your environment.

Perform these configuration steps each time you perform an initial installation or an upgrade to a new release.

To configure system settings:

1. Update the **/etc/resolv.conf** file with name server information specific to your system.
2. Update the **/etc/sysconfig/network** file with the hostname and domain information specific to your system.
3. Configure the LAN port with network information specific to your system:
 - a. Use the **show ifconfig -a** command to determine which LAN port you are using, as this might not be obvious on some systems due to the ways interfaces can be named.
 - b. Update the appropriate interface configuration file in **/etc/sysconfig/network-scripts/ifcfg-*<int name>*** using the following guidelines:
 - **IPADDR** = *<IP of the host you want to assign>*
 - **NETMASK** = *<e.g. 255.255.255.0>*
 - **GATEWAY** = *<gateway router address>*

- **BOOTPROTO** — delete this, or change **dhcp** to **static**
- Other settings can remain unchanged.

**Related
Documentation**

- [Installing the Contrail Packages, Part One \(CentOS or Ubuntu\) on page 15](#)
- [Setting Up the Testbed Definitions File on page 17](#)

Installing the Contrail Packages, Part One (CentOS or Ubuntu)

This procedure includes instructions for installing Contrail for either a CentOS-based system or an Ubuntu-based system. In each step, be sure to follow the instructions for your operating system type.

All installation files are available from

<http://www.juniper.net/support/downloads/?p=contrail#sw> .

CentOS Systems Contrail packages for CentOS are provided either as part of the Contrail ISO installation or separately in an RPM file with the format: **contrail-install-packages-x.xx-xxx~openstack_version.el6.noarch.rpm**, where **x.xx-xxx~openstack_version** represents the release number, build number, and OpenStack common version name for the included Contrail install packages.

If you already have a compatible operating system installed, you can choose to copy only the Contrail packages after the base operating system installation is complete. The base operating system can be installed using netboot or a USB, using installation instructions for that operating system.

Ubuntu Systems Contrail packages for Ubuntu are provided only as packages in a Debian file of the format: **contrail-install-packages-x.xx-xxx~openstack_version_all.deb**, where **x.xx-xxx~openstack_version** represents the release number, build number, and OpenStack common version name for the included Contrail install packages.

It is expected that you already have a compatible Ubuntu operating system installed before installing the Contrail packages. For more information, see [“Supported Platforms” on page 9](#).



NOTE: If the stock kernel version of your Ubuntu system is older than the required version, the following Fabric task can be used to upgrade the kernel version in all nodes:

```
cd /opt/contrail/utils; fab upgrade_kernel_all
```

Installing Contrail Packages for CentOS or Ubuntu

This procedure provides instructions for installing Contrail packages onto either a CentOS-based system or an Ubuntu-based system.

1. Ensure that a compatible base operating system has been installed, using the installation instructions for that system.

2. Download the appropriate Contrail install packages file from <http://www.juniper.net/support/downloads/?p=contrail#sw> :

CentOS: **contrail-install-packages-x.xx-xxx~openstack_version.el6.noarch.rpm**

Ubuntu: **contrail-install-packages-x.xx-xxx~openstack_version_all.deb**

3. Copy the downloaded Contrail install packages file to **/tmp/** on the first server for your system installation.

4. On one of the config nodes in your cluster, copy the Contrail packages as follows:

```
CentOS: scp  
<id@server>:/path/to/contrail-install-packages-x.xx-xxx~openstack_version.el6.noarch.rpm  
/tmp
```



```

Ubuntu: scp
<id@server>:/path/to/contrail-install-packages-x.xx-xxx~openstack_version_all.deb
/tmp

```

5. Install the Contrail packages:

```

CentOS: yum localinstall
/tmp/contrail-install-packages-x.xx-xxx~openstack_version.el6.noarch.rpm

```

```

Ubuntu: dpkg -i /tmp/contrail-install-packages-x.xx-xxx~openstack_version_all.deb

```

6. Run the **setup.sh** script. This step creates the Contrail packages repository as well as the Fabric utilities (located in **/opt/contrail/utls**) needed for provisioning:

```
cd /opt/contrail/contrail_packages; ./setup.sh
```

7. Populate the **testbed.py** definitions file, see [“Setting Up the Testbed Definitions File” on page 17](#).



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

Related Documentation

- [Download Software](#)
- [Supported Platforms on page 9](#)
- [Setting Up the Testbed Definitions File on page 17](#)
- [Supporting Multiple Interfaces on Servers and Nodes on page 22](#)
- [Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines on page 27](#)
- [Configuring the Control Node on page 30](#)

Setting Up the Testbed Definitions File

Populate a testbed definitions file, **/opt/contrail/utls/fabfile/testbeds/testbed.py**, with parameters specific to your system, then run the fab commands as provided in [“Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines” on page 27](#) to launch the role-based provisioning script tasks.

You can view *example* testbed files on any node in the controller at:

- `/opt/contrail/utils/fabfile/testbeds/testbed_multibox_example.py` for a multiple server system

For a list of all available Fabric commands, refer to the `/opt/contrail/utils/README.fabric` file.

To define the following parameters within the `testbed.py` file:

1. Provide host strings for the nodes in the cluster. Replace the addresses shown in the example with the actual IP addresses of the hosts in your system.

```
host1 = 'root@1.1.1.1'
host2 = 'root@1.1.1.2'
host3 = 'root@1.1.1.3'
host4 = 'root@1.1.1.4'
host5 = 'root@1.1.1.5'
```

2. Define external routers (MX Series routers and the like) to which the virtual network controller control nodes are peered.

```
ext_routers = [('mx1', '1.1.1.253'), ('mx2', '1.1.1.252')]
```

If there are no external routers, define

```
ext_routers = []
```

3. Provide the BGP autonomous system number.

```
router_asn = 64512
```



NOTE: The default ASN 64512 is a private ASN number. A private ASN should be used if an AS is only required to communicate via BGP with a single provider. As the routing policy between the AS and the provider is not visible in the Internet, a private ASN can be used for this purpose. IANA has reserved AS 64512 through to AS 65535 to be used as private ASNs. If these circumstances do not apply, you cannot use the default or any other private ASN number.

4. Define the host on which the Fabric tasks are invoked. Replace the address shown in the example with the actual IP address of the host in your system.

```
host_build = 'user@10.10.10.10'
```

5. Define which hosts operate with which roles.

For multinode setups:

```
env.roledefs = {
    'all': [host1, host2, host3, host4, host5],
    'database': [host1, host2, host3],
```

```

'cfgm': [host1, host2],
'control': [host1, host2],
'compute': [host4, host5],
'collector': [host1, host2, host3],
'webui': [host1],
'build': [host_build],
}

```

For single node all-in-one setups:

```

env.roledefs = {
  'all': [host1],
  'database': [host1],
  'cfgm': [host1],
  'control': [host1],
  'compute': [host1],
  'collector': [host1],
  'webui': [host1],
  'build': [host_build],
}

```

6. Define password credentials for each of the hosts.

```

env.password = 'secret' # Required only for releases prior to 1.10
env.passwords = {
  host1: 'secret',
  host2: 'secret',
  host3: 'secret',
  host4: 'secret',
  host5: 'secret',
}

```



NOTE: In releases earlier than Contrail Release 1.10, ensure that *both* the `env.password` and `env.passwords` variables are set.



NOTE: Set appropriate permissions for the `testbed.py` file because it contains host credentials.

If your system servers and nodes have multiple interfaces, refer to [“Supporting Multiple Interfaces on Servers and Nodes” on page 22](#) for information about setting up the `testbed.py` file for your system.

To deploy a Contrail High Available cluster, refer to [“Juniper OpenStack High Availability” on page 395](#) for information about setting up the `testbed.py` file for your system.

To deploy with an existing OpenStack, refer to [“Testbed Definitions File Settings for Deploying Contrail with an Existing OpenStack Node” on page 20](#) for `testbed.py` file definitions.

When you are finished, continue on to [“Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines” on page 27](#).

- Related Documentation**
- [Supporting Multiple Interfaces on Servers and Nodes on page 22](#)
 - [Testbed Definitions File Settings for Deploying Contrail with an Existing OpenStack Node on page 20](#)
 - [Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines on page 27](#)

Testbed Definitions File Settings for Deploying Contrail with an Existing OpenStack Node

It is possible to deploy Contrail when there is already an existing OpenStack node on your system.

The following shows additional **testbed.py** definitions that are required to deploy Contrail when there is already an existing OpenStack node.

1. Update the OpenStack admin password in the **testbed.py**.

For example, use the following to update the OpenStack admin password.

```
env.openstack_admin_password = '<password>'
```

2. Update the Keystone environment section.

In the following example, the entries shown in the **env.keystone** section override the following previously-supported options:

- **service_token**
- **keystone_ip**
- **keystone_admin_user**
- **keystone_admin_password**
- **region_name**

Options include the following:

- **keystone_ip**—IP Address of the Keystone server. If using Openstack high availability (HA), provide the OpenStack VIP.
- **auth_protocol**—The authentication protocol used by Keystone.
- **auth_port**—The authentication port used by Keystone.
- **admin_token**—The admin token of Keystone.
- **admin_user**—The admin user name of Keystone.
- **admin_password**—The password of the admin user of Keystone.
- **nova_password**—The password of the Nova service.
- **neutron_password**—The password of the Neutron networking service.

- **service_tenant**—The tenant name of services such as Nova, Neutron, Glance, and so on.
- **admin_tenant**—The name of the tenant admin user.
- **region_name**—The OpenStack region to use. The default is RegionOne.
- **insecure**—The insecure option set for Keystone. The default is False.



NOTE: The option "insecure" is applicable only when the protocol is https.

- **manage_neutron**—Option to configure a Neutron user or role in the Keystone server. The default = 'yes'.

The following is a sample configuration:

```
env.keystone = {
    'keystone_ip'      : '<ip address>',
    'auth_protocol'    : 'http',
    'auth_port'        : '35357',
    'admin_token'      : '$ABC123',
    'admin_user'       : 'admin',
    'admin_password'   : '<password>',
    'nova_password'    : '$ABC123',
    'neutron_password' : '$ABC123',
    'service_tenant'   : 'services',
    'admin_tenant'     : 'admin',
    'region_name'      : 'RegionOne',
    'insecure'         : 'False',
    'manage_neutron'   : 'no',
}
```

3. Update the OpenStack environment section.

Options include the following:

- **service_token**—the tenant name of services such as Nova, Neutron, Glance, and so on.
- **amqp_host**—the IP address of the Advanced Message Queuing Protocol (AMQP) server to be used in the OpenStack node. If using OpenStack high availability, provide the OpenStack VIP.
- **manage_amqp**—Option to manage separate AMQP for OpenStack services in OpenStack nodes. The default = 'no'. If set to 'yes', AMQP is provisioned in OpenStack nodes, and OpenStack services use the AMQP in OpenStack nodes instead of config nodes. The **amqp_host** is neglected if the **manage_amqp** value is set.
- **osapi_compute_workers**—The default is 40. For a low memory system, reduce the osapi compute workers thread.
- **conductor_workers**—The default is 40. For a low memory system, reduce the conductor workers thread.

The following is a sample configuration:

```
env.openstack = {
  'service_token'      : '$ABC123',
  'amqp_host'          : '<ip address>',
  'manage_amqp'        : 'no',
  'osapi_compute_workers' : 40,
  'conductor_workers'  : 40,
}
```

4. Update the configuration options for the Contrail config node in the OpenStack node.

Options include the following:

- **amqp_hosts**—List of customer-deployed AMQP servers to be used by config services.
- **amqp_port**—Port of the customer-deployed AMQP servers.

The following is a sample configuration:

```
env.cfgm = {
  'amqp_hosts' : ['<ip address>'],
  'amqp_port'  : '5672'
}
```

Related Documentation

- [Setting Up the Testbed Definitions File on page 17](#)
- [Supporting Multiple Interfaces on Servers and Nodes on page 22](#)
- [Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines on page 27](#)

Supporting Multiple Interfaces on Servers and Nodes

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

- [Support for Multiple Interfaces on page 22](#)
- [Server Interface Examples on page 24](#)
- [Interface Naming and Configuration Management on page 24](#)
- [Setting Up Interfaces and Installing on page 25](#)
- [Sample testbed.py File With Exclusive Interfaces on page 25](#)

Support for Multiple Interfaces

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

Examples of control traffic include the following:

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

In Contrail Release 1.10 and later, control and data must share the same interface, configured in the **testbed.py** file in a section named **control_data**.

Number of cfm Nodes Supported

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

Uneven Number of Database Nodes Required

In Contrail Release 1.10 and later, Apache ZooKeeper resides on the database node. Because a ZooKeeper ensemble operates most effectively with an odd number of nodes, it is required to have an odd number (3, 5, 7, and so on) of database nodes in a Contrail system.

Support for VLAN Interfaces

A VLAN ID can also be specified in the **testbed.py** file under the **control_data** section, similar to the following example:

```
control_data= { host1: { 'ip': '<ip address>', 'gw': '<ip address>', 'device':
                    'bond0', 'vlan': '20'},
                host2: { 'ip': '<ip address>', 'gw': '<ip
                    address>', 'device': 'bond0', 'vlan': '20'} }
```

Support for Bonding Options

Contrail provides support for bond interface options.

The default bond interface options are:

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

In the **testbed.py** bond section, anything other than name and member are treated as a bond interface option, and provisioned as such. The following is an example:

```
bond= { host1: { 'name': 'bond0', 'member': ['p2p0p2', 'p2p0p3'], 'lacp_rate': 'slow' }
```

Support for Static Route Options

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

The following shows the use of the **static_route** stanza in the **testbed.py** file to configure static routes in host2 and host5.

```
static_route = {
    host2 : [{ 'ip': '<ip address>', 'netmask' : '<ip address>',
               'gw': '<ip address>', 'intf': 'bond0' },
```

```
{ 'ip': '<ip address>', 'netmask' : '<ip address>',  
  
  'gw': '<ip address>', 'intf': 'bond0' }],  
  host5 : [{ 'ip': '<ip address>', 'netmask' : '<ip address>',  
            'gw': '<ip address>', 'intf': 'bond0'  
  }],  
}
```

Server Interface Examples

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

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

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

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

Interface Naming and Configuration Management

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

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

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

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

Setting Up Interfaces and Installing

As part of the provisioning scheme, there are two additional commands that the administrator can use to set up control and data interfaces.

The **fab setup_interface** command creates bond interface configurations, if there is a corresponding configuration in the **testbed.py** file (see the sample **testbed.py** file in “Sample testbed.py File With Exclusive Interfaces” on page 25).

When you use the **fab setup_interface** command, the interface configurations are generated with the syntax (**ifcfg-* files**), which is needed for the **network service**.

The **fab add_static_route** command creates static routes in a node, if there is a corresponding configuration in the **testbed.py** file (see the sample **testbed.py** file in “Sample testbed.py File With Exclusive Interfaces” on page 25).

The following is a typical work flow for setting up a cluster with multiple interfaces:

```
Set env.interface_rename = True in the testbed.py file (meaning: install the
contrail-interface-name package on compute nodes)

fab install_contrail (meaning: change the testbed.py file with the renamed interface
name)

fab setup_interface
fab add_static_route
fab setup_all
```



NOTE: The **fab setup_interface** command and **fab add_static_route** command can be executed simultaneously by using the **fab setup_network** command.

In cases where the **fab setup_interface** command is not used for setting up the interfaces, configurations for the data interface are migrated as part of the **vrouter** installation on the compute nodes.

If the data interface is a bond interface, the bond member interfaces are reconfigured into network service based configurations using appropriate **ifcfg** script files.

Sample testbed.py File With Exclusive Interfaces

The following is a sample **testbed.py** definitions file that shows the configuration for exclusive interfaces for management and control and for data networks.

```
#testbed file
from fabric.api import env
os_username = 'admin'
os_password = '<password>'
os_tenant_name = 'demo'

host1 = 'host@<ip address>'
host2 = 'host@<ip address>'
host3 = 'host@<ip address>'
```

```

host4 = 'host@<ip address>'
host5 = 'host@<ip address>'
host6 = 'host@<ip address>'
host7 = 'host@<ip address>'
host8 = 'host@<ip address>'

ext_routers = [('mx1', '<ip address>')] router_asn = <asn> public_vn_rtgt =
10003 public_vn_subnet = '<ip address>'

host_build = 'host@<ip address>'

env.roledefs = {
    'all': [host1, host2, host3, host4, host5, host6, host7, host8],
    'cfgm': [host1],
    'openstack': [host6],
    'webui': [host7],
    'control': [host4, host3],
    'compute': [host2, host5],
    'collector': [host2, host3],
    'database': [host8],
    'build': [host_build],
}

env.hostnames = {
    'all': ['nodea10', 'nodea4', 'nodea2', 'nodeb2',
'nodeb12', 'nodea32', 'nodec36', 'nodec31']
}

bond= {
    host2 : { 'name': 'bond0', 'member': ['p2p0p0', 'p2p0p1', 'p2p0p2', 'p2p0p3'],
'mode': 'balance-xor' },
    host5 : { 'name': 'bond0', 'member': ['p4p0p0', 'p4p0p1', 'p4p0p2', 'p4p0p3'],
'mode': 'balance-xor' }, }

control_data = {
    host1 : { 'ip': '<routing prefix address>', 'gw' : '<ip address>',
'device': 'eth0' },
    host2 : { 'ip': '<routing prefix address>', 'gw' : '<ip address>',
'device': 'p0p25p0' },
    host3 : { 'ip': '<routing prefix address>', 'gw' : '<ip address>',
'device': 'eth0' },
    host4 : { 'ip': '<routing prefix address>', 'gw' : '<ip address>',
'device': 'eth3' },
    host5 : { 'ip': '<routing prefix address>', 'gw' : '<ip address>',
'device': 'p6p0p1' },
    host6 : { 'ip': '<routing prefix address>', 'gw' : '<ip address>',
'device': 'eth0' },
    host7 : { 'ip': '<routing prefix address>', 'gw' : '<ip address>',
'device': 'eth1' },
    host8 : { 'ip': '<routing prefix address>', 'gw' : '<ip address>',
'device': 'eth1' }, }

env.password = 'secret' #Required only for releases prior to 1.10

env.passwords = {
    host1: 'secret',
    host2: 'secret',
    host3: 'secret',
    host4: 'secret',
    host5: 'secret',
    host6: 'secret',

```

```

    host7: 'secret',
    host8: 'secret',

    host_build: 'secret'
}

```

Related Documentation • [Juniper OpenStack High Availability on page 395](#)

Installing the Contrail Packages, Part Two (CentOS or Ubuntu) — Installing on the Remaining Machines

Preinstallation Checklist



NOTE: This procedure assumes that you have first completed the following procedures:

- [Installing the Contrail Packages, Part One \(CentOS or Ubuntu\) on page 15](#)
- [Setting Up the Testbed Definitions File on page 17](#)

And the following system tasks are accomplished:

- All of the servers are time synced.
- All servers can ping from one to another, both on management and on data and control, if part of the system.
- All servers can **ssh** and **scp** between one another.
- All host names are resolvable.
- If using CentOS or RHEL, SELinux has been disabled (**/etc/sysconfig/selinux**).

Each step in this procedure contains instructions for installing on a CentOS system or an Ubuntu system. Be sure to follow the instructions specific to your operating system.

To copy and install Contrail packages on the remaining machines in your cluster, you can use **scp** and **yum localinstall** as on the first server (for CentOS) or **scp** and **dpkg -i** (for Ubuntu), as in “[Installing the Contrail Packages, Part One \(CentOS or Ubuntu\)](#)” on page 15, or you can use a Fabric utility to copy onto all machines at once, as follows:

1. Ensure that the **testbed.py** file has been created and populated with information specific to your cluster at **/opt/contrail/utlils/fabfile/testbeds**.

See “[Setting Up the Testbed Definitions File](#)” on page 17.

2. Run Fabric commands to install packages as follows:

```

CentOS: /opt/contrail/utlils/fab
install_pkg_all:/tmp/contrail-install-packages-1.xx-xxx~openstack_version.el6.noarch.rpm

```

Ubuntu: `/opt/contrail/utils/fab`

`install_pkg_all:/tmp/contrail-install-packages-1.xx-xxx~openstack_version_all.deb`



NOTE: Fab commands are always run from `/opt/contrail/utils/`.

3. *Ubuntu:* The recommended Kernel version for Ubuntu based system is 3.13.0-40. Nodes can be upgraded to kernel version 3.13.0-40 using below fabric-utils command :

`fab upgrade_kernel_all`



NOTE: This step upgrades the kernel version to 3.13.0-40 in all nodes and performs reboot. Reconnect to perform remaining tasks.

4. Install the required Contrail packages in each node of the cluster:

`fab install_contrail`



NOTE: To install Contrail with an existing OpenStack node:

`fab install_without_openstack` # Script will install nova-compute in the compute node

or

`fab install_without_openstack:no` # User installs nova-compute in the compute node

5. If your installation has multiple interfaces (see [“Supporting Multiple Interfaces on Servers and Nodes” on page 22](#)), run `setup_interface`:

`fab setup_interface`

6. Provision the entire cluster:

`fab setup_all`



NOTE: To provision Contrail with an existing OpenStack node, use one of the following:

- `fab setup_without_openstack` # Script provisions vrouter and nova-compute services in the compute nodes and the compute nodes are rebooted on completion
- `fab setup_without_openstack:no` # Only vrouter services are provisioned, the nova-compute service is not provisioned and compute nodes are rebooted on completion
- `fab setup_without_openstack:no,False` # Only vrouter services are provisioned, the nova-compute service is not provisioned and the compute nodes are not rebooted on complet

7. *CentOS only:* Alternatively, if the **contrail-install-packages** is already installed (as part of installing the Contrail ISO via **netboot**), follow steps 2, 4, and 5.

When finished, you can proceed to [“Configuring the Control Node” on page 30](#).

Related Documentation

- [Configuring the Control Node on page 30](#)

Configuring the Control Node

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

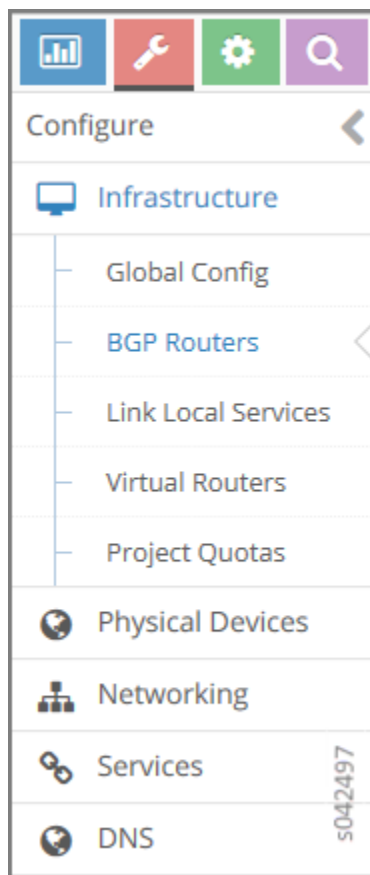
Before you begin, ensure that the following tasks are completed:

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

To configure BGP peering in the control node:

1. From the Contrail Controller module control node (<http://nn.nn.nn.nn:8080>), select **Configure > Infrastructure > BGP Routers**; see [Figure 1 on page 31](#).

Figure 1: Configure > Infrastructure > BGP Routers



A summary screen of the control nodes and BGP routers is displayed; see [Figure 2 on page 31](#).

Figure 2: 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 3 on page 32](#).

Figure 3: 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 3 on page 32](#).

Table 3: Create BGP Router Fields

Field	Description
Hostname	Enter a name for the node being added.
Vendor ID	Required for external peers. Populate with a text identifier, for example, "MX-0". (BGP peer only)
IP Address	The IP address of the node.
Router ID	Enter the router ID.
Autonomous System	Enter the AS number for the node. (BGP peer only)

Table 3: Create BGP Router Fields (continued)

Field	Description
Address Families	Enter the address family, for example, inet-vpn
Hold Time	BGP session hold time. The default is 90 seconds; change if needed.
BGP Port	The default is 179; change if needed.
Authentication Mode	Enable MD5 authentication if desired.
Authentication key	Enter the Authentication Key value.
Physical Router	The type of the physical router.
Available Peers	Displays peers currently available.
Configured Peers	Displays peers currently configured.

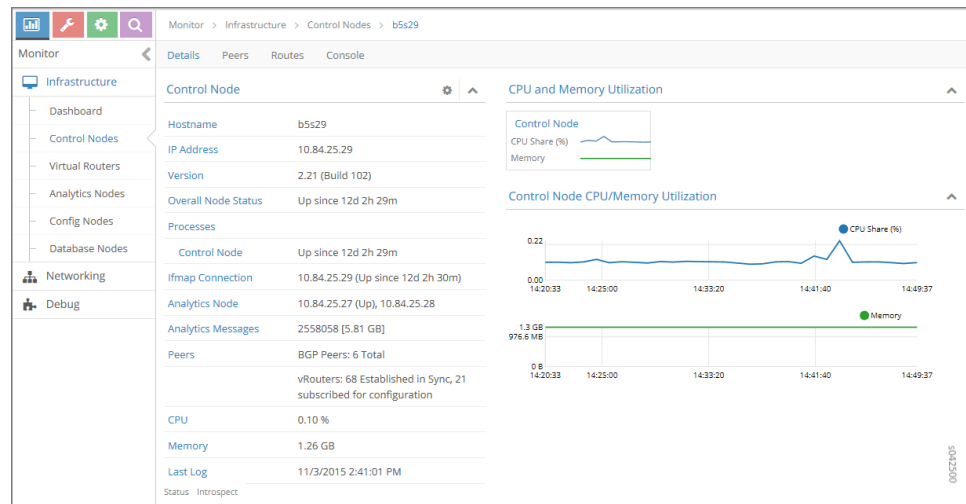
- Click **Save** to add each node that you create.
- To configure an existing node as a peer, select it from the list in the **Available Peers** box, then click **>>** to move it into the **Configured Peers** box.
Click **<<** to remove a node from the **Configured Peers** box.
- You can check for peers by selecting **Monitor > Infrastructure > Control Nodes**; see [Figure 4 on page 33](#).

Figure 4: Control Nodes



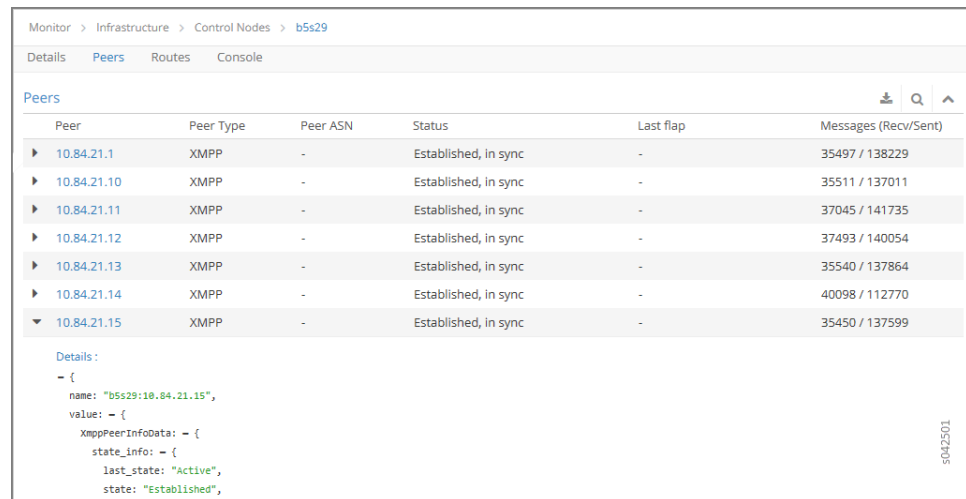
In the **Control Nodes** window, click any hostname in the memory map to view its details; see [Figure 5 on page 34](#).

Figure 5: Control Node Details



8. Click the **Peers** tab to view the peers of a control node; see [Figure 6 on page 34](#).

Figure 6: Control Node Peers Tab



Related Documentation

- [Creating Virtual Networks and Policies in Juniper Networks Contrail on page 239](#)
- [Creating Virtual Networks and Policies in OpenStack Contrail on page 247](#)

Adding or Removing a Compute Node in an Existing Contrail Cluster

Use the following procedure to add one or more new compute nodes to an existing Contrail cluster.

1. Add the new information about the new compute node(s) into your existing **testbed.py** file.



NOTE: For convenience, this procedure assumes you are adding a node @10.1.1.1, however, replace the 10.1.1.1 with the correct IP for the node or nodes that you are adding.

2. Copy the **contrail-install-packages** file for CentOS or Ubuntu to the **/tmp** directory of the **cfgm** node where the **fab** commands are triggered:

CentOS: `scp <id@server>:/path/to/contrail-install-packages-xxx-xxx.el6.noarch.rpm /tmp`

Ubuntu: `scp <id@server>:/path/to/contrail-install-packages_xxx-xxx~havana_all.deb /tmp`

3. Install the **contrail-install-packages** on to the new compute node (or nodes):

CentOS: **fab**

install_pkg_node:/tmp/contrail-install-packages_x.xx-xxx.xxx.noarch.rpm,host@10.1.1.1

Ubuntu: **fab**

install_pkg_node:/tmp/contrail-install-packages_x.xx-xxx~havana_all.deb,host@10.1.1.1

4. For Ubuntu 12.04.4 or 12.04.3 server with a kernel version older than 3.13.0-34, upgrade the kernel by using the following **fab** command:

cd /opt/contrail/utils; fab upgrade_kernel_node:host@10.1.1.1

where 10.1.1.1 must be replaced with the server's actual IP address.

5. Use **fab** commands to add the new compute node (or nodes):

fab add_vrouter_node:host@10.1.1.1

6. Manually reboot the compute node after kernel upgrade.

Removing a Node

Use the following procedure to remove one or more compute nodes from an existing Contrail cluster.



NOTE: For convenience, this procedure assumes you are adding a node @10.1.1.1, however, replace the 10.1.1.1 with the correct IP address for the node or nodes that you are adding.

1. Use the following fab command to remove the new compute node:

```
fab detach_vrouter_node:root@10.1.1.1
```

2. Remove the information about this detached compute node from the existing **testbed.py** file.

CHAPTER 4

Installation and Configuration Scenarios

- [Setting Up and Using a Simple Virtual Gateway with Contrail on page 37](#)
- [Configuring MD5 Authentication for BGP Sessions on page 48](#)
- [Installing Contrail with Red Hat OpenStack on page 50](#)
- [Configuring OpenStack Nova Docker with Contrail on page 56](#)
- [Configuring the Data Plane Development Kit \(DPDK\) Integrated with Contrail vRouter on page 58](#)
- [Configuring Single Root I/O Virtualization \(SR-IOV\) on page 61](#)
- [Configuring Virtual Networks for Hub-and-Spoke Topology on page 65](#)
- [Configuring Transport Layer Security-Based XMPP in Contrail on page 70](#)

Setting Up and Using a Simple Virtual Gateway with Contrail

- [Introduction to the Simple Gateway on page 37](#)
- [How the Simple Gateway Works on page 38](#)
- [Setup Without Simple Gateway on page 38](#)
- [Setup With a Simple Gateway on page 39](#)
- [Simple Gateway Configuration Features on page 40](#)
- [Packet Flows with the Simple Gateway on page 41](#)
- [Packet Flow Process From the Virtual Network to the Public Network on page 41](#)
- [Packet Flow Process From the Public Network to the Virtual Network on page 42](#)
- [Four Methods for Configuring the Simple Gateway on page 43](#)
- [Using Fab Provisioning to Configure the Simple Gateway on page 43](#)
- [Using the vRouter Configuration File to Configure the Simple Gateway on page 44](#)
- [Using Thrift Messages to Dynamically Configure the Simple Gateway on page 45](#)
- [Common Issues with Simple Gateway Configuration on page 48](#)

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. The simple gateway provides the Contrail virtual networks with access to the public network, and is represented as **vgw**.

How the Simple Gateway Works

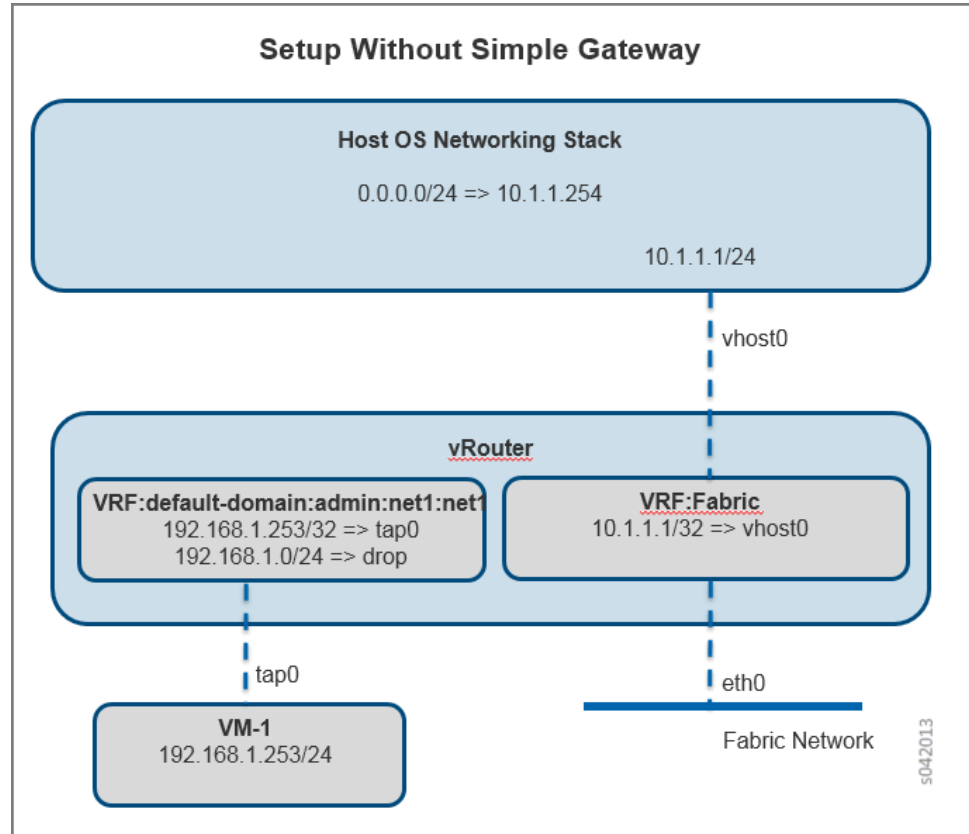
The following sections illustrate how the simple gateway works, first, by showing a virtual network setup with no simple gateway, then illustrating the same setup with a simple gateway configured.

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

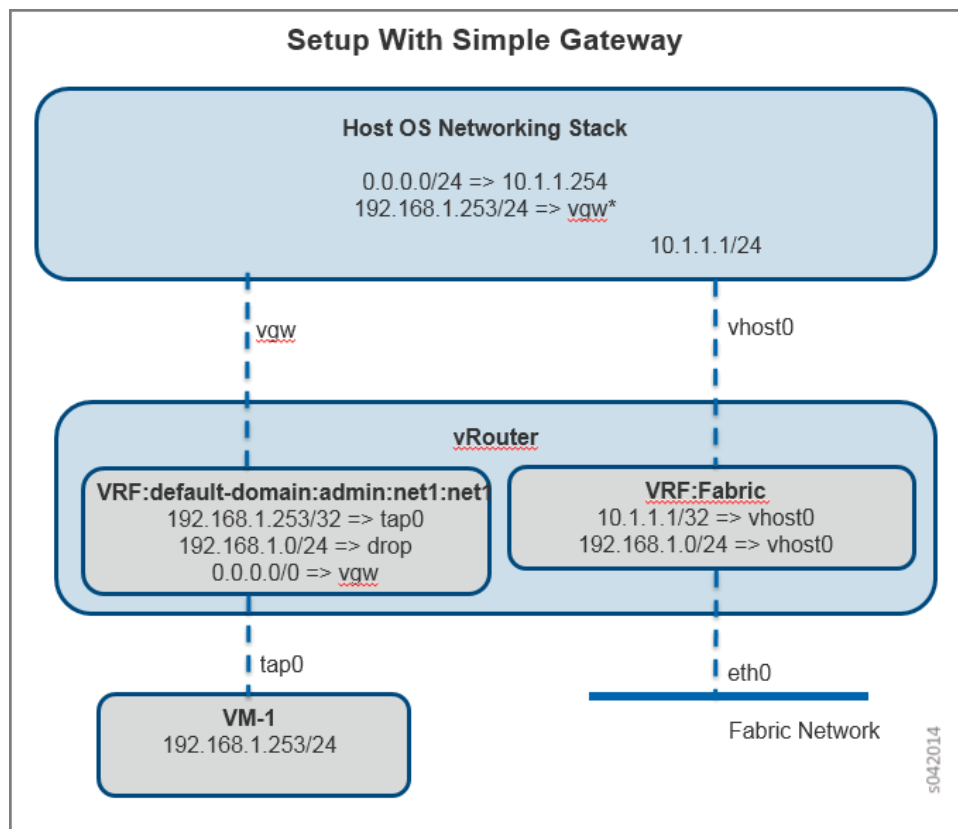
Figure 7 on page 40 shows a virtual network setup with the simple gateway configured for the **default-domain:admin:net1** virtual network.

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 7 on page 40 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 7: 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.

Four Methods for Configuring the Simple Gateway

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

Using Fab Provisioning to Configure the Simple Gateway

You can provision the simple virtual gateway (vgw) during system provisioning with fab commands by enabling the **vgw** parameter in the Contrail **testbed.py** file. Select some or all of the compute nodes to be configured as **vgw** by identifying **vgw** roles in the **env.roledefs** section, along with other role definitions.

The following example configuration shows three host nodes (**host4**, **host5**, and **host6**) configured as compute nodes. Two of the compute nodes (**host4** and **host5**) are also configured for vgw.

In the file section **env.vgw**, two vgw interfaces (**vgw1**, **vgw2**) are configured in **host4**, and the two interfaces are associated with virtual network **public** and **public1**, respectively.

For each vgw interface, the **ipam-subnets** key designates the subnets used by each virtual network. If the same vgw interface is configured in a different compute node, it must be associated with the same **ipam-subnets** virtual network. This is illustrated in the following example, where vgw2 is configured in two compute nodes, **host4** and **host5**. In both **host4** and **host5**, vgw2 is associated with the same **ipam-subnets** virtual network. .

The key gateway-routes is an optional parameter. If gateway-routes is configured, the corresponding vgw will only publish the list of routes identified for gateway routes.

If the vgw interfaces are defined in the **env.roledefs** variable, and you provision the system nodes with the **fab setup_all** command, the vgw interfaces are provisioned, along with all of the other nodes.

Example: Testbed.py Env.roledefs for vgw

```
env.roledefs = { 'all': [host1, host2, host3, host4, host5, host6],
'cfgm': [host1, host2, host3],
'openstack': [host2],
'webui': [host3],
'control': [host1, host3],
'compute': [host4, host5, host6],
'vgw': [host4, host5], >>>>>>>>Add section VGW in one or multiple compute
node
'collector': [host1, host3],
'database': [host1],
```

```
'build': [host_build],
}

env.vgw = {
    host4: {
        'vgw1': {
            'vn': 'default-domain:admin:public:public',
            'ipam-subnets': ['10.204.220.128/29',
'10.204.220.136/29']
            'gateway-routes': ['8.8.8.0/24', '1.1.1.0/24']
        },
        'vgw2': {
            'vn': 'default-domain:admin:public1:public1',
            'ipam-subnets': ['10.204.220.144/29']}},
    host5: {
        'vgw2': {
            'vn': 'default-domain:admin:public1:public1',
            'ipam-subnets': ['10.204.220.144/29']
        }
    }
}
```

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

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 on page 45](#)
 - [How to Dynamically Delete a Virtual Gateway on page 46](#)
 - [Using Devstack to Configure the Simple Gateway on page 47](#)

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.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 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 Release 2.20 and later implements 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.

There are 3 ways to enable BGP MD5 authentication and set the keys on the Contrail node.

1. To configure MD5 authentication for a BGP peer using an environment dictionary:

Before provisioning the node, include an environment dictionary (env dict) in the `testbed.py` file as shown. In this example, *juniper* is the `md5` key that is configured on the host1 and host2 nodes.

```
env.md5 = {  
    host1: 'juniper',  
    host2: 'juniper',  
}
```


Specify the desired key value on the node. The key should only be of type **string**.

2. Alternately, if the **md5** key is not included in the **testbed.py** file and the node is already provisioned, you can run the following script with an argument for md5:

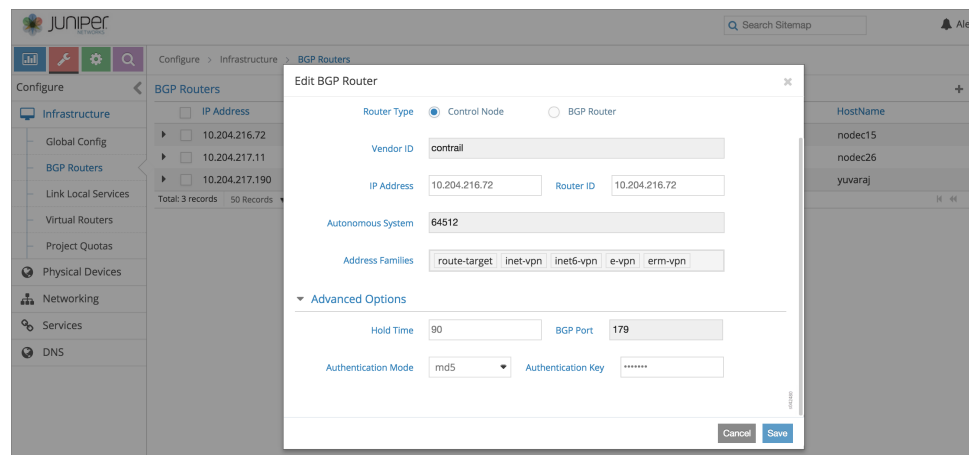
```
contrail-controller/src/config/utils/provision_control.py
```

```
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
```

3. Another alternative is to use the web user interface.

- a. Connect to the node's IP address at port 8080 (<node_ip>:8080) and select **Configure->Infrastructure->BGP Routers**. As shown in [Figure 8 on page 49](#), a list of BGP peers is displayed.

Figure 8: Edit BGP Router Window



- b. 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.
- c. Scroll down the window and select **Advanced Options**.
- d. Configure the MD5 authentication by selecting **Authentication Mode>MD5** and entering the **Authentication Key** value.

Related Documentation

- [Configuring the Authentication Key Update Mechanism for BGP and LDP Routing Protocols](#)
- [Creating Virtual Networks and Policies in Juniper Networks Contrail on page 239](#)
- [Creating Virtual Networks and Policies in OpenStack Contrail on page 247](#)

Installing Contrail with Red Hat OpenStack

- [Overview: Contrail with Red Hat OpenStack on page 50](#)
- [Procedure for Installing RHOSP5 on page 50](#)
- [Install and Configure Contrail on page 51](#)
- [Download and Install Contrail-Install-Packages to the First Config Node on page 52](#)
- [Update the Testbed.py File on page 52](#)
- [Complete the Installation on Remaining Nodes on page 54](#)
- [Appendix: Installing with RDO on page 55](#)
- [Install All-in-One OpenStack on page 56](#)

Overview: Contrail with Red Hat OpenStack

If you are planning to use Contrail with Red Hat OpenStack, be sure to first install Red Hat OpenStack, using either RDO or RHOSP packages.

Procedure for Installing RHOSP5

The following provides general steps for installing Red Hat OpenStack and configuring the setup for Contrail.

1. Install OpenStack.

To provision an OpenStack node with RHOSP5 packages, refer to the official Red Hat documents:

DEPLOYING OPENSTACK: LEARNING ENVIRONMENTS (MANUAL SETUP)



NOTE: Configure a password for Keystone, the same password must be used in the Contrail `testbed.py` file.

2. Copy the OpenStack Keystone password to the `testbed.py` file. Refer to *Update the Testbed.py*.

3. Stop the **nova-compute** and Neutron services in the OpenStack node:

```
# service openstack-nova-compute stop
# service neutron-server stop
# nova service-disable $(hostname) nova-compute
```

4. Update the **nova.conf** file as follows:

```
# openstack-config --set /etc/nova/nova.conf DEFAULT network_api_class
nova.network.neutronv2.api.API
# openstack-config --set /etc/nova/nova.conf DEFAULT neutron_url
http://<FIRST_CFGM_IP>:9696
(FIRST_CFGM_IP is the IP address of the first node in the CFGM role defined
in the testbed.py.)
# openstack-config --set /etc/nova/nova.conf DEFAULT neutron_admin_auth_url
```

```

http://<KEYSTONE_IP_ADDRESS>:35357/v2.0
# openstack-config --set /etc/nova/nova.conf DEFAULT compute_driver
nova.virt.libvirt.LibvirtDriver

```

5. Restart the Nova services:

```

# service openstack-nova-api restart
# service openstack-nova-conductor restart
# service openstack-nova-scheduler restart
# service openstack-nova-consoleauth restart

```

6. (optional) Configure the **novncproxy_port** value.

Contrail uses port 5999 for the **novncproxy_port** value. If the same port is preferred for any OpenStack node, update the **novncproxy_port** value as shown in the following:

```

# openstack-config --set /etc/nova/nova.conf DEFAULT novncproxy_port 5999

# service openstack-nova-novncproxy restart

```

Install and Configure Contrail

After Red Hat OpenStack is installed and the **testbed.py** file has been configured with items for Red Hat, you can install Contrail.

Repositories for Third Party Packages

The Contrail installation depends on a number of third party open source packages that are not included in the **contrail-install-packages** file, however, they are downloaded and installed through the Internet when respective repositories have been enabled in the nodes.

- For Red Hat registering of third party applications and subscribing, refer to Red Hat documentation: *How to register and subscribe a system to the Red Hat Customer Portal using Red Hat Subscription-Manager* at: <https://access.redhat.com/solutions/253273>
- For enabling Extra Packages for Enterprise Linux (EPEL) repositories, the respective EPEL packages might require installation. Refer to EPEL documentation at: <https://fedoraproject.org/wiki/EPEL>.

Ensure that the **contrail_install_repo** has the highest priority.

The following are the least-required RHEL repositories to be enabled in all nodes.

```
subscription-manager repos --enable=rhel-7-server-extras-rpms
```

```
subscription-manager repos --enable=rhel-7-server-optional-rpms
```

```
subscription-manager repos --enable=rhel-7-server-rpms
```

```
subscription-manager repos --enable=rhel-7-server-openstack-5.0-rpms
```

```
rpm -ivh http://yum.puppetlabs.com/puppetlabs-release-el-7.noarch.rpm
```

```
yum -y install http://dl.fedoraproject.org/pub/epel/7/x86\_64/e/epel-release-7-5.noarch.rpm
```

Priority for a repository can be set manually by editing the corresponding sections in the `/etc/yum.repos.d/*repo` files or use the `yum-config-manager` command.

To change priority of repos using the `yum-config-manager` command:

```
yum install yum-plugin-priorities
```

```
yum-config-manager --enable [reponame] --setopt="[reponame].priority=1"
```

Download and Install Contrail-Install-Packages to the First Config Node

The following steps show how to copy and install the `contrail-install-packages`, before updating the `testbed.py` file.

1. Copy the `contrail-install-packages` to the `/root/` directory of the first config node. The first config node is the first Configuration Manager (cfgm) node defined in the `env.roledefs['cfgm']` section roles in the `testbed.py` file.

2. Install the `contrail-install-packages`:

```
# yum --disablerepo=* localinstall <package file>
```

3. Set up `contrail_install_repo` and install `fabric-utils`:

```
# cd /opt/contrail/contrail_packages/
```

```
# ./setup.sh
```

4. Create the `testbed.py` file in the `testbeds` directory, with host details under different Contrail roles.

```
cd /opt/contrail/contrail_packages/testbeds/
```

Refer to example the `testbed.py` files available in the `testbeds` directory.

Update the Testbed.py File

The OpenStack node is not provisioned by using the Contrail `fabric-utils` utilities. It is the `testbed.py` file that carries node information.

Use the following steps to update the `testbed.py` file to make Contrail nodes aware of the OpenStack node information.

1. Update the OpenStack admin password in the `testbed.py` file:

```
#Openstack admin password
```

```
env.openstack_admin_password = '<password>'
```

```
and
```

```
env.keystone = {  
    'keystone_ip' : '10.84.12.15',  
    'auth_protocol' : 'http',  
    'auth_port' : '35357',  
    #Default is http  
    #Default is 35357
```

```

    'admin_token' : '$ABC123',
    'admin_user'   : 'admin',           #Default is admin
    'admin_password': '<password>',
    'service_tenant': 'service',       #Default is service
    'admin_tenant' : 'admin',          #Default is admin
    'region_name'  : 'RegionOne',      #Default is RegionOne
    'insecure'     : 'True',           #Default = False
}

```

2. Update the **keystone_ip** value in the **testbed.py** file. The Keystone IP address is the same as the OpenStack IP address.

```

env.keystone = {
    'keystone_ip'   : '10.xx.xx.xx',
    'auth_protocol' : 'http',          #Default is http
    'auth_port'     : '35357',         #Default is 35357
    'admin_token'   : '$ABC123',
    'admin_user'    : 'admin',         #Default is admin
    'admin_password': '<password>',
    'service_tenant': 'service',       #Default is service
    'admin_tenant'  : 'admin',         #Default is admin
    'region_name'   : 'RegionOne',     #Default is RegionOne
    'insecure'      : 'True',          #Default = False
}

```

3. Update the **admin_token** value in the **testbed.py** file.

The **admin_token** value is available in the **/etc/keystone/keystone.conf** file of the OpenStack node.

```

env.keystone = {
    'keystone_ip'   : '10.xx.xx.xx',
    'auth_protocol' : 'http',          #Default is http
    'auth_port'     : '35357',         #Default is 35357
    'admin_token'   : '$ABC123',
    'admin_user'    : 'admin',         #Default is admin
    'admin_password': '<password>',
    'service_tenant': 'service',       #Default is service
    'admin_tenant'  : 'admin',         #Default is admin
    'region_name'   : 'RegionOne',     #Default is RegionOne
    'insecure'      : 'True',          #Default = False }

```

4. (optional) If a different Keystone user or tenant for Neutron service is preferred, update the Keystone settings as in the following:

```

env.keystone = {
    'keystone_ip'   : '10.xx.xx.xx',
    'auth_protocol' : 'http',          #Default is http
    'auth_port'     : '35357',         #Default is 35357
    'admin_token'   : '$ABC123',
    'admin_user'    : 'admin',         #Default is admin
    'admin_password': '<password>',
    'service_tenant': 'service',       #Default is service
    'admin_tenant'  : 'admin',         #Default is admin
    'region_name'   : 'RegionOne',     #Default is RegionOne
    'insecure'      : 'True',          #Default = False
    'manage_neutron': 'no',            #Default = 'yes', Does configure
    neutron user/role in keystone required.
}

```

5. Update the **service_token** value in the **testbed.py** file.

Copy the **admin_token** value from the **/etc/keystone/keystone.conf** file of the OpenStack node, and enter it as the **service_token** value:

```
env.openstack = {
    'service_token' : '$ABC123',
    'amqp_host' : '<ip address>',
}
```

6. Update the **amqp_host** value in the **testbed.py** file.

Make the **amqp_host** value the same as the IP address of the OpenStack node:

```
env.openstack = {
    'service_token' : '$ABC123',
    'amqp_host' : '<ip address>',
}
```

7. Verify that all nodes are reachable and properly updated in the **testbed.py** file.

To verify that all nodes are reachable, use the following command and see if it passes.

```
# fab all_command:"uname -a"
```

Complete the Installation on Remaining Nodes

Use the following procedure to complete the installation.

1. Copy and install **contrail-install-packages** to all other nodes, except the OpenStack node:

```
# fab install_pkg_all_without_openstack:</path/to/contrail-install-packages.rpm>
```

2. Disable IP tables. You must permanently disable IP tables.

IP tables can be disabled by issuing the following fab commands. The **fab all_command** command, as used in the following example, executes the given command in all nodes configured in the **testbed.py** file.

```
# fab all_command:"iptables --flush"
# fab all_command:"sudo service iptables stop; echo pass"
# fab all_command:"sudo service ip6tables stop; echo pass"
# fab all_command:"sudo systemctl stop firewalld; echo pass"
# fab all_command:"sudo systemctl status firewalld; echo pass"
# fab all_command:"sudo chkconfig firewalld off; echo pass"
# fab all_command:"sudo /usr/libexec/iptables/iptables.init stop; echo pass"

# fab all_command:"sudo /usr/libexec/iptables/ip6tables.init stop; echo pass"

# fab all_command:"sudo service iptables save; echo pass"
# fab all_command:"sudo service ip6tables save; echo pass"
```

3. Install Contrail without OpenStack.

Because the OpenStack node is set up with RDO or RHOSP, and the relevant details are updated in the `testbed.py` file, Contrail must be installed without OpenStack. The following fab command is used to set up Contrail without OpenStack, while also assuming that IP tables are disabled permanently in all nodes.

```
# fab install_without_openstack
```

4. Set up Contrail.

Use the following fab command to set up Contrail without OpenStack, and with the assumption that IP tables are permanently disabled in all nodes.

```
# fab setup_without_openstack
```

5. Verify the setup.

Use the `contrail-status` and `openstack-status` commands to verify the setup status.

```
# fab all_command:"contrail-status; echo pass"
```

```
# fab all_command:"openstack-status"
```



NOTE: The `contrail-status` command is not available from the Red Hat OpenStack node.

Appendix: Installing with RDO

You can provision the OpenStack node by using RDO packages instead of RHOSP. This section provides guidelines for installing and provisioning when using RDO instead of RHOSP.

For more RDO installation information, refer to the Red Hat OpenStack documentation at <https://openstack.redhat.com/Quickstart>.

The following are the prerequisite steps.

1. You must already have the Red Hat OpenStack repositories set up and enabled.

2. Install RDO.

```
# yum install -y https://rdo.fedorapeople.org/rdo-release.rpm
```

3. Install Packstack.

```
# yum install -y openstack-packstack
```

Install All-in-One OpenStack

Use the following procedure to prepare to install Contrail when using RDO instead of RHOSP for Red Hat.

1. Install as an all-in-one node using Packstack.

```
# packstack --allinone --mariadb-pw=<password> --use-epel=y
```



NOTE: Packstack can use an answers file to determine where a service can be enabled or disabled based on your preference. Refer to Packstack documentation for more information.

2. Update the Keystone password to use a predictable password.

Packstack usually sets up Keystone with a random password. Use the following to set a predictable password for Keystone.

```
# source /root/keystonerc_admin && keystone user-password-update --pass  
<password> admin and update the same password in "OS_PASSWORD" in  
/root/keystonerc_admin
```

Configuring OpenStack Nova Docker with Contrail

- [Nova Docker Support in Contrail Overview on page 56](#)
- [Deploying a Contrail Compute Node to Work With OpenStack Nova Docker on page 57](#)
- [Launching a Docker Container on page 58](#)
- [Limitations on page 58](#)

Nova Docker Support in Contrail Overview

In Contrail Release 2.20 and later, OpenStack Nova Docker containers can be used instead of virtual machines for specific use cases. It is possible to configure a compute node in a Contrail cluster to support Docker containers. **DockerDriver** is a driver for launching Docker containers.

This topic describes how to set up a compute node in a Contrail cluster to support Docker containers.

For more details about OpenStack Nova Docker, refer to the OpenStack wiki at: <https://wiki.openstack.org/wiki/Docker>.

Contrail and OpenStack Nova Docker Platform Support

[Table 4 on page 57](#) shows the platforms and versions supported for Contrail and OpenStack Nova Docker.

Table 4: Contrail and OpenStack Nova Docker Support

Platform	Version	OpenStack SKU	Contrail Release
Ubuntu	Precise	Icehouse	2.20 and greater
Ubuntu	Trusty	Icehouse	2.20 and greater
Ubuntu	Trusty	Juno	2.20 and greater

Deploying a Contrail Compute Node to Work With OpenStack Nova Docker

To deploy a compute node to work with Nova Docker in a Contrail cluster, the Nova DockerDriver is defined in place of the **LibvirtDriver** in the **env.hypervisor** dictionary in the **testbed.py** file.

Example: The following example uses a **testbed.py** file with the compute nodes listed in the **env.roledefs** dictionary.

```
env.roledefs = {
    'compute' : [host5, host6, host7],
}
```

Changes are made to the **env.hypervisor** dictionary, to deploy Nova DockerDriver in host7 and deploy the **host5** and **host6** nodes with the default LibvirtDriver.

After the settings are changed in the **testbed.py** file as in the following example, use the steps in [“Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines” on page 27](#), to have the **host7** compute node deployed with the Nova DockerDriver.

```
env.hypervisor = {
    host7 : 'docker',
}
```

Saving the Docker Image to Glance

The Docker image must be pulled from the compute node that is deployed with DockerDriver (host7) and added to the Glance configuration, as in the following.

```
host7$ source /etc/contrail/openstackrc
```

```
host7$ docker pull cirros
```

```
host7$ docker save cirros | glance image-create --is-public=True --container-format=docker
--disk-format=raw --name cirros
```



NOTE: Always use the same name for the Docker image and its configuration in Glance. Otherwise, the Docker image cannot be launched.

Launching a Docker Container

Contrail creates a separate Nova availability zone (nova/docker) for compute nodes deployed with DockerDriver.

To launch a Docker container, use a Nova command that includes the **--availability-zone** option, as in the following.

```
host7$ source /etc/contrail/openstackrc

host7$ nova boot --flavor 1 --nic net-id=<netId> --image <imageId>
--availability-zone nova/docker <dockerContainerName>
```

Limitations

The following are limitations of the deployment of Nova Docker and Contrail.

- The Docker containers cannot be seen in the virtual network controller console because console access is not supported with Nova Docker, see <https://bugs.launchpad.net/nova-docker/+bug/1321818>.
- Docker images must be saved in Glance using the same name as used in the **docker images** command. If the same name is not used, you are not able to launch the Docker container.

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

- [DPDK Support in Contrail on page 58](#)
- [Preparing the testbed.py File for Provisioning a Contrail Cluster Node with DPDK on page 59](#)
- [Provisioning a Contrail Cluster with DPDK on page 60](#)

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, one or more Contrail compute nodes are provisioned with DPDK during installation. An entry in the **testbed.py** file specifies which nodes are to be configured to use the DPDK vRouter mode instead of the regular kernel mode. This allows for a mixed setup, where different nodes use different modes of the vRouter.

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

Preparing the testbed.py File for Provisioning a Contrail Cluster Node with DPDK

The **testbed.py** file is a Python file that is configured to specify all of the options necessary for the installation of a Contrail cluster, including whether any node should be configured to use DPDK.

An **env.dpdk** entry in the **testbed.py** file is used to specify which node(s) (hosts) will be configured upon installation to use the DPDK vRouter mode instead of the regular kernel module.

Each node to be configured with the DPDK vRouter must be listed in the **env.dpdk** dictionary entry, along with the percentage of memory for DPDK huge pages and the CPUs to be used, as in the following example:

```
env.dpdk = {
    host1: { 'huge_pages' : '50', 'coremask' : '0xf' },
    host2: { 'huge_pages' : '50', 'coremask' : '0xf' },
}
```

The following are required entries for **env.dpdk**:

- **hostx**—Specify the numerical name of each node host (**host1**, **host2**, **host3**, etc.) that is to be configured with a DPDK vRouter. Any nodes not listed here will use the kernel vRouter module.
- **huge_pages**—Specify the percentage of host memory to be reserved for the DPDK huge pages. The reserved memory will be used by the vRouter and the Quick Emulator (QEMU) for allocating memory resources for the virtual machines (VMs) spawned on that host.



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

- **coremask**—Specify a CPU affinity mask with which vRouter will run. vRouter will use only the CPUs specified for its threads of execution.

Supported formats include:

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

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

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

Provisioning a Contrail Cluster with DPDK

Use the same process for provisioning a regular kernel vRouter module using fab tools, Server Manager, or Server Manager Lite to provision a cluster with DPDK vRouter mode.

The DPDK nodes are specified in the **testbed.py** before cluster installation with fab tools.

The following is a summary of the fab tools provisioning steps. For more details about provisioning, see [“Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines”](#) on page 27.

1. Copy the **contrail-install-packages*.deb** file into the host that has been configured as a **host_build** in your **testbed.py** file.

2. Log in to the **host_build** node and install the package (assuming it is in your **\$HOME** directory).

```
# dpkg -i ~/contrail-install-packages_3.0-*~kilo_all.deb
```

3. Run the **setup.sh** script from **/opt/contrail/contrail_packages**.

```
# cd /opt/contrail/contrail_packages/
```

```
# ./setup.sh
```

4. Copy your **testbed.py** file into the **/opt/contrail/utils/fabfile/testbeds/** directory.

5. From the **/opt/contrail/utils** directory, issue the following fab commands:

```
# cd /opt/contrail/utils
```

```
# fab install_pkg_all:~/contrail-install-packages_3.0-*~icehouse_all.deb
```

```
# fab upgrade_kernel_all
```

```
# fab install_contrail
```

```
# fab setup_interface
```

```
# fab setup_all
```

Upon completion, the cluster with nodes using the DPDK vRouter implementation is ready to use.

Creating a Flavor for DPDK in OpenStack Kilo

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

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

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

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

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

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

Related Documentation

- [Installing the Contrail Packages, Part One \(CentOS or Ubuntu\) on page 15](#)
- [Setting Up the Testbed Definitions File on page 17](#)
- [Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines on page 27](#)
- DPDK official page: <http://www.dpdk.org>

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

- [Overview: Configuring SR-IOV on page 61](#)
- [Preparing the testbed.py File for Provisioning a Contrail Cluster with SR-IOV on page 61](#)
- [Enabling ASPM in BIOS on page 62](#)
- [Configuring SR-IOV Features Without env.sriov in testbed.py on page 62](#)
- [Launching SR-IOV Virtual Machines on page 63](#)

Overview: Configuring SR-IOV

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

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.

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

The **testbed.py** file is a Python file that is configured to specify all of the options necessary for the installation of a Contrail cluster.

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

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

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

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

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

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

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

Enabling ASPM in BIOS

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



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

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

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

1. Enable the Intel IOMMU (Input-Output Memory Management Unit) on Linux, by doing the following:
 - a. Make sure the IOMMU is turned on, using the following option line in **/etc/default/grub**:
GRUB_CMDLINE_LINUX_DEFAULT="nomdmonddf nomdmonisw intel_iommu=on"
 - b. Enter the **update-grubls /sys** command
 - c. Reboot the compute node.
2. Enable the required number of VFs on the selected NIC.
The following example enables seven VFs on the **eth0** interface. Check the configuration by using the commands **lspci -nn** or **ip link**.

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

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

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

4. Reboot Nova compute.

```
service nova-compute restart
```

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

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

6. Restart Nova Scheduler.

```
service nova-scheduler restart
```

Launching SR-IOV Virtual Machines

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

- [Using the Contrail UI to Enable and Launch an SR-IOV Virtual Machine on page 63](#)
- [Using the CLI to Enable and Launch SR-IOV Virtual Machines on page 64](#)

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. Specify the physical network already enabled for SR-IOV (in **testbed.py** or **nova.conf**) and its VLAN ID.

- On the virtual network, create a Neutron port, and in the **Port Binding** section, define a **Key** value of SR-IOV and a **Value** of direct.

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

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

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

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

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

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

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

- Create a subnet in **vn1**.

```
neutron subnet-create vn3 a.b.c.0/24
```

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

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


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

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

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

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

Related Documentation

- [Installing the Contrail Packages, Part One \(CentOS or Ubuntu\) on page 15](#)
- [Setting Up the Testbed Definitions File on page 17](#)
- [Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines on page 27](#)
- [Configuring the Data Plane Development Kit \(DPDK\) Integrated with Contrail vRouter on page 58](#)

Configuring Virtual Networks for Hub-and-Spoke Topology

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 on page 65](#)
- [Example: Configuring Hub-and-Spoke Virtual Networks on page 66](#)
- [Troubleshooting Hub-and-Spoke Topology on page 66](#)

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/utls# 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 9 on page 70 shows the import and export targets for **hub-vn** and **spoke-vns**, by invoking **contrail-control-introspect**.

Figure 9: Contrail Introspect

The screenshot shows the Contrail Introspect web interface. The browser address bar displays 'nodec13.englab.juniper.net:8083/ShowRoutingInstanceSummary?search_string='. The page title is 'Contrail'. Below the title, there is a search bar and a table of routing instances. The table has columns: name, virtual_network, vn_index, vxlan_id, import_target, and export_target. The table lists several instances, including 'default-domain:admin:hub-vn:hub-vn' and 'default-domain:admin:spoke-vn1:spoke-vn1'. The right side of the interface shows a sidebar with a search bar and a list of instances.

name	virtual_network	vn_index	vxlan_id	import_target	export_target
default-domain:admin:hub-vn:hub-vn	default-domain:admin:hub-vn	15	0	import_target	export_target
default-domain:admin:spoke-vn1:spoke-vn1	default-domain:admin:spoke-vn1	11	0	import_target	export_target
default-domain:admin:spoke-vn2:spoke-vn2	default-domain:admin:spoke-vn2	12	0	import_target	export_target
default-domain:admin:spoke-vn3:spoke-vn3	default-domain:admin:spoke-vn3	14	0	import_target	export_target
default-domain:admin:spoke-vn4:spoke-vn4	default-domain:admin:spoke-vn4	13	0	import_target	export_target

Configuring Transport Layer Security-Based XMPP in Contrail

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
xmpp_server_cert	Path to the node's public certificate	/etc/contrail/ssl/certs/server.pem
xmpp_server_key	Path to server's or 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	Enables SSL based XMPP	Default is disabled

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	Enables SSL based XMPP	Default is disabled

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

Parameter	Description	Default
<code>xmpp_auth_enable=true</code> <code>xmpp_dns_auth_enable</code>	Enables SSL based XMPP	Default is set to false, XMPP is disabled

CHAPTER 5

Using Contrail with VMware vCenter

- [Installing and Provisioning VMware vCenter with Contrail on page 73](#)
- [Underlay Network Configuration for ContrailVM on page 82](#)
- [Sample Testbed.py Files for Contrail vCenter on page 90](#)
- [Using the Contrail and VMWare vCenter User Interfaces to Manage the Network on page 95](#)

Installing and Provisioning VMware vCenter with Contrail

- [Overview: Integrating Contrail with vCenter Server on page 73](#)
- [Different Modes of vCenter Integration with Contrail on page 74](#)
- [vCenter-Only Mode on page 74](#)
- [vCenter-as-Compute Mode on page 75](#)
- [Preparing the Installation Environment on page 76](#)
- [Installation for vCenter-Only Mode on page 76](#)
- [Installing the vCenter-Only Components on page 77](#)
- [Installation for vCenter-as-Compute Mode on page 78](#)
- [Installing the vCenter-as-Compute Components on page 80](#)
- [Verification on page 81](#)
- [Adding Hosts or Nodes on page 81](#)
- [Adding an ESXi Host to an Existing vCenter Cluster on page 81](#)
- [Adding a vCenter Compute Node on page 82](#)

Overview: Integrating Contrail with vCenter Server

This topic describes how to install and provision Contrail Release 2.20 and later so that it works with existing or already provisioned vSphere deployments that use VMware vCenter as the main orchestrator.

The Contrail VMware vCenter solution is comprised of the following main components:

- Control and management that runs the following components as needed per Contrail system:
 - A VMware vCenter Server independent installation that is not managed by Juniper Networks Contrail. The Contrail software provisions vCenter with Contrail components and creates entities required to run Contrail.
 - The Contrail controller, including the configuration nodes, control nodes, analytics, database, and Web UI, which are installed, provisioned, and managed by Contrail software.
 - A VMware vCenter plugin provided with Contrail.
- VMware ESXi virtualization platforms forming the compute cluster, with Contrail data plane (vRouter) components running inside an Ubuntu-based virtual machine. The virtual machine, named *ContrailVM*, forms the compute personality while performing Contrail installs. The ContrailVM is set up and provisioned by Contrail. There is one ContrailVM running on each ESXi host.

Different Modes of vCenter Integration with Contrail

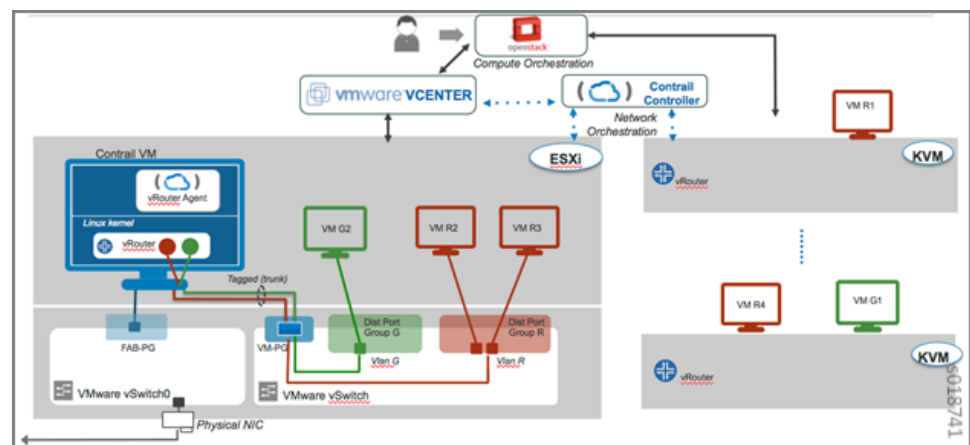
The vCenter integrated Contrail solution has the following modes:

- vCenter-only
- vCenter-as-compute

vCenter-Only Mode

In the vCenter-only mode, vCenter is the main orchestrator, and Contrail is integrated with vCenter for the virtual networking.

[Figure 10 on page 75](#) shows the Contrail vCenter-only solution.

75

Preparing the Installation Environment

Use the standard Contrail installation procedure to install Contrail on one of the target boxes or servers, so that Fabric (fab) scripts can be used to install and provision the entire cluster.

Follow the steps in the *Installing Contrail Packages for Ubuntu* section in [“Installing the Contrail Packages, Part One \(CentOS or Ubuntu\)”](#) on page 15



.....

NOTE: The fab scripts require a file named `testbed.py`, that holds all of the key attributes for fab to begin provisioning, including the IP addresses of the Contrail roles. Refer to the sample `testbed.py` file for Contrail vCenter in [“Sample Testbed.py Files for Contrail vCenter”](#) on page 90.

.....

Installation for vCenter-Only Mode

This section lists the basic installation procedure and the assumptions and prerequisites necessary before starting the installation of any VMware vCenter Contrail integration.

**Installation:
Assumptions and
Prerequisites**

The following assumptions and prerequisites are required for a successful installation of a VMware vCenter Contrail integrated system:

- VMware vCenter Server (version 5.5 or 6.0)
- A cluster of ESXi hosts with VMware version 5.5 or 6.0
- The following software installation packages:
 - `contrail-vcenter-install` “*.deb package for Ubuntu 14.04
- VMDK image of ContrailVM
- Because a Contrail vRouter runs as a virtual machine on each ESXi host, it needs an IP address assigned from the same underlay network as the host, all of which must be specified appropriately in the `testbed.py` file. Refer to the section “[Underlay Network Configuration for ContrailVM](#)” on page 82 for ContrailVM IP fabric connectivity.

Installing the vCenter-Only Components

Follow the steps in this section to install the Contrail for vCenter-only components. Refer to the sample `testbed.py` file for Contrail vCenter for specific examples: “[Sample Testbed.py Files for Contrail vCenter](#)” on page 90.

1. Ensure that all information in the `esxi_hosts` section of the `testbed.py` file is accurate, then provision the ESXi hosts using the following command:

```
fab prov_esxi
```

The `esxi_hosts = { }` section of the `testbed.py` file spawns the ContrailVM from the bundled VMDK file.

- Ensure that all required information in the section is specific to your environment and that the VMDK file can be accessed by the machine running the `fab` task.
- If the IP address and the corresponding MAC address of the ContrailVM are statically mapped in the DHCP server, specify the static IP address in the `host` field and the MAC address in the `mac` field in the `contrail_vm` subsection.
- ContrailVM IP fabric connectivity can be configured in various ways; refer to the section “[Underlay Network Configuration for ContrailVM](#)” on page 82 for details.

When finished, ping each of the ContrailVMs to make sure they respond.

2. Set up vCenter.

```
fab setup_vcenter
```

Specify the orchestrator to be vCenter for proper provisioning of vCenter-related components, as in the following:

```
env.orchestrator = 'vcenter'
```

When finished, verify that you can see the ESXIs and Contrail VMs on the vCenter user interface, refer to [“Using the Contrail and VMWare vCenter User Interfaces to Manage the Network”](#) on page 95.

3. Ensure that the **contrail-install-vcenter** Debian package is available on all the nodes.

```
fab install_pkg_all:<Contrail vcenter deb package>
```

4. Install the Contrail components into the desired roles on the specified nodes.

```
fab install_contrail
```

This also installs the **contrail-install-vcenter-plugin** on the contrail config nodes.

5. Set up the management and control data interfaces. Perform this step ONLY if the management and control_data interfaces are separate.

```
fab setup_interface_node
```

6. Provision all of the Contrail components and the vCenter plugin.:

```
fab setup_all
```

This step also creates the required configuration files on the system.

Installation for vCenter-as-Compute Mode

This section lists the basic installation procedure and the assumptions and prerequisites necessary before starting the installation of any VMware vCenter-as-compute Contrail integration.

Installation: The following assumptions and prerequisites are required for a successful installation of a VMware vCenter Contrail integrated system:
Assumptions and Prerequisites

- VMware vCenter Server (version 5.5 or 6.0)
- A cluster of ESXi hosts with VMware version 5.5 or 6.0
- The following software installation packages:
 - The OpenStack contrail-install *.deb package for Ubuntu 14.04, Juno, or Kilo
 - The **contrail-install-vcenter-plugin** *.deb package
- VMDK image of ContrailVM
- Because a Contrail vRouter runs as a virtual machine on each ESXi host, it needs an IP address assigned from the same underlay network as the host, all of which must be specified appropriately in the **testbed.py** file. Refer to [“Underlay Network Configuration for ContrailVM” on page 82](#) for ContrailVM IP fabric connectivity.

For the vCenter-as-compute mode, an additional role of **'vcenter-compute'** is required, specified as **['vcenter_compute']** in the **env.roledefs** section of the **testbed.py** file. Nodes configured as **vcenter_compute** act as the **nova-compute** nodes in this mode.

For specific examples, refer to the sample **testbed.py** file in [“Sample Testbed.py Files for Contrail vCenter” on page 90](#).

Installing the vCenter-as-Compute Components

To install the vCenter-as-compute components:

1. Provision the ESXi hosts.

fab prov_esxi

Before performing this, ensure that all information in the **esxi_hosts** section of the **testbed.py** file is accurate.

The **esxi_hosts = { }** section of the **testbed.py** file spawns the ContrailVM from the bundled VMDK file.

Ensure all required information in the section is specific to your environment and the VMDK file can be accessed by the machine running the fab task.

If the IP address and the corresponding MAC address of the ContrailVM are statically mapped in the DHCP server, specify the static IP address in the **host** field and the MAC address in the **mac** field in the **contrail_vm** subsection.

ContrailVM IP fabric connectivity can be configured in various ways, refer to [“Underlay Network Configuration for ContrailVM” on page 82](#) for details.

When finished, ping each of the ContrailVMs to make sure they respond.

2. Set up vCenter.

fab setup_vcenter

Specify the orchestrator to be OpenStack for proper provisioning of vCenter-as-compute, as in the following:

```
env.orchestrator = 'openstack'
```

In the vCenter-as-compute mode, you can have multiple **vcenter_servers** specified in the **testbed.py**.

Refer to [“Sample Testbed.py Files for Contrail vCenter” on page 90](#).

When finished, verify that you can see the ESXIs and ContrailVMs on the vCenter user interface. Refer to [“Using the Contrail and VMWare vCenter User Interfaces to Manage the Network” on page 95](#)

3. Ensure that the contrail-install *.deb package is available on all nodes.

```
fab install_pkg_all: <Contrail deb package>
```

4. Install the Contrail components into the desired roles on the specified nodes.

```
fab install_contrail
```

5. Install the vcenter-plugin *.deb file on the nodes specified under the vcenter-compute role.


```
fab install_contrail_vcenter_plugin:<vcenter-plugin-deb>
```

6. Set up the management and control data interfaces. Run this step ONLY if the management and control_data interfaces are separate.

```
fab setup_interface_node
```

7. Provision all of the Contrail components and the vCenter plugin.

```
fab setup_all
```

This step also creates the required configuration files on the system.

Verification

When the provisioning step completes, run the **contrail-status** command on all nodes to view a health check of the Contrail configuration and control components. All components display as active when provisioned successfully.

Adding Hosts or Nodes

You can add some vCenter features to existing installations, including:

- Adding an ESXi host
- Adding a vCenter compute node

Adding an ESXi Host to an Existing vCenter Cluster

You can provision and add an ESXi host to an existing vCenter cluster.

To add an ESXi host by using the following commands (which spawn the compute VM on an ESXi host) install and set up Contrail roles, and add the ESXi host to the vCenter cluster and switch:

1. Spawn the ContrailVM.

```
fab prov_esxi: <esxi_host>
```

where **<esxi_host>** is the ESXi hostname as specified in the **esxi_hosts{}**.

2. Install the Contrail *.deb package on the ContrailVM in the ESXi host.

```
fab install_pkg_node:<contrail-deb>,root@ContrailVM-ip
```

3. Add the ESXi to the vCenter cluster and put it into the switch, as specified in the configuration in the **testbed.py** file.

```
fab add_esxi_to_vcenter:esxi_host
```

This

4. Install and set up the Contrail vRouter on the ContrailVM in the ESXi host.

```
fab add_vrouter_node:root@ContrailVM-ip
```

Adding a vCenter Compute Node

To add a vCenter compute node:

1. Install the Contrail *.deb package on the ContrailVM in the ESXi host.

```
fab install_pkg_node:<contrail-deb>,root@ContrailVM-ip
```

2. Install the vcenter-plugin in the vcenter_compute node.

```
fab install_contrail_vcenter_plugin:<vcenter-plugin-deb>,root@vcenter_compute-ip
```

3. Provision the vcenter_compute node and set up Nova configuration files.

```
fab add_vcenter_compute_node:root@vcenter_compute-ip
```

Related Documentation

- [Underlay Network Configuration for ContrailVM on page 82](#)
- [Sample Testbed.py Files for Contrail vCenter on page 90](#)
- [Using the Contrail and VMWare vCenter User Interfaces to Manage the Network on page 95](#)

Underlay Network Configuration for ContrailVM

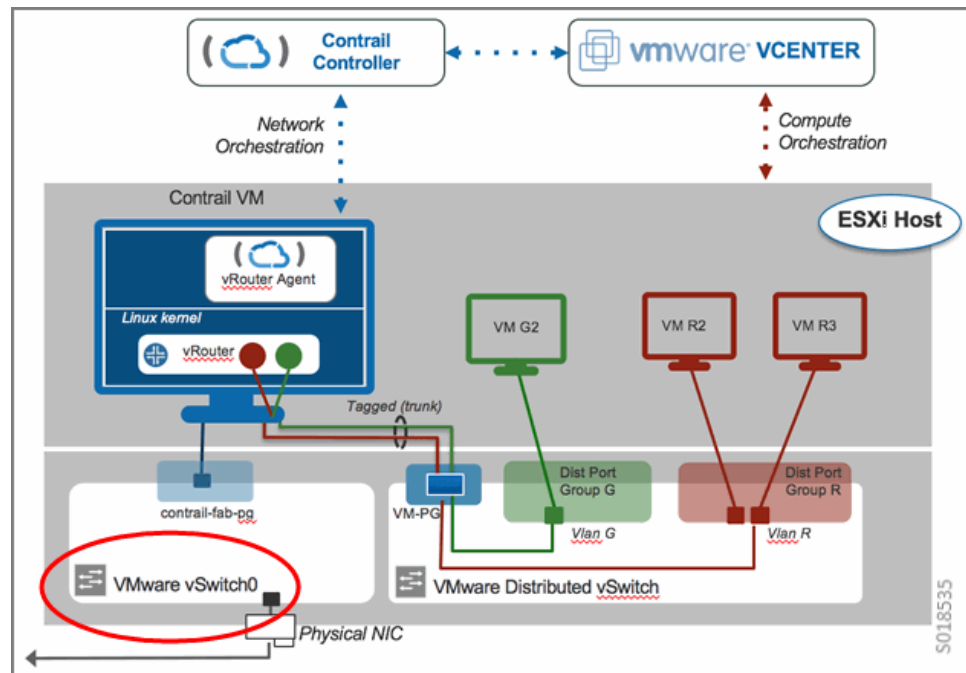
When using vCenter as compute, the ContrailVM can be configured in several different ways for the underlay (**ip-fabric**) connectivity:

- [Standard Switch Setup on page 82](#)
- [Distributed Switch Setup on page 83](#)
- [PCI Pass-Through Setup on page 85](#)
- [SR-IOV Setup on page 87](#)

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 12 on page 83](#).

Figure 12: Standard Switch Setup



To set up the ContrailVM in this mode, the standard switch and port group must be configured in the **contrail_vm** section in **testbed.py**.

If not configured, the default values of **vSwitch0** and **contrail-fab-pg** are used for the standard switch and port group, respectively.

The following is an example of a standard switch and port group configuration in **testbed.py**.

```
'contrail_vm': {
    'name' : "computevm-24-6",
    'mac' : "00:50:56:05:ba:ba",
    'host' : "root@10.84.24.32",
    'fabric_vswitch' : "vSwitch0",
    'fabric_port_group' : "contrail-fab-pg",
}
```



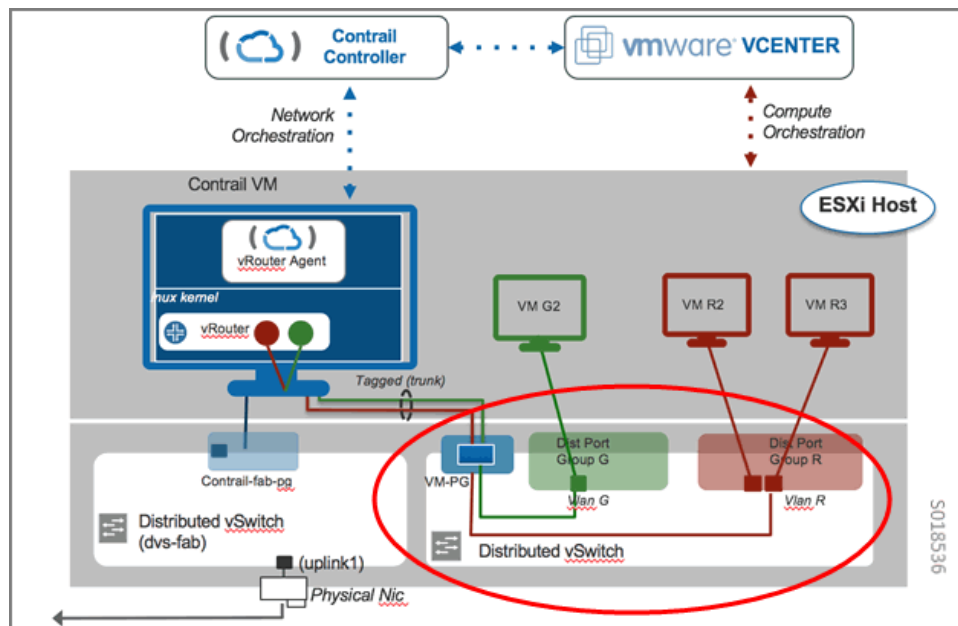
NOTE: By default, the management and control_data interfaces are the same in this configuration. To have separate a path for control_data, a VMXNET3 interface must be added manually in the ContrailVM, and control_data must be configured in the global section in **testbed.py**.

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 13 on page 84](#).

Figure 13: 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 **testbed.py**.



NOTE: The uplink can be a link aggregation group (LAG).

The following are required before using **testbed.py** to set up a distributed switch:

- Data center
- Cluster
- Distributed switch
- Host associated with the distributed switch

The following is an example distributed switch configuration in **testbed.py**.

```
env.vcenter: {
    'server': '<ip address>',
    'port': '443',
    'username': 'administrator@vsphere.local',
    'password': '<password>',
    'datacenter': 'kd_dc',
    'cluster': ['kd_cluster1', 'kd_cluster2'],
    'dv_switch_fab': {'dv_switch_name': 'dvs-fab'},
    'dv_port_group_fab': {
```

```

        'dv_portgroup_name': 'contrail-fab-pg',
        'number_of_ports': '3',
        'uplink': 'uplink11',
    }
}

```



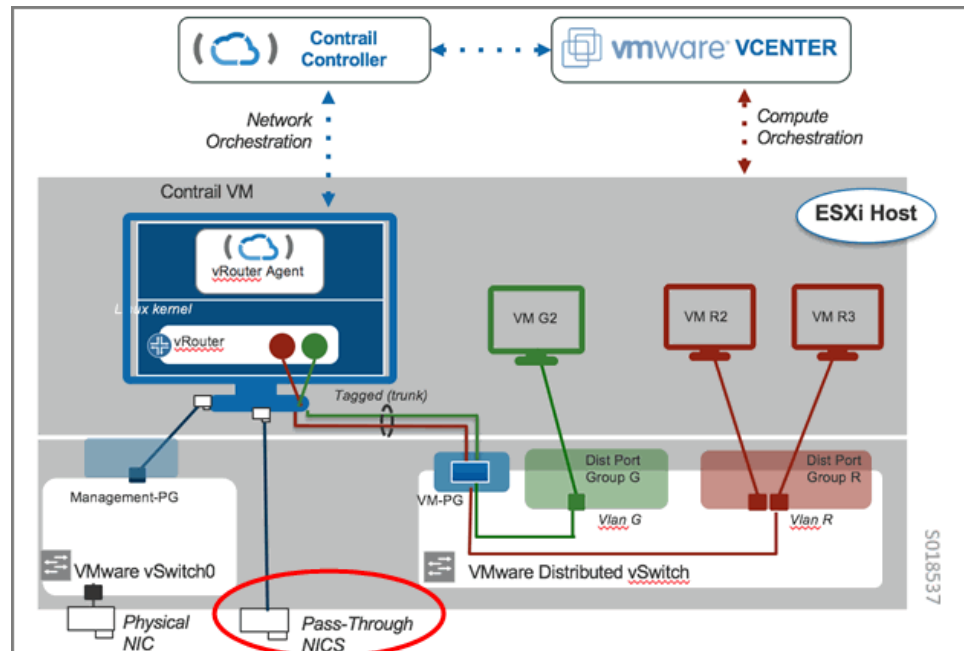
NOTE: By default, the management and control_data interfaces are the same in this configuration. To have a separate path for control_data, the VMXNET3 interface must be added manually in the ContrailVM, and control_data must be configured in the global section of testbed.py.

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 14 on page 85](#) shows a PCI pass-through setup with a single control_data interface.

Figure 14: PCI Pass-Through with Single Control Data Interface



To set up the ContrailVM with pass-through interfaces, all **testbed.py** configurations used for the standard switch setup are also used for the pass-through setup, providing the management connectivity to the ContrailVM.

To provide the **control_data** interfaces, configure the **pci_id** of the pass-through interfaces in the **contrail_vm** section, and configure **control_data** in the global section of **testbed.py**.

Upon provisioning ESXi hosts in the installation process, the PCI pass-through interfaces are exposed as Ethernet interfaces in the ContrailVM, and are identified in the **control_data** device field.

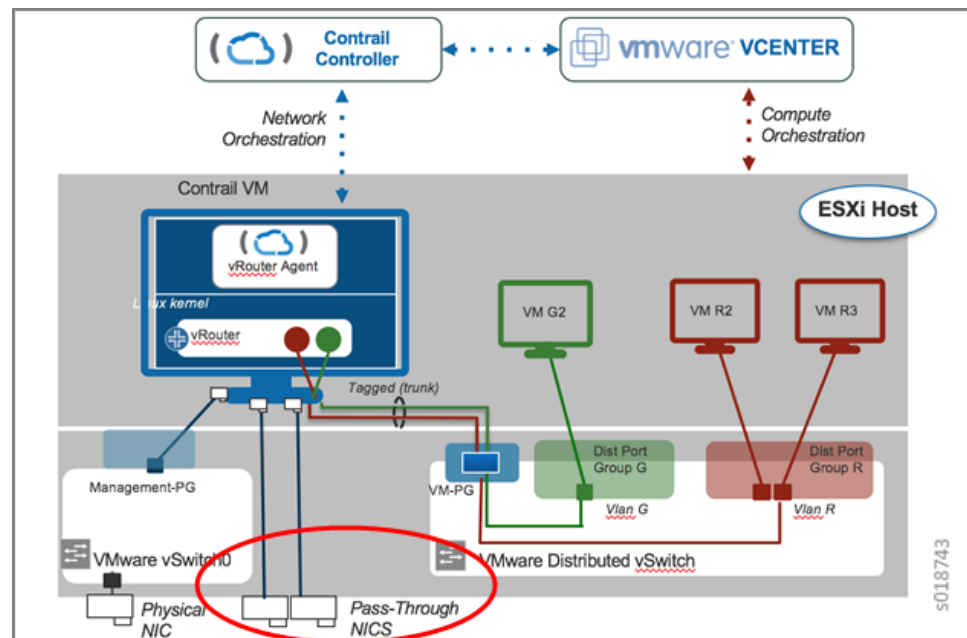
The following is an example PCI pass-through configuration with a single **control_data** interface:

```
'contrail_vm': {
    'name': "computevm-24-6",
    'mac': "00:50:56:05:ba:ba",
    'host': "root@10.84.24.232", #this is mgmt intf
    'pci_devices': {
        'nic': ["04:00.0"],
    },
    'vmdk_download_path':
"http://10.84.5.120/cs-shared/contrail-vmcenter/vmdk/LATEST/ContrailVM-disk1.vmdk",
}

control_data = {
    host4: { 'ip': '10.84.20.232/24', 'gw': '10.84.20.254', 'device': 'eth20' },
}
```

Figure 15 on page 86 shows a PCI pass-through setup with a bond_control data interface, which has multiple pass-through NICs.

Figure 15: PCI Pass-Through Setup with Bond Control Interface



The following is an example PCI pass-through configuration with a bond **control_data** interface:

```
'contrail_vm': {
    'name': "computevm-24-6",
    'mac': "00:50:56:05:ba:ba",
    'host': "root@10.84.24.232", #this is mgmt intf
    'pci_devices': {
        'nic': ["04:00.0", "04:00.1"],
    },
    'vmdk_download_path':
"http://10.84.5.120/cs-shared/contrail-vcenter/vmdk/LATEST/ContrailVM-disk1.vmdk",
}

control_data = {
    host4: { 'ip': '10.84.20.232/24', 'gw': '10.84.20.254', 'device': 'bond0'},
}

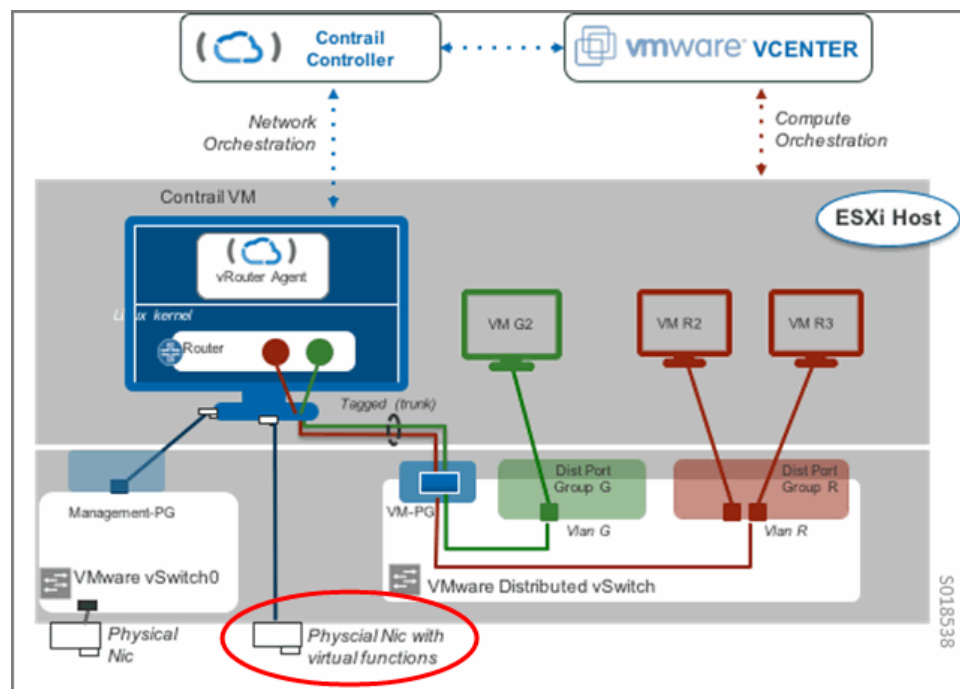
bond= {
    host2: { 'name': 'bond0', 'member': ['eth20', 'eth21'], 'mode': '802.3ad',
'xmit_hash_policy': 'layer3+4' },
}
```

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 16 on page 87](#).

Figure 16: SR-IOV Setup



In VMware, the **port-group** is mandatory for SR-IOV interfaces because the ability to configure the networks is based on the active policies for the port holding the virtual

machines. For more information, refer to VMware's [SR-IOV Component Architecture and Interaction](#).

The **port-group** is created as part of provisioning; however, before the provisioning, the distributed virtual switch (DVS) for the **port-group** should be created by the user.

To set up the ContrailVM with SR-IOV interfaces, all **testbed.py** configurations used for the standard switch setup are also used for the pass-through setup, providing the management connectivity to the ContrailVM.

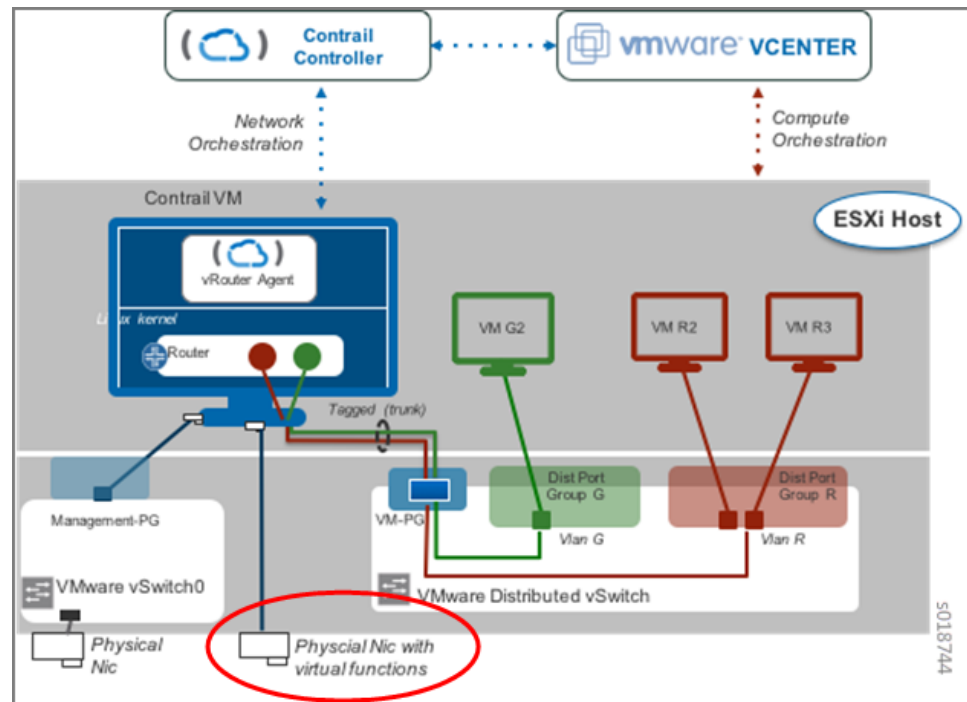
To provide the **control_data** interfaces, configure the SR-IOV-enabled physical interfaces in the **contrail_vm** section, and configure the **control_data** in the global section of **testbed.py**.

Configure the port group (**dv_port_group_sr_iov**) and the DVS (**dv_switch_sr_iov**) in the **env.vcenter_servers** section in the **testbed.py**.

Upon provisioning ESXi hosts in the installation process, the SR-IOV interfaces are exposed as Ethernet interfaces in the ContrailVM, and are identified in the **control_data** device field.

Figure 17 on page 88 shows a SR-IOV setup with a single **control_data** interface.

Figure 17: SR-IOV With Single Control Data Interface



The following is an example SR-IOV configuration with a single **control_data** interface:

```
'contrail_vm': {
  'name': "computevm-24-6",
  'mac': "<mac address>",
```



```

    'host': "host@<ip address>", #this is mgmt intf
    'sr_iov_nics': ['vmnic0'],
    'vmdk_download_path': "http://<ip
address>/cs-shared/contrail-vcenter/vmdk/LATEST/ContrailVM-disk1.vmdk",
  }
  control_data = {
    host4: { 'ip': '<ip address>', 'gw': '<ip address>', 'device': 'eth20'},
  }

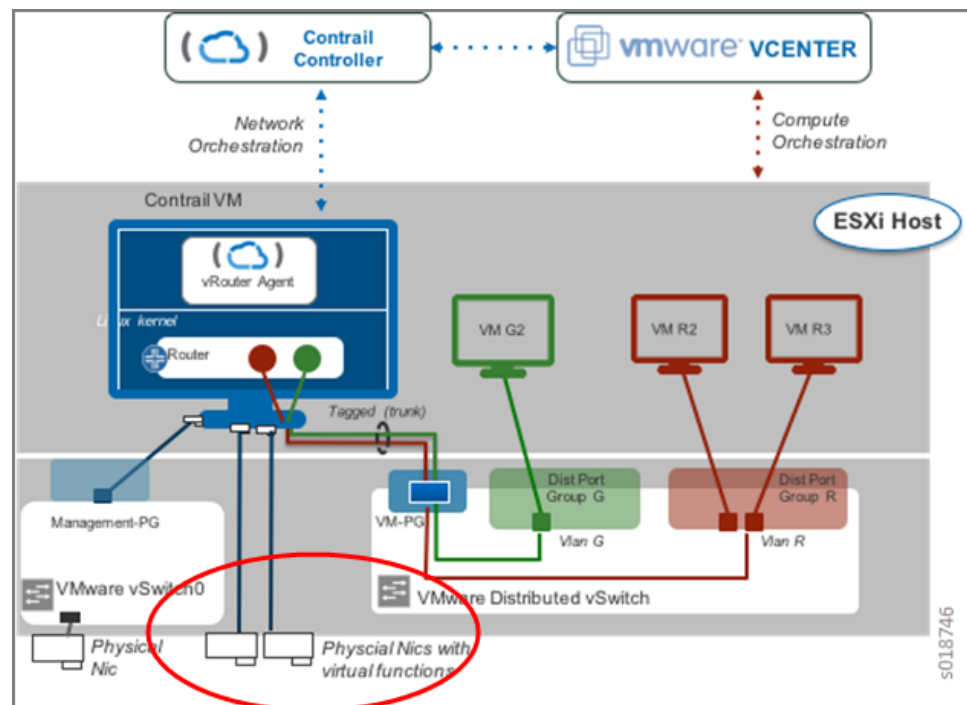
  env.vcenter = {
    'server': '<ip address>',
    'port': '443',
    'username': 'administrator@vsphere.local',
    'password': '<password>',
    ...
    ...
    ...
    ...

    'dv_switch_sr_iov': {
    'dv_switch_name': 'dvs-sriov',
    },
    'dv_port_group_sr_iov': {
    'dv_portgroup_name': 'dvs-sriov-pg',
    'number_of_ports': '2',
  },
},
}

```

Figure 18 on page 89 shows an SR-IOV configuration with a bond **control_data** interface, which has multiple SR-IOV NICs.

Figure 18: SR-IOV With Bond Control Data Interface



The following is an example SR-IOV configuration with a bond **control_data** interface:

```
'contrail_vm': {
    'name': "computevm-24-6",
    'mac': "<mac address>",
    'host': "host<ip address>", #this is mgmt intf
    'sr_iov_nics': ['vmnic0', 'vmnic1'],
    'vmdk_download_path': "http://<ip
address>/cs-shared/contrail-vcenter/vmdk/Ubuntu-14.04/Ubuntu-14.04-disk1.vmdk",
}
control_data = {
    host4 : { 'ip': '<ip address>', 'gw' : '<ip address>', 'device': 'bond0' },
}

bond= {
    host2: { 'name': 'bond0', 'member': ['eth20', 'eth21'], 'mode': 'active-backup',
'fail_over_mac': '1'},
}

env.vcenter = {
    'server': '<ip address>',
    'port': '443',
    'username': 'administrator@vsphere.local',
    'password': '<password>',
...
...
...
...
'dv_switch_sr_iov': {
    'dv_switch_name': 'dvs-sriov',
},
'dv_port_group_sr_iov': {
    'dv_portgroup_name': 'dvs-sriov-pg',
    'number_of_ports': '2',
},
}
```

- Related Documentation**
- [Installing and Provisioning VMware vCenter with Contrail on page 73](#)
 - [Sample Testbed.py Files for Contrail vCenter on page 90](#)
 - [Using the Contrail and VMware vCenter User Interfaces to Manage the Network on page 95](#)

Sample Testbed.py Files for Contrail vCenter

- [Sample Testbed.py File for vCenter-Only Mode on page 90](#)
- [Sample Testbed.py File for vCenter-as-Compute Mode on page 93](#)

Sample Testbed.py File for vCenter-Only Mode

```
from fabric.api import env

#Management ip addresses of hosts in the cluster
host1 = 'user@10.xx.xx.xx' #Contrail Controller
host2 = 'user@10.xx.xx.xx' #ContrailVM on ESXi

#External routers if any
ext_routers = []
```

```

#Autonomous system number
router_asn = 64512

#Host from which the fab commands are triggered to install and provision
host_build = 'user@10.xx.xx.xx'

minimum_diskGB=32

env.ntp_server = 'ntp.juniper.net'

#Role definition of the hosts.
env.roledefs = {
    'all': [host1, host2],
    'cfgm': [host1],
    'control': [host1],
    'compute': [host2],
    'collector': [host1],
    'webui': [host1],
    'database': [host1],
    'build': [host_build],
}

#Openstack admin password
env.openstack_admin_password = 'secret123'
env.orchestrator = 'vcenter' #other values are 'vcenter' default:openstack

#Disable multi-tenancy feature
multi_tenancy = False

env.password = 'secret'
#Passwords of each host
env.passwords = {
    host1: '<password>',
    host2: '<password>',
    host_build: '<password>',
}

#For reimage purpose
env.ostypes = {
    host1: 'ubuntu',
    host2: 'ubuntu',
}

#####
#vcenter provisioning
#server is the vcenter server ip
#port is the port on which vcenter is listening for connection
#username is the vcenter username credentials
#password is the vcenter password credentials
#auth is the authentication type used to talk to vcenter, http or https
#datacenter is the datacenter name we are operating on
#cluster is the clustername we are operating on
#dvswitch section contains distributed switch related para,s
#    dv_switch_name is the name of the dvswitch
#dvportgroup section contains the distributed port group info
#    dv_portgroupname and the number of ports the group has
#####
env.vcenter_servers = {
    'vcenter1': {
        'server': '10.xx.xx.xx',
        'port': '443',

```

```

        'username': 'administrator@vsphere.local',
        'password': '<password>!',
        'auth': 'https',
        'datacenter': 'test_dc',
        'cluster': ['test_cluster'],
        'dv_switch': { 'dv_switch_name': 'test_dvswitch', },
        'dv_port_group': { 'dv_portgroup_name': 'test_dvportgroup',
                           'number_of_ports': '3',
                           },
    },
}

#####
# The compute vm provisioning on ESXI host
# This section is used to copy a vmdk on to the ESXI box and bring it up
# the contrailVM which comes up will be setup as a compute node with only
# vrouter running on it. Each host has an associated esxi to it.
#
# esxi_host information:
#   ip: the esxi ip on which the contrailvm(host/compute) runs
#   username: username used to login to esxi
#   password: password for esxi
#   fabric_vswitch: the name of the underlay vswitch that runs on esxi
#                   optional, defaults to 'vswitch0'
#   fabric_port_group: the name of the underlay port group for esxi
#                      optional, defaults to 'contrail-fab-pg'
#   uplink_nic: the nic used for underlay
#               optional, defaults to None
#   data_store: the datastore on esxi where the vmdk is copied to
#   vcenter_server: the vcenter_server name which manages this esxi
#   cluster: name of the cluster to which this esxi is added
#   contrail_vm information:
#     mac: the virtual mac address for the contrail vm
#     host: the contrail_vm ip in the form of 'user@contrailvm_ip'
#     mode: the mode is 'openstack' or 'vcenter'
#           optional, defaults to env.orchestrator value
#     pci_devices: pci_devices information
#     nic: pci_id of the pass-through interfaces
#     sr_iov_nics: virtual functions enabled physical interface's name
#     vmdk: the absolute path of the contrail-vmdk used to spawn vm
#           optional, if vmdk_download_path is specified
#     vmdk_download_path: download path of the contrail-vmdk.vmdk used to spawn
#                         vm
#                         optional, if vmdk is specified
#####
esxi_hosts = {
    'b4s4': {
        'ip': '10.xx.xx.xx',
        'username': 'root',
        'password': '<password>',
        'datastore': '/vmfs/volumes/b4s4-ds1',
        'vcenter_server': 'vcenter1',
        'cluster': 'test_cluster',
        'contrail_vm': {
            'mac': "00:50:56:05:ba:ba",
            'host': host2,
            'mode': "vcenter",
            'vmdk': "/tmp/14-04-ContrailVM-disk1.vmdk",
        }
    },
}

```

```
}
```

Sample Testbed.py File for vCenter-as-Compute Mode

```
from fabric.api import env

#Management ip addresses of hosts in the cluster
host1 = 'user@10.xx.xx.xx' #Contrail Controller
host2 = 'user@10.xx.xx.xx' #ContrailVM on ESXi
host3 = 'user@10.xx.xx.xx' #vcenter-compute
host4 = 'user@10.xx.xx.xx' #KVM Compute

#External routers if any
ext_routers = []

#Autonomous system number
router_asn = 64512

#Host from which the fab commands are triggered to install and provision
host_build = user@10.xx.xx.xx4'

minimum_diskGB=32

env.ntp_server = 'ntp.juniper.net'

#Role definition of the hosts.
env.roledefs = {
    'all': [host1, host2, host3, host4 ],
    'cfgm': [host1],
    'control': [host1],
    'compute': [host2, host4],
    'collector': [host1],
    'webui': [host1],
    'database': [host1],
    'openstack': [host1],
    'vcenter_compute': [host3 ],
    'build': [host_build],
}

#Openstack admin password
env.openstack_admin_password = 'secret123'

env.password = 'secret'
#Passwords of each host
env.passwords = {
    host1: '<password>',
    host2: '<password>',
    host3: '<password>',
    host4: '<password>',
    host_build: '<password>',
}

#For reimage purpose
env.ostypes = {
    host1: 'ubuntu',
    host2: 'ubuntu',
    host3: 'ubuntu',
    host4: 'ubuntu',
```

```

}

#To enable multi-tenancy feature
multi_tenancy = True

#####
#vcenter provisioning
#server is the vcenter server ip
#port is the port on which vcenter is listening for connection
#username is the vcenter username credentials
#password is the vcenter password credentials
#auth is the authentication type used to talk to vcenter, http or https
#datacenter is the datacenter name we are operating on
#cluster is the clustername we are operating on
#vcenter_compute is the nova-compute node for this vcenter server
#dvswitch section contains distributed switch related para,s
#    dv_switch_name is the name of the dvswitch
#dvportgroup section contains the distributed port group info
#    dv_portgroupname and the number of ports the group has
#####
env.vcenter_servers = {
    'vcenter1': {
        'server': '10.xx.xx.xx',
        'port': '443',
        'username': 'administrator@vsphere.local',
        'password': '<password>',
        'auth': 'https',
        'datacenter': 'test_dc',
        'cluster': ['test_cluster'],
        'vcenter_compute': '10.xx.xx.xx',
        'dv_switch': { 'dv_switch_name': 'test_dvswitch', },
        'dv_port_group': { 'dv_portgroup_name': 'test_dvportgroup',
                           'number_of_ports': '3',
                           },
    },
}

#####
# The compute vm provisioning on ESXI host
# This section is used to copy a vmdk on to the ESXI box and bring it up
# the contrailVM which comes up will be setup as a compute node with only
# vrouter running on it. Each host has an associated esxi to it.
#
# esxi_host information:
#   ip: the esxi ip on which the contrailvm(host/compute) runs
#   username: username used to login to esxi
#   password: password for esxi
#   fabric_vswitch: the name of the underlay vswitch that runs on esxi
#                   optional, defaults to 'vswitch0'
#   fabric_port_group: the name of the underlay port group for esxi
#                       optional, defaults to 'contrail-fab-pg'
#   uplink_nic: the nic used for underlay
#               optional, defaults to None
#   data_store: the datastore on esxi where the vmdk is copied to
#   vcenter_server: the vcenter_server name which manages this esxi
#   cluster: name of the cluster to which this esxi is added
#   contrail_vm information:
#       mac: the virtual mac address for the contrail vm
#       host: the contrail_vm ip in the form of 'user@contrailvm_ip'
#       mode: the mode is 'openstack' or 'vcenter'
#             optional, defaults to env.orchestrator value

```

```

#      pci_devices: pci_devices information
#      nic: pci_id of the pass-through interfaces
#      sr_iov_nics: virtual functions enabled physical interface's name
#      vmdk: the absolute path of the contrail-vmdk used to spawn vm
#      optional, if vmdk_download_path is specified
#      vmdk_download_path: download path of the contrail-vmdk.vmdk used to spawn
vm
#      optional, if vmdk is specified
#####
esxi_hosts = {
    'b4s4': {
        'ip': '10.xx.xx.xx',
        'username': 'user',
        'password': '<password>',
        'datastore': "/vmfs/volumes/b4s4-ds1",
        'vcenter_server': 'vcenter1',
        'cluster': 'test_cluster',
        'contrail_vm': {
            'mac': "00:50:56:05:ba:ba",
            'host': host2,
            'mode': "vcenter",
            'vmdk': "/tmp/14-04-ContrailVM-disk1.vmdk",
        }
    },
}

```

- Related Documentation**
- [Installing and Provisioning VMware vCenter with Contrail on page 73](#)
 - [Underlay Network Configuration for ContrailVM on page 82](#)

Using the Contrail and VMWare vCenter User Interfaces to Manage the Network

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 on page 95](#)
- [Contrail User Interface on page 96](#)
- [Contrail vCenter User Interface on page 96](#)
- [Feature Configuration for Contrail vCenter on page 97](#)
- [Creating a Virtual Machine on page 104](#)
- [Configuring the vCenter Network in Contrail UI on page 110](#)

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” on page 73](#).

When Contrail is integrated with VMware vCenter, the following two user interfaces are used to manage and configure features of the system.

1. Contrail administration user interface (Contrail UI).
2. Contrail vCenter user interface (vCenter UI).

Contrail 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.

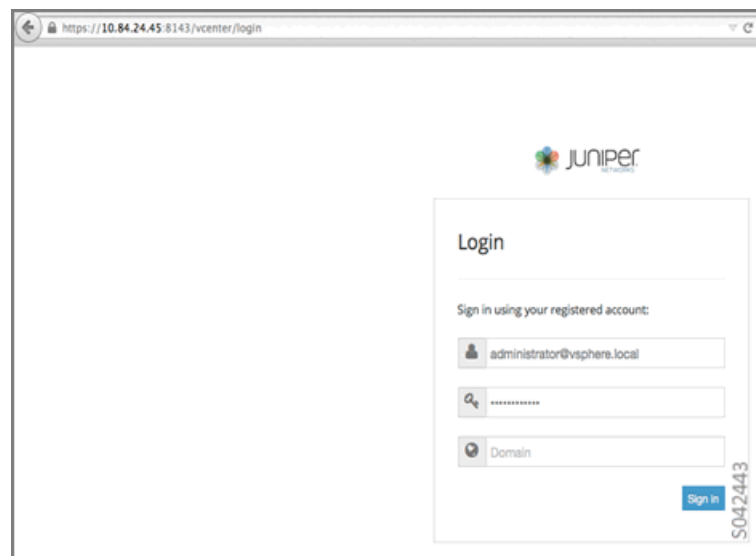


NOTE: This is applicable only to the vCenter-only mode.

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, as in the following.



Upon successful login, the Contrail vCenter user interface is displayed, as in the following example.



Feature Configuration for Contrail vCenter

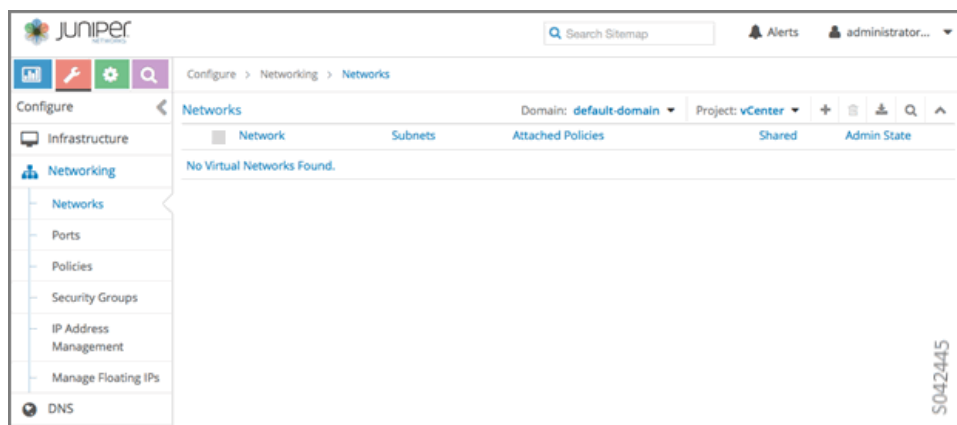
This section shows how to use the Contrail UI and the Contrail vCenter UI to configure features for the Contrail vCenter integrated system.

Creating a Virtual Network

This section describes how to create a virtual network using the Contrail UI and the Contrail vCenter UI.

Create Virtual Network – Contrail UI

After logging in to the Contrail UI, select **Configure > Networking > Networks** to access the Networks window.



At Networks, click the **+** (plus icon) to access the **Create Network** window.

Complete the fields in the **Create Network** window. Provide a **Primary VLAN** value and a **Secondary VLAN** value as part of a **Private VLAN** configuration. **Private VLAN** pairs are configured on a Distributed Virtual Switch. Select the values for the Primary and Secondary VLANs from one of the configured, isolated, private-vlan pairs.

The following figure shows the creation of a virtual network named Green-VN.

Click **Save** to create the virtual network.

The virtual network just created (Green-VN) is displayed with its details, as in the following figure.

Network	Subnets	Attached Policies	Shared	Admin State
Green-VN	192.168.1.0/24	Disabled	Disabled	Up

Subnets	IPAM	CIDR	Gateway	DNS	DHCP	Allocation Pools
vCenter-ipam	vCenter-ipam	192.168.1.0/24	192.168.1.1	Enabled	Enabled	

Network Properties

Display Name	: Green-VN
Admin State	: Up
Shared	: Disabled
External	: Disabled
Attached Network Policies	: -
Floating IP Pools	: -
Route Targets	: -
VxLAN Identifier	: Automatic
Allow Transit	: Disabled
DNS Servers	: -
Host Routes	: -

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.

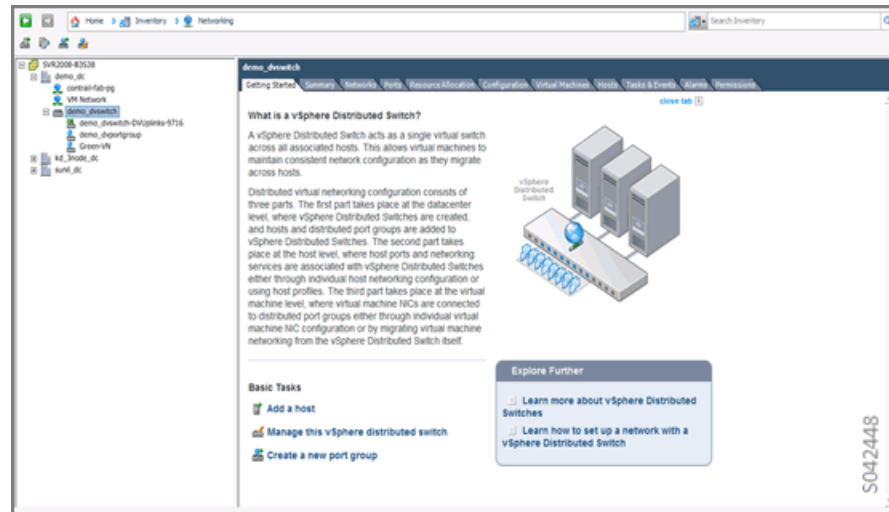
In vCenter, a virtual network is called a port group, which is part of a distributed switch.

Log in to the vCenter client UI (<https://<Contrail URL>:8143/vcenter>).

To start creating a virtual network (distributed port group), click the distributed virtual switch (dvswitch) on the left panel.

The following figure shows the *demo_dvswitch* has been selected for this example.

To create a virtual network (vCenter port group), at the bottom of the window, click **Create a new port group**.

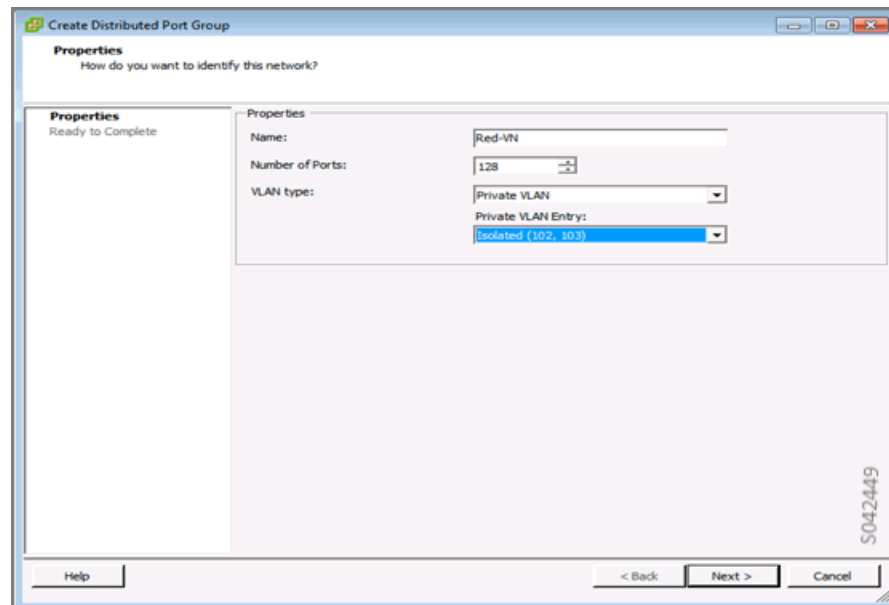


When you click **Create a new port group**, the **Create Distributed Port Group** window is displayed, as in the following figure.

Enter the name of the virtual network. Select the **VLAN type**, then select other details for the selected VLAN type.

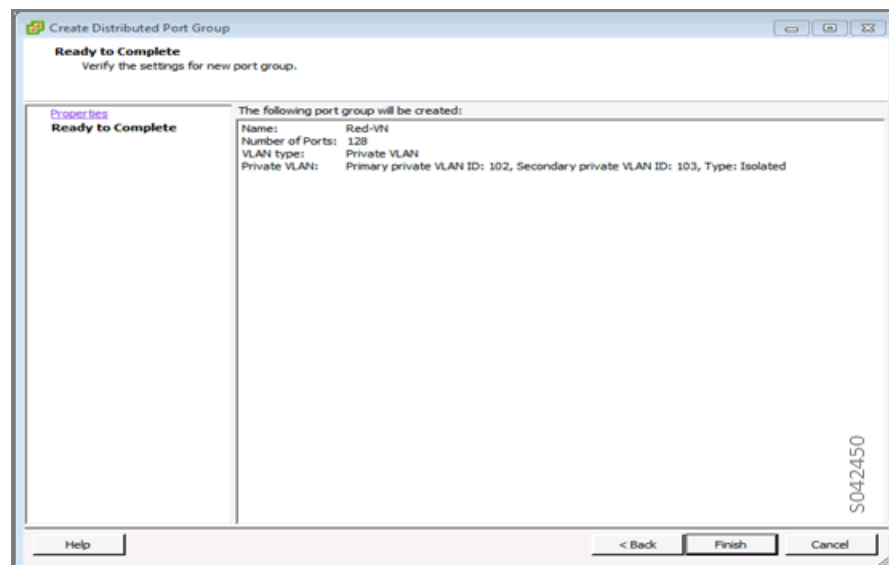
The following figure shows the **Create Distributed Port Group** window with the example creation of a virtual network named Red-VN, with a Private VLAN and isolated private VLAN ports 102, 103.

When you are finished, click **Next**.



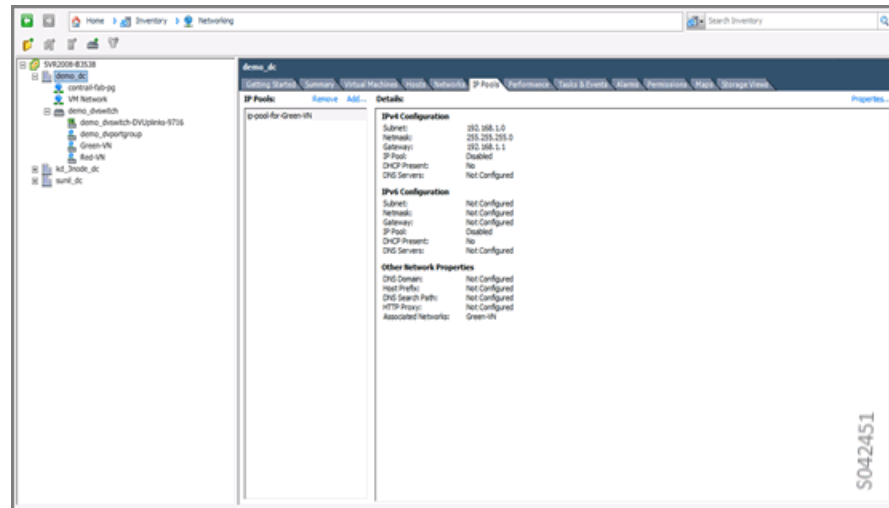
The **Ready to Complete** window is displayed, see the following figure. It shows the details entered for the virtual network (distributed port group).

If changes are needed, click **Back**. If the details are correct, click **Finish** to verify the port group details and complete its creation.

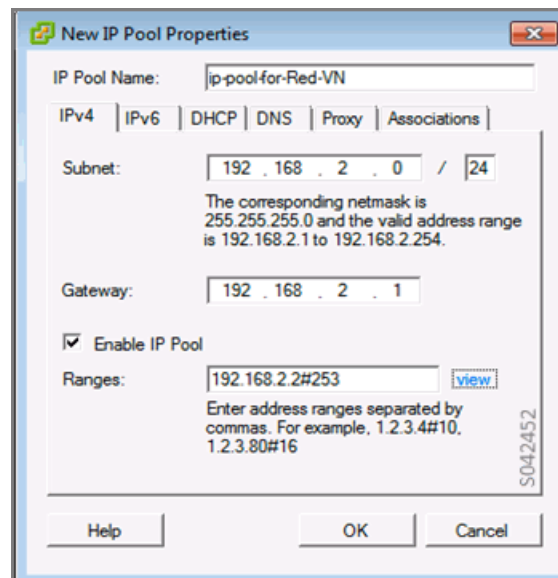


Next, create IP pools for the virtual network port group. Select the datacenter name in the left side panel, then click the **IP Pools** tab.

The following figure shows the **IP Pools** tab for the datacenter named **demo_dc**.



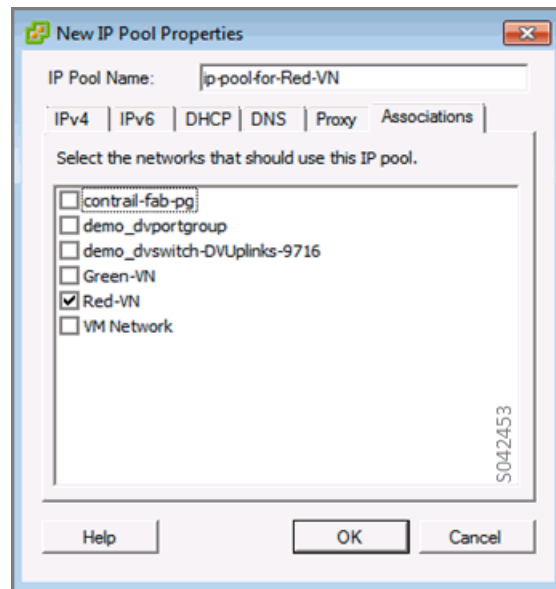
Near the top of the **IP Tools** window, click **Add** to open the **New IP Pool Properties** window, as in the following figure. The **IP Pool Properties** window has several tabs across the upper area. Ensure the **IPv4** tab is selected, and enter a name for the IP pool. Then enter the IP pool IPv4 details, including subnet, gateway, and IP address ranges. To enable IP address pools, select **Enable IP Pool**.



In the **New IP Pool Properties** window, click the **Associations** tab to select the networks that should use the IP address pool you are creating. This tab enables you to associate the IP pool with the port group.

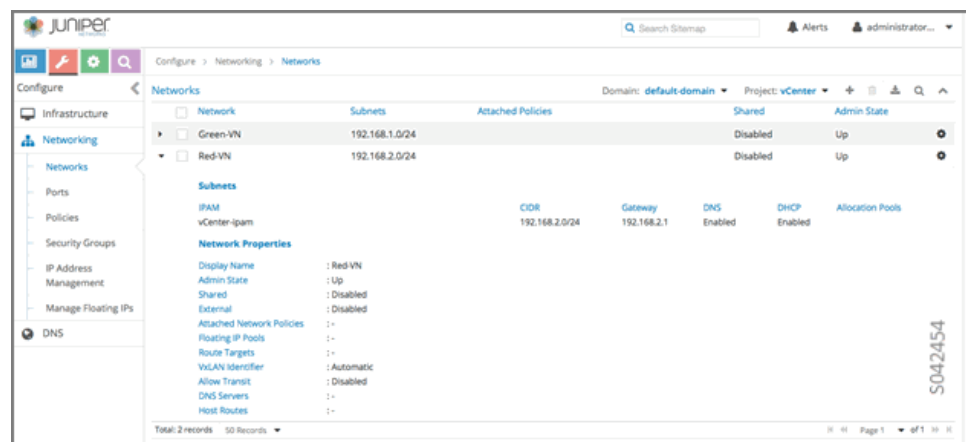
The following figure of the **Associations** tab shows that the IP pool being created should be associated with the virtual network port group named **Red-VN**.

When you are finished, click **OK**.



To verify that the virtual network is created and visible to Contrail, in the Contrail UI, select **Configure > Networking > Networks** to display Contrail network information.

The virtual network just created (Red-VN in this example) is displayed in the **Networks** window, see the following.

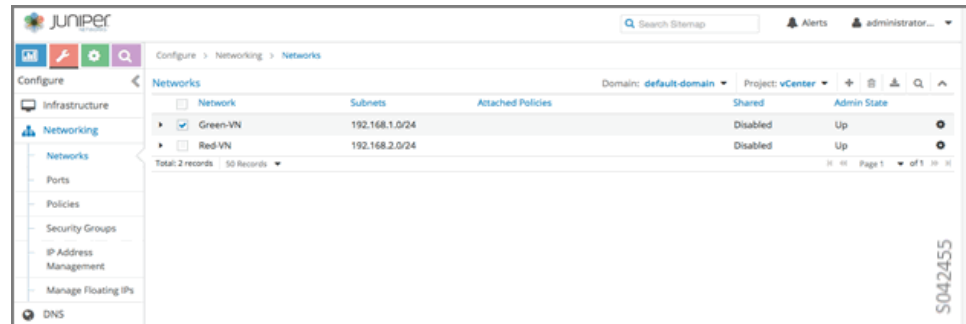


Delete Virtual Network – Contrail UI

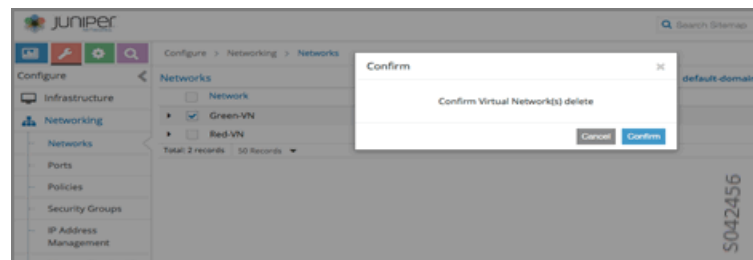
You can delete a virtual network in either the Contrail UI or in the vCenter UI. This section shows you how to delete a virtual network in the Contrail UI.

In the Contrail UI, select **Configure > Networking > Networks** to display Contrail network information.

Select the network you want to delete, then click the trashcan icon.



A Confirm window is displayed. Click **Confirm** to delete the selected network.



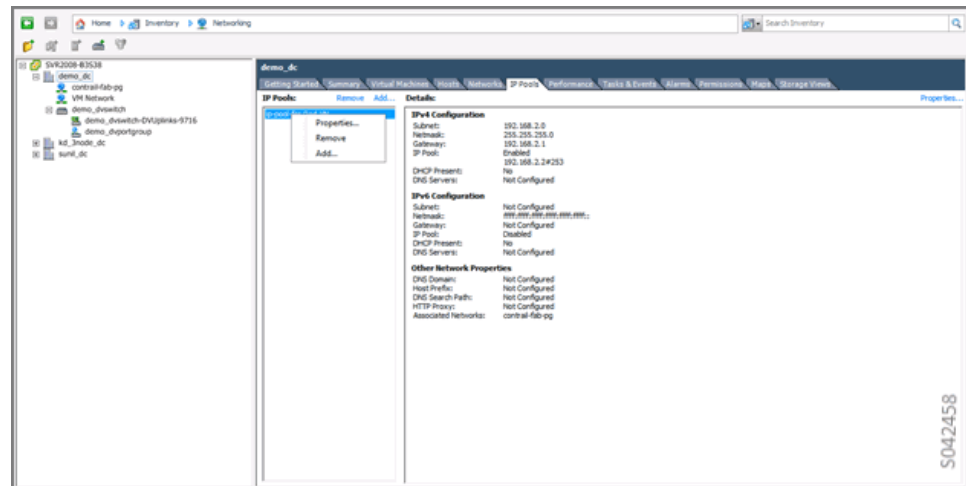
Delete Virtual Networks – vCenter UI

You can also delete a virtual network from the vCenter UI. From the vCenter UI, in the left side panel, right-click the port-group (virtual-network) you want to delete. In the menu, select **Delete** to delete the selected port group. An example is shown in the following.



When deleting a port group (virtual network) using the vCenter UI, you must also delete the IP pool associated with the port group. Select the **IP Pools** tab, and right click the name of the IP pool associated with the port group being deleted. From the menu, select **Remove** to delete the IP pool.

The following shows the deletion of the IP pool associated with the Red-VN from the vCenter UI.



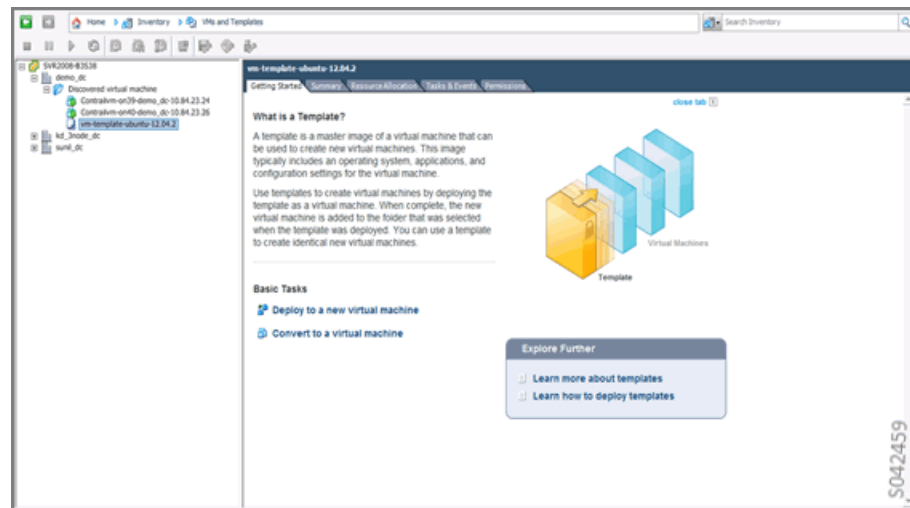
Creating a Virtual Machine

Use the vCenter client interface to create a virtual machine for your VMware vCenter Contrail integrated system. This section describes how to create a virtual machine using a virtual machine template from the vCenter client interface.

Create a Virtual Machine – vCenter UI

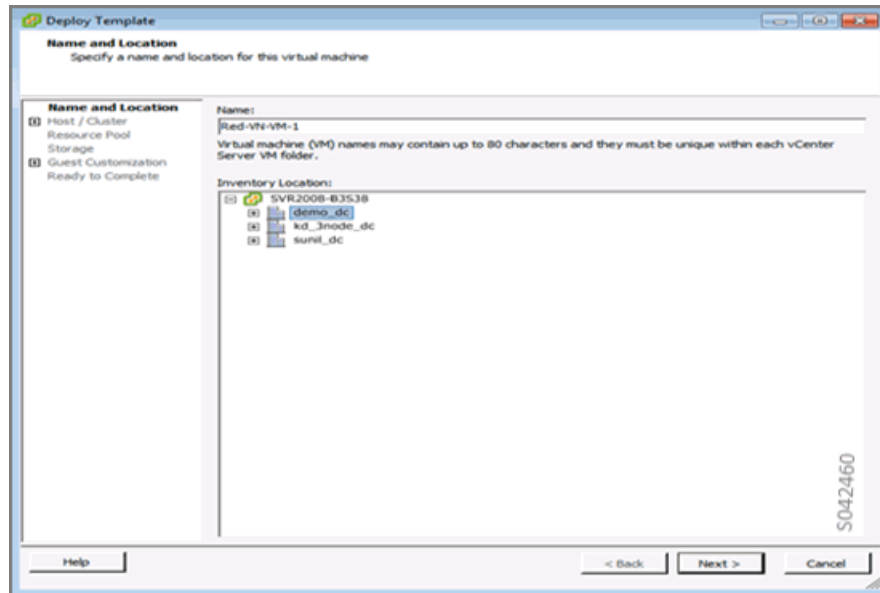
From the vCenter UI, select the virtual machine template from the left side panel. At the bottom of the right side pane, click **Deploy** to deploy a new virtual machine.

The following figure shows the **vm-template-ubuntu-12.04.2** virtual machine selected.



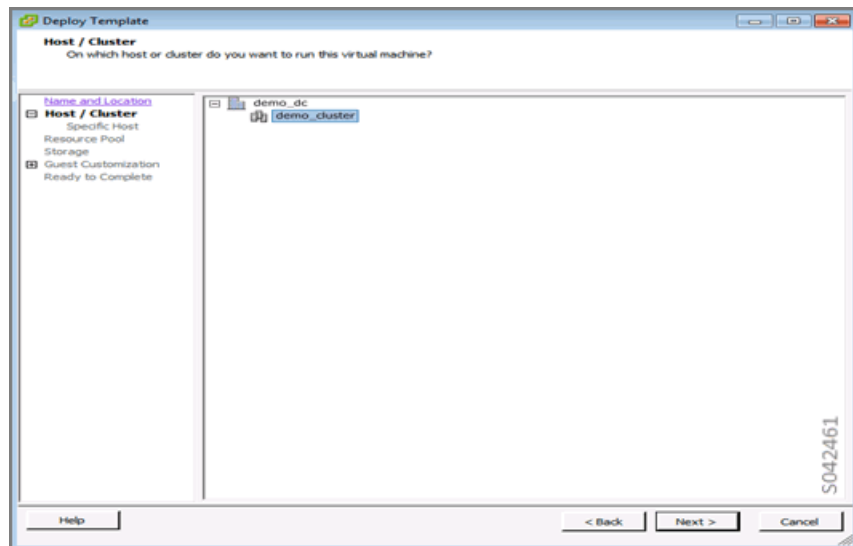
The **Deploy Template** Name and Location window is displayed, as in the following. Specify a name for the virtual machine and select the datacenter on which the virtual machine is to be spawned.

When you are finished, click **Next**.



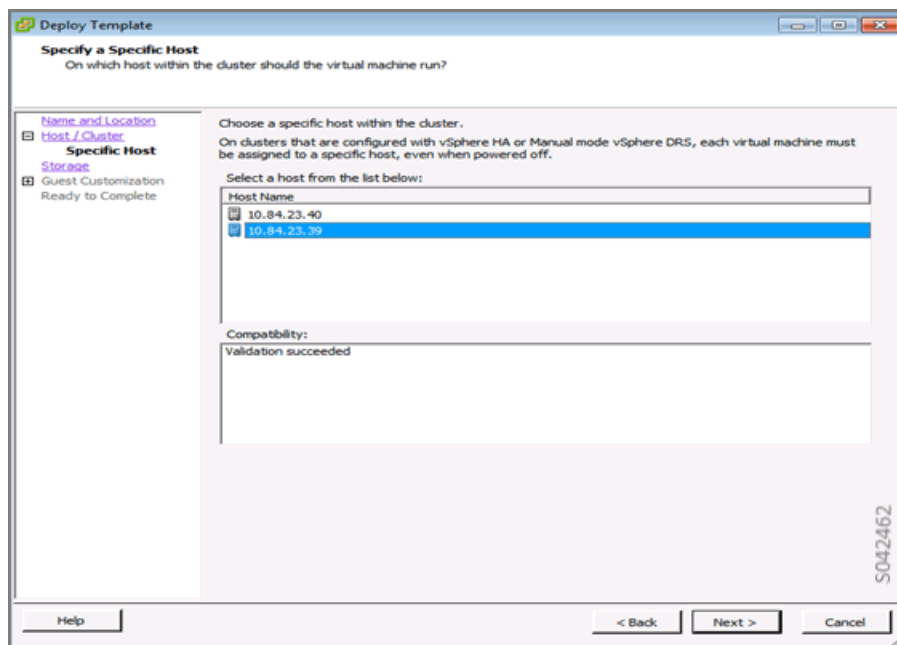
The **Host/Cluster** window is displayed, as in the following. Select the cluster on which to spawn the virtual machine.

When you are finished, click **Next**.



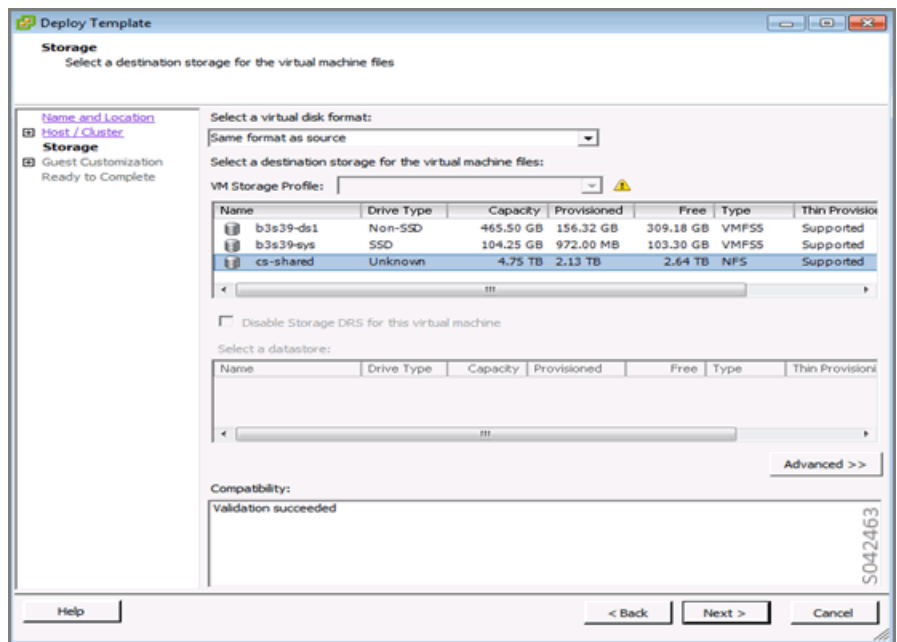
The **Specify a Specific Host** window is displayed, as in the following. Select the ESXi host on which to spawn the virtual machine.

When you are finished, click **Next**.



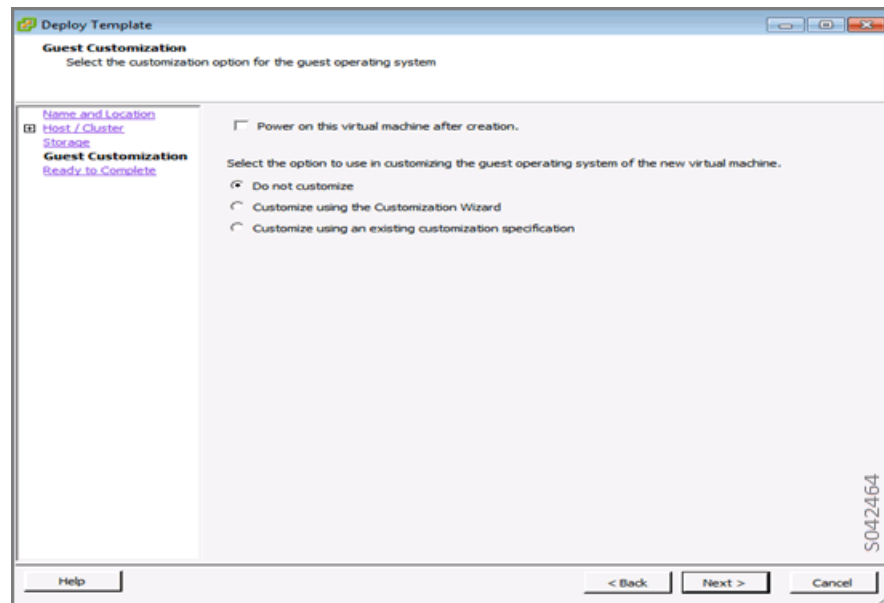
In the **Storage** window, select the destination storage location for the virtual machine.

When you are finished, click **Next**.



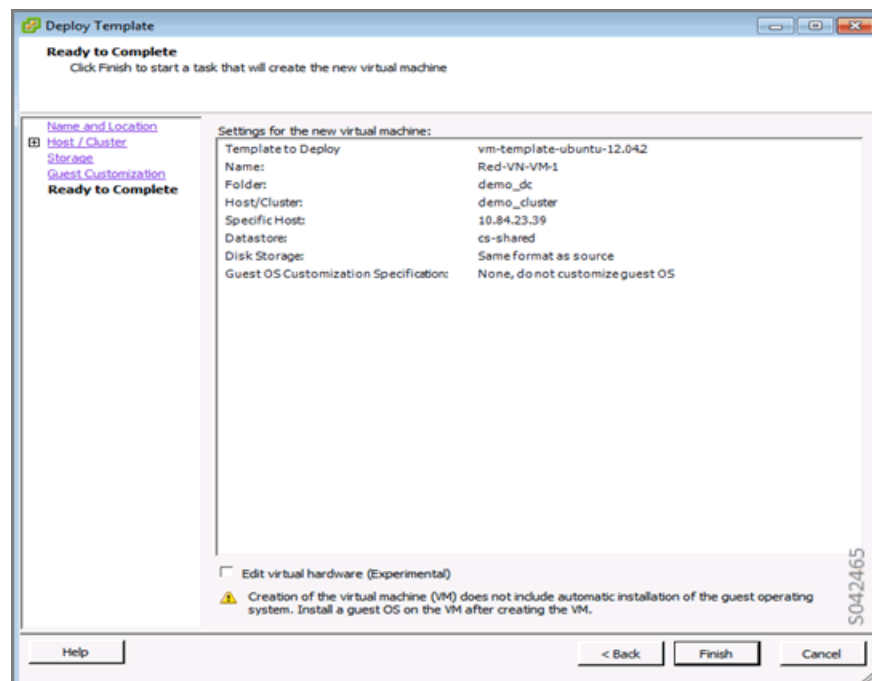
On the **Guest Customization** window, the typical selection is Do not customize. Select **Do not customize**.

When you are finished, click **Next**.



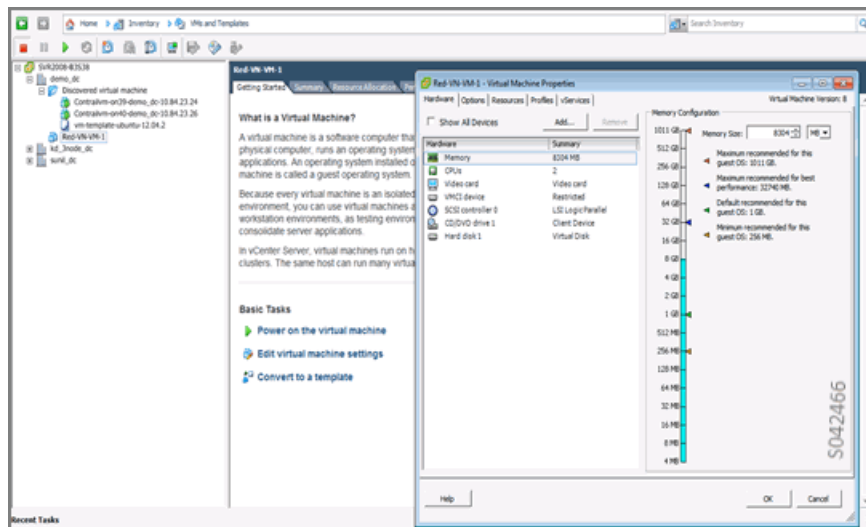
On the **Ready to Complete** window, review all of the virtual machine definitions that you have selected for the template.

If all the selections are correct, click **Finish**. This spawns the virtual machine.



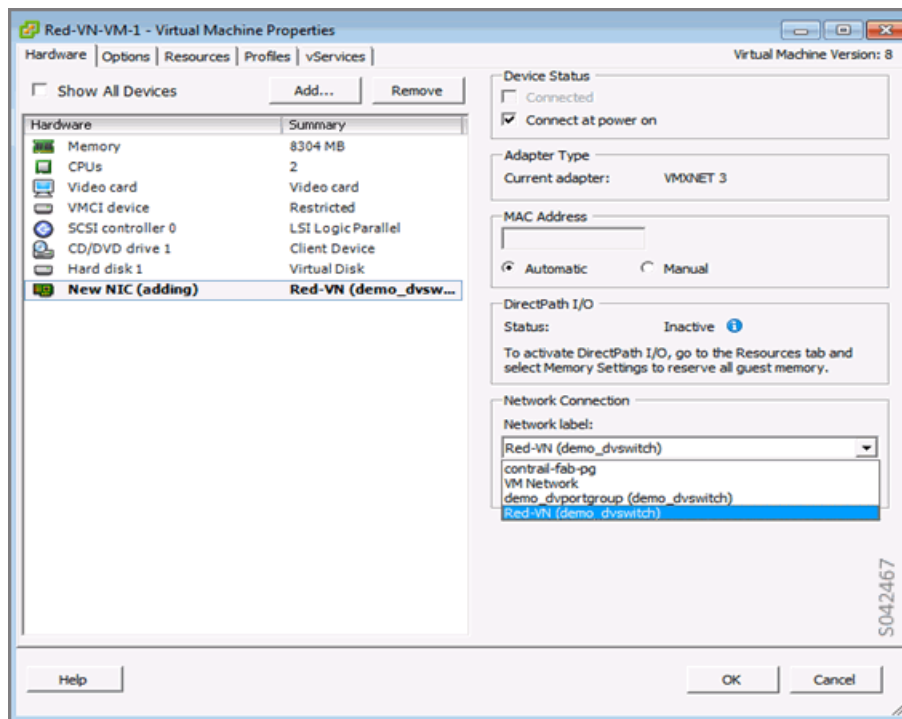
To complete the settings for the virtual machine, select the virtual machine to be edited in the left column of the main window of the vCenter UI. Then click **Edit virtual machine settings**.

The **Virtual Machine Properties** window is displayed, as in the following. From here you can update the virtual machine properties.

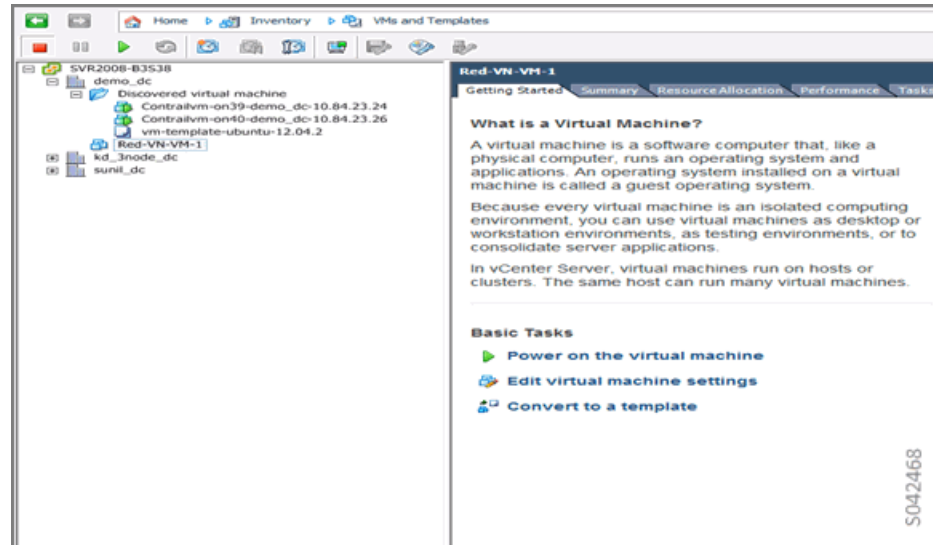


Click the **Hardware** tab in the **Virtual Machine Properties** window. Next, click **Add** to add a NIC and select the appropriate network. Select **Connect at power on**, as in the following.

When you are finished, click **OK**.



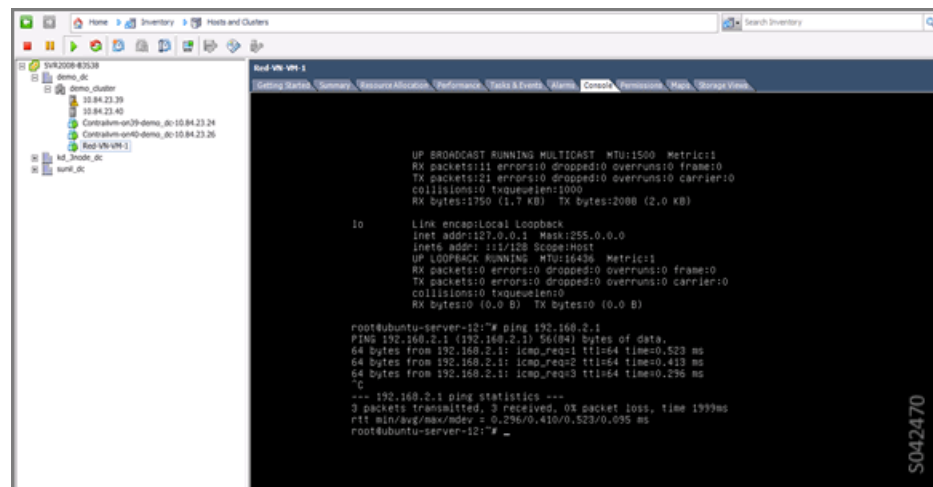
You are returned to the main vCenter UI window. Select the **Getting Started** tab. Select **Power on the virtual machine**. The virtual machine launches.



Once the virtual machine is launched, you can view it from the Contrail UI. Select **Monitor > Networking > Instances**. The virtual machines are displayed in the **Instances Summary** window, as in the following.



You can also see real-time running information for the virtual machine in the vCenter UI. Select the virtual machine and the **Console** tab. Real-time information is displayed, including ping statistics, as in the following.



Configuring the vCenter Network in Contrail UI

The following items can be configured for the vCenter network by using the Contrail UI.

- Network policy is configured by using the Contrail UI.
- Security policy is configured by using the Contrail UI.
- Public networks, floating IP address pools, and floating IP addresses are configured using the Contrail Administrator UI.

When you configure a virtual network using the administrator UI, the network is a Contrail-only network. No resources are consumed on vCenter to implement this type of network. You can configure a floating IP address pool on the network, allocate floating IP addresses, and associate floating IP addresses to virtual machine interfaces (ports).

Related Documentation

- [Installing and Provisioning VMware vCenter with Contrail on page 73](#)

CHAPTER 6

Using Server Manager to Automate Provisioning

- [Installing Server Manager on page 111](#)
- [Using Server Manager to Automate Provisioning on page 116](#)
- [Using the Server Manager Web User Interface on page 151](#)
- [Installing and Using Server Manager Lite on page 165](#)

Installing Server Manager

- [Installation Requirements for Server Manager on page 111](#)
- [Installing Server Manager on page 112](#)
- [Upgrading Server Manager Software on page 114](#)
- [Server Manager Installation Completion Checks on page 114](#)
- [Sample Configurations for Server Manager Templates on page 115](#)

Installation Requirements for Server Manager

This document provides details for installing Server Manager.

Platform Support

Server Manager can be installed on, and used to reimage and provision, the following platform operating systems:

- Ubuntu 14.04.4
- Ubuntu 14.04.2
- Ubuntu 14.04.1
- Ubuntu 14.04
- Ubuntu 12.04.3

As of Contrail 3.0, Server Manager installation supports Contrail provisioning for only the following OpenStack versions:

- Juno

- Kilo

Additionally, Server Manager can be used to reimage and provision:

- VMware ESXi 5.5

Installation Prerequisites

Before installing Server Manager ensure the following prerequisites are met.

- The system has Internet access to get dependent packages. Ensure access is available to the Ubuntu archive **mirrors/repos** at **/etc/apt/sources.list**.



NOTE: Server Manager is tested only with the following versions of dependent packages: Puppet 3.7.3-1 and Cobbler 2.6.3-1. The tested versions are installed during the Server Manager installation.

- Puppet Master requires the fully-qualified domain name (FQDN) of the Server Manager for key generation. The domain name is taken from the **/etc/hosts** file. If the server is part of multiple domains, specify the domain name by using the **–domain** option during the installation.
- On multi-interface systems, specify the interface to which Server Manager needs to listen by using the **–hostip** option. If the listening interface is not specified, the first available interface from the **ifconfig** list is used.

Installing Server Manager

Server Manager and all of its components (Server Manager, monitoring, Server Manager client, Server Manager Web user interface) are provided together in a wrapper installation package:

Ubuntu: **contrail-server-manager-installer_<version~sku>.deb**

You can choose to install all components at once or install individual components one at a time.

Use the following steps to install and set up Server Manager and its components.

1. Install the Server Manager packages:

Ubuntu: **dpkg –i contrail-server-manager–installer_<version-sku>.deb**



NOTE: Make sure to select the correct version package that corresponds to the platform for which you are installing.

2. Set up the Server Manager components. Use the **setup.sh** command to install all of the components, or you can install individual components.


```
cd /opt/contrail/contrail_server_manager; ./setup.sh [--hostip=<ip address>]
[--domain=<domain name>]
```

- To set up all components:

```
./setup.sh --all
```

- To set up only the Server Manager server:

```
./setup.sh --sm=contrail-server-manager_<version-sku>.deb
```

- To set up only the Server Manager client:

```
setup.sh --sm-client=contrail-server-manager_<version-sku>.deb
```

- To set up only the Server Manager user interface:

```
setup.sh --webui=contrail-server-manager_<version-sku>.deb
```

- To set up only Server Manager monitoring:

```
setup.sh --sm-mon=contrail-server-manager_<version-sku>.deb
```

Other options include:

- --sm-cliff-client
- --nowebui
- --nosm-mon

3. Installation logs are located at `/var/log/contrail/install_logs/`.

Finishing the Provisioning

The Server Manager service does not start automatically upon successful installation. You must finish the provisioning by modifying the following templates. Refer to the sample configuration section included in this topic for details about configuring these files.

```
/etc/cobbler/dhcp.template
/etc/cobbler/named.template
/etc/bind/named.conf.options
/etc/cobbler/setting
/etc/cobbler/modules.conf
/etc/mail/sendmail.cf
```

Starting the Server Manager Service

When you are finished modifying the templates to match your environment, start the Server Manager service using the following command:

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

Upgrading Server Manager Software

If you are upgrading Server Manager software from a previous version to the current version, use the following guidelines to ensure successful installation.

Prerequisite to Upgrading

Before upgrading, you must remove the previous version of the Server Manager installer. Remove any existing Server Manager installer package from the system using the following steps.

1. **`dpkg -P contrail-server-manager-installer`**
2. **`rm -rf /opt/contrail/contrail-server-manager`**

Steps for a New Installation

After the existing Server Manager installer package has been removed, use the installation steps for a new installation, see details in previous section:

1. **`dpkg -i <contrail-server-manager-installer*.deb>`**
2. **`cd /opt/contrail/contrail-server-manager`**
3. **`./setup.sh -all`**
4. After the setup script has completed running, you can restart Server Manager by issuing:

`service contrail-server-manager restart`

It is not necessary to reconfigure the templates of DHCP, bind, and so on. Previous template configurations and configured data are preserved during the upgrade.

Server Manager Installation Completion Checks

The following are various checks you can use to investigate the status of your Server Manager installation.

Server Manager Checks

Use the following to check that the Server Manager installation is complete.

- Use the following commands to verify that the services are running:

`service contrail-server-manager status`

`service cobblerd status`

`service bind9 status`

`service isc-dhcp-server status`

- Also verify processes using the following command:

```
ps auwx | grep Passenger
```

Server Manager Client Checks

- Verify the items listed using the following command:

```
which server-manager
```

- Check the client configuration at
`/opt/contrail/contrail-server-manager/client/sm-client-config.ini`
- Make sure `listen_ip_addr` is configured with the Server Manager IP address.

Server Manager Webui Checks

- Verify the status of the Server Manager webui using the following command:

```
service supervisor-webui status
```

- Check the webui access from the browser:
 - Contrail release 2.2 and lower—`http:<server manager> :8080`
 - Contrail release 3.0 and greater—`http:<server manager> :9080`.

Sample Configurations for Server Manager Templates

The following are sample parameters for the Server Manager templates. Use settings specific for your environment. Typically you configure to parameters for DHCP, bind, and e-mail services.

Sample Settings

```
bind_master: 10.84.11.6
```

```
manage_forward_zones: ['contrail.juniper.net']
```

```
manage_reverse_zones: ['10.84.11']
```

```
next_server: 10.84.11.6
```

```
server: 10.84.11.6
```

The dhcp.template File

Add Server Manager hooks into the DHCP template. When DHCP commit, release, or expire actions occur, the Server Manager is notified. The DHCP servers are detected on the Server Manager and the *Discovered* status is maintained.

Define subnet blocks that the DHCP server needs to support, using the sample format given in the `/etc/cobbler/dhcp.template` file.

The `named.conf.options` File

You must configure the following:

```
forwarders {  
    0.0.0.0;  
};  
  
allow-query { any; };  
  
recursion yes;
```

The `named.template` File

Include the following in the beginning of the `named.template` file:

```
"/etc/bind/named.conf.options";  
  
"/etc/bind/named.conf.local";
```

The `sendmail.cf` File

The `sendmail.cf` template is present with a juniper.net configuration. Populate it with configuration specific to your environment. The Server Manager uses the template to generate e-mails when reimaging or provisioning is completed.

Related Documentation

- [Using Server Manager to Automate Provisioning on page 116](#)
- [Using the Server Manager Web User Interface on page 151](#)

Using Server Manager to Automate Provisioning

- [Overview of Server Manager on page 117](#)
- [Server Manager Requirements and Assumptions on page 117](#)
- [Server Manager Component Interactions on page 118](#)
- [Configuring Server Manager on page 119](#)
- [Configuring the Cobbler DHCP Template on page 120](#)
- [User-Defined Tags for Server Manager on page 121](#)
- [Server Manager Client Configuration File on page 121](#)
- [Restart Services on page 123](#)
- [Accessing Server Manager on page 123](#)
- [Communicating with the Server Manager Client on page 123](#)
- [Server Manager Commands for Configuring Servers on page 124](#)
- [Server Manager REST API Calls on page 142](#)
- [Example: Reimaging and Provisioning a Server on page 148](#)

Overview of Server Manager

The Contrail Server Manager can be used to provision, configure, and reimage a Contrail virtual network system of servers, clusters, and nodes. Server Manager is an alternative to using Fabric commands to provision a Contrail system.

This section describes the functions and usage guidelines for the Contrail Server Manager.

The Server Manager provides a simple, centralized way for users to manage and configure components of a virtual network system running across multiple physical and virtual servers in a cloud infrastructure.

You can use Server Manager to configure, provision, and reimage servers with the correct software version and packages for the nodes that are running on each server in multiple virtual network system clusters.

The Server Manager:

- Provides REST APIs to handle customer requests.
- Manages its own database to store information about the servers.
- Interacts with other open source products such as Cobbler and Puppet to configure servers based on user requests.

Server Manager Requirements and Assumptions

The following are requirements and assumptions for the Server Manager:

- The Server Manager runs on a Linux server (bare metal or virtual machine) and assumes availability of several software products with which it interacts to provide the functionality of managing servers.
- The Server Manager has network connectivity to the servers it is trying to manage.
- The Server Manager has access to a remote power management tool to power cycle the servers that it manages.
- The Server Manager uses Cobbler software for Linux provisioning to configure and download software to physical servers. Cobbler resides on the same server that is running the Server Manager daemon.
 - Server Manager assumes that DNS and DHCP servers embedded with Cobbler provide IP addresses and names to the servers being managed, although it is possible to use external DNS and DHCP servers.
- The Server Manager uses Puppet software, an open source configuration management tool, to accomplish the configuration management of target servers, including the installation and configuration of different software packages and the launching of various services.
- SQLite3 database management software is used to maintain and manage server configurations and it runs on the same machine where the Server Manager daemon is running.

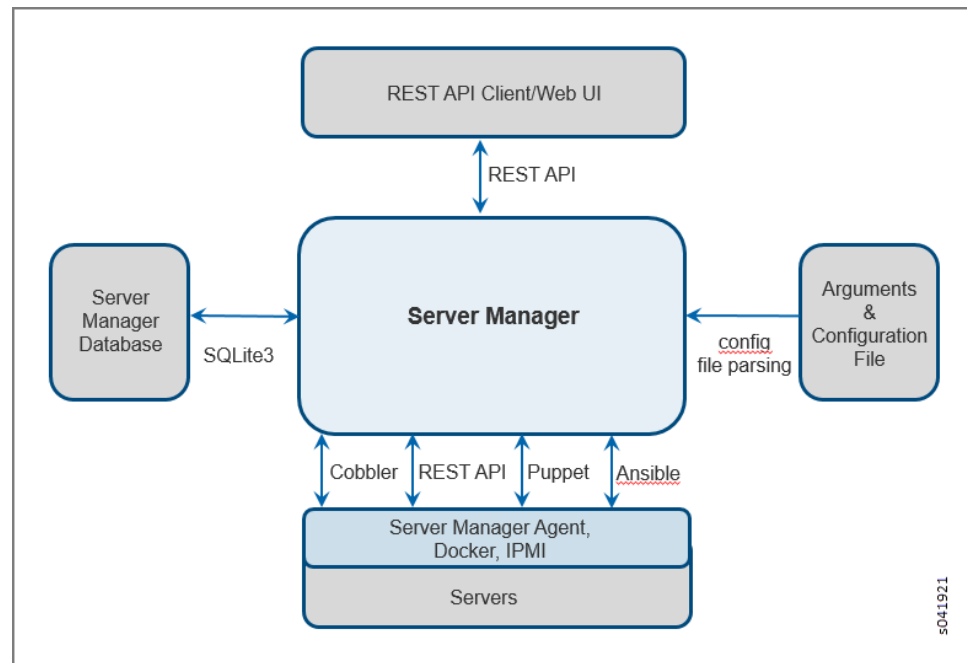
Server Manager Component Interactions

The Server Manager runs as a daemon and provides REST APIs for interaction with the client. The Server Manager accepts user input in the form of REST API requests, performs the requested function on the resources, and responds with a REST API response.

Configuration parameters required by the Server Manager are provided in the Server Manager configuration file, however, the parameters can be overridden by Server Manager command line parameters.

Figure 19 on page 118 illustrates several high-level components with which the Server Manager interacts.

Figure 19: Server Manager Component Interactions



Internally, the Server Manager uses a SQLite3 database to hold server configuration information. The Server Manager coordinates the database configuration information and user requests to manage the servers defined in the database.

While managing the servers, the Server Manager also communicates with other software components. It uses Cobbler for reimagining target servers and it uses Puppet for provisioning, thereby ensuring necessary software packages are installed and configured, required services are running, and so on.

A Server Manager agent runs on each of the servers and communicates with the Server Manager, providing the information needed to monitor the operation of the servers. The Server Manager agent also uses REST APIs to communicate with the Server Manager, and it can use other software tools to fetch other information, such as Nagios infrastructure monitoring, Razor spam filtering, Intelligent Platform Interface (IPMI), and so on. Monitoring functionality is not part of Server Manager.

Configuring Server Manager

When the installation of all Server Manager components and dependent packages is finished, configure the Server Manager with parameters that identify your environment and make it available for clients to serve REST API requests.

Upon installation, a sample Server Manager configuration file is created at:

`/opt/contrail/server_manager/sm-config.ini`

Modify the **sm-config.ini** configuration file to include parameter values specific to your environment.

The environment-specific configuration section of the **sm-config.ini** file is named **SERVER-MANAGER**.

The following example shows the format and parameters of the **SERVER-MANAGER** section. Typically, only the **listen_ip_addr**, **cobbler_username**, and **cobbler_passwd** values need to be modified.

```
[SERVER-MANAGER]

listen_ip_addr = <IP-Address-of-SM>

listen_port    = <port-number>

database_name  = <database-file-name>

server_manager_base_dir = <base-dir-where-SM-files-are-created>

html_root_dir  = <html-root-dir>

cobbler_ip_address = <cobbler-ip-address>

cobbler_port    = <cobbler-port-number>

cobbler_username = <cobbler-username>

cobbler_password = <cobbler-password>

puppet_dir     = <puppet-directory>

ipmi_username  = <IPMI username>

ipmi_password  = <IPMI password>

ipmi_type      = <IPMI type>
```

[Table 5 on page 120](#) provides details for each of the parameters in the **SERVER-MANAGER** section.

Table 5: Server Manager Parameters

Parameter	Configuration
<code>listen_ip_addr</code>	Specify the IP address of the server on which the Server Manager is listening for REST API requests.
<code>listen_port</code>	The port number on which the Server Manager is listening for REST API requests. The default is 9001.
<code>database_name</code>	The name of the database file where the Server Manager stores configuration information. This file is created under <code>server_manager_base_dir</code> .
<code>server_manager_base_dir</code>	The base directory where all of the Server Manager configuration files are created. The default is <code>/etc/contrail_smgr</code> .
<code>html_root_dir</code>	The HTML root directory (<code>/var/www/html</code>).
<code>cobbler_ip_address</code>	The IP address used to access Cobbler. This address MUST be the same address as the <code>listen_ip_address</code> . The Server Manager assumes that the Cobbler service is running on the same server as the Server Manager service.
<code>cobbler_port</code>	The port on which Cobbler listens for user requests. Leave this field blank.
<code>cobbler_username</code>	Specify the user name to access the Cobbler service. Specify testing unless your Cobbler settings have been modified to use a different user name.
<code>cobbler_password</code>	Specify the password to access the Cobbler service. Specify testing unless your Cobbler settings have been modified to use a different password.
<code>puppet_dir</code>	The directory where the Puppet manifests and templates are created. This is <code>/etc/puppet</code> , unless your Puppet configuration has been modified to use another directory.
<code>ipmi_username</code>	The IPMI username for power management.
<code>ipmi_password</code>	The IPMI password for power management.
<code>ipmi_type</code>	The IPMI type (ipmilan, or other cobbler supported types).

Configuring the Cobbler DHCP Template

In addition to configuring the `server_config.ini` file, you must manually change the settings in the `/etc/cobbler/dhcp.template` file to use the correct subnet address, mask, and DNS domain name for your environment. Optionally, you can also restrict the use of the current instance of Server Manager and Cobbler to a subset of servers in the network.

Below is a snippet from the `dhcp.template` file showing the fields to be modified.



NOTE: The IP addresses and other values in the following are shown for example purposes only. Be sure to use values that are correct for your environment.


```

subnet <subnet address> netmask <netmask address> {
    option routers                <ip-address>;
    option subnet-mask            <subnet-address>;
    option domain-name-servers    $next_server, 8.8.8.8 ;
    option domain-search          "<domain name1>t", "<domain name 2>";
    option domain-name            "<domain name>" ;
    option ntp-servers            $next_server;
    default-lease-time            21600;
    max-lease-time                43200;
    next-server                   $next_server;
    filename                      "/pxelinux.0";
}

```

User-Defined Tags for Server Manager

Server Manager allows you to define tags that can be used to group servers for performing a particular operation such as show information, reimage, provision, and so on. The Server Manager supports up to seven different tags that can be configured and used for grouping servers.

The names of user-defined tags are kept in the **tags.ini** file, at **/etc/contrail_smgr/tags.ini**.

It is possible to modify tag names, and add or remove tags dynamically using the Server Manager REST API interface. However, if a tag is already being used to group servers, the tag must be removed from the servers before tag modification is allowed.

The following is a sample **tags.ini** file that is copied on installation. In the sample file, five tags are defined – **datacenter**, **floor**, **hall**, **rack**, and **user_tag**. Use the tags to group servers together.

```

[TAGS]
tag1 = datacenter
tag2 = floor
tag3 = hall
tag4 = rack
tag5 = user_tag

```

Server Manager Client Configuration File

The Server Manager client application installation copies the **/opt/contrail/server_manager/client/sm-client-config.ini**, sample configuration file. The sample file contains parameter values such as the IP address to reach the Server Manager, the port used by Server Manager, default values for cluster table entries, default values for server table entries, and so on. You must modify the values in the **sm-client-config.ini** file to match your environment.

Use values from the **CLUSTER** and **SERVER** subsections in the **ini** file during creation of new clusters and servers, unless you explicitly override the values at the time of creation.

The **CLUSTER** and **SERVER** subsections have fields that represent the password for a host or a service. If a value for the password field is not explicitly provided, the Server Manager selects a default password.

Starting with Contrail Release 3.0.2, if you don't explicitly specify a password, a password is automatically generated by the system. This makes the clusters provisioned by Server Manager more secure.

The following fields now autogenerate a password whenever an explicit password is not provided.

- Previous Parameters:
 - mysql_root_password
 - keystone_password
 - heat_encryption_key
- New Parameters:
 - mysql:root_password
 - mysql:service_password
 - keystone:admin_password
 - heat_encryption_key

The following is a sample client configuration file.

```
[SERVER-MANAGER]
; ip address of the server manager
; replace the following with proper server manager address
listen_ip_addr = 10.xx.xx.xx
; server manager listening port
listen_port    = 9001
[CLUSTER]
subnet_mask = <subnet address>
domain = <domain name>
database_dir = /home/cassandra
encapsulation_priority = MPLSoUDP,MPLSoGRE,VXLAN
router_asn = 64512
keystone_username = admin
keystone_password = <password>
password = <password>
analytics_data_ttl = 168
haproxy = disable
use_certificates = False
multi_tenancy = False
database_token =
service_token = <password>
analytics_data_ttl = 168
[SERVER]
```

Restart Services

When all user changes have been made to the configuration files, restart the Server Manager by issuing the following command so that it runs with the modifications :

```
service contrail-server-manager restart
```

Accessing Server Manager

When the Server Manager configuration has been customized to your environment, and the required daemon services are running, clients can request and use services of the Server Manager by using REST APIs. Any standard REST API client can be used to construct and send REST API requests and process Server Manager responses.

The following steps are typically required to fully implement a new cluster of servers being managed by the Server Manager.

1. Configure elements such as servers, clusters, and images in the Server Manager.

Before managing servers, the Server Manager needs to have configuration details of the servers that the Server Manager is managing.

2. Specify the name and location of boot images, packages, and repositories used to bring up the servers with needed software.

Currently, the servers can be imaged with CentOS, Ubuntu Linux, or VMWare ESXi distributions.

3. Provision or configure the servers by installing necessary packages, creating configuration files, and bringing up the correct services so that each server can perform the functions or role(s) configured for that server.

A Contrail system of servers has several components or roles that work together to provide the functionality of the virtual network system, including: control, config, analytics, compute, web-ui, OpenStack, and database. Each of the roles has different requirements for the software and services needed. The provisioning REST API enables the client to configure the roles on servers using the Server Manager.

4. Set up API calls for monitoring servers.

Once the servers in the Contrail system are correctly reimaged and provisioned to run configured roles, the server monitoring REST API calls allow clients to monitor performance of the servers as they provide one or more role functions. Monitoring functionality is not available in Server Manager for Contrail Release 1.10.

Communicating with the Server Manager Client

Server Manager provides a REST API interface for clients to talk to the Server Manager software. Any client that can send and receive REST API requests and responses can be used to communicate with Server Manager, for example, Curl or Postman. Additionally, the Server Manager software provides a client with a simplified CLI interface, in a separate

package. The Server Manager client can be installed and run on the Server Manager machine itself or on another server with an IP connection to the Server Manager machine.

Prior to using the Server Manager client CLI commands, you need to modify the **sm-client-config.ini** file to specify the IP address and the port for the Server Manager, along with other parameters.

Each of the commands described in this section takes a set of parameters you specify, constructs a REST API request to the Server Manager, and provides the server's response.

The following describes each Server Manager client CLI command in detail.

Server Manager Commands for Configuring Servers

This section describes commands that are used to configure servers and server parameters in the Server Manager database. These commands allow you to add, modify, delete, or view servers.

- [Create New Servers or Update Existing Servers on page 124](#)
- [Delete Servers on page 126](#)
- [Show Server Configuration on page 127](#)
- [Server Manager Commands for Managing Clusters on page 127](#)
- [Server Manager Commands for Managing Tags on page 130](#)
- [Server Manager Commands for Managing Images on page 132](#)
- [Server Manager Operational Commands for Managing Servers on page 136](#)
- [Reimaging Server\(s\) on page 136](#)
- [Provisioning and Configuring Roles on Servers on page 137](#)
- [Adding and Deleting Roles on page 139](#)
- [Restarting Server\(s\) on page 140](#)
- [Show Status of Server\(s\) on page 141](#)

Create New Servers or Update Existing Servers

Use the **server-manager add** command to create a new server or update a server in the Server Manager database.

```
server-manager [-h] [--config_file CONFIG_FILE] server [-f FILE_NAME]
```

[Table 6 on page 124](#) lists the optional arguments.

Table 6: Server Manager Add Server Command Options

Option	Description
-h, --help	Show the options available for the current command and exit.
--config_file CONFIG_FILE, -c CONFIG_FILE	The name of the Server Manager client configuration file. The default file is: /opt/contrail/server_manager/client/sm-client-config.ini

Table 6: Server Manager Add Server Command Options (continued)

Option	Description
<code>--file_name FILE_NAME, -f FILE_NAME</code>	The JSON file that contains the server parameter values.

If no JSON file is specified, the client program accepts all the needed server parameter values interactively, then builds a JSON file and makes a REST API call to the Server Manager. The JSON file contains a number of server entries, in the format shown in the following example:

```
{
  "server": [
    {
      "id": "demo2-server",
      "mac_address": "<mac address>",
      "ip_address": "<ip address>",
      "parameters" : {
        "interface_name": "eth1",
        "partition": ""
      },
      "roles" :
["config","openstack","control","compute","collector","webui","database"],
      "cluster_id": "demo-cluster",
      "subnet_mask": "<subnet mask address>",
      "gateway": "<ip address>",
      "password": "<password>",
      "domain": "<domain name>",
      "ipmi_address": "<ipmi address>",
      "tag" : {
        "datacenter" : "demo-dc",
        "floor" : "demo-floor",
        "hall" : "demo-hall",
        "rack" : "demo-rack",
        "user_tag" : "demo-user"
      }
    }
  ]
}
```

```

    }
  ]
}
```

Most of the parameters in the JSON sample file are self-explanatory. **Cluster_id** defines the cluster to which the server belongs. The **interface_name** is the Ethernet interface name on the server used to configure the server and **roles** define the roles that can be configured for the server. The sample **roles** array in the example lists all valid role values. **Tag** defines the list of tag values for grouping and classifying the server.

The **server-manager add** command adds a new entry if the server with the given ID or **mac_address** does not exist in the Server Manager database. If an entry already exists, the add command modifies the fields in the existing entry with any new parameters specified.

Delete Servers

Use the **server-manager delete** command to delete one or more servers from the Server Manager database.

[Table 7 on page 126](#) lists the optional arguments.

Table 7: Server Manager Delete Server Command Options

Option	Description
-h, --help	Show the options available for the current command and exit.
--config_file CONFIG_FILE, -c CONFIG_FILE	The name of the Server Manager client configuration file. The default file is: /opt/contrail/server_manager/client/sm-client-config.ini
--server_id SERVER_ID	The server ID for the server or servers to be deleted.
--mac MAC	The MAC address for the server or servers to be deleted.
--ip IP	The IP address for the server or servers to be deleted.
--cluster_id CLUSTER_ID	The cluster ID for the server or servers to be deleted.
--tag TagName=TagValue	The TagName that is to be matched with the Tagvalue. Up to seven TagName and Tagvalue pairs separated by commas can be provided.

The criteria for identifying servers to be deleted can be specified by providing the **server_id** or the server: **mac address**, **ip**, **cluster_id**, or the **TagName = TagValue**.

Provide one of the server matching criteria to display a list of servers available to be deleted.

Show Server Configuration

Use the **server-manager show** command to display the configuration of servers from the Server Manager database.

```
server-manager show [--config_file CONFIG_FILE]
                    server (--server_id SERVER_ID | --mac MAC | --ip IP |
                    --cluster_id CLUSTER_ID | --tag <tag_name=tag_value>.. ) [--detail]
```

Table 8 on page 127 lists the optional arguments.

Table 8: Server Manager Show Server Command Options

Option	Description
-h, --help	Show the options available for the current command and exit.
--config_file CONFIG_FILE, -c CONFIG_FILE	The name of the Server Manager client configuration file. The default file is: /opt/contrail/server_manager/client/sm-client-config.ini
--server_id SERVER_ID	The server ID for the server or servers to be deleted.
--mac MAC	The MAC address for the server or servers to be displayed.
--ip IP	The IP address for the server or servers to be displayed.
--cluster_id CLUSTER_ID	The cluster ID for the server or servers to be displayed.
--tag TagName=TagValue	The TagName that is to be matched with the Tagvalue. Up to seven TagName and Tagvalue pairs separated by commas can be provided.
--detail, -d	Flag to indicate if details are requested.

The criteria for identifying servers to be displayed can be specified by providing the **server_id** or the server: **mac address**, **ip**, **VNS_id**, **cluster_id**, or **TagName=TagValue**.

Provide one or more of the server matching criteria to display a list of servers.

Server Manager Commands for Managing Clusters

A cluster is used to store parameter values that are common to all servers belonging to that cluster. The commands in this section facilitate managing clusters in the Server Manager database, enabling you to add, modify, delete, and view clusters.



NOTE: Whenever a server is created with a specific `cluster_id`, Server Manager checks to see if a cluster with that ID has already been created. If there is no matching `cluster_id` already in the database, an error is returned.

- [Create a New Cluster or Update an Existing Cluster on page 128](#)
- [Delete a Cluster on page 129](#)
- [Show Cluster Configuration on page 130](#)

Create a New Cluster or Update an Existing Cluster

Use the **server-manager add** command to create a new cluster or update an existing cluster in the Server Manager database.

```
server-manager [-h] [--config_file CONFIG_FILE]
               cluster [--file_name FILE_NAME]
```

[Table 9 on page 128](#) lists the optional arguments.

Table 9: Server Manager Add Cluster Command Options

Option	Description
<code>-h, --help</code>	Show the options available for the current command and exit.
<code>--config_file CONFIG_FILE, -c CONFIG_FILE</code>	The name of the Server Manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>
<code>--file_name FILE_NAME, -f FILE_NAME</code>	The JSON file that contains the cluster parameter values.

If no JSON file is specified, the client program accepts all the needed cluster parameter values interactively, then builds a JSON file and makes a REST API call to the Server Manager. The JSON file contains a number of cluster entries, in the format shown in the following example:

```
{
  "cluster" : [
    {
      "id" : "demo-cluster",
      "parameters" : {
        "router_asn": "<asn number>",
        "database_dir": "/home/cassandra",
        "database_token": "",
        "use_certificates": "False",
        "multi_tenancy": "False",
```



```

        "encapsulation_priority": "MPLSoUDP,MPLSoGRE,VXLAN",
        "service_token": "<password>",
        "keystone_username": "admin",
        "keystone_password": "<password>",
        "keystone_tenant": "admin",
        "analytics_data_ttl": "168",
        "haproxy": "disable",
        "subnet_mask": "<subnet mask address>",
        "gateway": "<ip address>",
        "password": "<password>",
        "external_bgp": "",
        "domain": "<domain name>"
    }
}
]
}

```

Server membership to a cluster is determined by specifying the ID corresponding to the cluster when defining the server. All of the cluster parameters are available to the server when provisioning roles on the server.

Delete a Cluster

Use the **server-manager delete** command to delete a cluster from the Server Manager database that are no longer needed. Use this command after all servers in the cluster have been deleted.

```

server-manager delete [-h] [--config_file CONFIG_FILE]
                    cluster [--cluster_id CLUSTER_ID]

```

Table 10 on page 129 lists the optional arguments.

Table 10: Server Manager Delete Cluster Command Options

Option	Description
-h, --help	Show the options available for the current command and exit.
--config_file CONFIG_FILE, -c CONFIG_FILE	The name of the Server Manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>

Show Cluster Configuration

Use the **server-manager show** command to list the configuration of a cluster.

```
server-manager show [-h] [ --config_file CONFIG_FILE]
                    cluster [--cluster_id CLUSTER_ID] [--detail]
```

[Table 11 on page 130](#) lists the optional arguments.

Table 11: Server Manager Show Cluster Command Options

Option	Description
-h, --help	Show the options available for the current command and exit.
--config_file CONFIG_FILE, -c CONFIG_FILE	The name of the Server Manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>
--detail, -d	Flag to indicate if details are requested.
--cluster_id CLUSTER_ID	The cluster ID for the cluster or clusters.

You can optionally specify a cluster ID to get information about a particular cluster. If the optional parameter is not specified, information about all clusters in the system is returned.

Server Manager Commands for Managing Tags

Tags are used for grouping servers together so that an operation such as get, reimage, provision, status, and so on can be easily performed on servers that have matching tags. The Server Manager provides a flexible way for you to define your own tags, then use those tags to assign values to servers. Servers with matching tag values can be easily grouped together. The Server Manager can store a maximum of seven tag values. At initialization, the Server Manager reads the tag names from the configuration file. The tag names can be retrieved or modified using CLI commands. When modifying tag names, the Server Manager ensures that the tag name being modified is not used by any of the server entries.

- [Create a New Tag or Update an Existing Tag on page 130](#)
- [Show Tag Configuration on page 131](#)

Create a New Tag or Update an Existing Tag

Use the **server-manager add** command to create a new tag or update an existing tag in the Server Manager database.

```
server-manager add [-h] [--config_file CONFIG_FILE]
                  tag [--file_name FILE_NAME]
```

Optional arguments include the following:

Option	Description
<code>-h, --help</code>	Show the options available for the current command and exit.
<code>--config_file CONFIG_FILE, -c CONFIG_FILE</code>	The name of the Server Manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>
<code>--file_name FILE_NAME, -f FILE_NAME</code>	The JSON file that contains the tag names.

If no JSON file is specified, the client program prompts you for tag names, then builds a JSON file and makes a REST API call to the Server Manager. The JSON file contains a number of tag entries, in the format shown in the following example:

```
{
    "tag1" : "data-center",
    "tag2" : "floor",
    "tag3" : "",
    "tag4" : "pod",
    "tag5" : "rack",
}
```

In the example, you specify a JSON file to add or modify the tags, tag1 thru tag5. For tag3, the "" value specifies that if the tag is defined prior to the CLI command, it is removed on execution of the command. The tag name for tag1 is set to data-center. This is allowed if, and only if, none of the server entries are using tag1.

Show Tag Configuration

Use the **server-manager show** command to list the configuration of a tag.

```
server-manager show [-h] [ --config_file CONFIG_FILE] tag
```

Optional arguments include the following:

Option	Description
<code>-h, --help</code>	Show the options available for the current command and exit.
<code>--config_file CONFIG_FILE, -c CONFIG_FILE</code>	The name of the Server Manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>

The following is sample output for the **show tag** command.

```
{
    "tag1": "datacenter",
```

```

    "tag2": "floor",
    "tag3": "hall",
    "tag4": "rack",
    "tag5": "user_tag"
  }

```

Server Manager Commands for Managing Images

In addition to servers and clusters, the Server Manager also manages information about images and packages that can be used to reimage and configure servers. Images and packages are both stored in the database as images. When new images are added to the database, or existing images are modified or deleted, the Server Manager interfaces with Cobbler to make corresponding modifications in the Cobbler distribution profile for the specified image.

The image types supported are summarized in the following table:

Image Type	Description
centos	Manages the CentOS stock ISO, and does not include the Contrail packages repository packaged with the ISO.
contrail-centos-package	Maintains a repository of the package to be installed on the CentOS system image.
ubuntu	Manages the base Ubuntu ISO.
contrail-ubuntu-package	Maintains a repository of packages that contain Contrail and dependent packages to be installed on an Ubuntu base system.
ESXi5.1/ESXi5.5	Manages VMware ESXi 5.1 or 5.5 ISO.

- [Creating New Images or Updating Existing Images on page 132](#)
- [Add an Image on page 133](#)
- [Upload an Image on page 134](#)
- [Delete an Image on page 135](#)
- [Show Image Configuration on page 135](#)

Creating New Images or Updating Existing Images

The Server Manager maintains five types of images – CISO, Ubuntu ISO, ESXi hypervisor ISO, Contrail CentOS package, and Contrail Ubuntu package.

Use the **server-manager add** command or the **server-manager upload** command to add new images to the Server Manager database.

- Use **add** when the new image is present locally on the Server Manager machine. The path provided is the image path on the Server Manager machine.

- Use **upload_image** when the new image is present on the machine where the client program is being invoked. The path provided is the image path on the client machine.

Add an Image

```
server-manager add [-h] [ --config_file CONFIG_FILE]
                  image [--file_name FILE_NAME]
```

Optional arguments include the following:

Option	Description
-h, --help	Show the options available for the current command and exit.
--config_file CONFIG_FILE, -c CONFIG_FILE	The name of the Server Manager client configuration file. The default file is: /opt/contrail/server_manager/client/sm-client-config.ini
--file_name FILE_NAME, -f FILE_NAME	The name of the JSON file that contains the image parameter values.

If no JSON file is specified, the client program accepts parameter values interactively, then builds a JSON file and makes a REST API call to the Server Manager.

The JSON file contains an array of possible entries, in the following sample format. The sample shows three images: one CentOS ISO containing Contrail packages, one Ubuntu base ISO, and one Contrail Ubuntu package. When the images are added, corresponding distribution, profile, and repository entries are created in Cobbler by the Server Manager.

```
{
  "image": [
    {
      "id": "ubuntu-12.04.3",
      "type": "ubuntu",
      "version": "ubuntu-12.04.3",
      "path": "/iso/ubuntu-12.04.3-server-amd64.iso"
    },
    {
      "id": "centos-6.4",
      "type": "centos",
      "version": "centos-6.4",
      "path": "/iso/CentOS-6.4-x86_64-minimal.iso"
    },
  ],
}
```

```

    {
        "id": "contrail-ubuntu-r11-b33",
        "type": "contrail-ubuntu-package",
        "version": "contrail-ubuntu-r11-b33",
        "path": "/iso/contrail-install-packages_x.xx-xx_all.deb"
    }
]
}

```

Upload an Image

The server-manager **upload_image** command is similar to the **server-manager add** command, except that the path provided for the image being added is the local path on the client machine. This command is useful if the client is being run remotely, not on the Server Manager machine, and the image being added is not physically present on the Server Manager machine.

```

server-manager upload_image [-h]
                           --config_file CONFIG_FILE]
                           image_id image_version image_type file_name

```

Positional arguments include the following:

Option	Description
image_id	Name of the new image.
image_version	Version number of the new image.
image_type	Type of image: fedora , centos , ubuntu , contrail-ubuntu-package , contrail-centos-package
file_name	Complete path for the file.

Optional arguments include the following:

Option	Description
-h, --help	Show the options available for the current command and exit.
--config_file CONFIG_FILE, -c CONFIG_FILE	The name of the Server Manager client configuration file. The default file is: /opt/contrail/server_manager/client/sm-client-config.ini

Delete an Image

Use the **server-manager delete** command to delete an image from the Server Manager database. When an image is deleted from the Server Manager database, the corresponding distribution, profile, or repository for the image is also deleted from the Cobbler database.

```
server-manager delete [-h] [ --config_file CONFIG_FILE]
image image_id
```

Positional arguments include the following:

Option	Description
image_id	The image ID for the image to be deleted.

Optional arguments include the following:

Option	Description
-h, --help	Show the options available for the current command and exit.
--config_file CONFIG_FILE, -c CONFIG_FILE	The name of the Server Manager client configuration file. The default file is: /opt/contrail/server_manager/client/sm-client-config.ini

Show Image Configuration

Use the **server-manager show** command to list the configuration of images from the Server Manager database. If the detail flag is specified, detailed information about the image is returned. If the optional **image_id** is not specified, information about all the images is returned.

```
server-manager [-h] [--config_file CONFIG_FILE] image [--image_id IMAGE_ID]
[--detail]
```

Optional arguments include the following:

Option	Description
-h, --help	Show the options available for the current command and exit.
--config_file CONFIG_FILE, -c CONFIG_FILE	The name of the Server Manager client configuration file. The default file is: /opt/contrail/server_manager/client/sm-client-config.ini
image_id	The image ID for the image or images.
--detail, -d	Flag to indicate if details are requested.

Server Manager Operational Commands for Managing Servers

The Server Manager commands in the following sections are operational commands for performing a specific operation on a server or a group of servers. These commands assume that the base configuration of entities required to execute the operational commands is already completed using configuration CLI commands.

Reimaging Server(s)

Use the **server-manager reimage** command to reimage a server or servers with a provided base ISO and package. Servers are specified by providing match conditions to select them from the database.

Before issuing the **reimage** command, the images must be added to the Server Manager using the **create image** command, which also adds the images to Cobbler. The set of servers to be reimaged can be specified by providing match criteria for servers already added to the Server Manager database. Use the **server_id** or server: **vns_id**, **cluster_id**, **pod_id**, or **rack_id**.

You must identify the base image ID to be used to reimage, plus any optional Contrail package to be used. When a Contrail package is provided, a local repository is created that can be used for subsequent provisioning of reimaged servers.

The command prompts for a confirmation before making the REST API call to the Server Manager to start reimaging the servers. This confirmation message can be bypassed by specifying the optional **--no_confirm** or **-F** parameter on the command line.

```
server-manager reimage [-h]
                        [ --config_file CONFIG_FILE]
                        [--package_image_id PACKAGE_IMAGE_ID]
                        [--no_reboot]
                        (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag
                        <tag_name=tag_value>)
                        [--no_confirm]
                        base_image_id
```

Positional arguments include the following:

Option	Description
base_image_id	The image ID of the base image to be used.

Optional arguments include the following:

Option	Description
-h, --help	Show the options available for the current command and exit.

Option	Description
<code>--config_file CONFIG_FILE, -c CONFIG_FILE</code>	The name of the Server Manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>
<code>--package_image_id PACKAGE_IMAGE_ID, -p PACKAGE_IMAGE_ID</code>	The optional Contrail package to be used to reimage the server or servers.
<code>--no_reboot, -n</code>	Optional parameter to indicate that the server should not be rebooted following the reimage setup.
<code>--server_id SERVER_ID</code>	The server ID for the server or servers to be reimaged.
<code>--cluster_id CLUSTER_ID</code>	The cluster ID for the server or servers to be reimaged.
<code>--tag TagName=TagValue</code>	TagName which is to be matched with Tagvalue
<code>--no_confirm, -F</code>	Flag to bypass confirmation message, default = do NOT bypass.

Provisioning and Configuring Roles on Servers

Use the **server-manager provision** command to provision identified server(s) with configured roles for the virtual network system. The servers can be selected from the database configuration (using standard server match criteria), identified in a JSON file, or provided interactively.

From the configuration of servers in the database, the Server Manager determines which roles to configure on which servers and uses this information along with other server and VNS parameters from the database to achieve the task of configuring the servers with specific roles.

When the **server-manager provision** command is used, the Server Manager builds the manifest files corresponding to each of the servers and pushes them to the Puppet agent for execution upon the servers.

```
server-manager provision [-h]
                        [--config_file CONFIG_FILE]
                        (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag
<tag_name=tag_value> | --provision_params_file PROVISION_PARAMS_FILE |
--interactive)
                        [--no_confirm]
                        package_image_id
```

Positional arguments include the following:

Option	Description
<code>package_image_id</code>	The Contrail package image ID to be used for provisioning.

Optional arguments include the following:

Option	Description
<code>-h, --help</code>	Show the options available for the current command and exit.
<code>--config_file CONFIG_FILE, -c CONFIG_FILE</code>	The name of the Server Manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>
<code>--server_id SERVER_ID</code>	The server ID for the server or servers to be provisioned.
<code>--cluster_id CLUSTER_ID</code>	The cluster ID for the server or servers to be provisioned.
<code>--tag TagName=TagValue</code>	TagName to be matched with Tagvalue.
<code>--provision_params_file PROVISION_PARAMS_FILE, -f PROVISION_PARAMS_FILE</code>	Optional JSON file containing the parameters for provisioning the server(s).
<code>--interactive, -l</code>	Flag indicating that you are manually entering the server parameters for provisioning.
<code>--no_confirm, -F</code>	Flag to bypass confirmation message, default = do NOT bypass.

You can specify roles different from what is configured in the database by using the JSON file option parameter. When using the file option, the rest of the server parameters, the cluster parameters, and the list of servers must be configured before using the provision command. The following is a sample format for the file option:

```
{
  "roles" : {
    "database" : ["demo2-server"],
    "openstack" : ["demo2-server"],
    "config" : ["demo2-server"],
    "control" : ["demo2-server"],
    "collector" : ["demo2-server"],
    "webui" : ["demo2-server"],
    "compute" : ["demo2-server"]
  }
}
```

The final option for specifying roles for provisioning servers is to specify the `--interactive` option flag. When the provision command is used, you are prompted to enter role definitions interactively.

Adding and Deleting Roles

You can add or delete roles through server manager to expand or shrink your cluster. For example, you might want to expand the number of controllers or compute nodes when there is more traffic. Conversely, you could shrink the number of controllers or compute nodes when there is less traffic.

Adding and deleting roles is done by modifying the roles section in the server's JSON and issuing the provision command.

The following example shows the JSON to remove all of the controller roles from a server cluster.

```
{
  "cluster_id": "test-cluster",
  "id": "<ID>",
  "ip_address": "<ip address>",
  "roles": [
    "config",
    "control",
    "collector",
    "webui",
    "database",
    "openstack"
  ]
},
```

The resulting role changes are shown in the following.

```
{
  "cluster_id": "test-cluster",
  "id": "<ID>",
  "ip_address": "<ip address>",
  "roles": [
  ]
},
```

Use **server-manager add** and **server-manager provision** for the whole cluster. Provisioning the whole cluster enables the references to the server to be added or removed in the other controllers of the cluster.

The following example shows how to add roles to a server cluster, assuming the server initially has no roles, as shown.

```
{
  "cluster_id": "test-cluster",
  "id": "<ID>",
  "ip_address": "<ip address>",
  "roles": [
  ]
},
```

The following shows the addition of a controller and its roles.

```
{
  "cluster_id": "test-cluster",
  "id": "<ID>",
  "ip_address": "<ip address>",
  "roles": [
```

```
        "config",  
        "control",  
        "collector",  
        "webui",  
        "database",  
        "openstack"  
    ],  
    },  
}
```

Use **server-manager add** and **server-manager provision** for the whole cluster. Provisioning the whole cluster enables the references to the server to be added or removed in the other controllers of the cluster.

A similar procedure can be followed to add a compute node.



NOTE: The following are caveats for adding and deleting roles:

- Adding and deleting roles is supported only on a high availability (HA) cluster.
- You cannot add and delete in a single provision command. You can perform all adds in a single provision or all deletes in a single provision.
- A node that is down or unreachable cannot be deleted, because the Puppet agent cannot run to perform the add or delete.

Restarting Server(s)

Use the **server-manager restart** command to reboot identified server(s). Servers can be specified from the database by providing standard match conditions. The **restart** command provides a way to reboot or power-cycle the servers, using the Server Manager REST API interface. If reimaging is intended, use the **restart** command with the **net-boot** flag enabled. When netbooted, the Puppet agent is also installed and configured on the servers. If there are Puppet manifest files created for the server prior to rebooting, the agent pulls those from the Server Manager and executes the configured Puppet manifests. The **restart** command uses an IPMI mechanism to power cycle the servers, if available and configured. Otherwise, the **restart** command uses SSH to the server and the existing reboot command mechanism is used.

```
server-manager restart [-h]  
  
    [ --config_file CONFIG_FILE]  
  
    (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag  
<tag_name=tag_value>)  
  
    [--net_boot]  
    [--no_confirm]
```

Optional arguments include the following:

Option	Description
<code>-h, --help</code>	Show the options available for the current command and exit.
<code>--config_file CONFIG_FILE, -c CONFIG_FILE</code>	The name of the Server Manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code> .
<code>--server_id SERVER_ID</code>	The server ID for the server or servers to be restarted.
<code>--cluster_id CLUSTER_ID</code>	The cluster ID for the server or servers to be restarted.
<code>--tag TagName=TagValue</code>	TagName to be matched with Tagvalue.
<code>--net_boot, -n</code>	Optional parameter to indicate if the server should be netbooted.
<code>--no_confirm, -F</code>	Flag to bypass confirmation message, default = do NOT bypass.

Show Status of Server(s)

Use the **server-manager status** command to view the reimaging or provisioning status of server(s).

```
server-manager status server [-h]
                             [--config_file CONFIG_FILE]
                             (--server_id SERVER_ID | --cluster_id CLUSTER_ID
                             | --tag <tag_name=tag_value>)
```

Optional arguments include the following:

Option	Description
<code>-h, --help</code>	Show the options available for the current command and exit.
<code>--config_file CONFIG_FILE, -c CONFIG_FILE</code>	The name of the Server Manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>
<code>--server_id SERVER_ID</code>	The server ID for the server whose status is to be fetched.
<code>--cluster_id CLUSTER_ID</code>	The cluster ID for the server or servers to be restarted.
<code>--tag TagName=TagValue</code>	TagName to be matched with Tagvalue.

The status command provides a way to fetch the current status of a server.

Status outputs include the following:

restart_issued
reimage_started
provision_started
provision_completed
database_started
database_completed
openstack_started
openstack_completed
config_started
config_completed
control_started
control_completed
collector_started
collector_completed
webui_started
webui_completed
compute_started
compute_completed

Server Manager REST API Calls

This section describes all of the REST API calls to the Server Manager. Each description includes an example configuration.

- [REST APIs for Server Manager Configuration Database Entries on page 143](#)
- [API: Add a Server on page 143](#)
- [API: Delete Servers on page 144](#)
- [API: Retrieve Server Configuration on page 144](#)
- [API: Add an Image on page 144](#)
- [API: Upload an Image on page 145](#)
- [API: Get Image Information on page 145](#)
- [API: Delete an Image on page 145](#)
- [API: Add or Modify a Cluster on page 146](#)
- [API: Delete a Cluster on page 146](#)
- [API: Get Cluster Configuration on page 147](#)
- [API: Get All Server Manager Configurations on page 147](#)
- [API: Reimage Servers on page 147](#)

- [API: Provision Servers on page 147](#)
- [API: Restart Servers on page 148](#)

REST APIs for Server Manager Configuration Database Entries

The REST API calls in this section help in configuring different elements in the Server Manager database.



NOTE: The IP addresses and other values in the following are shown for example purposes only. Be sure to use values that are correct for your environment.

API: Add a Server

To add a new server to the service manager configuration database:

URL: `http://<SM-IP-Address>:<SM-Port>/server`

Method: **PUT**

Payload: JSON payload containing an array of servers to be added. For each server in the array, all the parameters are specified as JSON fields. The mask, gateway, password, and domain fields are optional, and if not specified, the values of these fields are taken from the cluster to which the server belongs.

The following is a sample JSON file for adding a server.

```
{
  "server": [
    {
      "id": "demo2-server",
      "mac_address": "<mac address>",
      "ip_address": "<ip address>",
      "parameters": {
        "interface_name": "eth1"
      },
      "roles": [
        "config",
        "openstack",
        "control",
        "compute",
        "collector",
        "webui",
        "database"
      ],
      "cluster_id": "<cluster name>",
      "mask": "<mask address>",
      "gateway": "<ip address>",
      "password": "<password>",
      "domain": "<domain name>",
      "email": "id@company.net",
      "tag": {
        "datacenter": "demo-dc",
        "rack": "demo-rack"
      },
    },
  ],
}
```

```
    }
  ]
}
```

API: Delete Servers

Use one of the following formats to delete a server.

URL: `http://<SM-IP-Address>:<SM-Port>/server?server_id=SERVER_ID`

`http://<SM-IP-Address>:<SM-Port>/server?cluster_id=CLUSTER_ID`

`http://<SM-IP-Address>:<SM-Port>/server?mac=MAC`

`http://<SM-IP-Address>:<SM-Port>/server?ip=IP`

`http://<SM-IP-Address>:<SM-Port>/server[?tag=<tag_name>=<tag_value>,,]`

Method : DELETE

Payload : None

API: Retrieve Server Configuration

Use one of the following methods to retrieve a server configuration. The detail argument is optional, and specified as part of the URL if details of the server entry are requested.

URL: `http://<SM-IP-Address>:<SM-Port>/server[?server_id=SERVER_ID&detail]`

`http://<SM-IP-Address>:<SM-Port>/server[?cluster_id=CLUSTER_ID&detail]`

`http://<SM-IP-Address>:<SM-Port>/server[?tag=<tag_name>=<tag_value>,,]`

`http://<SM-IP-Address>:<SM-Port>/server[?mac=MAC&detail]`

`http://<SM-IP-Address>:<SM-Port>/server[?ip=IP&detail]`

`http://<SM-IP-Address>:<SM-Port>/server[?tag=<tag_name>=<tag_value>,,]`

Method : GET

Payload : None

API: Add an Image

Use the following to add a new image to the Server Manager configuration database from the Server Manager machine.

An image is either an ISO for a CentOS or Ubuntu distribution or an Ubuntu Contrail package repository. When adding an image, the image file is assumed to be available on the Server Manager machine.

URL : `http://<SM-IP-Address>:<SM-Port>/image`

Method: PUT

Payload: Specifies all the parameters that define the image being added.

```
{
  "image": [
    {
      "id": "Image-id",
      "type": "image_type", <ubuntu or centos or esxi5.1 or esxi5.5 or
      contrail-ubuntu-package or contrail-centos-package>
      "version": "image_version",
      "path": "path-to-image-on-server-manager-machine"
    }
  ]
}
```

API: Upload an Image

Use the following to upload a new image from a client to the Server Manager configuration database.

An image is an ISO for a CentOS or Ubuntu distribution or an Ubuntu Contrail package repository. Add image assumes the file is available on the Server Manager, whereas upload image transfers the image file from the client machine to the Server Manager machine.

URL : **http://<SM-IP-Address>:<SM-Port>/image/upload**

Method: **PUT**

Payload: Specifies all the parameters that define the image being added.

```
{
  "image": [
    {
      "id": "Image-id",
      "type": "image_type", <ubuntu or centos or esxi5.1 or esxi5.5 or
      contrail-ubuntu-package or contrail-centos-package>
      "version": "image_version",
      "path": "path-to-image-on-client-machine"
    }
  ]
}
```

API: Get Image Information

Use the following to get image information.

URL : **http://<SM-IP-Address>:<SM-Port>/image[?image_id=IMAGE_ID&detail]**

Method: **GET**

Payload: Specifies criteria for the image being sought. If no match criteria is specified, information about all the images is provided. The details field specifies if details of the image entry in the database are requested.

API: Delete an Image

Use the following to delete an image.

URL: **http://<SM-IP-Address>:<SM-Port>/image?image_id=IMAGE_ID**

Method: **DELETE**

Payload: Specifies criteria for the image being deleted.

API: Add or Modify a Cluster

Use the following to add a cluster to the Server Manager configuration database. A cluster maintains parameters for a set of servers that work together in different roles to provide complete functions for a Contrail cluster.

URL: **http://<SM-IP-Address>:<SM-Port>/cluster**

Method: **PUT**

Payload: Contains the definition of the cluster, including all the global parameters needed by all the servers in the cluster. The subnet_mask, gateway, password, and domain fields define parameters that apply to all servers in the VNS. These parameter values can be individually overridden for a server by specifying different values in the server entry.

```
{
  "cluster" : [
    {
      "id" : "demo-cluster",
      "parameters" : {
        "router_asn": "<asn number>",
        "database_dir": "/home/cassandra",
        "database_token": "",
        "use_certificates": "False",
        "multi_tenancy": "False",
        "encapsulation_priority": "MPLSoUDP,MPLSoGRE,VXLAN",
        "service_token": "<password>",
        "keystone_user": "admin",
        "keystone_password": "<password>",
        "keystone_tenant": "admin",
        "analytics_data_ttl": "168",
        "subnet_mask": "<subnet mask address>",
        "gateway": "<ip address>",
        "password": "<password>",
        "haproxy": "disable",
        "external_bgp": "",
        "domain": "<domain name>"
      }
    }
  ]
}
```

API: Delete a Cluster

Use this API to delete a cluster from the Server Manager database.

URL: **http://<SM-IP-Address>:<SM-Port>/cluster?cluster_id=CLUSTER_ID**

Method: **DELETE**

Payload: None

API: Get Cluster Configuration

Use this API to get a cluster configuration.

URL: `http://<SM-IP-Address>:<SM-Port>/cluster[?cluster_id=CLUSTER_ID&detail]`

Method: **GET**

Payload: None

The optional detail argument is specified as part of the URL if details of the VNS entry are requested.

API: Get All Server Manager Configurations

Use this API to get all configurations of Server Manager objects, including servers, clusters, images, and tags.

URL: `http://<SM-IP-Address>:<SM-Port>/all[?detail]`

Method: **GET**

Payload: None

The optional detail argument is specified as part of the URL if details of the Server Manager configuration are requested.

API: Reimage Servers

Use one of the following API formats to reimage one or more servers.

URL: `http://<SM-IP-Address>:<SM-Port>/server/reimage?server_id=SERVER_ID`

`http://<SM-IP-Address>:<SM-Port>/server/reimage?cluster_id=CLUSTER_ID`

`http://<SM-IP-Address>:<SM-Port>/server/reimage?mac=MAC`

`http://<SM-IP-Address>:<SM-Port>/server/reimage?ip=IP`

`http://<SM-IP-Address>:<SM-Port>/server/reimage [?tag=<tag_name>=<tag_value>,.]`

Method: **POST**

Payload: None

API: Provision Servers

Use this API to provision or configure one or more servers for roles configured on them.

URL: `http://<SM-IP-Address>:<SM-Port>/server/provision`

Method: **POST**

Payload: Specifies the criteria to be used to identify servers which are being provisioned. The servers can be identified by server_id, mac, cluster_id or tags. See the following example.

```
{
  server_id : <server_id> OR
```

```

mac : <server_mac_address> OR
cluster_id : <cluster_id> OR
tag : {"data-center" : "dc1"} OR
provision_parameters = {
  "roles" : {
    "database" : ["demo2-server"],
    "openstack" : ["demo2-server"],
    "config" : ["demo2-server"],
    "control" : ["demo2-server"],
    "collector" : ["demo2-server"],
    "webui" : ["demo2-server"],
    "compute" : ["demo2-server"]
  }
}
}

```

API: Restart Servers

This REST API is used to power cycle the servers and reboot either with net-booting enabled or disabled.

If the servers are to be reimaged and reprovisioned, the **net-boot** flag should be set.

If servers are only being reprovisioned, the **net-boot** flag is not needed, however, the Puppet agent must be running on the target systems with the correct puppet configuration to communicate to the puppet master running on the Server Manager.

URL: `http://<SM-IP-Address>:<SM-Port>/server/restart?server_id=SERVER_ID`
`http://<SM-IP-Address>:<SM-Port>/server/restart?[netboot&]cluster_id=CLUSTER_ID`
`http://<SM-IP-Address>:<SM-Port>/server/restart? [netboot&]mac=MAC`
`http://<SM-IP-Address>:<SM-Port>/server/restart? [netboot&]ip=IP`
`http://<SM-IP-Address>:<SM-Port>/server/restart ?`
`[netboot&]tag=<tag_name>=<tag_value>`

Method: **POST**

Payload: Specifies the criteria to be used to identify servers which are being restarted. The servers can be identified by their **server_id**, **mac**, **cluster_id**, or **tag**. The netboot parameter specifies if the servers being power-cycled are to be booted from Cobbler or locally.

Example: Reimaging and Provisioning a Server

This example shows the steps used in Server Manager software to configure, reimage, and provision a server running all roles of the Contrail system in a single-node configuration.



NOTE: Component names and IP addresses in the following are used for example only. To use this example in your own environment, be sure to use addresses and names specific to your environment.

The Server Manager client configuration file used for the following CLI commands, is `/opt/contrail/server_manager/client/sm-client-config.ini`. It contains the values for the server IP address and port number as follows:

[SERVER-MANAGER]

listen_ip_addr = 192.168.1.10 (Server Manager IP address)

listen_port = 9001

The steps to be followed include:

1. Configure cluster.
2. Configure servers.
3. Configure images.
4. Reimage servers (either using servers configured above or using explicitly specified reimage parameters with the request).
5. Provision servers (either using servers configured above or using explicitly specified provision parameters with the request).

1. Configure a cluster.

server-manager add cluster -f cluster.json

Where cluster.json contains :

```
{
  "cluster" : [
    {
      "id" : "demo-cluster",
      "parameters" : {
        "router_asn": "<asn number>",
        "database_dir": "/home/cassandra",
        "database_token": "",
        "use_certificates": "False",
        "multi_tenancy": "False",
        "encapsulation_priority": "MPLSoUDP,MPLSoGRE,VXLAN",
        "service_token": "<password>",
        "keystone_user": "admin",
        "keystone_password": "<password>",
        "keystone_tenant": "admin",
        "analytics_data_ttl": "168",
        "subnet_mask": "<subnet mask address>",
        "gateway": "<ip address>",
        "password": "<password>",
        "haproxy": "disable",
        "external_bgp": "",
        "domain": "<domain name>"
      }
    }
  ]
}
```

```
}
```

2. Configure the server.

server-manager add server -f server.json

Where server.json contains :

```
{
  "server": [
    {
      "id": "demo2-server",
      "mac_address": "<mac address>",
      "ip_address": "<ip-address>",
      "parameters" : {
        "interface_name": "eth1",
      },
      "roles" :
      ["config","openstack","control","compute","collector","webui","database"],
      "cluster_id": "demo-cluster",
      "subnet_mask": "<subnet mask address>",
      "gateway": "<ip-address>",
      "password": "<password>",
      "domain": "<domain name>t",
      "ipmi_address": "<ip-address>"
    }
  ]
}
```

3. Configure images.

In the example, the image files for **ubuntu-12.04.3** and **contrail-ubuntu-164** are located at the corresponding image path specified on the Server Manager.

server-manager add -c smgr_client_config.ini image -f image.json

Where image.json contains:

```
{
  "image": [
    {
      "id": "ubuntu-12.04.3",
      "type": "ubuntu",
      "version": "ubuntu-12.04.3",
      "path": "/iso/ubuntu-12.04.3-server-amd64.iso"
    },
    {
      "id": "contrail-ubuntu-xxx",
      "type": "contrail-ubuntu-package",
      "version": "contrail-ubuntu-xxx",
      "path": "/iso/contrail-install-packages_x.xx-xxx~xxxxxxxxx_all.deb"
    }
  ]
}
```

4. Reimage servers.

This step can be performed after the configuration from the previous steps is in the Server Manager database.

```
server-manager reimage --server_id demo-server -r contrail-ubuntu-164 ubuntu-12.04.3
```

5. Provision servers.

```
server-manager provision --server_id demo-server contrail-ubuntu-164
```



NOTE: Optionally, the Contrail package to be used can be specified with the `reimage` command, in which case the repository with the Contrail packages is created and made available to the target nodes as part of the `reimage` process. The repository is a mandatory parameter of the `provision` command.

See Also • [Using the Server Manager Web User Interface on page 151](#)

Using the Server Manager Web User Interface

When the Server Manager is installed on your Contrail system, you can also install a Server Manager Web user interface that you can use to access the features of Server Manager.

- [Log In to Server Manager on page 151](#)
- [Create a Cluster for Server Manager on page 152](#)
- [Working with Servers in the Server Manager User Interface on page 159](#)
- [Add a Server on page 159](#)
- [Edit Tags for Servers on page 161](#)
- [Using the Edit Config Option for Multiple Servers on page 161](#)
- [Filter Servers by Tag on page 162](#)
- [Viewing Server Details on page 162](#)
- [Configuring Images and Packages on page 163](#)
- [Add New Image or Package on page 163](#)
- [Selecting Server Manager Actions for Clusters on page 164](#)
- [Reimage a Cluster on page 164](#)
- [Provision a Cluster on page 164](#)

Log In to Server Manager

The Server Manager user interface can be accessed using:

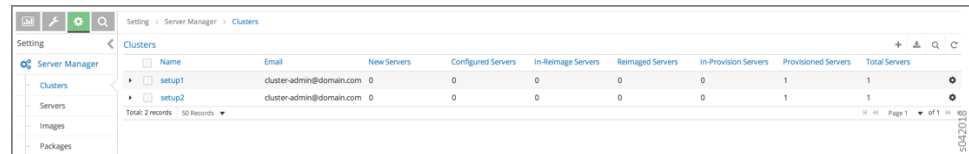
```
http://<server-manager-user-interface-ip>:9080
```

Where `<server-manager-user-interface-ip>` is the IP address of the server on which the Server Manager web user interface is installed.

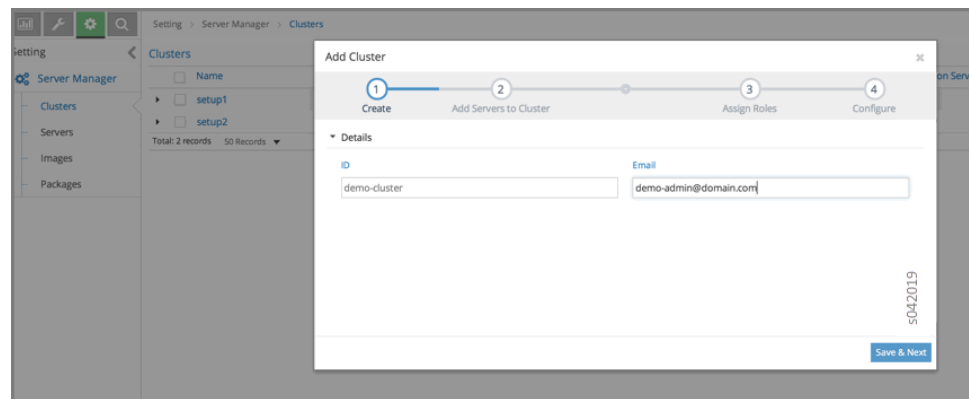
From the Contrail user interface, select **Setting > Server Manager** to access the Server Manager home page. From this page you can manage Server Manager settings for clusters, servers, images, and packages.

Create a Cluster for Server Manager

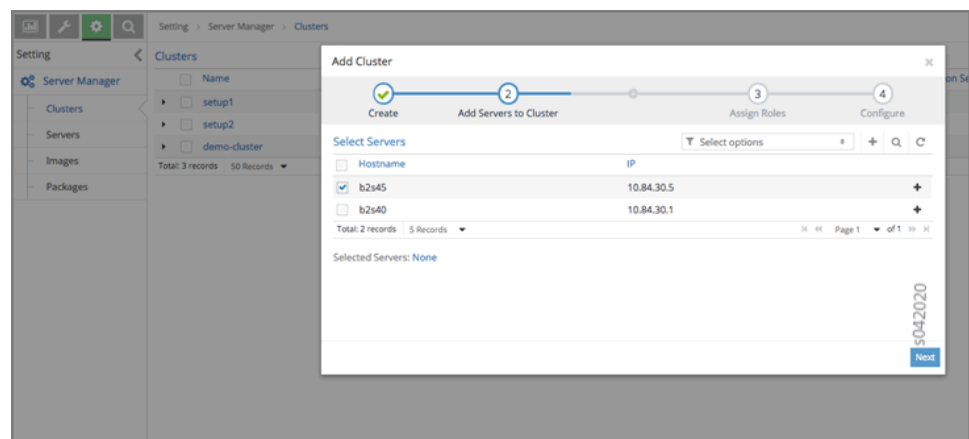
Select **Add Cluster** to identify a cluster to be managed by the Server Manager. Select **Setting > Server Manager > Clusters**, to access the **Clusters** page, as shown:



To create a new cluster, click the plus icon in the upper right of the **Clusters** page. The **Add Cluster** window is displayed. In the **Add Cluster** window, you can add a new cluster ID and the domain e-mail address of the cluster.

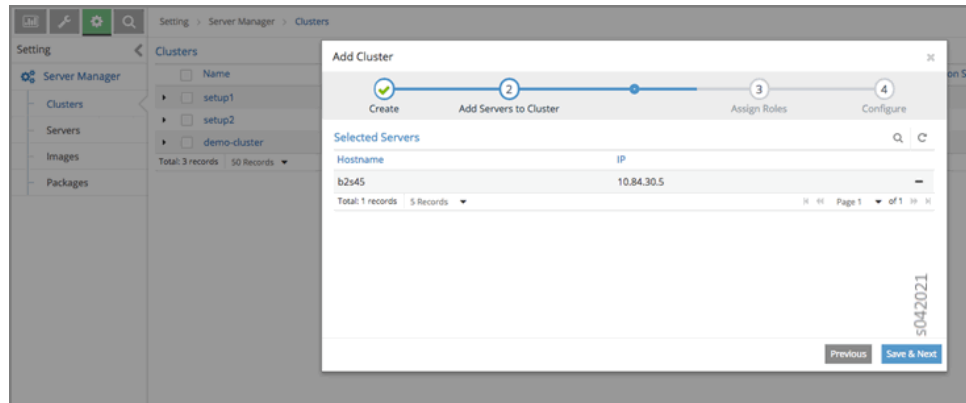


When you are finished adding information about the new cluster in the **Add Clusters** window, click **Save & Next**. Now you can add servers to the cluster, as shown in the following.

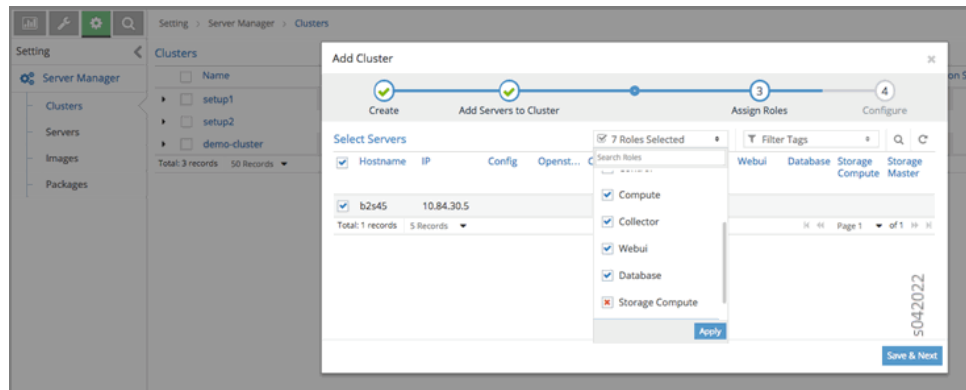


Click the check box of each server to be added to the cluster.

When you are finished, click **Next**. The selected servers are added to the cluster.

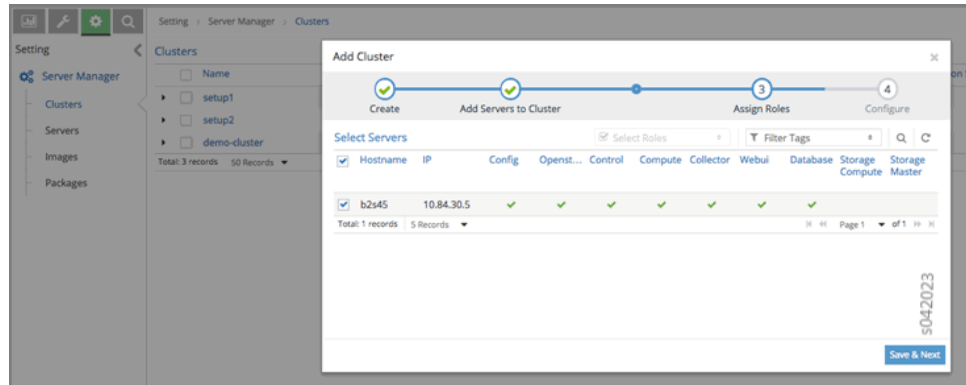


When you are finished adding servers, click **Save & Next**. Now you can assign Contrail roles to servers that you select in the cluster. Roles available are Config, OpenStack, Control, Compute, and Collector. Select each role assignment for the selected server. You can also unselect any assigned role. The assigned roles correspond to the role functions in operation on the server.

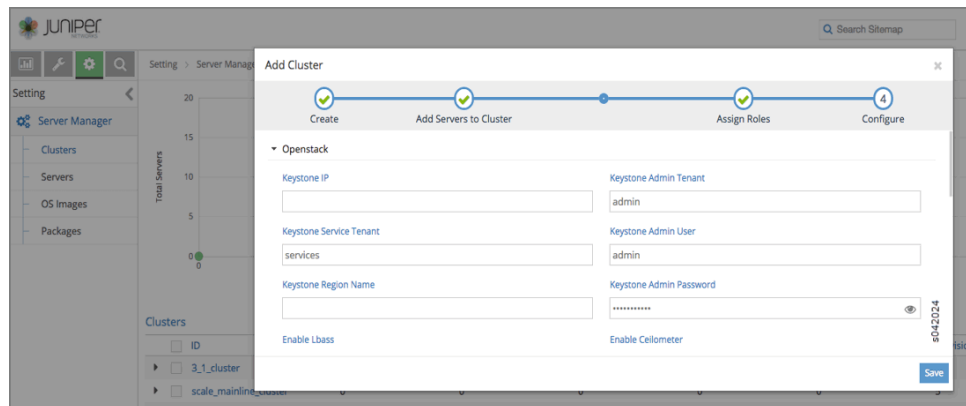


When you are finished selecting roles for the selected server in the **Roles** window, click **Apply** to save your choices.

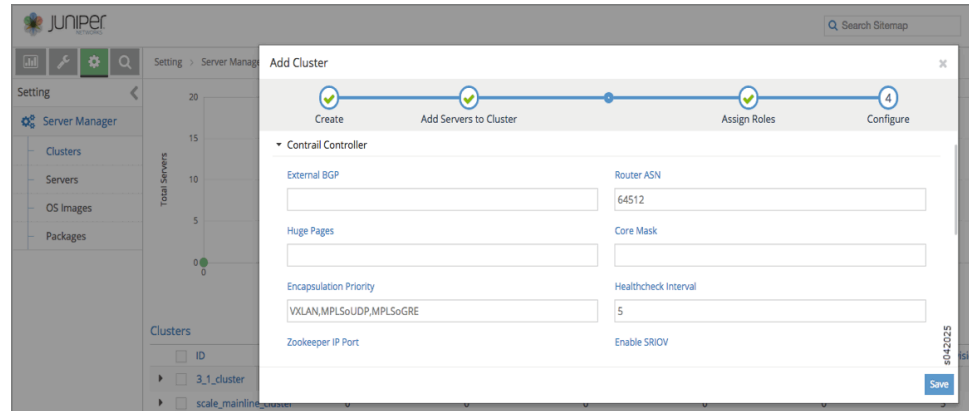
Click **Save & Next** to view your selections. Check marks are displayed in the columns of the **Add Cluster** window, see the following image.



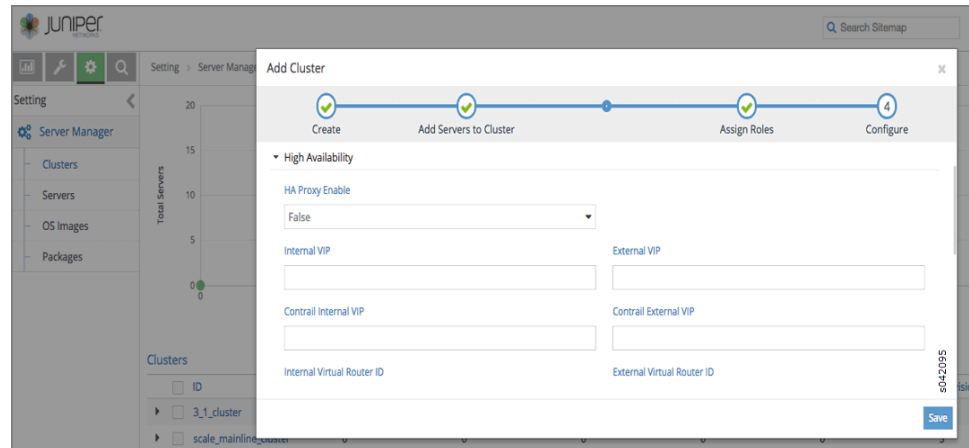
The next step after roles are assigned is to enter the cluster configuration information for OpenStack. After viewing the assigned roles, click **Save & Next**. The **Add Cluster** window is displayed. Click an icon that opens a set of fields where you can enter OpenStack or Contrail configuration information for the cluster. In the following image, the **Openstack** icon is selected. You can enter configuration information, such as OpenStack or Keystone passwords and usernames in the fields.



In the following image, the Contrail controller icon is selected. You can enter configuration information for Contrail, such as **External BGP**, **Multi Tenancy**, **Router ASN**, **HA Proxy**, and so on.



In the following image, the High Availability (HA) icon is selected. You can configure high availability parameters such as **HA Proxy**, **Internal** and **External VIP**, and so on.



In the following image, the **Analytics** icon is selected. Here the user can configure parameters for Contrail Analytics, including **TTL**, **Syslog Port**, **Data Dir**, and so on.

Add Cluster

Progress: Create (✓), Add Servers to Cluster (✓), Assign Roles (✓), Configure (4)

Analytics

Syslog Port: -1

Topology Scan Frequency (Seconds): 60

SNMP Scan Frequency (Seconds): 600

SNMP Fast Scan Frequency (Seconds): 60

Analytics Data TTL: 48

Analytics Flow TTL: 2

Analytics Config Audit TTL:

Analytics Stats TTL:

Save

In following image, the **Contrail Storage** icon is selected. You can configure parameters for Contrail Storage, including **Monitoring Key**, **OSD Bootstrap Key**, **Admin Key**, and so on.

Add Cluster

Progress: Create (✓), Add Servers to Cluster (✓), Assign Roles (✓), Configure (4)

Contrail Storage

Storage Monitor Secret:

OSD Bootstrap Key:

Storage Admin Key:

Live Migration Storage Scope: Select

Live Migration IP:

Live Migration Host:

Storage Enabled: ☐

Save

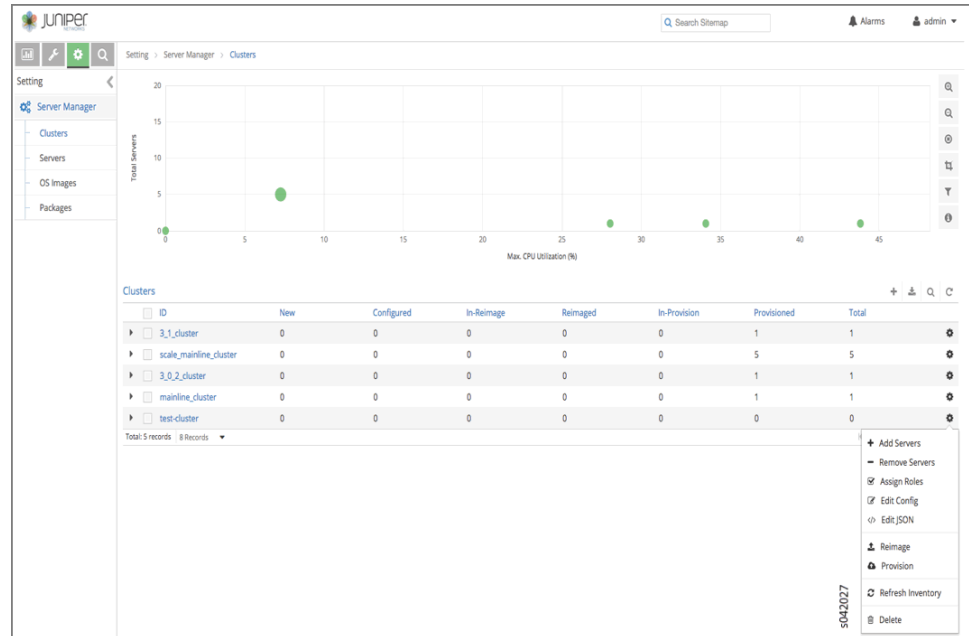
When you are finished entering all of the cluster configuration information, click **Save** to submit the configurations. You can view all configured clusters on the **Clusters** window by selecting **Setting > Server Manager > Clusters**.

Clusters

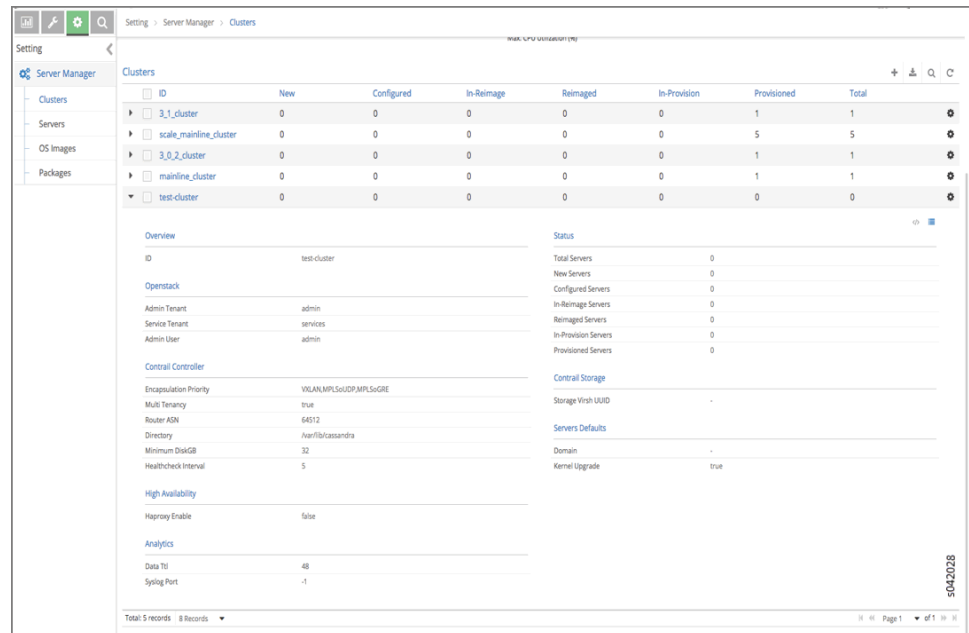
Name	Email	New Servers	Configured Servers	In-Reimage Servers	Reimage Servers	In-Provision Servers	Provisioned Servers
setup1	cluster-admin@domain.com	0	0	0	0	0	0
setup2	cluster-admin@domain.com	0	0	0	0	0	1
demo-cluster	demo-admin@domain.com	0	0	0	0	0	1

Total: 3 records 50 Records

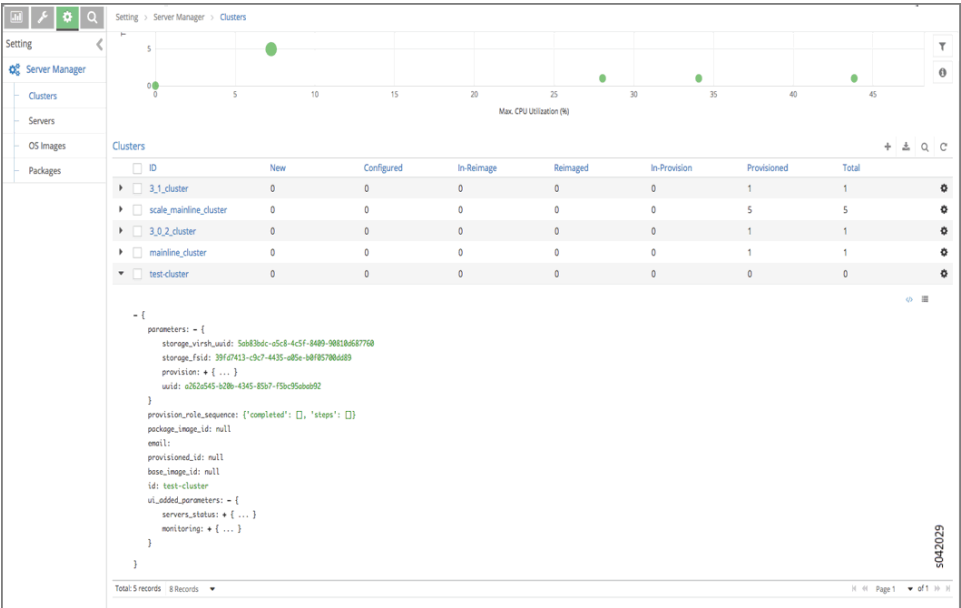
To perform an action on one of the configured clusters, click the gear wheel icon at the right to select from a menu of actions available for that cluster, including **Add Servers**, **Remove Servers**, **Assign Roles**, **Edit Config**, **Reimage**, **Provision**, and **Delete**, as shown.



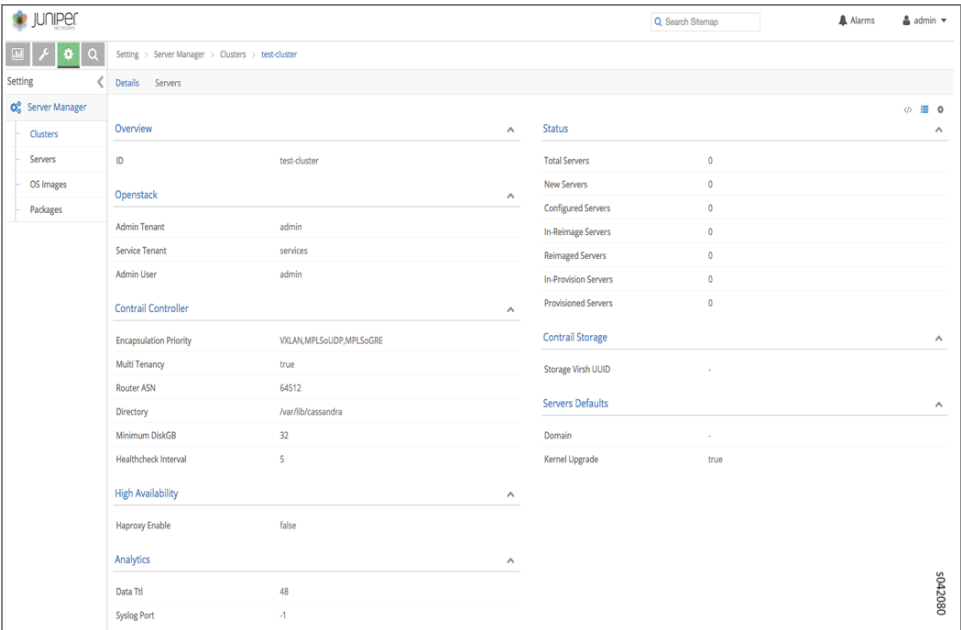
You can also click the expansion icon on the left side of the cluster name to display the details of that cluster in an area below the name line, as shown.



Click the upper right icon to switch to the JSON view to see the contents of the JSON file for the cluster.

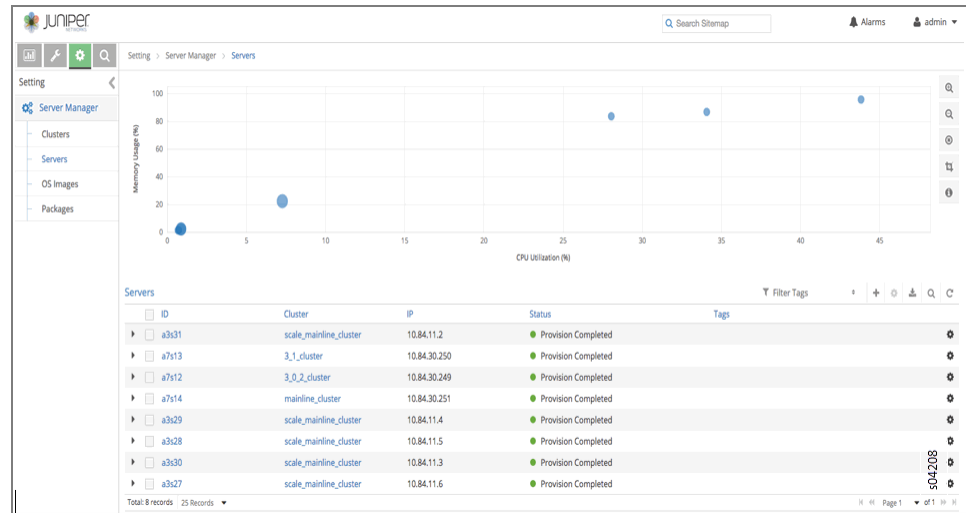


The cluster name is a link, click the cluster name to display the cluster **Details** page, as shown.



Working with Servers in the Server Manager User Interface

Select **Setting > Server Manager** and click the **Servers** link in the left sidebar at to view a list of all servers.



Add a Server

To add a new server, select **Setting > Server Manager > Servers** and click the plus (+) icon at the upper right side in the header line. The **Add Server** window is displayed, as in the following.

The screenshot shows the 'Add Server' window in the Juniper Server Manager interface. The window is titled 'Add Server' and has a close button (X) in the top right corner. It contains two main sections: 'System Management' and 'Interfaces'. The 'System Management' section includes fields for ID (demo-server), Password, Host Name (demo-server), Domain (englab.juniper.net), Static IP (10.84.60.148), IPMI Username (ADMIN), and IPMI Password (*****). There is also a 'Partition' field. The 'Interfaces' section is currently collapsed. At the bottom right, there are 'Cancel' and 'Save' buttons. A date stamp 's042082' is visible on the right side of the window.

In the following image, the **Interfaces** icon is selected. You can add new interfaces or edit existing interfaces. To enable editing for any field, hover the cursor on any selected field to open it.

The screenshot shows the 'Add Server' window with the 'Physical Interfaces' section expanded. It displays a table with columns: Name, IP/Mask, MAC Address, Gateway, DHCP, TOR, and TOR Port. The first row contains the values: eth01, 1.2.3.4, aa:aa:aa:aa:aa:aa, 2.2.3.4, a checked checkbox, an empty field, and a dash. Below the table is an '+Add' button. Other sections like 'Bond Interfaces', 'OVS Type Switches', 'Contrail Storage', and 'Provisioning' are collapsed. The bottom status bar shows '3_1_cluster', '10.84.30.250', and 'Provision Completed'.

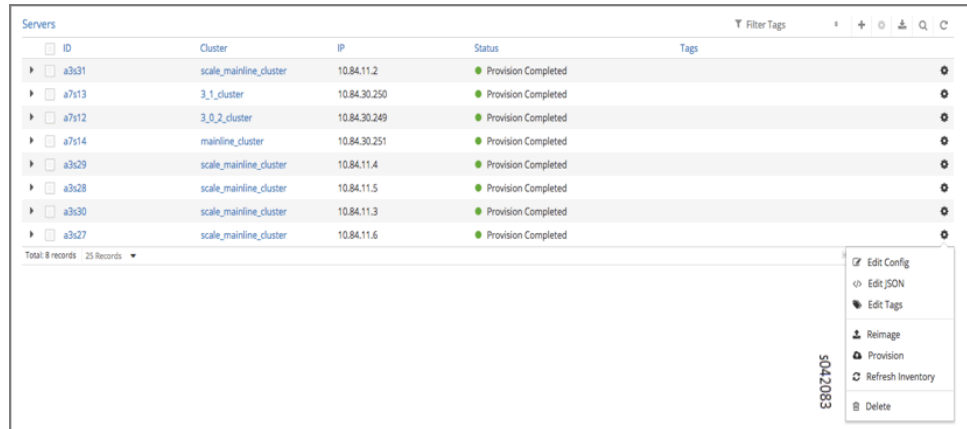
Name	IP/Mask	MAC Address	Gateway	DHCP	TOR	TOR Port
eth01	1.2.3.4	aa:aa:aa:aa:aa:aa	2.2.3.4	<input checked="" type="checkbox"/>		-

In the following image, the **Contrail Storage** icon is selected. You can configure parameters for Contrail Storage, including selecting a package and adding storage disks locations.

The screenshot shows the 'Add Server' window with the 'Contrail Storage' section expanded. It contains a 'Storage Repo ID' dropdown menu with 'Select Repo ID' as the current selection. Below it is a 'Chassis ID' dropdown menu with 'Select Chassis ID' as the current selection, and an 'Add New Chassis ID' button next to it. There is also a 'Storage Disks' section with an '+Add' button. The bottom status bar shows 'scale_mainline_cluster', '10.84.11.4', and 'Provision Completed'.

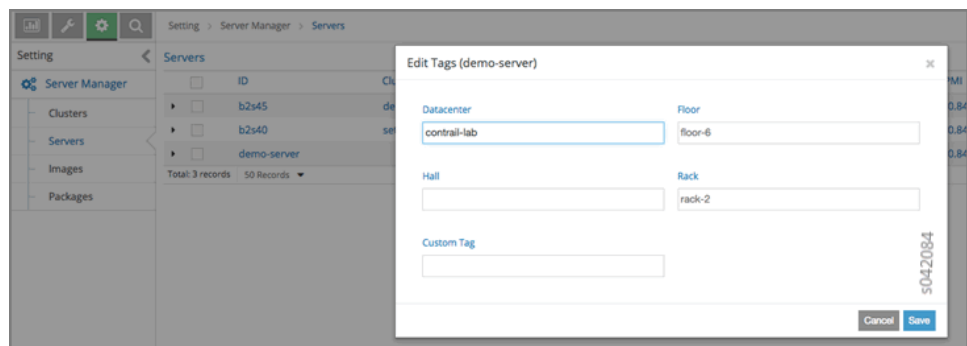
When you are finished entering new server details in the **Add Server** window, click **Save** to add the new server configuration to the list of servers.

You can change details of the new server by clicking the gear wheel icon to the right side to get a list of actions available, including **Edit Config**, **Edit Tags**, **Reimage**, **Provision**, and **Delete**, as shown in the following.



Edit Tags for Servers

Select **Edit Tags** from the gear wheel icon menu. The **Edit Tags** window is displayed. Enter any user-defined tags to be associated with the selected server, then click **Save** to add the tags to the server configuration.

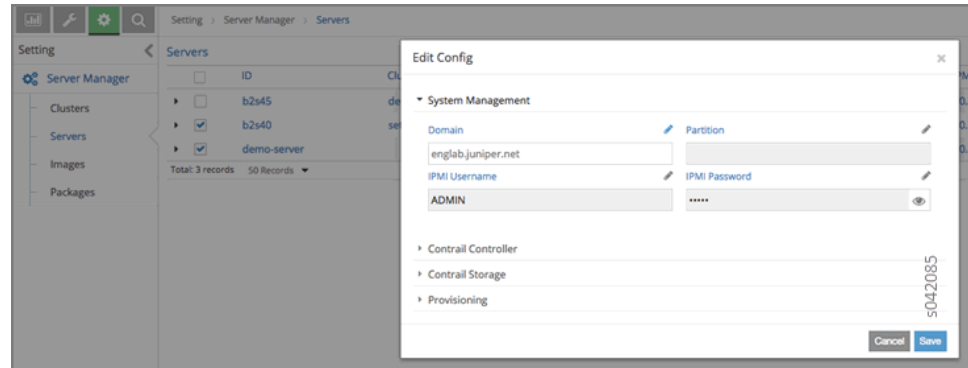


Using the Edit Config Option for Multiple Servers

You can also edit the configuration of multiple servers at one time. From the **Servers** window at **Setting > Server Manager > Servers**, select the servers you want to edit, then click a gear wheel icon at the right to open the action menu, and select **Edit Config**.

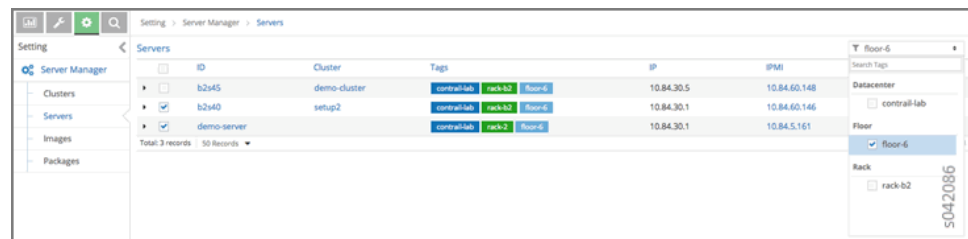
The **Edit Config** window is displayed, as shown.

Click a pencil icon to open configuration fields that can be edited. Fields include **System Management**, **Contrail Controller**, **Contrail Storage**, and so on.



Filter Servers by Tag

You can filter servers according to the tags defined for them. In the **Servers** window, click the **Filter Tags** field in the upper right heading. A list of configured tags is displayed. Select a tag by which to filter the list of servers.



Viewing Server Details

Each server name on the **Servers** page is a link to the details page for that server. Click any server name to open the details for that server, as shown in the following image.

For each server, the **Sensors Information** area shows the **Name**, **Type**, **Reading**, and **Status** for the **temperature** (degrees Celsius), **fan** (rpm), and **power** (watts) sensor types.

The screenshot displays the Juniper Server Manager interface. The left sidebar shows the navigation menu with 'Servers' selected. The main content area is divided into three sections: 'CPU/Memory Information', 'Chassis State', and 'Sensors'. The 'Sensors' section is expanded, showing a table of sensor data.

Name	Type	Reading	Status
CPU1 Temp	temperature	40 C	ok
CPU2 Temp	temperature	39 C	ok
PCH Temp	temperature	41 C	ok
System Temp	temperature	27 C	ok
Peripheral Temp	temperature	38 C	ok

Below the sensors table, there are sections for 'Interface Monitoring' and 'File System'. The 'Interface Monitoring' section shows a table of network interfaces with columns for Name, TX Bytes, TX Packets, RX Bytes, and RX Packets. The 'File System' section shows a table of file systems with columns for Name, Type, Size, and Used.

Configuring Images and Packages

Use the sidebar **Images and Packages** options to configure the software images and packages to be used by the Server Manager. Images are typically used to reimage clusters with an operating system version. Packages are used to provision clusters with a Contrail setup.

Both areas of the Server Manager user interface operate in a similar fashion. The figure shows the **Images** section. The **Packages** section has similar options.

Select **Images**. The Images page is displayed, as shown.

The screenshot displays the Juniper Server Manager interface with the 'OS Images' section selected in the sidebar. The main content area shows a table of OS images with columns for ID, Category, Type, Version, and Path. A table below shows details for the selected image 'newubuntu'.

ID	Category	Type	Version	Path
newubuntu	image	ubuntu	14.04.4	/root/ubuntu-14.04.4-server-amd64.iso

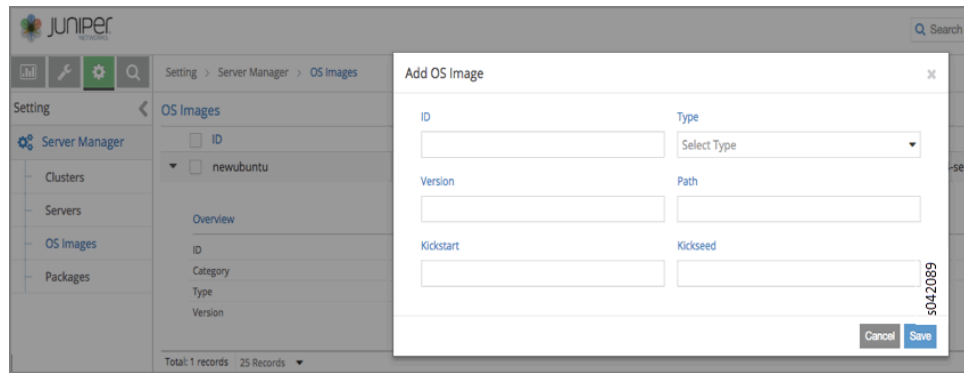
Overview		Details	
ID	newubuntu	Path	/root/ubuntu-14.04.4-server-amd64.iso
Category	image	Kickstart	/usr/share/contrail/images/newubuntu.iso
Type	ubuntu	Kickseed	/usr/share/contrail/images/newubuntu.seed
Version	14.04.4		

Add New Image or Package

To add a new image or package, on the respective **Images** or **Packages** page, click the plus (+) icon in the upper right header. The **Add Image** window is displayed. Enter the information for the new image (or package) and click **Save** to add the new item to the list of configured items.



NOTE: The path field requires the path of the image where it is located on the server upon which the server-manager process is running.



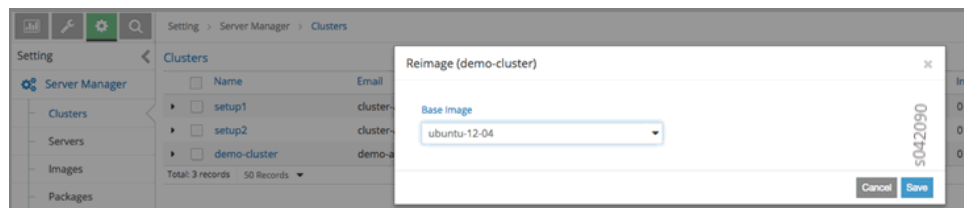
Selecting Server Manager Actions for Clusters

After all aspects of a cluster are configured, you can select actions for the Server Manager to perform on the cluster, such as **Reimage** or **Provision**.

Reimage a Cluster

Select **Setting > Servers > Clusters**. The **Clusters** window is displayed. Click the right side gear wheel icon of the cluster to be reimaged, then select **Reimage** from the action menu.

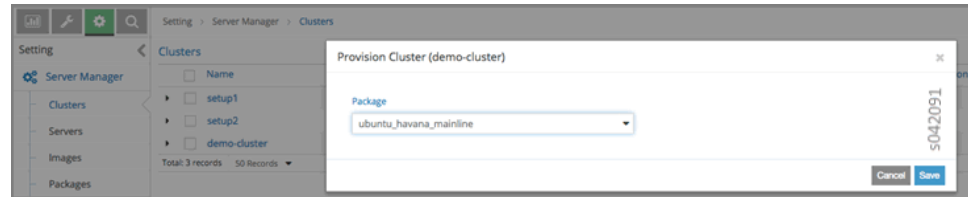
The **Reimage** dialog box is displayed, as shown. Verify that the correct image is selected in the **Default Image** field, then click **Save** to initiate the reimage action.



Provision a Cluster

The process to provision a cluster is similar to the process to reimage a cluster. Select **Setting > Servers > Clusters**. The **Clusters** window is displayed. Click the right side gear wheel icon of the cluster to be provisioned, then select **Provision** from the action menu.

The **Provision Cluster** dialog box is displayed, as shown. Verify that the correct package for provisioning is selected in the **Default Package** field, then click **Save** to initiate the provisioning action.



See Also • [Using Server Manager to Automate Provisioning on page 116](#)

Installing and Using Server Manager Lite

This topic describes how to install and troubleshoot Server Manager Lite.

Server Manager Lite Overview

Server Manager Lite (SM-Lite), is a streamlined version of the Server Manager software that does not include the reimaging function.

SM-Lite supports the Server Manager functions of provisioning, monitoring, inventory, and webui. SM-Lite is intended to replace fab command provisioning. It allows easy deployment of Contrail provisioning and enables developers to work in isolated environments for Contrail provisioning.

SM-Lite eliminates installation and configuration of DHCP, DNS, and Cobbler services. Additionally, SM-Lite installation setup scripts are enhanced to reduce installation time.

SM-Lite provides a single command to install SM-Lite and provision a Contrail cluster.

SM-Lite introduces additional capabilities into Server Manager. The SM-Lite package is part of the Contrail Server Manager installer Debian package (**contrail-server-manager-installer_<version string>.deb**).

SM-Lite works with or without having a separate node for the SM-Lite installation, it can be installed on any Contrail node, but it is recommended to install it on the config node.

SM-Lite preserves the existing Server Manager webui functionality and it can be run on the same node as the Contrail webui. Because of that, the default port for the Server Manager webui has been changed to port **9080**.

It is important to note that the code base used for SM-Lite and Server Manager is common. Therefore, any changes or enhancements made to Server Manager provisioning functionality are automatically available in the SM-Lite software.

Installing Server Manager Lite

The SM-Lite package is included as part of the Server Manager installer package.

The installer package also has other packages such as Server Manager, Server Manager client, Server Manager webui, and Server Manager inventory. Before provisioning commands can be executed using SM-Lite, you need to install the Server Manager installer package.

Use the following command to install the Server Manager installer package.

```
dpkg -i <contrail-server-manager-installer-deb>
```

After the Server Manager installer package is installed, all necessary Server Manager packages, scripts, and so on are made available on the server where it is installed. You can then start using Server Manager Lite commands.

Provisioning Using SM-Lite

To provisioning the target systems, use the **provision.sh** script.

The full syntax and available options of the **provision.sh** script is:

```
/opt/contrail/contrail_server_manager/provision.sh --testbed <testbed>  
--contrail-package <contrail-package> --cluster-id <cluster-id>
```

When you include the **help** option the system displays the following:

```
/opt/contrail/contrail_server_manager/provision.sh --testbed <testbed>  
--contrail-package <contrail-package> --cluster-id <cluster-id>  
-h --help  
-c | --contrail-package <pkg>  
-t | --testbed <testbed.py>  
-cid | --cluster-id <cluster-id>
```

The **provision.sh** script performs the following functions:

1. Installs SM-Lite.

Uses the **setup.sh** installation script with the **-smlite** option to install the SM-Lite package. (**contrail-server-manager-lite_<version-sku>_all.deb**) and all other needed packages on the system.

2. Prepares the cluster for Contrail provisioning.

Translates the parameters in the **testbed.py** file into Server Manager objects and stores them in the Server Manager database. This specifies the servers in the cluster and the configuration parameters. The cluster-id value is used, if it is specified.

3. Performs a pre-check on the target systems to ensure that they are ready for running provisioning via Puppet. Uses the Puppet modules and manifests from the Contrail package to provision the Contrail cluster.

This step issues provisioning commands for the cluster with the given Contrail package.

Server Manager Lite can be installed on any node. We recommend that you install it on the config node. Server Manager Lite can be installed on a separate node other than the Contrail cluster nodes.

With the introduction of Server Manager Lite in Contrail Release 3.0, the Server Manager webui default port is changed to **9080**. You can change the port by editing the `/etc/contrail/config.global.sm.js` file, and then restarting the `supervisor-webui-sm` process.



NOTE: It is also still possible to use the `provision.sh` script to install the full Server Manager (not SM-Lite) software to provision a target cluster.

Displaying the Cluster Status

The `server-manager show cluster -detail` command displays the provisioning status of a cluster by role and by role progress.

Use the `server-manager status server` command to display the current status of the servers.

Displaying the SM-Lite Installation and Provisioning Log Files

Log files that provide information during installation and use of SM-Lite software are available at:

- `/var/log/contrail/install_logs/install_<timestamp>.log` (SM-Lite install)
- `/var/log/contrail/install_logs/provision_<timestamp>.log` (provisioning command logs)
- `testbed_parser.log` and `preconfig.log`

Contrail Provisioning Log Files

For each Puppet run, log files are automatically uploaded to the Server Manager at the following locations:

- `http:<sm-lite-ip-address>/logs`
- `/var/log/contrail_server_manager/<target>/<timestamp>.log`
- `/var/log/contrail/*`

You can also display the status of the processes and services using the `contrail-status` command.

Related Documentation

- [Using Server Manager to Automate Provisioning on page 116](#)
- [Using the Server Manager Web User Interface on page 151](#)
- [Installing Server Manager on page 111](#)

CHAPTER 7

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

- [Using ToR Switches and OVSDb to Extend the Contrail Cluster to Other Instances on page 169](#)
- [Configuring High Availability for the Contrail OVSDb ToR Agent on page 179](#)
- [Using Device Manager to Manage Physical Routers on page 184](#)
- [REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces on page 210](#)

Using ToR Switches and OVSDb to Extend the Contrail Cluster to Other Instances

- [Support for ToR Switch and OVSDb Overview on page 169](#)
- [ToR Services Node \(TSN\) on page 170](#)
- [Contrail ToR Agent on page 170](#)
- [Using the Web Interface to Configure ToR Switch and Interfaces on page 172](#)
- [Provisioning with Fab Commands on page 174](#)
- [Prerequisite Configuration for QFX5100 Series Switch on page 176](#)
- [Changes to Agent Configuration File on page 178](#)
- [REST APIs on page 179](#)

Support for ToR Switch and OVSDb Overview

Contrail Release 2.1 and later supports extending a cluster to include bare metal servers and other virtual instances connected to a top-of-rack (ToR) switch that supports the Open vSwitch Database Management (OVSDb) Protocol. The bare metal servers and other virtual instances can belong to any of the virtual networks configured in the Contrail cluster, facilitating communication with the virtual instances running in the cluster. Contrail policy configurations can be used to control this communication.

The OVSDb protocol is used to configure the ToR switch and to import dynamically-learned addresses. VXLAN encapsulation is used in the data plane communication with the ToR switch.

ToR Services Node (TSN)

A ToR services node (TSN) can be provisioned as a role in the Contrail system. The TSN acts as the multicast controller for the ToR switches. The TSN also provides DHCP and DNS services to the bare metal servers or virtual instances running behind ToR switch ports.

The TSN receives all the broadcast packets from the ToR switch, and replicates them to the required compute nodes in the cluster and to other EVPN nodes. Broadcast packets from the virtual machines in the cluster are sent directly from the respective compute nodes to the ToR switch.

The TSN can also act as the DHCP server for the bare metal servers or virtual instances, leasing IP addresses to them, along with other DHCP options configured in the system. The TSN also provides a DNS service for the bare metal servers. Multiple TSN nodes can be configured in the system based on the scaling needs of the cluster.

Contrail ToR Agent

A ToR agent provisioned in the Contrail cluster acts as the OVSDB client for the ToR switch, and all of the OVSDB interactions with the ToR switch are performed by using the ToR agent. The ToR agent programs the different OVSDB tables onto the ToR switch and receives the local unicast table entries from the ToR switch.

The ToR agent receives the configuration information for the ToR switch. The ToR agent translates the Contrail configuration to OVSDB and populates the relevant OVSDB table entries in the ToR switch.

The typical practice is to run the ToR agent on the TSN node.

Configuration Model

[Figure 20 on page 171](#) depicts the configuration model used in the system.

Figure 20: Configuration Model

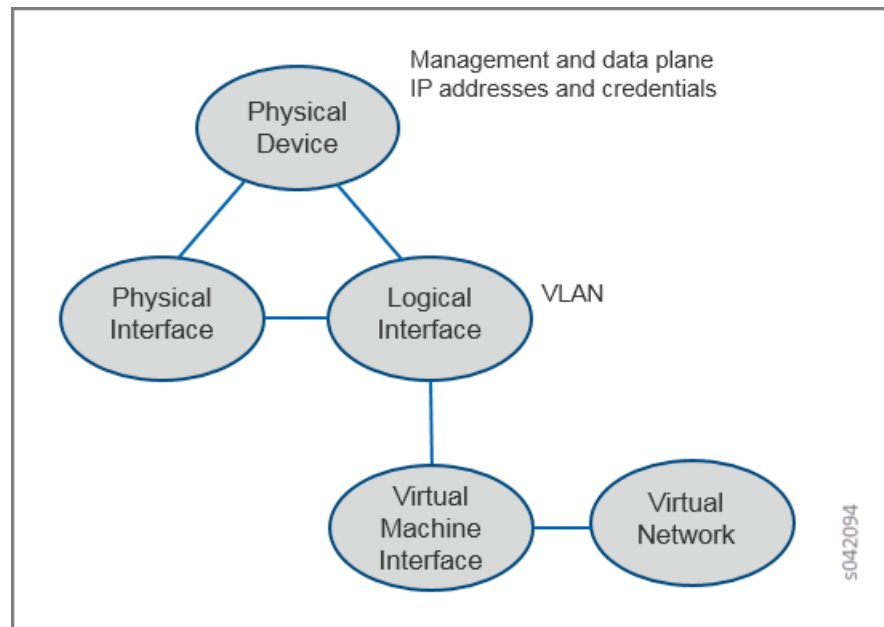


Table 12 on page 171 maps the Contrail configuration objects to the OVSDb tables.

Table 12: Contrail Objects in the OVSDb

Contrail Object	OVSDb Table
Physical device	Physical switch
Physical interface	Physical port
Logical interface	<VLAN physical port> binding to logical switch
Virtual networks	Logical switch
Layer 2 unicast route table	Unicast remote and local table
	Multicast remote table
	Multicast local table
	Physical locator table
	Physical locator set table

Control Plane

The ToR agent receives the EVPN route entries for the virtual networks in which the ToR switch ports are members, and adds the entries to the unicast remote table in the OVSDb.

MAC addresses learned in the ToR switch for different logical switches (entries from the local table in OVSDb) are propagated to the ToR agent. The ToR agent exports the addresses to the control node in the corresponding EVPN tables, which are further distributed to other controllers and subsequently to compute nodes and other EVPN nodes in the cluster.

The TSN node receives the replication tree for each virtual network from the control node. It adds the required ToR addresses to the received replication tree, forming its complete replication tree. The other compute nodes receive the replication tree from the control node, whose tree includes the TSN node.

Data Plane

The data plane encapsulation method is VXLAN. The virtual tunnel endpoint (VTEP) for the bare metal end is on the ToR switch.

Unicast traffic from bare metal servers is VXLAN-encapsulated by the ToR switch and forwarded, if the destination MAC address is known within the virtual switch.

Unicast traffic from the virtual instances in the Contrail cluster is forwarded to the ToR switch, where VXLAN is terminated and the packet is forwarded to the bare metal server.

Broadcast traffic from bare metal servers is received by the TSN node. The TSN node uses the replication tree to flood the broadcast packets in the virtual network.

Broadcast traffic from the virtual instances in the Contrail cluster is sent to the TSN node, which replicates the packets to the ToR switches.

Using the Web Interface to Configure ToR Switch and Interfaces

The Contrail Web user interface can be used to configure a ToR switch and the interfaces on the switch. To add a switch, select **Configure > Physical Devices > Physical Routers**.

The Physical Routers list is displayed.

Click the + symbol to open the Add menu. From the Add menu you can select one of the following:

- Add OVSDb Managed ToR
- Add Netconf Managed Physical Router
- CPE Router
- Physical Router

To add a physical router select **Physical Router**. The **Add Physical Router** window is displayed as shown.

Figure 21: Add Physical Router Window

Create

OVSDDB Managed ToR Permissions

Name

Vendor

Model

Management IP

VTEP Address

TOR Agent(s)

Select or Enter TOR Agent Name

Select or Enter TOR Agent Name

TSN(s)

Select or Enter TSN Name

Select or Enter TSN Name

SNMP Monitored

Cancel Save

s042093

Enter the ToR switch's IP address and VTEP address. Also configure the TSN and ToR agent addresses for the ToR.

To add the logical interfaces to be configured on the ToR switch, select **Configure > Physical Devices > Interfaces**.

The **Physical Routers** list is displayed. Click the + symbol. The **Add Interface** window is displayed as shown.

Figure 22: Add Interface Window

Enter the name of the logical interface. The name must match the name on the ToR, for example, ge-0/0/0.10. Also enter other logical interface configuration parameters, such as VLAN ID, MAC address, and IP address of the bare metal server and the virtual network to which it belongs.

Provisioning with Fab Commands

The TSN can be provisioned using fab commands.

To provision with fab commands, edit the `testbed.py` file.

1. In `env.roledefs`, add hosts for the roles `tsn` and `toragent`.
2. Configure the TSN node in the compute role.
3. Use the following example to configure the ToR agent. The ToR agent node should also be configured into the compute node.

```
env.tor_agent = {
    host2: [{
        'tor_ip': '<ip address>',          # IP address of the TOR
        'tor_id': '<1>',                    # Numeric value to uniquely identify
        TOR
        'tor_type': 'ovs',                  # always ovs
        'tor_ovs_port': '9999',             # the TCP port to connect on the
        TOR
        'tor_ovs_protocol': 'tcp',          # always tcp, for now
        'tor_tsn_ip': '<ip address>'        # IP address of the TSN for this
        TOR
```

```

        'tor_tsn_name': '<name>',          # Name of the TSN node
        'tor_name': '<switch name>',      # Name of the TOR switch
        'tor_tunnel_ip': '<ip address>',  # IP address of Data tunnel endpoint

        'tor_vendor_name': '<name>',      # Vendor name for TOR switch
        'tor_http_server_port': '<port>', # HTTP port for TOR Introspect
    }
}
}

```

4. Two ToR agents provisioned on different hosts are considered redundant to each other if the **tor_name** and **tor_ovs_port** in their respective configurations are the same, which means the ToR agents are listening on the same port for SSL connections on both the nodes.

Use the tasks **add_tsn** and **add_tor_agent** values to provision the TSN and ToR agents.

To configure an existing compute node as a TSN or a ToR Agent, use the following fab tasks:

Command	Description
add_tsn	Provision all the TSNs configured in the testbed.py .
add_tor_agent	Add all the tor-agents configured in the testbed.py .
add_tor_agent_node	Add all tor-agents in specified node (e.g., fab add_tor_agent_node:root@<ip>)
add_tor_agent_by_id	Add the specified tor-agent, identified by tor_agent_id (e.g., fab add_tor_agent_by_id:1,root@<ip>).
add_tor_agent_by_index	Add the specified tor-agent, identified by index/position in testbed.py (e.g., fab add_tor_agent_by_index:0,root@<ip>).
add_tor_agent_by_index_range	Add a group of tor-agents, identified by indices in the testbed.py (e.g., fab add_tor_agent_by_index_range:0-2,root@<ip>).
delete_tor_agent	Remove all tor-agents in all nodes.
delete_tor_agent_node	Remove all tor-agents in specified node (e.g., fab delete_tor_agent_node:root@<ip>).
delete_tor_agent_by_id	Remove the specified tor-agent, identified by tor-id (e.g., fab delete_tor_agent_by_id:2,root@<ip>).
delete_tor_agent_by_index	Remove the specified tor-agent, identified by index/position in testbed.py. (e.g., fab delete_tor_agent_by_index:0,root@<ip>).
delete_tor_agent_by_index_range	Remove a group of tor-agents, identified by indices in testbed.py. (e.g., fab delete_tor_agent_by_index_range:0-2,root@<ip>).
setup_haproxy_config	Provision HA Proxy.

To configure an existing compute node as a TSN or a ToR agent, use the following fab tasks:

```
fab add_tsn_node:True,user@<ip>
```

```
fab add_tor_agent_node:True,user@<ip>
```



NOTE: Use `fab setup_all` to provision appropriately when run with the updated `testbed.py` file.

5. Configure the vRouter limits on the TSN node to match the scaling requirements in the setup.

The following example shows an update in the `testbed.py` made before the setup, so that appropriate vrouter options are configured by the `fab` task.

```
env.vrouter_module_params = {
    host4: {'mpls_labels': '196000', 'nexthops': '521000', 'vrfs': '65536',
    'macs': '1000000'},
    host5: {'mpls_labels': '196000', 'nexthops': '521000', 'vrfs': '65536',
    'macs': '1000000'}
}
```

In the example, the following applies:

- `mpls_labels` = (maximum number of VNs * 3) + 4000
- `nexthops` = (maximum number of VNs * 4) + number of ToRs + number of compute nodes + 100
- `vrfs` = maximum number of VNs
- `macs` = maximum number of MACs in a VN

On a TSN node or on a compute node, the currently configured limits can be displayed using `vrouter --info`.

If the configured limits need to be changed, update by editing the options in the `/etc/modprobe.d/vrouter.conf` file with the desired values and restarting the node. An example of an updated options statement is the following:

```
options vrouter vr_mpls_labels=196000 vr_nexthops=521000 vr_vrfs=65536
vr_bridge_entries=1000000
```

Prerequisite Configuration for QFX5100 Series Switch

When using the Juniper Networks QFX5100 Series switches, ensure the following configurations are made on the switch before extending the Contrail cluster.

1. Enable OVSDDB.
2. Set the connection protocol.
3. Identify the interfaces that are managed by means of OVSDDB.

4. Configure the controller (in case pssl is used). If HA Proxy is used, use the address of the HA Proxy node and use the VIP when VRRP is used between multiple nodes running HA Proxy. The following is an example:

```
set interfaces lo0 unit 0 family inet address  
  
set switch-options ovssdb-managed  
  
set switch-options vtep-source-interface lo0.0  
  
set protocols ovssdb interfaces  
  
set protocols ovssdb passive-connection protocol tcp port  
  
set protocols ovssdb controller <tor-agent-ip> inactivity-probe-duration 10000  
protocol ssl port <tor-agent-port>
```

5. When using SSL to connect, CA-signed certificates must be copied to the `/var/db/certs` directory in the QFX device. The following example shows one way to get the certificates. The following commands could be run on any server.

```
apt-get install openvswitch-common  
ovs-pki init  
ovs-pki req+sign vtep  
scp vtep-cert.pem root@<qfx>:/var/db/certs  
scp vtep-privkey.pem root@<qfx>:/var/db/certs  
cacert.pem file will be available in /var/lib/openvswitch/pki/switchca, when  
the above are done. This is the file to be provided in the above testbed (in  
env.ca_cert_file).
```

Debug QFX5100 Configuration

You can use the following commands on the QFX switch to show the OVSSDB configuration.

```
show ovssdb logical-switch  
  
show ovssdb interface  
  
show ovssdb mac  
  
show ovssdb controller  
  
show vlans
```

You can use the agent introspect on the ToR agent and the TSN nodes to show the configuration and operational state of these modules.

- The TSN module is like any other **contrail-vrouter-agent** on a compute node, with introspect access available on port 8085 by default. Use the introspect on port 8085 to view operational data such as interfaces, virtual network, and VRF information, along with their routes.

- The port on which the ToR agent introspect access is available is in the configuration file provided to the contrail-tor-agent. This provides the OVSDB data available through the client interface, apart from the other data available in a Contrail Agent.

Changes to Agent Configuration File

You can make changes to the agent features by making changes in the configuration file.

In the `/etc/contrail/contrail-vrouter-agent.conf` file for TSN, the `agent_mode` option is available in the `DEBUG` section to configure the agent to be in TSN mode.

agent_mode = tsn

The following are typical configuration items in a ToR agent configuration file.

[DEFAULT]

`agent_name = noded2-1 # Name (formed with hostname and TOR id from below)`

`agent_mode = tor # Agent mode`

`http_server_port=9010 # Port on which Introspect access is available`

[DISCOVERY]

`server=<ip> # IP address of discovery server`

[TOR]

`tor_ip=<ip> # IP address of the TOR to manage`

`tor_id=1 # Identifier for ToR Agent.`

`tor_type=ovs # ToR management scheme - only "ovs" is supported`

`tor_ovs_protocol=tcp # IP-Transport protocol used to connect to TOR, can be tcp or pssl`

`tor_ovs_port=port # OVS server port number on the ToR`

`tsn_ip=<ip> # IP address of the TSN`

`tor_keepalive_interval=10000 # keepalive timer in ms`

`ssl_cert=/etc/contrail/ssl/certs/tor.1.cert.pem # path to SSL certificate on TOR Agent, needed for pssl`

`ssl_privkey=/etc/contrail/ssl/private/tor.1.privkey.pem # path to SSL private key on TOR Agent, needed for pssl`

`ssl_cacert=/etc/contrail/ssl/certs/cacert.pem # path to SSL CA cert on the node, needed for pssl`

REST APIs

For information regarding REST APIs for physical routers and physical and logical interfaces, see [“REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces” on page 210](#).

Related Documentation

- [REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces on page 210](#)
- [Using Device Manager to Manage Physical Routers on page 184](#)
- [Configuring High Availability for the Contrail OVSDb ToR Agent on page 179](#)

Configuring High Availability for the Contrail OVSDb ToR Agent

This topic describes how high availability can be configured for the Contrail ToR agent.

- [Overview: High Availability for a ToR Switch on page 179](#)
- [High Availability Solution for Contrail ToR Agent on page 179](#)
- [Failover Methodology Description on page 181](#)
- [Failure Scenarios on page 181](#)
- [Redundancy for HAProxy on page 183](#)
- [Configuration for ToR Agent High Availability on page 183](#)
- [Testbed.py and Provisioning for High Availability on page 184](#)

Overview: High Availability for a ToR Switch

In Contrail Release 2.20 and later, high availability can be configured for the Contrail ToR agent.

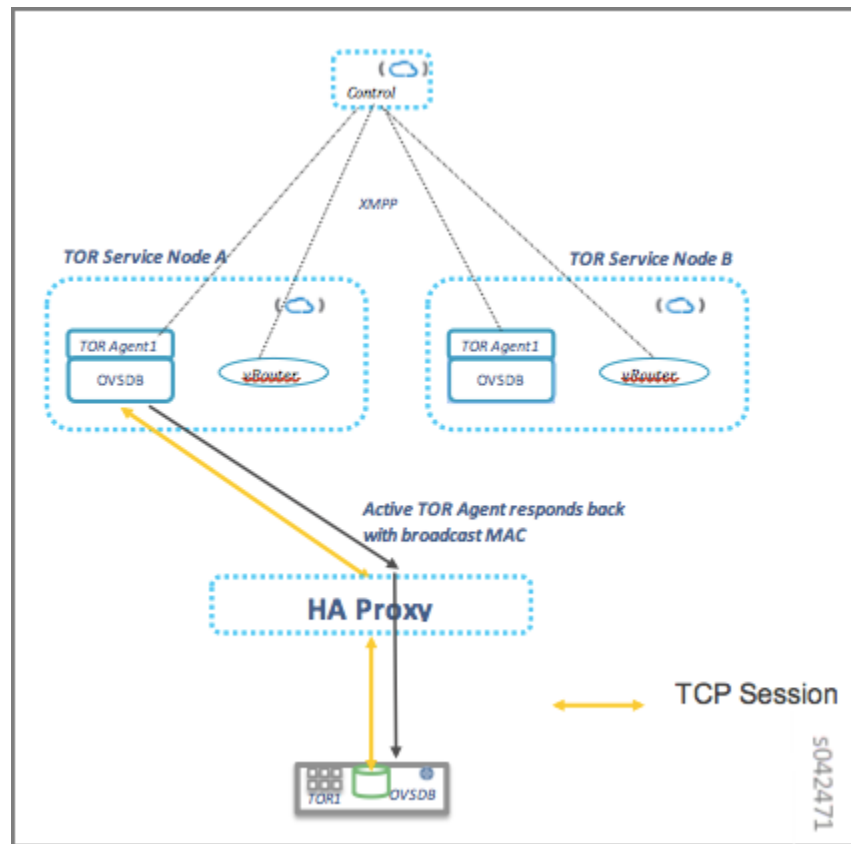
When a top-of-rack (ToR) switch is managed through the Open vSwitch Database (OVSDb) protocol by using a ToR agent on Contrail, a high availability configuration is necessary to maintain ToR agent redundancy. With ToR agent redundancy, if the ToR agent responsible for a ToR switch is unable to act as 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 23: 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 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.

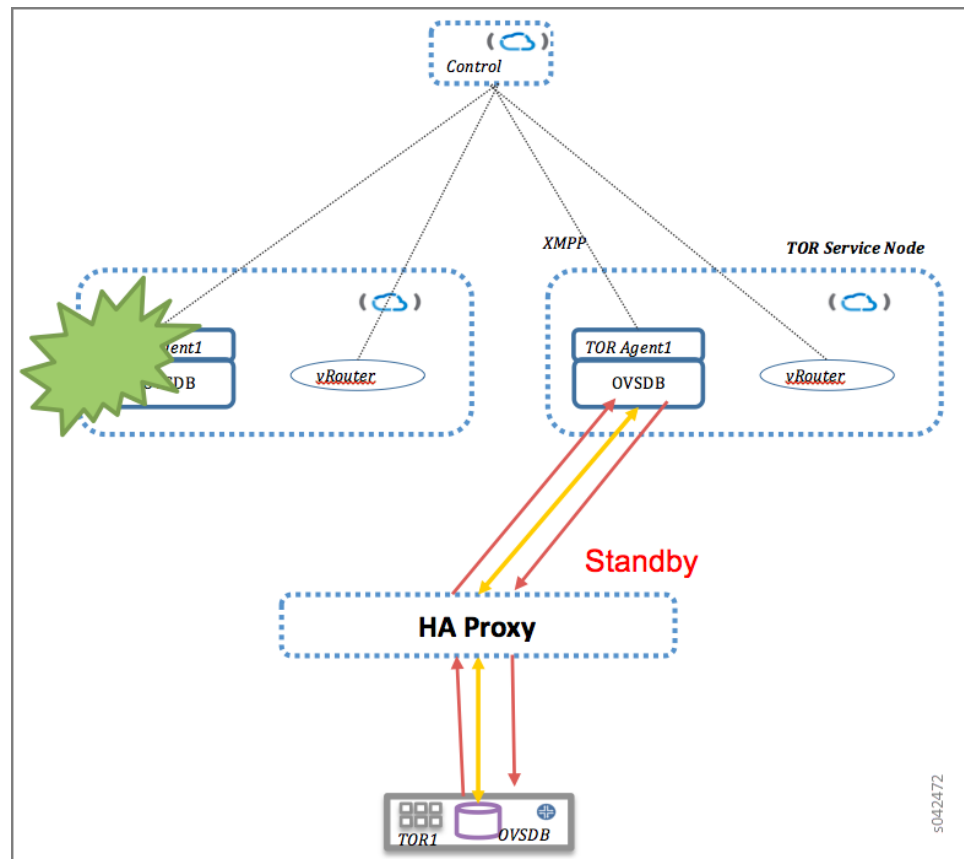
Failure Scenarios

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 24: Failure Scenarios



When a connection is established to the other ToR agent, the new ToR agent does the following:

- Updates the multicast route in OVSDB to point to the new TSN.
- Gets all of the OVSDB entries.
- Audits the data with the configurations available.
- Updates the database.
- Exports entries from the OVSDB local table to the control node.

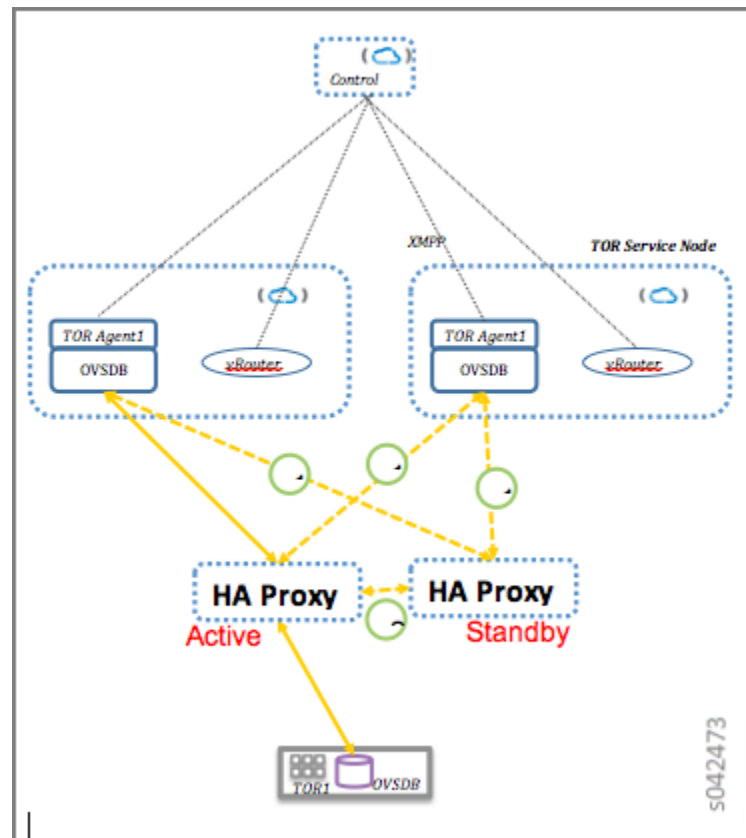
Because the configuration and routes from the control node are already synchronized to the new ToR Services Node (TSN), the new TSN can immediately act on the broadcast traffic from the ToR switch. Any impact to the service is only for the time needed for the SSL connection to be set up and for programming the multicast and unicast routes in the OVSDB.

When the SSL connection goes down, the ToR agent retracts the routes exported. Also, if the Extensible Messaging and Presence Protocol (XMPP) connection between the ToR agent and the control node goes down, the control node removes the routes exported by the ToR agent. In these scenarios, the entries from the OVSDB local table are retracted and then added back from the new ToR agent.

Redundancy for HAProxy

In a high availability configuration, multiple HAProxy nodes are configured, with Virtual Router Redundancy Protocol (VRRP) running between them. The ToR agents are configured to use the virtual IP address of the HAProxy nodes to make the SSL connection to the controller. The active TCP connections go to the virtual IP master node, which proxies them to the chosen ToR agent. A ToR agent is chosen based on the number of connections from the HA Proxy to that node (the node with lower number of connections gets the new connection) and can be controlled through configuration of the HAProxy.

Figure 25: Redundancy for HAProxy



If the HAProxy node fails, a standby node becomes the virtual IP master and sets up the connections to the ToR agents. The SSL connections are reestablished, following the same methods discussed earlier.

Configuration for ToR Agent High Availability

To get the required configuration downloaded from the control node to the TSN agent and to the ToR agent, the physical router node must be linked to the virtual router nodes that represent the two ToR agents and the two TSNs.

In the Contrail Web user interface select **Configure > Physical Devices > Physical Routers**. In the **Physical Routers** window, click the + icon. The **Add OVSDb Managed ToR** window is displayed. See [Figure 26 on page 184](#).

Figure 26: Add OVSDb Managed ToR Window

The screenshot shows the 'Add OVSDb Managed ToR' window. It has a title bar with a close button. The form includes 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 placeholder text 'Select or Enter Name'.
- TSN(s):** Two dropdown menus, each with the placeholder text 'Select or Enter Name'.
- SNMP Monitored:** A checkbox that is currently checked.
- SNMP Settings:** A section header with a dropdown arrow.
- Version:** Two radio buttons labeled '2' and '3', with '2' selected.
- Buttons:** 'Cancel' and 'Save' buttons at the bottom right.

Enter a name to create an entry for the ToR switch, enter the ToR switch management IP address, and enter the virtual tunnel endpoint (VTEP) address. The router name should match the hostname of the ToR switch. Both ToR agents and their respective TSN nodes can be configured here.

Testbed.py and Provisioning for High Availability

The same testbed.py configuration used for provisioning the TSN and ToR agents is used to provision high availability. To be considered a pair, the redundant ToR agents must have the same **tor_name** and **tor_ovs_port** in their respective stanzas.

- Related Documentation**
- [Using ToR Switches and OVSDb to Extend the Contrail Cluster to Other Instances on page 169](#)

Using Device Manager to Manage Physical Routers

- [Support for Physical Routers Overview on page 185](#)
- [Configuration Model on page 185](#)
- [Alternate Ways to Configure a Physical Router on page 188](#)
- [Device Manager Configurations on page 188](#)

- [Prerequisite Configuration Required on MX Series Device on page 189](#)
- [Configuration Scenarios on page 189](#)
- [Device Manager Functionality on page 189](#)
- [Dynamic Tunnels on page 190](#)
- [Extending the Public Network on page 196](#)
- [Ethernet VPN Configuration on page 197](#)
- [Floating IP Addresses and Source Network Address Translation for Guest Virtual Machines and Bare Metal Servers on page 198](#)
- [Samples of Generated Configurations for an MX Series Device on page 206](#)

Support for Physical Routers Overview

A configuration node daemon named Device Manager can be used to manage physical routers in the Contrail system.

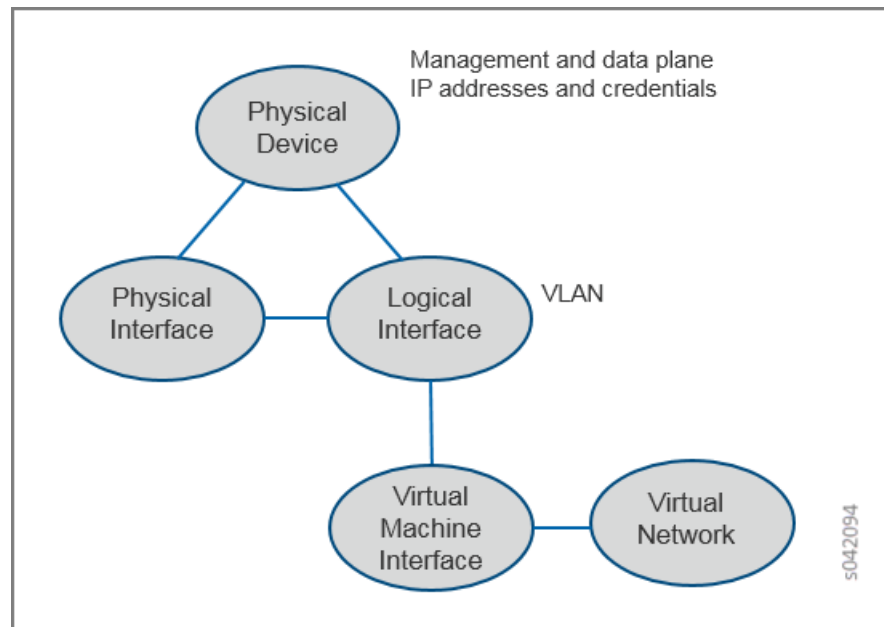
The Device Manager daemon listens to configuration events from the API server, creates any necessary configurations for all physical routers it is managing, and programs those physical routers.

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 27 on page 186](#) depicts the configuration model used in the system. The Physical Router, Physical Interface, and Logical Interface all represent physical router entities.

Figure 27: Contrail Configuration Model

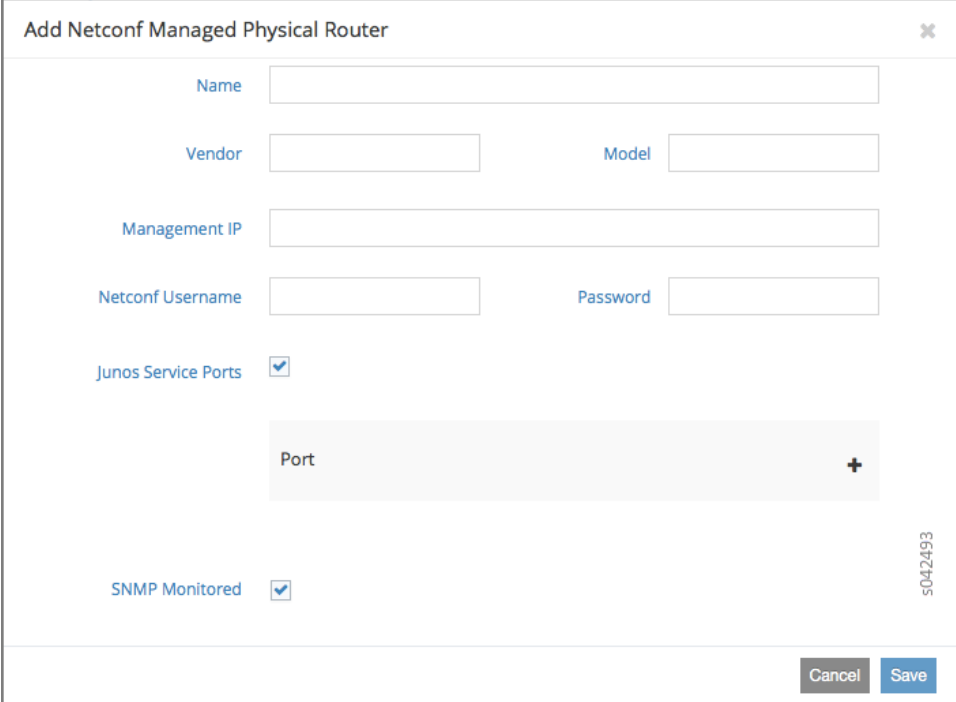


Configuring a Physical Router

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 28: Add Physical Router Window



The image shows a web-based configuration window titled "Add Netconf Managed Physical Router". The window contains several input fields and checkboxes. The fields are: "Name", "Vendor", "Model", "Management IP", "Netconf Username", and "Password". There are checkboxes for "Junos Service Ports" and "SNMP Monitored". Below the "Junos Service Ports" checkbox is a section labeled "Port" with a plus sign icon. At the bottom right, there are "Cancel" and "Save" buttons. A small number "5042493" is visible on the right side of the window.

Name			
Vendor		Model	
Management IP			
Netconf Username		Password	
Junos Service Ports	<input checked="" type="checkbox"/>		
Port +			
SNMP Monitored	<input checked="" type="checkbox"/>		

5042493

Cancel Save

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 29: Add Interface Window

Alternate Ways to Configure a Physical Router

You can also configure a physical router by using a Contrail REST API, see [“REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces”](#) on page 210.

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 210.

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/utlils/provision_physical_router.py

Device Manager Functionality

Device Manager auto configures physical routers when it detects associations in the Contrail database.

The following naming conventions are used for generating MX Series router configurations:

- Device Manager generated configuration group name: `__contrail__`
- BGP groups:
 - Internal group name: `__contrail__`
 - External group name: `__contrail_external`
- VRF name: `_contrai_{l2|l3}_{vn-id}_{vn-name}`
- NAT VRF name: `_contrai_{l2|l3}_{vn-id}_{vn-name}-nat`

- Import policy: **[vrf-name]—import**, Export policy: **[vrf-name]—export**
- Service set: **sv-[vrf-name]**
- NAT rules, SNAT: **sv-[vrf-name]-sn-rule**, DNAT: **sv-[vrf-name]-dn-rule**
- SNAT term name: **term_[private_ip]**, DNAT term name: **term_[public_ip]**
- Firewall filters:
 - Public VRF filter: **redirect_to_public_vrf_filter**
 - Private VRF filter: **redirect_to_[vrf_name]_vrf**
- Logical interface unit numbers:
 - Service ports: **2*vn_id -1, 2*vn_id**
 - IRB interface: **vn_id**

Dynamic Tunnels

Dynamic tunnel configuration in Contrail allows you to configure GRE tunnels on the Contrail Web user interface. When Contrail detects this configuration, the Device Manager module constructs GRE tunnel configuration and pushes it to the MX Series router. A property named **ip-fabric-subnets** is used in the global system configuration of the Contrail schema. Each IP fabric subnet and BGP router is configured as a dynamic tunnel destination point in the MX Series router. The physical router data plane IP address is considered the source address for the dynamic tunnel. You must configure the data plane IP address for auto configuring dynamic tunnels on a physical router. The IP fabric subnets is a global configuration; all of the subnets are configured on all the physical routers in the cluster that have data plane IP configuration.

The following naming conventions are used in the API configuration:

- Global System Config: **ip-fabric-subnets**
- Physical Router: **data-plane-ip**

Web UI Configuration

[Figure 30 on page 191](#) shows the web user interface used to configure dynamic tunnels.

Figure 30: Edit Global Config Window

In the **Edit Global Config** window, the VTEP address is used for the **data-plane-ip** address.

The following is an example of the MX Series router configuration generated by the Device Manager.

```
root@host# show groups __contrail__ routing-options
router-id 172.16.184.200;
route-distinguisher-id 10.87.140.107;
autonomous-system 64512;
dynamic-tunnels {
  __contrail__ {
    source-address 172.16.184.200;
    gre;
    destination-networks {
      172.16.180.0/24;
      172.16.180.8/32;
    }
  }
}
```

```

172.16.185.200/32;
172.16.184.200/32;
172.16.180.5/32;
172.16.180.7/32;
}
}
}

```

BGP Groups

When Device Manager detects BGP router configuration and its association with a physical router, it configures BGP groups on the physical router.

Figure 31 on page 192 shows the web user interface used to configure BGP groups.

Figure 31: Edit BGP Router Window

The screenshot shows the 'Edit BGP Router' window with the following configuration:

- 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 x, inet-vpn, e-vpn x, inet6-vpn x
- Advanced Options:**
 - Peer(s) Selection:** (dropdown menu)
- Available Peer(s):** (empty list)
- Configured Peer(s):** (empty list)
- Buttons:** Cancel, Save

Figure 32 on page 193 shows the web user interface used to configure the physical router.

Figure 32: Edit Physical Router Window for BGP Groups

Edit Physical Router	
Name	tasman
Vendor	juniper
Model	mx
Management IP	10.87.140.107
VTEP Address	172.16.184.200
BGP Gateway	tasman

The following is an example of the MX Series router configuration generated by the Device Manager.

```

root@host show groups __contrail__ protocols bgp
group __contrail__ {
    type internal;
    multihop;
    local-address 172.16.184.200;
    hold-time 90;
    keep all;
    family inet-vpn {
        unicast;
    }
    family inet6-vpn {
        unicast;
    }
    family evpn {
        signaling;
    }
    family route-target;
    neighbor 172.16.180.8;
    neighbor 172.16.185.200;
    neighbor 172.16.180.5;
    neighbor 172.16.180.7;
}

group __contrail_external__ {
    type external;
    multihop;
    local-address 172.16.184.200;
    hold-time 90;
    keep all;
    family inet-vpn {
        unicast;
    }
    family inet6-vpn {
        unicast;
    }
    family evpn {
        signaling;
    }
    family route-target;
}

```

Extending the Private Network

Device Manager allows you to extend a private network and ports to a physical router. When Device Manager detects a VNC configuration, it pushes Layer 2 (EVPN) and Layer 3 VRF, import and export rules and interface configuration to the physical router.

Figure 33 on page 194 shows the web user interface for configuring the physical router for extending the private network.

Figure 33: Edit Physical Router Window for Extending Private Networks

Edit Physical Router	
Name	tasman
Vendor	juniper
Model	mx
Management IP	10.87.140.107
VTEP Address	172.16
BGP Gateway	tasman
Virtual Networks	<div>vn_private-x1-63 (default-domain:default-project) ×</div> <div>vn_private-x2-17 (default-domain:default-project) ×</div>

The following is an example of the MX Series router configuration generated by the Device Manager.

```
/* L2 VRF */

root@host# show groups __contrail__ routing-instances
_contrail_l2_147_vn_private-x1-63
vtep-source-interface lo0.0;
instance-type virtual-switch;
vrf-import _contrail_l2_147_vn_private-x1-63-import;
vrf-export _contrail_l2_147_vn_private-x1-63-export;
protocols {
    evpn {
        encapsulation vxlan;
        extended-vni-list all;
    }
}
bridge-domains {
    bd-147 {
        vlan-id none;
        routing-interface irb.147;
        vxlan {
            vni 147;
        }
    }
}
```

```

}

/* L3 VRF */
root@host# show groups __contrail__ routing-instances
_contrail_l3_147_vn_private-x1-63
instance-type vrf;
interface irb.147;
vrf-import _contrail_l3_147_vn_private-x1-63-import;
vrf-export _contrail_l3_147_vn_private-x1-63-export;
vrf-table-label;
routing-options {
    static {
        route 1.0.63.0/24 discard;
    }
    auto-export {
        family inet {
            unicast;
        }
    }
}

/* L2 Import policy */

root@host# ...cy-options policy-statement
_contrail_l2_147_vn_private-x1-63-import
term t1 {
    from community target_64512_8000066;
    then accept;
}
then reject;

/* L2 Export Policy */
root@host# ...ail__ policy-options policy-statement
_contrail_l2_147_vn_private-x1-63-export
term t1 {
    then {
        community add target_64512_8000066;
        accept;
    }
}

/* L3 Import Policy */

root@host# ...ail__ policy-options policy-statement
_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 34 on page 196 shows the web user interface for extending the public network.

Figure 34: 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 text input field containing 'DNS Servers' and a plus icon to add more servers.
- VxLAN Identifier:** A text input field 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 input field containing 'bali' with a close icon (x).

A vertical ID 's042478' is visible on the right side of the form.

The following is an example of the MX Series router configuration generated by the Device Manager.

```
/* forwarding options */

root@host show groups __contrail__ forwarding-options
family inet {
    filter {
        input redirect_to_public_vrf_filter;
    }
}
```

```

/* firewall filter configuration */

root@host# show groups __contrail__ firewall family inet filter
redirect_to_public_vrf_filter

term term-_contrail_l3_184_vn_public-x1- {

    from {

        destination-address {

            20.1.0.0/16;

        }

    }

    then {

        routing-instance _contrail_l3_184_vn_public-x1-;

    }

}

term default-term {

    then accept;

}

/* L3 VRF static route 0.0.0.0/0 configuration */

root@host# ...instances _contrail_l3_184_vn_public-x1- routing-options static
route 0.0.0.0/0
next-table inet.0;

```

Ethernet VPN Configuration

For every private network, a Layer 2 Ethernet VPN (EVPN) instance is configured on the MX Series router. If any Layer 2 interfaces are associated with the virtual network, logical interfaces are also created under the bridge domain.

The following is an example of the MX Series router configuration generated by the Device Manager.

```

root@host# show groups __contrail__ routing-instances
_contrail_l2_147_vn_private-x1-63
vtep-source-interface lo0.0;
instance-type virtual-switch;
vrf-import _contrail_l2_147_vn_private-x1-63-import;
vrf-export _contrail_l2_147_vn_private-x1-63-export;
protocols {
    evpn {
        encapsulation vxlan;
        extended-vni-list all;
    }
}
bridge-domains {
    bd-147 {

```

```
        vlan-id none;

        interface ge-1/0/5.0;
        routing-interface irb.147;
        vxlan {
            vni 147;
        }
    }
}
```

Floating IP Addresses and Source Network Address Translation for Guest Virtual Machines and Bare Metal Servers

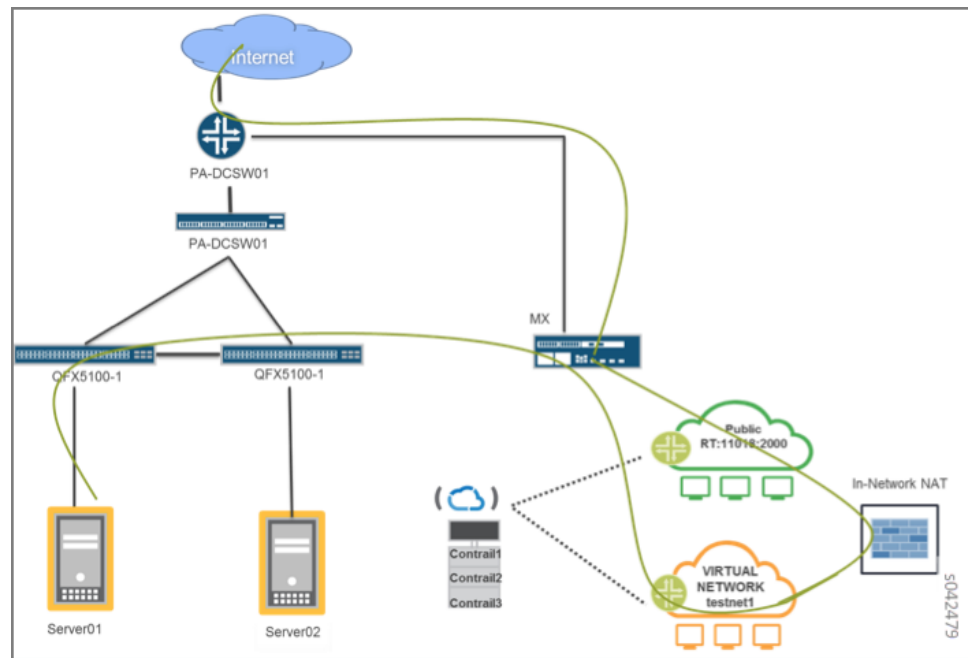
This section describes a bare metal server deployment scenario in which servers are connected to a TOR QFX device inside a private network and an MX Series router is the gateway for the public network connection.

The MX Series router provides the NAT capability that allows traffic from a public network to enter a private network and also allows traffic from the private network to the public network. To do this, you need to configure NAT rules on the MX Series router. The Device Manager is responsible for programming these NAT rules on MX Series routers when it detects that a bare metal server is connected to a public network.

You must configure virtual network computing for the TOR device, the MX Series router, the private network, and the public network, including the address pool. When a logical interface on the TOR device is associated with the virtual machine interface and a floating IP address is assigned to the same virtual machine interface (VMI), Contrail detects this and the Device Manager configures the necessary floating IP NAT rules on each of the MX Series routers associated with the private network.

[Figure 35 on page 199](#) 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 35: 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 35 on page 199](#), 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_13_147_vn_private-x1-63-nat

instance-type vrf;

interface si-2/0/0.293;

vrf-import _contrail_13_147_vn_private-x1-63-nat-import;

vrf-export _contrail_13_147_vn_private-x1-63-nat-export;

vrf-table-label;

routing-options {
    static {
        route 0.0.0.0/0 next-hop si-2/0/0.293;
```



```
    }

    auto-export {

        family inet {

            unicast;

        }

    }

}

/* NAT VRF import policy */

root@host# ...y-statement _contrail_13_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_13_147_vn_private-x1-63-nat-export

term t1 {

    then reject;

}

/* The following additional config is generated for public 13 vrf */

root@host# show groups __contrail__ routing-instances
_contrail_13_184_vn_public-x1-

interface si-2/0/0.294;

routing-options {

    static {

        route 20.1.252.8/32 next-hop si-2/0/0.294;

        route 20.1.252.9/32 next-hop si-2/0/0.294;

    }

}

/* Services set configuration */
```

```
root@host# show groups __contrail__

services {

    service-set sv-_contrail_l3_147_vn_ {

        nat-rules sv-_contrail_l3_147_vn_-sn-rule;

        nat-rules sv-_contrail_l3_147_vn_-dn-rule;

        next-hop-service {

            inside-service-interface si-2/0/0.293;

            outside-service-interface si-2/0/0.294;

        }

    }

}

/* Source Nat Rules*/

root@host# show groups __contrail__ services nat rule
sv-_contrail_l3_147_vn_-sn-rule

match-direction input;

term term_1_0_63_248 {

    from {

        source-address {

            1.0.63.248/32;

        }

    }

    then {

        translated {

            source-prefix 20.1.252.8/32;

            translation-type {

                basic-nat44;

            }

        }

    }

}

term term_1_0_63_249 {
```

```
from {
    source-address {
        1.0.63.249/32;
    }
}
then {
    translated {
        source-prefix 20.1.252.9/32;
        translation-type {
            basic-nat44;
        }
    }
}
}

/* Destination NAT rules */

root@host# show groups __contrail__ services nat rule
sv-__contrail__l3_147_vn_-dn-rule

match-direction output;

term term_20_1_252_8 {
    from {
        destination-address {
            20.1.252.8/32;
        }
    }
    then {
        translated {
            destination-prefix 1.0.63.248/32;
            translation-type {
                dnat-44;
            }
        }
    }
}
```

```
    }
  }
  term term_20_1_252_9 {
    from {
      destination-address {
        20.1.252.9/32;
      }
    }
    then {
      translated {
        destination-prefix 1.0.63.249/32;
        translation-type {
          dnat-44;
        }
      }
    }
  }
}

/* Public VRF Filter */
root@host# show groups __contrail__ firewall family inet filter
redirect_to_public_vrf_filter
term term-_contrail_l3_184_vn_public-x1- {
  from {
    destination-address {
      20.1.0.0/16;
    }
  }
  then {
    routing-instance _contrail_l3_184_vn_public-x1-;
  }
}
```

```
term default-term {
    then accept;
}

/* NAT Vrf filter */

root@host# ...all family inet filter
redirect_to__contrail_l3_147_vn_private-x1-63-nat_vrf

term term-_contrail_l3_147_vn_private-x1-63-nat {
    from {
        source-address {
            1.0.63.248/32;
            1.0.63.249/32;
        }
    }
    then {
        routing-instance _contrail_l3_147_vn_private-x1-63-nat;
    }
}

term default-term {
    then accept;
}

/* IRB interface for NAT VRF */

root@host# show groups __contrail__ interfaces

irb {
    gratuitous-arp-reply;
    unit 147 {
        family inet {
            filter {
                input redirect_to__contrail_l3_147_vn_private-x1-63-nat_vrf;
            }
            address 1.0.63.254/24;
        }
    }
}
```

```
    }

/* Service Interfaces config */

root@host# show groups __contrail__ interfaces si-2/0/0

unit 293 {

    family inet;

    service-domain inside;

}

unit 294 {

    family inet;

    service-domain outside;

}
```

Samples of Generated Configurations for an MX Series Device

This section provides several scenarios and samples of MX Series device configurations generated using Python script.

Scenario 1: Physical Router With No External Networks

The following describes the use case of basic **vn**, **vmi**, **li**, **pr**, **pi** configuration with no external virtual networks. When the Python script shown in the following is executed with the parameters of this use case, the configuration is applied on the MX Series physical router.

Script executed on the Contrail controller:

```
# python provision_physical_router.py --api_server_ip 127.0.0.1 --api_server_port
8082 --admin_user user1 --admin_password password1 --admin_tenant_name
default-domain --op add_basic
```

Generated configuration for MX Series device:

```
root@host# show groups __contrail__
routing-options {
    route-distinguisher-id 10.84.63.133;
    autonomous-system 64512;
}
protocols {
    bgp {
        group __contrail__ {
            type internal;
            multihop;
            local-address 10.84.63.133;
            keep all;
            family inet-vpn {
                unicast;
            }
            family inet6-vpn {
                unicast;
            }
        }
    }
}
```

```

    }
    family evpn {
        signaling;
    }
    family route-target;
}
group __contrail_external__ {
    type external;
    multihop;
    local-address 10.84.63.133;
    keep all;
    family inet-vpn {
        unicast;
    }
    family inet6-vpn {
        unicast;
    }
    family evpn {
        signaling;
    }
    family route-target;
}
}
}
policy-options {
    policy-statement __contrail__default-domain_default-project_vn1-export {
        term t1 {
            then {
                community add target_64200_8000008;
                accept;
            }
        }
    }
    policy-statement __contrail__default-domain_default-project_vn1-import {
        term t1 {
            from community target_64200_8000008;
            then accept;
        }
        then reject;
    }
    community target_64200_8000008 members target:64200:8000008;
}
routing-instances {
    __contrail__default-domain_default-project_vn1 {
        instance-type vrf;
        interface ge-1/0/5.0;
        vrf-import __contrail__default-domain_default-project_vn1-import;
        vrf-export __contrail__default-domain_default-project_vn1-export;
        vrf-table-label;
        routing-options {
            static {
                route 10.0.0.0/24 discard;
            }
            auto-export {
                family inet {
                    unicast;
                }
            }
        }
    }
}
}

```

Scenario 2: Physical Router With External Network, Public VRF

This section describes the use case of **vn**, **vmi**, **li**, **pr**, **pi** configuration with an external virtual network, public VRF. When the Python script shown is executed with the parameters of this use case, the configuration is applied on the MX Series physical router.

This example assumes that the configuration already described in Scenario 1 has been executed.

Script executed on the Contrail controller:

```
# python provision_physical_router.py --api_server_ip 127.0.0.1 --api_server_port
8082 --admin_user user1 --admin_password password1 --admin_tenant_name
default-domain --op add_basic --public_vrf_test True
```

Generated configuration for MX Series device:

The following additional configuration is pushed to the MX Series device, in addition to the configuration generated in Scenario 1.

```
forwarding-options {
  family inet {
    filter {
      input redirect_to__contrail__default-domain_default-project_vn1_vrf;
    }
  }
}
firewall {
  filter redirect_to__contrail__default-domain_default-project_vn1_vrf {
    term t1 {
      from {
        destination-address {
          10.0.0.0/24;
        }
      }
      then {
        routing-instance __contrail__default-domain_default-project_vn1;
      }
    }
    term t2 {
      then accept;
    }
  }
}
routing-instances {
  __contrail__default-domain_default-project_vn1 {
    routing-options {
      static {
        route 0.0.0.0/0 next-table inet.0;
      }
    }
  }
}
```


Scenario 3: Physical Router With External Network, Public VRF, and EVPN

The scenario in this section describes the use case of **vn, vmi, li, pr, pi** physical router configuration with external virtual networks (public VRF) and EVPN configuration. When the Python script (as in the previous examples) is executed with the parameters of this scenario, the following configuration is applied on the MX Series physical router.

This example assumes that the configuration already described in Scenario 1 has been executed.

Script executed on the Contrail controller:

```
# python provision_physical_router.py --api_server_ip 127.0.0.1 --api_server_port
8082 --admin_user user1 --admin_password password1 --admin_tenant_name
default-domain --op add_basic --public_vrf_test True -vxlan 2002
```

Generated configuration for MX Series device:

The following additional configuration is pushed to the MX Series device, in addition to the configuration generated in Scenario 1.

```
protocols {
  mpls {
    interface all;
  }
}
firewall {
  filter redirect_to___contrail___default-domain_default-project_vn1_vrf {
    term t1 {
      from {
        destination-address {
          10.0.0.0/24;
        }
      }
      then {
        routing-instance ___contrail___default-domain_default-project_vn1;
      }
    }
    term t2 {
      then accept;
    }
  }
}
routing-instances {
  ___contrail___default-domain_default-project_vn1 {
    vtep-source-interface lo0.0;
    instance-type virtual-switch;
    vrf-target target:64200:8000008;
    protocols {
      evpn {
        encapsulation vxlan;
        extended-vni-all;
      }
    }
    bridge-domains {
      bd-2002 {
```

```
        vlan-id 2002;
        interface ge-1/0/5.0;
        routing-interface irb.2002;
        vxlan {
            vni 2002;
            ingress-node-replication;
        }
    }
}
}
```

Scenario 4: Physical Router With External Network, Public VRF, and Floating IP Addresses for a Bare Metal Server

The scenario in this section describes the user case of **vn**, **vmi**, **li**, **pr**, **pi** physical router configuration with external virtual networks (public VRF) and floating IP addresses for bare metal server configuration.

Script executed on the Contrail controller:

```
#python provision_physical_router.py --api_server_ip <ip address> --api_server_port
8082 --admin_user admin --admin_password <password> --admin_tenant_name
default-domain --op {fip_test|delete_fip_test}
```

- Related Documentation**
- [REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces on page 210](#)
 - [Using Device Manager to Manage Physical Routers on page 184](#)

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

Introduction: REST APIs for Extending Contrail Cluster

Use the following REST APIs when extending the Contrail cluster to include physical routers, physical interfaces, and logical interfaces.

REST API for Physical Routers

Use the following REST API when extending the Contrail cluster to include physical routers.

```
{
  u'physical-router': {
    u'physical_router_management_ip': u'100.100.100.100',
    u'virtual_router_refs': [],
    u'fq_name': [
      u'default-global-system-config',
```

```
    u'test-router'
  ],
  u'name': u'test-router',
  u'physical_router_vendor_name': u'juniper',
  u'parent_type': u'global-system-config',
  u'virtual_network_refs': [],
  'id_perms': {
    u'enable': True,
    u'uuid': None,
    u'creator': None,
    u'created': 0,
    u'user_visible': True,
    u'last_modified': 0,
    u'permissions': {
      u'owner': u'cloud-admin',
      u'owner_access': 7,
      u'other_access': 7,
      u'group': u'cloud-admin-group',
      u'group_access': 7
    },
    u'description': None
  },
  u'bgp_router_refs': [],
  u'physical_router_user_credentials': {
    u'username': u'',
    u'password': u''
  },
  'display_name': u'test-router',
  u'physical_router_dataplane_ip': u'101.1.1.1'
}
```

REST API for Physical Interfaces

Use the following REST API when extending the Contrail cluster to include physical interfaces.

```
{
  u'physical-interface': {
    u'parent_type': u'physical-router',
    'id_perms': {
      u'enable': True,
      u'uuid': None,
      u'creator': None,
      u'created': 0,
      u'user_visible': True,
      u'last_modified': 0,
      u'permissions': {
        u'owner': u'cloud-admin',
        u'owner_access': 7,
        u'other_access': 7,
        u'group': u'cloud-admin-group',
        u'group_access': 7
      },
      u'description': None
    },
    u'fq_name': [
      u'default-global-system-config',
      u'test-router',
      u'ge-0/0/1'
    ],
    u'name': u'ge-0/0/1',
    'display_name': u'ge-0/0/1'
  }
}
```

```
}
```

REST API for Logical Interfaces

Use the following REST API when extending the Contrail cluster to include logical interfaces.

```
{
  u'logical-interface': {
    u'fq_name': [
      u'default-global-system-config',
      u'test-router',
      u'ge-0/0/1',
      u'ge-0/0/1.0'
    ],
    u'parent_uuid': u'6608b8ef-9704-489d-8cbc-fed4fb5677ca',
    u'logical_interface_vlan_tag': 0,
    u'parent_type': u'physical-interface',
    u'virtual_machine_interface_refs': [
      {
        u'to': [
          u'default-domain',
          u'demo',
          u'4a2edbb8-b69e-48ce-96e3-7226c57e5241'
        ]
      }
    ],
    u'id_perms': {
      u'enable': True,
      u'uuid': None,
      u'creator': None,
      u'created': 0,
      u'user_visible': True,
    }
  }
}
```

```
    u'last_modified': 0,
    u'permissions': {
        u'owner': u'cloud-admin',
        u'owner_access': 7,
        u'other_access': 7,
        u'group': u'cloud-admin-group',
        u'group_access': 7
    },
    u'description': None
},
u'logical_interface_type': u'l2',
'display_name': u'ge-0/0/1.0',
u'name': u'ge-0/0/1.0'
}
}
```

**Related
Documentation**

- [Using ToR Switches and OVSDB to Extend the Contrail Cluster to Other Instances on page 169](#)
- [Using Device Manager to Manage Physical Routers on page 184](#)

CHAPTER 8

Installing and Using Contrail Storage

- [Installing and Using Contrail Storage on page 215](#)

Installing and Using Contrail Storage

- [Overview of the Contrail Storage Solution on page 215](#)
- [Basic Storage Functionality with Contrail on page 216](#)
- [Ceph Block and Object Storage Functionality on page 216](#)
- [Using the Contrail Storage User Interface on page 217](#)
- [Hardware Specifications on page 218](#)
- [Software Files for Compute Storage Nodes on page 218](#)
- [Contrail OpenStack Nova Modifications on page 218](#)
- [Installing the Contrail Storage Solution on page 219](#)
- [Using Fabric Commands to Install and Configure Storage on page 219](#)
- [Fabric Installation Procedure on page 220](#)
- [Using Server Manager to Install and Configure Storage on page 222](#)
- [Server Manager Installation Procedure for Storage on page 222](#)
- [Example: Configurations for Storage for Reimaging and Provisioning a Server on page 223](#)
- [Storage Installation Limits on page 228](#)

Overview of the Contrail Storage Solution

Contrail provides a storage support solution using OpenStack Cinder configured to work with Ceph. Ceph is a unified, distributed storage system whose infrastructure provides storage services to Contrail. This Contrail storage solution provides a validated Network File System (NFS) storage service, however, it is not the Ceph FS distributed file system.

The Contrail storage solution has the following features:

- Provides storage class features to Contrail clusters, including replication, reliability, and robustness.
- Uses open source components.
- Uses Ceph block and object storage functionality.

- Integrates with OpenStack Cinder functionality.
- Does not require virtual machines (VMs) to configure mirrors for replication.
- Allows nodes to provide both compute and storage services.
- Provides easy installation of basic storage functionality based on Contrail roles.
- Provides services necessary to perform virtual machine migrations between compute nodes, and supports both migratable and non-migratable virtual machines.
- Provides a Contrail-integrated user interface from which the user can monitor Ceph components and drill down for more information about components.

Basic Storage Functionality with Contrail

The following are basic interaction points between Contrail and the storage solution.

- Cinder volumes must be manually configured prior to installing the Contrail storage solution. The Cinder volumes can be attached to virtual machines (VMs) to provide additional storage.
- The storage solution stores virtual machine boot images and snapshots in Glance, using Ceph object storage functionality.
- All storage nodes can be monitored through a graphical user interface (GUI).
- It is possible to migrate virtual machines that have ephemeral storage in Ceph.

Ceph Block and Object Storage Functionality

Installing the Contrail storage solution creates the following Ceph configurations.

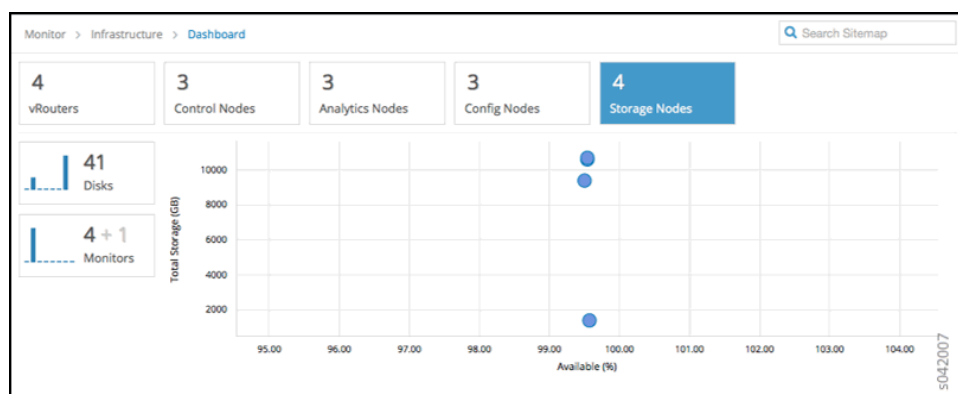
- Each disk is configured as a standalone storage device, enhancing optimal performance and creating proper failure boundaries. Ceph allocates and assigns a process called object storage daemon (OSD) to each disk.
- A replication factor of 2 is configured, consisting of one original instance plus one replica copy. Ceph ensures that each replica is on a different storage node.
- A Ceph monitor process (mon) is configured on each storage node.
- The correct number of placement groups are automatically configured, based on the number of disk drives in the cluster.
- Properly identified SSD drives are set up for use as Ceph OSD journals to reduce write latencies.
- An NFS server is created in a virtual machine within the cluster to support virtual machine migration. The NFS file system is mounted on all storage nodes, and every storage node has a shared Nova directory under the `/var/lib/nova/instances` directory. By default, this NFS file system is configured to utilize 30% of the total initial Contrail storage capacity.

Using the Contrail Storage User Interface

The Contrail storage solution provides a user interface integrated into the Contrail user interface. The storage solution user interface displays the following:

- Customer usable space, which is different from Ceph total space. The displayed usable space does not display the space used by replication and other Ceph functions.
- Monitor OSDs (disks), monitoring processes (MON), and state changes, enabling quick identification of resource failures within storage components.
- Total cluster I/O statistics and individual drive statistics.
- Ceph-specific information about each OSD (disk).
- Ceph logs, Ceph nodes, and Ceph alerts.

Select **Monitor > Infrastructure > Dashboard** to display an at-a-glance view of the system infrastructure components, including the numbers of virtual routers, control nodes, analytics nodes, config nodes, and storage nodes currently operational, and a bubble chart of storage nodes showing the Available (%) and Total Storage (GB). See the following figure.



Bubble charts use the following color-coding scheme for storage nodes:

- Blue—working as configured.
- Red—error, node is down.
- Yellow—one of the node disks is down.

Select **Monitor > Storage > Dashboard** to see a summary of cluster health, usage, pools, and disk status, and to gain insight into activity statistics for all nodes. See the following figure.



Hardware Specifications

The following are additional hardware specifications needed for the Contrail storage solution.

Additional minimum specifications:

- Two 500 GB, 7200 RPM drives in the server 4 and server 5 cluster positions (those with the compute storage role) in the Contrail installation. This configuration provides 1 TB of clustered, replicated storage.

Recommended compute storage configuration:

- For every 4-5 HDD devices on one compute storage node, use one SSD device to provide the OSD journals for that set of HDD devices.

Software Files for Compute Storage Nodes

The Contrail storage solution is only supported with the Ubuntu operating system.

For each compute storage node, ensure the following software is downloaded:

- The storage Debian package: **contrail-storage-packages_x.xx-xx~xxxxxx_all.deb**.
- NFS VM **qcow2** image from Juniper Networks.

Contrail OpenStack Nova Modifications

Contrail's OpenStack Nova function has been modified to spawn both migratable and non-migratable virtual machines.

- Nova's typical virtual machine storage directory, `/var/lib/nova/instances`, is used for non-migratable virtual machine ephemeral storage.
- Contrail storage creates a new directory, `/var/lib/nova/instances/global`, used for the ephemeral storage for migratable virtual machines. The `/var/lib/nova/instances/global` must be mounted on a shared storage device (NFS with Contrail Storage), accessible from all the compute nodes.
- To start a non-migratable virtual machine with the Nova CLI command `nova boot`, the additional argument `"--meta storage_scope=local"` must be provided.
- To start a migratable virtual machine with `nova boot`, the additional argument `"--meta storage_scope=global"` must be provided. To force Nova and the Horizon UI to spawn migratable virtual machines by default, the storage scope must be set to global. This task is described in the next section.

Installing the Contrail Storage Solution

The Contrail storage solution can be installed using the same tools used to install Contrail, either by using Fabric (fab) commands or by using the Contrail Server Manager.

Both installation methods are described in the following sections.

Installation Notes

- When installing a base operating system on any compute storage node, the operating system must be installed only on a single drive. The other drives must be configured as individual devices, and should not be concatenated together in a logical volume manager (LVM) device.
- For best performance, it is recommended to use solid state devices (SSD) for the Ceph OSD journals. Each SSD device can provide OSD journal support to 3-6 HDD OSD devices, depending on the model of SSD device. Most SSD devices can support up to 4 HDDs, assuming the HDDs are running at capacity.

Using Fabric Commands to Install and Configure Storage

Use the information in this section to install storage using Fabric (fab) commands.

When installing the operating system on a compute storage node, install the operating system on a single drive and leave all other drives as unbundled.

Installing the Contrail storage solution with Fabric commands gives the following:

- Base Ceph block device and object support.
- Easy configuration of SSD devices for OSD journals.
- Virtual machine migration support.
- Limited Cinder multi-backend support.

Cautions

Before installing, ensure the following:

- Manually ensure that the UID or GID of the Nova user is identical on all compute nodes before provisioning any software.
- Manually ensure that the time is identical on all nodes by configuring NTP.

Fabric Installation Procedure

This section provides guidelines and steps for using Fabric (fab) commands to install the Contrail storage solution. The installation is similar to a regular Contrail fab installation, however, you define additional storage information in the **testbed.py** file, including:

- Define new roles: **storage-master** and **compute-storage**.
 - Define how each additional non-root drive is used in the cluster.
 - Define potential additional virtual machine migration variables.
 - Copy and install the additional storage package to systems.
1. Install the storage Debian package on all nodes:
fab install_storage_pkg_all:/YYYY/contrail-storage-package-XXX.deb
 2. After using the **fab install_contrail** command, use the **fab install_storage** command.
 3. After using the **fab setup_all** command, use the **fab setup_storage** command.
 4. If you need to enable Contrail-based live virtual machine migration, use the **fab setup_nfs_livem** command or the **fab setup_nfs_livem_global** command, as described in the following.



NOTE: If virtual machine migration is not needed, do not use either command.

- Use the **fab setup_nfs_livem** command to store the virtual machine's ephemeral storage on local drives.
 - Use the **fab setup_nfs_livem_global** command to store the virtual machine's ephemeral storage within Contrail's storage (using Ceph). This command sets the cluster storage scope to global.
5. Add two new Contrail storage roles: **compute-storage** and **storage-master**.
 - Define the **storage-master** role on all nodes running OpenStack. Although Ceph has no notion of a master, define this role because Ceph must be run on the node that runs the OpenStack software. OpenStack nodes typically do not have any cluster storage defined, only local storage.

- The **storage-compute** role is an add-on role. It means that compute nodes have the option of providing storage functionality. Standalone storage nodes are not supported.
6. Change the **testbed.py** file details as needed for your environment.

In the base configuration, define the **storage_node_config** values, which gives device details. See the following example.

```
storage_node_config = {

    host2 : { 'disks' : ['/dev/sdb', '/dev/sdc'], 'ssd-disks': ['/dev/sdd',
'/dev/sde'], 'local-disks' : ['/dev/sdf', '/dev/sdh'], 'nfs' :
['10.87.140.156:/test', '10.87.140.156:/test1']},

    host3 : { 'disks' : ['/dev/sdb:/dev/sde', '/dev/sdc:/dev/sde',
'/dev/sdd:/dev/sde', '/dev/sdf:/dev/sdj', '/dev/sdg:/dev/sdj',
'/dev/sdh:/dev/sdj', '/dev/sdi:/dev/sdj', 'local-ssd-disks' : ['/dev/sdk',
'/dev/sdl']},}
```

Available device details parameters include:

- **disks** and **ssd-disks** are Ceph disks.
 - **local-disk** and **local-ssd-disks** are LVM disks.
 - **host2** in the example shows all the storage types that can be configured using Cinder multi-backend.
 - **disks** is a list of HDD disks used for a Ceph HDD pool.
 - **ssd-disks** is a list of SSD disks used for a Ceph SSD pool.
 - **local-disks** is a list of disks used for local LVM storage.
 - **nfs** is an NFS device.
 - In the example, **host3** is a more typical configuration.
 - **/dev/sde** and **/dev/sdj** are SSD disks that are used as OSD journals for other HDD drives.
 - **local-ssd-disks** is a list of disks used for local SSD LVM storage.
7. Add virtual machine migration as needed, using the following parameters.

```
live_migration = True
```

```
ceph_nfs_livevm = True
```

```
ceph_nfs_livem_subnet = '192.168.10.0/24' # Private subnet to be provided for live
migration VM
```

```
ceph_nfs_livem_image = '/Ubuntu/libmnfs.qcow2' # path of live migration qcow2 image.
This image is provided by Juniper Networks.
```

```
ceph_nfs_livem_host = host3 # host in which the NFS VM will run
```

For external NFS server-based live migration, use the following configuration.

```
live_migration = True
```

```
ext_nfs_livevm = True
```

```
ext_nfs_livem_mount = '10.10.10.10:/nfsmount' # External NFS server mount path
```



NOTE: When using an external NFS server, make sure the NFS server maps the uids and gids correctly, and provides read and write access for all the uids. If there is any issue related to the permission, either the VM launch errors out or the live migration fails with permission-related errors.

Using Server Manager to Install and Configure Storage

This section provides notes and guidelines to install the storage solution using the Contrail Server Manager. Installing the Contrail Storage solution using Server Manager provides:

- Base Ceph block device and object support.
- Easy configuration of SSD journals.
- Support for live migration configuration.

Before installing the base operating system with Server Manager, ensure that the compute storage nodes have been configured with single operating system device installs.

Cautions

- Virtual machine migration support uses a fixed IP address (192.168.101.3) for the **livemnfs** virtual machine.
- There is no Cinder multi-backend support.
- There is no support for single server provisioning, the entire cluster must be provisioned.

Server Manager Installation Procedure for Storage

This section provides notes and guidelines if you choose to install the storage solution using the Contrail Server Manager.

1. Upload the storage package: **server-manager add image -f <filename.json>**

where <filename.json> has content similar to the following example:

```
{
  "image": [
    {
      "id": "contrail-storage-packages_1.10-xx~xxxxxx_all",
      "parameters": "{}",
      "path":
        "/store/contrail-storage-packages_1.10-xx~xxxxxx_all.deb",
      "type": "contrail-storage-ubuntu-package",
```

```

        "version": "1.10-xx"
    },
]
}

```

2. Use the **ceph-authtool** command if you need to generate unique keys for administration, monitor, and OSD.

- a. To install **ceph-authtool** on CentOS, use the following command:

```

yum install
http://ceph.com/rpm/el6/x86_64/ceph-common-0.80.5-0.el6.x86_64.rpm

```

- b. To install **ceph-authtool** on Ubuntu:

```

apt-get install ceph-common

```

- c. Use the following command once for each key:

```

ceph-authtool --gen-print-key

```

- d. Add the generated keys to the **cluster.json** file:

```

cluster.json:

"storage_mon_secret":

"AQBDCdpTsB5FChAA0zI2++uosfmtj7tjmhPu0g==",

"osd_bootstrap_key":

"AQBKCdpTmN+HGAA16rmStq5iYoPnANzSXLcXA==",

"admin_key":

"AQBLCdpTu0S6FhAAFDW0SsdzyDAUeuwOr/h61A=="

```

- e. In Contrail Release 2.10 and later, add live-migration configuration to the **cluster.json** file, if you are using live migration.

```

"live_migration": "enable",

"live_migration_nfs_vm_host": "compute-node-01",

"live_migration_storage_scope": "global",

```

Example: Configurations for Storage for Reimaging and Provisioning a Server

Use the following example configurations as guidelines for reimaging and provisioning a server for storage. Examples are given for configurations for releases prior to Release 2.10 and for configurations for Release 2.10 and later.

1. Define storage in the cluster. The following example configurations show new key-value pairs added to the configuration. The **cluster** section should appear similar to the following when storage is defined in a cluster.

Example: Storage and key-value pairs defined in releases prior to 2.10:

```
{
  "cluster" : [
    {
      "id" : "demo-cluster",
      "parameters" : {
        "router_asn": "<asn>",
        "database_dir": "/home/cassandra",
        "database_token": "",
        "use_certificates": "False",
        "multi_tenancy": "False",
        "encapsulation_priority": "MPLSoUDP,MPLSoGRE,VXLAN",
        "service_token": "<password>",
        "keystone_user": "admin",
        "keystone_password": "<password>",
        "keystone_tenant": "admin",
        "analytics_data_ttl": "168",
        "subnet_mask": "<ip address>",
        "gateway": "<ip address>",
        "password": "<password>",
        "haproxy": "disable",
        "external_bgp": "",
        "domain": "demo.company.net",
        "storage_mon_secret": "$ABC123",
        "osd_bootstrap_key": "$ABC123",
        "admin_key": "$ABC123"
      }
    }
  ]
}
```



```

    ]
}

```

Example: Storage and key-value pairs defined in releases 2.10 and later:

```

{
  "cluster" : [
    {
      "id" : "demo-cluster",
      "parameters" : {
        "router_asn": "<asn>",
        "database_dir": "/home/cassandra",
        "database_token": "",
        "use_certificates": "False",
        "multi_tenancy": "False",
        "encapsulation_priority": "MPLSoUDP,MPLSoGRE,VXLAN",
        "service_token": "<password>",
        "keystone_user": "admin",
        "keystone_password": "<password>",
        "keystone_tenant": "admin",
        "analytics_data_ttl": "168",
        "subnet_mask": "<ip address>",
        "gateway": "<ip address>",
        "password": "<password>",
        "haproxy": "disable",
        "external_bgp": "",
        "domain": "demo.company.net",
        "storage_mon_secret": "$ABC123",
        "osd_bootstrap_key": "$ABC123",
        "admin_key": "$ABC123",
        "live_migration" : "enable",
        "live_migration_nfs_vm_host": "compute-host-01,

```

```
    "live_migration_storage_scope": "global",  
  },  
},  
]
```

2. Add the **disks** key, the **storage-compute** role value, and the **storage_repo_id** key.

- The **storage_repo_id** key must be added to servers with the **storage-master** or **storage-compute** roles.
- The **disks** key-value pair must be added to servers with the **storage-compute** roles.
- The **storage-master** value must be added to the **roles** key for the server that has the **storage-master** role.
- The **storage-compute** value must be added to the **roles** key for the servers that have the **storage-compute** role.

The following server section is an example, showing the **storage_repo_id** and **disks** keys, and the **storage-compute** and **storage-master** values.

In the example, one server contains the **storage-compute** role and has 3 HDD drives (**/dev/sdb**, **/dev/sdc**, **/dev/sdd**), supporting 3 OSDs.

Each OSD uses one partition of an SSD drive (**/dev/sde**) as its OSD journal.

The server manager software correctly partitions **/dev/sdd** and assign one partition to each OSD. The **storage_repo_id** contains the base name of the Contrail storage package which has been added as an image to Server Manager.

Example: Server.json updates defined in releases earlier than 2.10:

```
{  
  "server": [  
    {  
      "id": "demo2-server",  
      "mac_address": "<mac address>",  
      "ip_address": "<password>",  
      "parameters" : {  
        "interface_name": "eth1",  
        "compute_non_mgmt_ip": "",  
        "compute_non_mgmt_gway": "",  
        "storage_repo_id": "contrail-storage-packages",  
        "disks": ["/dev/sdb:/dev/sde", "/dev/sdc:/dev/sde",
```

```

"/dev/sdd:/dev/sde"]
    },
    "roles" :
["config","openstack","control","compute","collector","webui","database","storage-compute","storage-master"],
    "cluster_id": "demo-cluster",
    "subnet_mask": "<ip address>",
    "gateway": "1<ip address>",
    "password": "<password>",
    "domain": "demo.company.net",
    "email": "id@company.net"
} ]
}

```

Example: Server.json updates defined in releases 2.10 and later:

```

Server.json :
{"server": [
    {
        "id": "demo2-server",
        "mac_address": "<mac address>",
        "ip_address": "<ip address>",
        "parameters" : {
            "interface_name": "eth1",
            "compute_non_mgmt_ip": "",
            "compute_non_mgmt_gway": "",
            "storage_repo_id": "contrail-storage-packages",
            "disks": ["/dev/sdb:/dev/sde", "/dev/sdc:/dev/sde",
"/dev/sdd:/dev/sde"]
        },
        "roles" :
["config","openstack","control","compute","collector","webui","database","storage-compute","storage-master"],
        "contrail": {
            "control_data_interface": "p3p2"

```

```

    },
    "network": {
      "interfaces": [
        {
          "default_gateway": "<ip address>",
          "dhcp": true,
          "ip_address": "<ip address>",
          "mac_address": "<mac address>",
          "member_interfaces": "",
          "name": "eth1",
          "tor": "",
          "tor_port": "",
          "type": "physical"
        },
        {
          "default_gateway": "<ip address>",
          "dhcp": "",
          "ip_address": "<ip address>",
          "mac_address": "<mac address>",
          "member_interfaces": "",
          "name": "p3p2",
          "tor": "",
          "tor_port": "",
          "type": "physical"
        }
      ],
      "management_interface": "eth1"
    },
    "cluster_id": "demo-cluster",
    "subnet_mask": "<ip address>",
    "gateway": "<ip address>",
    "password": "<password>",
    "domain": "demo.company.net",
    "email": "id@company.net"
  } ]
}

```

3. Use the following commands to provision the entire cluster:

```

# /opt/contrail/server_manager/client/server-manager -c

# /opt/contrail/server_manager/smgr_config.ini provision --cluster_id test-cluster
contrail_test_pkg

```

Storage Installation Limits

General Limitations

- Minimum number of storage nodes to configure: 2
- The number of storage nodes should always be an even number (2, 4, 12, 22, etc.).

**Fab Storage Install
Limitations**

There are no additional limitations to installation when using fab commands.

**Server Manager
Storage Install
Limitations**

- There is no integrated way to add OSDs or drives to a storage node.
- There is no integrated way to add new storage nodes to a cluster.
- Provisioning a single server is not supported. You can add a server to Server Manager and then provision the entire cluster.
- The live migration overlay network is preset to use 192.168.101.0/24.
- The user must copy the image **livemnfs.qcow2.gz** to the folder **/var/www/html/contrail/images** before provisioning live migration using Server Manager.

Upgrading Contrail Software

- Upgrading Contrail Software on page 231
- DKMS for vRouter Kernel Module on page 234

Upgrading Contrail Software

Use the following procedure to upgrade an installation of Contrail software from one release to a more recent release. This procedure is valid for Contrail Release 2.21 and later.



NOTE: If you are installing Contrail for the first time, refer to the full documentation and installation instructions in “Installing the Operating System and Contrail Packages” on page 13.

Instructions are given for both CentOS and Ubuntu versions. The only Ubuntu version supported for upgrading Ubuntu 4.04.2.

To upgrade Contrail software from Contrail Release 2.21 or later:

1. Download the **contrail-install-packages-x.xx-xxx.xxx.noarch.rpm | deb** file from <http://www.juniper.net/support/downloads/?p=contrail#sw> and copy it to the **/tmp** directory on the config node, as follows:

CentOS : **scp <id@server>:/path/to/contrail-install-packages-x.xx-xxx.xxx.noarch.rpm /tmp**

Ubuntu : **scp <id@server>:/path/to/contrail-install-packages-x.xx-xx~havana_all.deb /tmp**



NOTE: The variables xxx.-xxx and so on represent the release and build numbers that are present in the name of the installation packages that you download.

2. Install the **contrail-install-packages**, using the correct command for your operating system:

CentOS: `yum localinstall /tmp/contrail-install-packages-x.xx-xxx.xxx.noarch.rpm`

Ubuntu: `dpkg -i /tmp/contrail-install-packages_x.xx-xxx~_all.deb`

3. Set up the local repository by running the `setup.sh`:

`cd /opt/contrail/contrail_packages; ./setup.sh`

4. Ensure that the `testbed.py` file that was used to set up the cluster with Contrail is intact in the `/opt/contrail/utils/fabfile/testbeds/` directory.
 - Ensure that the `testbed.py` file has been set up with a combined `control_data` section (required in Contrail Release 1.10 and later).

See [“Setting Up the Testbed Definitions File” on page 17](#).

5. In release packages prior to Contrail Release 3.0, the packaged Cassandra version is 1.2.11. In the 3.0 release, the packaged Cassandra version is 2.1.9. Upgrading Cassandra from 1.2.11 to 2.1.9 is not supported by Cassandra. For more information, refer to [DataStax Upgrade Guide, Cassandra 2.1.x restrictions](#).

Consequently, during the Contrail upgrade procedure (`fab upgrade_contrail`), the Cassandra SSTables are upgraded, which takes a long time if the Cassandra data is huge (usually because the Contrail Analytics keyspace is huge).

There is an option to minimize upgrade down time by dropping the Contrail Analytics keyspace before the upgrade, by issuing the following fab command:

`fab drop_analytics_keyspace`

6. Upgrade the software, using the correct set of commands to match your operating system and vRouter, as described in the following:

Change directory to the `utils` folder:

`cd /opt/contrail/utils; \`

Select the correct upgrade procedure from the following to match your operating system and vRouter. In the following, `<from>` refers to the currently installed release number, such as 2.0, 2.01, 2.1, or 2.2:

CentOS Upgrade Procedure:

`fab upgrade_contrail:<from>,/tmp/contrail-install-packages-x.xx-xxx.xxx.noarch.rpm;`

Ubuntu 14.04 Upgrade, Two Procedures:

There are two different upgrade procedures for the upgrade to Contrail Release 3.0, depending on which vRouter (`contrail-vrouter-3.13.0--generic` or `contrail-vrouter-dkms`) is installed in your current setup.

In Contrail Release 3.0 and later, the recommended kernel version for an Ubuntu 14.04-based system is 3.13.0-. Both procedures can use the command `fab upgrade_kernel_all` to upgrade the kernel.

Ubuntu 14.04 Upgrade Procedure For a System With contrail-vrouter-3.13.0--generic:

Use the following upgrade procedure for Contrail Release 3.0 systems based on Ubuntu 14.04 with the **contrail-vrouter-3.13.0-35-generic** installed. The command sequence upgrades the kernel version and also reboots the compute nodes when finished.

```
fab install_pkg_all:/tmp/contrail-install-packages-x.xx-xxx~icehouse_all.deb;
```

```
fab migrate_compute_kernel;
```

```
fab upgrade_contrail:<from>,/tmp/contrail-install-packages-x.xx-xxx~icehouse_all.deb;
```

```
fab upgrade_kernel_all;
```

```
fab restart_openstack_compute;
```

Ubuntu 14.04 Upgrade Procedure For System with contrail-vrouter-dkms:

Use the following upgrade procedure for Contrail Release 3.0 systems based on Ubuntu 14.04 with **contrail-vrouter-dkms** installed. The command sequence upgrades the kernel version and also reboots the compute nodes when finished.

```
fab upgrade_contrail:
<from>,/tmp/contrail-install-packages-x.xx-xxx~icehouse_all.deb;
```

All nodes in the cluster can be upgraded to kernel version 3.13.0-40 by using the following **fab** command:

```
fab upgrade_kernel_all
```

7. (For Contrail Storage option, only.)

Contrail Storage has its own packages.

To upgrade Contrail Storage, download the file:

```
contrail-storage-packages_x.x-xx*.deb
```

from <http://www.juniper.net/support/downloads/?p=contrail#sw>

and copy it to the **/tmp** directory on the config node, as follows:

```
Ubuntu: scp <id@server>:/path/to/contrail-storage-packages_x.x-xx*.deb /tmp
```



NOTE: Use only Icehouse packages (for example, **contrail-storage-packages_2.0-22~icehouse_all.deb**) because OpenStack Havana is no longer supported.

Use the following statement to upgrade the software:

```
cd /opt/contrail/utlis; \
```

```
Ubuntu: fab
```

```
upgrade_storage:<from>,/tmp/contrail-storage-packages_2.0-22~icehouse_all.deb;
```

DKMS for vRouter Kernel Module

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.

This feature is supported in Contrail Release 1.10 on Ubuntu distributions.

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>

PART 3

Configuring Contrail

- [Configuring Virtual Networks on page 237](#)
- [Example of Deploying a Multi-Tier Web Application Using Contrail on page 281](#)
- [Configuring Services on page 293](#)
- [Configuring Service Chaining on page 319](#)
- [Adding Physical Network Functions in Service Chains on page 371](#)
- [Load Balancers on page 381](#)
- [Configuring High Availability on page 395](#)
- [Configuring Multitenancy Support on page 413](#)
- [Optimizing Contrail on page 419](#)

CHAPTER 10

Configuring Virtual Networks

- [Creating Projects in OpenStack for Configuring Tenants in Contrail on page 238](#)
- [Creating Virtual Networks and Policies in Juniper Networks Contrail on page 239](#)
- [Creating Virtual Networks and Policies in OpenStack Contrail on page 247](#)
- [Creating an Image and Launching a Virtual Machine on page 257](#)
- [Creating a Floating IP Address Pool and Allocating it to a Virtual Machine on page 262](#)
- [Using Security Groups with Virtual Machines \(Instances\) on page 266](#)
- [Support for IPv6 Networks in Contrail on page 269](#)
- [Configuring EVPN and VXLAN on page 272](#)

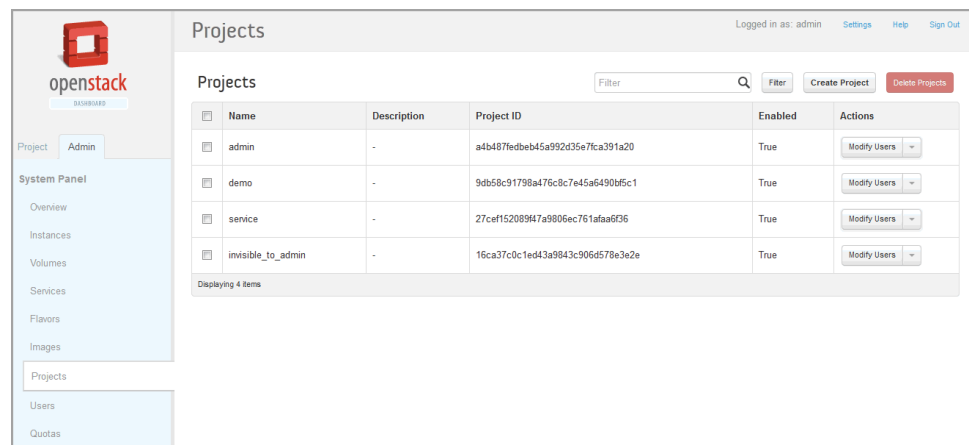
Creating Projects in OpenStack for Configuring Tenants in Contrail

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 36 on page 238](#).

Figure 36: OpenStack Projects



2. In the upper right, click the **Create Project** button to access the **Add Project** window; see [Figure 37 on page 238](#).

Figure 37: Add Project

Add Project

Project Info | Project Members | Quota

Name
customer 1

Description
Additional information here...

Enabled
☒

From here you can create a new project to organize users.

Cancel Finish

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 Virtual Networks and Policies in Juniper Networks Contrail on page 239](#)
 - [Creating Virtual Networks and Policies in OpenStack Contrail on page 247](#)
 - [OpenStack documentation](#)

Creating Virtual Networks and Policies in Juniper Networks Contrail

- [Creating a Virtual Network—Juniper Networks Contrail on page 240](#)
- [Deleting a Virtual Network with Juniper Networks Contrail on page 242](#)
- [Creating a Network Policy with Juniper Networks Contrail on page 244](#)
- [Associating a Network to a Policy—Juniper Networks Contrail on page 246](#)

Creating a Virtual Network—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. Before creating a virtual network, create an IP address management (IPAM) for your project. Select **Configure > Networking > IP Address Management**, then click the **Create** button.

The **Add IP Address Management** window appears, see [Figure 38 on page 240](#).

Figure 38: Add IP Address Management

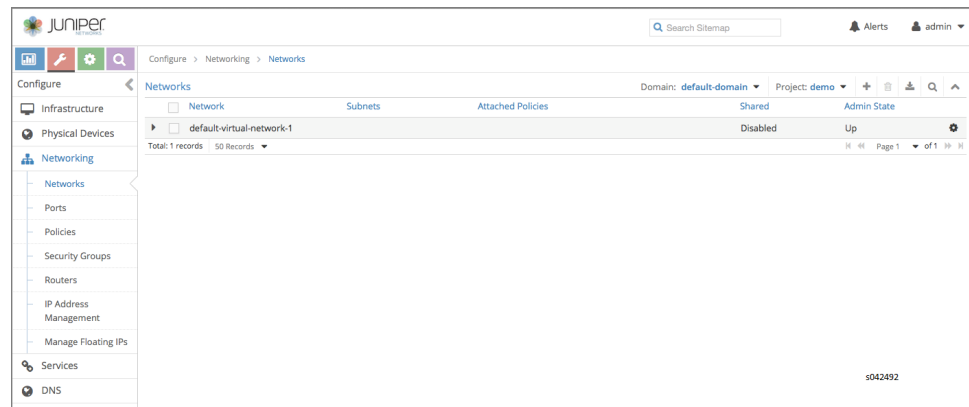
2. Complete the fields in **Add IP Address Management**: The fields are described in [Table 13 on page 240](#).

Table 13: Add IP Address Management Fields

Field	Description
Name	Enter a name for the IPAM you are creating.
DNS Method	Select from a drop-down list the domain name server method for this IPAM: Default , Virtual DNS , Tenant , or None .
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. Select **Configure > Networking > Networks** to access the **Configure Networks** screen; see [Figure 39 on page 241](#).

Figure 39: Configure Networks




- 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 40 on page 241](#). Use the scroll bar to access all sections of this window.

Figure 40: Create Network

- 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 14 on page 241](#).

Table 14: Create Network Fields

Field	Description
Name	Enter a name for the virtual network you are creating.
Network Policy(s)	Select the policy to be applied to this network from the drop-down list of available policies. You can select more than one policy by clicking each one needed.

Table 14: Create Network Fields (continued)

Field	Description
Subnets	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. Or click the - icon to remove the selections.
Host Routes	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.
Advanced Options	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.
Floating IP Pools	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.
Route Target(s)	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.

- To save your network, click the **Save** button, or click **Cancel** to discard your work and start over.

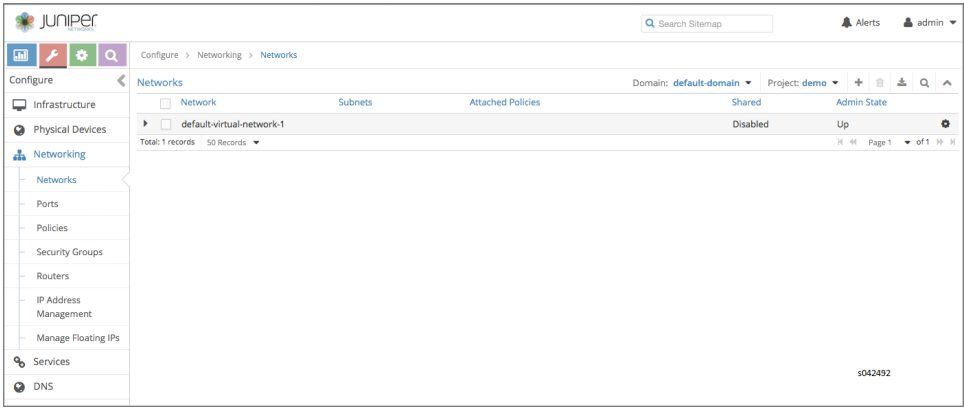
Now you can create a network policy, see [“Creating a Network Policy with Juniper Networks Contrail” on page 244](#).

Deleting a Virtual Network with Juniper Networks Contrail

You can delete any of the virtual networks in your system. However, you must first disassociate any virtual machines (instances) that are associated with that network. Use OpenStack to view and delete the virtual machines associated with a virtual network, see [“Deleting a Virtual Network–OpenStack Contrail” on page 250](#). When you have deleted the virtual machines associated with a virtual network, you can delete the network in OpenStack, or you can delete the network in Juniper Networks Contrail, using the following procedure.

- To view the virtual networks in the current project, select **Configure > Networks**. The **Configure Networks** window is displayed. See [Figure 41 on page 243](#).

Figure 41: Configure Networks



- 2. Select the network you want to delete, then click the Delete (trashcan) icon at the top right. A confirm window is displayed.
- 3. Click **Confirm** to delete the network, or click **Cancel** to quit the delete activity.

See Also • [Creating Virtual Networks and Policies in Juniper Networks Contrail on page 239](#)

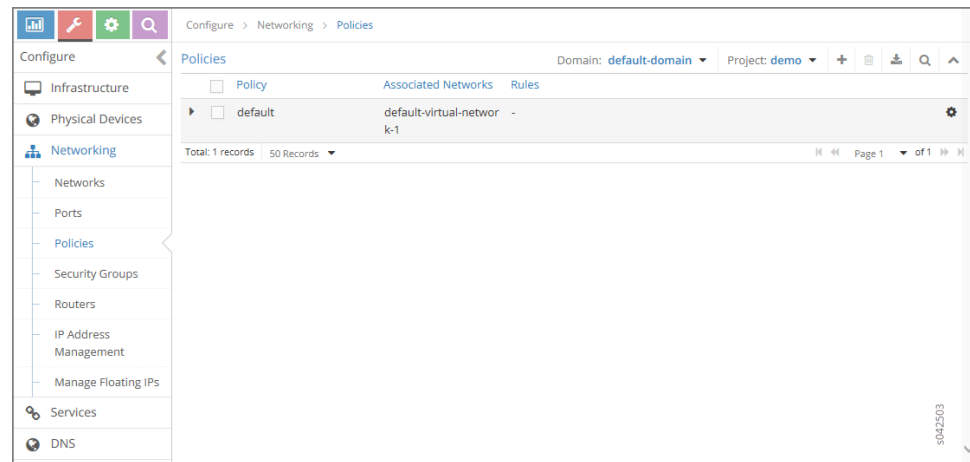
Creating a Network Policy with Juniper Networks Contrail

The Contrail Controller makes creating network traffic policies very simple. You work from the self-service user interface to define a policy, then define a rule or rules to be applied in that policy. You can define such parameters as the type and direction of traffic for the rule, the source and destination of that traffic, traffic originating from or destined for specific ports, the sequence in which to apply a rule, and so on.

To create a network policy when using Juniper Networks Contrail:

1. In the Contrail Web user interface, select **Configure > Networking > Policies**. The **Policies** window is displayed. See [Figure 42 on page 244](#).

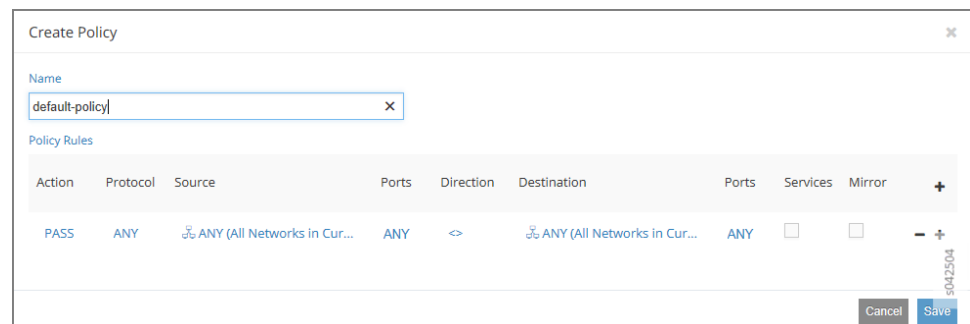
Figure 42: Policies Window



2. Click the + icon.

The **Create Policy** window is displayed. See [Figure 43 on page 244](#). Click the + icon in the Create Policy window.

Figure 43: Create Policy Window



3. Enter the policy name and select the values from the menus in the **Create Policy** window. [Table 15 on page 245](#) describes the selections.

Table 15: Create Policy Fields

Field	Description
Name	Enter a name for the policy you are creating.
Policy Rules	Use this area to define the rules for the policy you are creating. Click the + (plus sign) to open up the fields for defining the rules. Click the – (minus sign) to delete any rule. Multiple rules can be added to a policy. Each policy rule field is described in the following table rows.
Action	Define the action to take with traffic that matches the current rule. Select from a list: Pass, Deny .
Protocol	Define the protocol associated with traffic for this policy rule. Select from a list of available protocols (or ANY to use any protocol, not named here): ANY, TCP, UDP, ICMP, ICMP6 .
Source	Select the source network for traffic associated with this policy rule. Choose ANY or select from the menu list of all available sources. Sources are displayed in the form: <i>domain-name:project-name:network-name</i> .
Ports	Use this field to specify that traffic from particular source ports are associated with this policy rule. Identify traffic from any port or enter a specific port, a list of ports separated with commas, or a range of ports in the form <i>nnnn-nnnnn</i> .
Direction	Define the direction of traffic to match the rule. For traffic moving in and out, select <> (bidirectional). For traffic moving in one direction, select > (unidirectional).
Destination	Select the destination network for traffic to match this rule. Choose ANY or select from the menu of all available destinations. Destinations are displayed in the form: <i>domain-name:project-name:network-name</i> .
Destination	Select the destination port for traffic to match this rule. Enter ANY for any destination port or enter a specific port, a list of ports separated with commas, or a range of ports in the form <i>nnnn-nnnnn</i> .
Services	Select this check box to open a field where you can select from a list of available services to apply to this policy. The services are applied in the order in which they are selected. There is a restricted set of options that can be selected when applying services. For more information about services, see “Service Chaining” on page 319 .
Mirror	Select this check box to open a field where you can select from the list of configured services that you want to mirror in this policy. You can select a maximum of two services to mirror. For more information about mirroring, see “Configuring Traffic Analyzers and Packet Capture for Mirroring” on page 443 .

- When you are finished selecting the rules for this policy, click **Save**.

The policy you just defined is displayed in the **Policy** column.

Next, you can associate the policy to a network, see [“Associating a Network to a Policy—Juniper Networks Contrail” on page 246](#).

- See Also**
- [Associating a Network to a Policy—Juniper Networks Contrail on page 246](#)
 - [Creating Virtual Networks and Policies in Juniper Networks Contrail on page 239](#)

Associating a Network to a Policy—Juniper Networks Contrail

- [Associating Network Policies Overview on page 246](#)
- [Associating a Network Policy to a Network on page 246](#)

Associating Network Policies Overview

Contrail helps you create and manage virtual networks (VNs). By default, all traffic in a VN is isolated to that VN. Traffic can only leave a VN by means of network policies that are defined for the VN.

This procedure shows how to associate a network policy with a network, using the Juniper Networks Contrail interface.

If you did not associate an existing network policy when you created your virtual network, you can use the **Network Policy(s)** field in the **Edit Network** window, or you can use the **Associate Networks** field in the **Edit Policy** window to associate or disassociate network policies with networks. The following procedures demonstrate both methods.

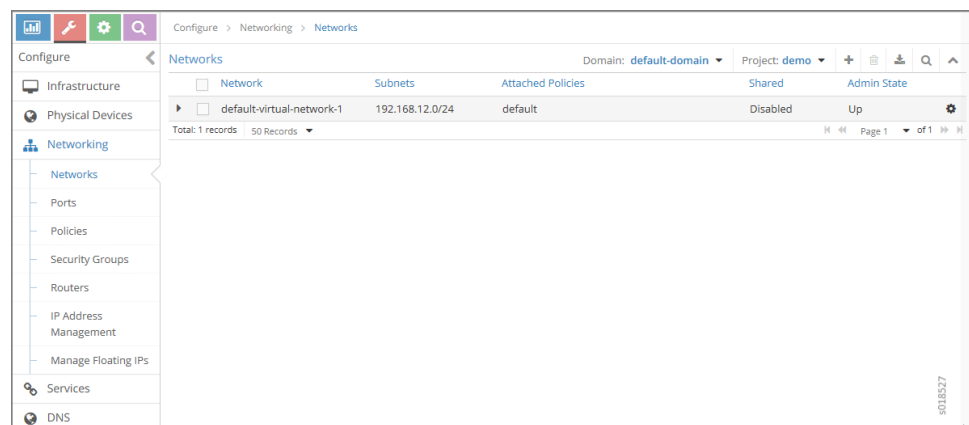
Associating a Network Policy to a Network

This procedure shows how to attach (associate) a network policy to a network when starting from the **Edit Network** window.

1. Select **Configure > Networking > Networks**; see [Figure 44 on page 246](#).

Make sure your project is the active project in the upper right.

Figure 44: Configure > Networking > Networks



2. Select the network you want to associate with a policy, then in the **Action** column, click the gear wheel icon and select **Edit**.

The **Edit Network** window for the selected network is displayed; see [Figure 45 on page 247](#).

Figure 45: Edit Network

Edit Network TestProject15-VN41E8cE1

Name: TestProject15-VN41E8cE1

Network Policy(s): vn1tovn2 (default-domain:admin) x

Subnets

IPAM	CIDR	Allocation Pools	Gateway	DNS	DHCP	
TestProject15-ipam9C87bE	220.123.235.0/24		<input checked="" type="checkbox"/> 220.123.235.1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	+ -

Host Routes

Advanced Options

Floating IP Pools

Route Targets

Cancel Save

- Click the **Network Policy(s)** field to show a list of existing policies, and then select a policy to associate with the selected network.

You can also disassociate a selected policy by clicking the - next to its name when it appears configured in the **Network Policy(s)** field.

- When you are finished, click **Save**, or click **Cancel** to undo your selections.

Related Documentation

- [Creating an Image and Launching a Virtual Machine on page 257](#)
- [Creating a Floating IP Address Pool and Allocating it to a Virtual Machine on page 262](#)

Creating Virtual Networks and Policies in OpenStack Contrail

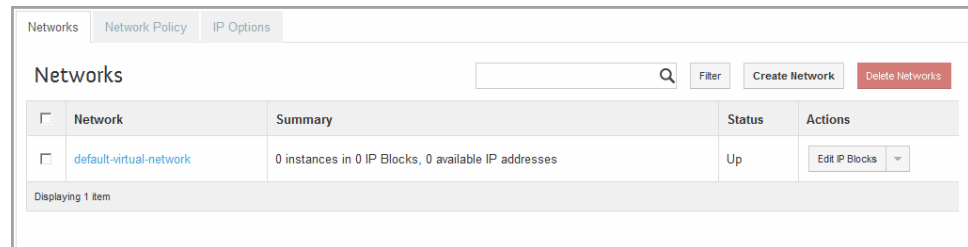
- [Creating a Virtual Network—OpenStack Contrail on page 248](#)
- [Deleting a Virtual Network with OpenStack Contrail on page 250](#)
- [Creating a Network Policy with OpenStack Contrail on page 252](#)
- [Associating a Network to a Policy—OpenStack Contrail on page 255](#)

Creating a Virtual Network—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. Select **Project > Other > Networking**. The **Networks** window is displayed. See [Figure 46 on page 248](#).

Figure 46: 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 47 on page 248](#) and [Figure 48 on page 249](#).

Figure 47: Create Network Window

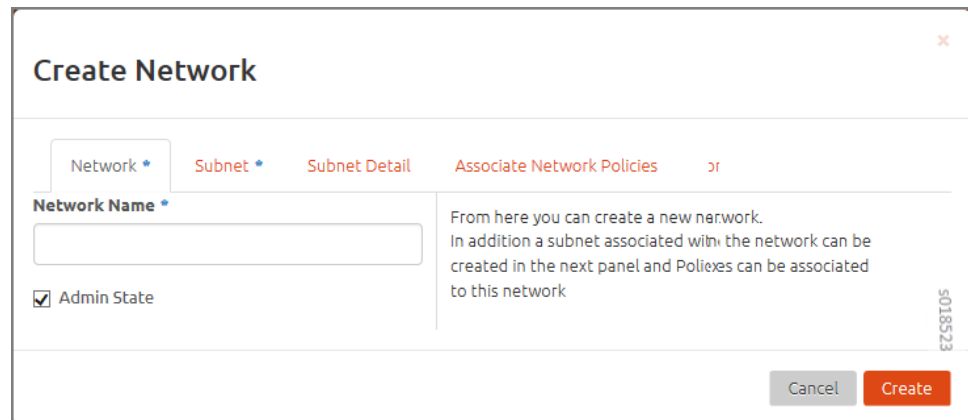


Figure 48: Create Network Window Subnet Tab

3. Click the **Network**, **Subnet**, **Subnet Detail**, and **Associate Network Policies** tabs to complete the fields in the **Create Network** window. See field descriptions in [Table 16 on page 249](#).

Table 16: Create Network Fields

Field	Description
Network Name	Enter a name for the network.
Subnet Name	Enter a name for the subnetwork.
IPAM	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.
Network Address	Enter the network address in CIDR format.
IP Version*	Select IPv4 or IPv6.

Table 16: Create Network Fields (continued)

Field	Description
Gateway IP	Optionally, enter an explicit gateway IP address for the IP address block. Check the Disable Gateway box if no gateway is to be used.
Network Policy	Any policies already created are listed. To select a policy, click the check box for the policy.

- Click the **Subnet Details** tab to specify the Allocation Pool, DNS Name Servers, and Host Routes.
- Click the **Associate Network Policies** tab to associate policies to the network.
- To save your network, click **Create Network**, or click **Cancel** to discard your work and start over.

Deleting a Virtual Network with OpenStack Contrail

You can delete any of the virtual networks in your system. However, you must first disassociate any virtual machines (instances) that are associated with that network.

To delete a virtual network when using OpenStack Contrail:

- To view virtual machines that are associated with a virtual network, in the OpenStack module, select **Project > Other > Networking**. The **Networks** window is displayed. See [Figure 49 on page 250](#).

Figure 49: OpenStack Networks

Networks

Network Policies

Network IPAMs

Networks

Filter

Filter

Create Network

Delete Networks

<input type="checkbox"/>	Name	Subnets Associated	Policies Associated	Shared	Admin State	Actions
<input type="checkbox"/>	default-virtual-network-1	192.168.12.0/24	default (demo)	No	UP	<div>Edit Network</div>

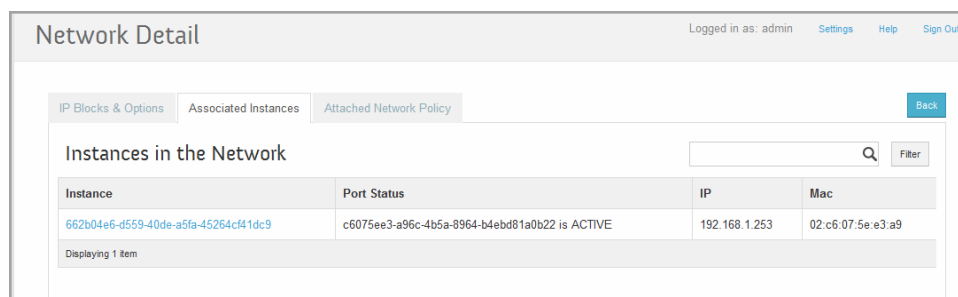
Displaying 1 item

5018511

- In the **Networks** window, select the network to be deleted.

The **Network Detail** window appears; see [Figure 50 on page 251](#).

Figure 50: OpenStack Network Detail, Associated Instances Tab

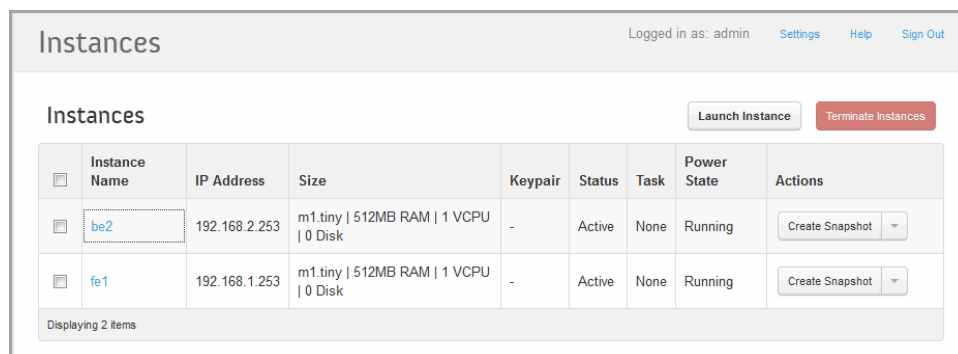


- Click the **Associated Instances** tab to see the instances associated with this network. Make note of the IP addresses of any instances that are associated with this network.

- On the **Project** tab, select **Instances**.

The **Instances** window appears, displaying the instances associated with the current project; see Figure 51 on page 251.

Figure 51: Instances



- On the **Instances** window, click the check box for any instance that is associated with the network that you want to delete, then click **Terminate Instances** to delete the instance.
- When all instances that are associated with the network to be deleted have been terminated, delete the network.

To delete a network, return to the **Networks** page (see Figure 49 on page 250), select the network to be deleted, then click **Delete Networks** in the upper right.

See Also • [Creating Virtual Networks and Policies in OpenStack Contrail on page 247](#)

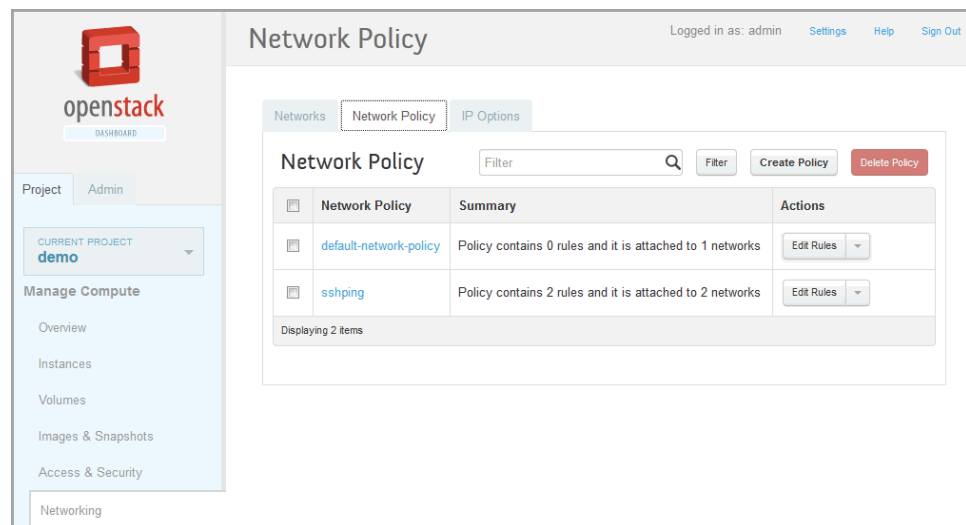
Creating a Network Policy with OpenStack Contrail

Contrail makes creating network traffic policies very simple. You work from the self-service user interface to define a policy, then define a rule or rules to be applied in that policy. You can define such parameters as the type and direction of traffic for the rule, the source and destination of that traffic, traffic originating from or destined for specific ports, the sequence in which to apply a rule, and so on.

To create a network policy when using OpenStack:

1. On the OpenStack dashboard, make sure your project is displayed in the **Current Project** box, click **Networking**, and then click the **Network Policy** tab to display the **Network Policy** window; see [Figure 52 on page 252](#).

Figure 52: Network Policy



2. Click **Create Policy** at the upper right.

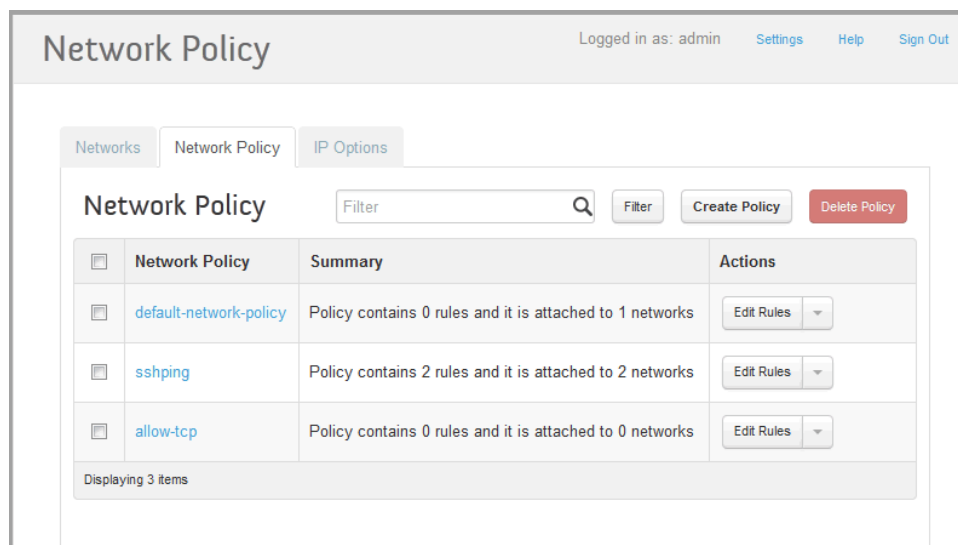
The **Create Network Policy** window is displayed; see [Figure 53 on page 252](#).

Figure 53: Create Network Policy

3. Enter a name and a description for this policy. Names cannot include spaces.
4. When finished, click **Create Policy** on the lower right.

Your policy is created and it appears in the **Network Policy** window; see [Figure 54 on page 253](#).

Figure 54: Network Policy



5. In the **Network Policy** window, select the check box for your new policy, then click **Edit Rules** for that policy.

The **Edit Policy Rules** window is displayed; see [Figure 55 on page 254](#).

Figure 55: Edit Policy Rules

6. Define the rules for your policy, using the guidelines in [Table 17 on page 254](#).

Table 17: Edit Policy Rules Fields

Field	Description
Policy Rules Details	This section of the window displays any rules that have already been created for this policy.
Id	Displays a sequential number identifier for each rule within a policy.
Rule Details	Displays a description of the rule on this line.
Actions	Available actions for the rule on this line appear in this column. Currently you can use the Delete button in this column to delete a rule.
Sequence Id	This field lets you define the order in which to apply the current rule. Select from a list: Last Rule, First Rule, After Rule .
Action	Define the action to take with traffic that matches the current rule. Select from a list: Pass, Deny .
Direction	Define the direction in which to apply the rule, for example, to traffic moving in and out, or only to traffic moving in one direction. Select from a list: Bidirectional, Unidirectional .
IP Protocol	Select from a list of available protocols (or ANY): ANY, TCP, UDP, ICMP .

Table 17: Edit Policy Rules Fields (continued)

Field	Description
Source Net	Select the source network for this rule. Choose Local (any network to which this policy is associated), Any (all networks created under the current project) or select from a list of all sources available displayed in the list, in the form: <i>domain-name:project-name:network-name</i> .
Source Ports	Accept traffic from any port or enter a specific port, a list of ports separated with commas, or a range of ports in the form <i>nnnn-nnnnn</i> .
Destination Net	Select the destination network for this rule. Choose Local (any network to which this policy is associated), Any (all networks created under the current project) or select from a list of all destinations available displayed in the list, in the form: <i>domain-name:project-name:network-name</i> .
Destination Ports	Send traffic to any port or enter a specific port, a list of ports separated with commas, or a range of ports in the form <i>nnnn-nnnnn</i> .

- After you select the rules for this policy, click **Add Rule** on the lower right of the **Edit Policy Rules** window.

Next, you can associate the policy to a network, see “[Associating a Network to a Policy—OpenStack Contrail](#)” on page 255.

See Also • [Associating a Network to a Policy—OpenStack Contrail on page 255](#)

Associating a Network to a Policy—OpenStack Contrail

- [Associating Network Policies Overview on page 256](#)
- [Associating a Network Policy to a Network on page 256](#)

Associating Network Policies Overview

Contrail helps you create and manage virtual networks (VNs). By default, all traffic in a VN is isolated to that VN. Traffic can only leave a VN by means of network policies that are defined for the VN.

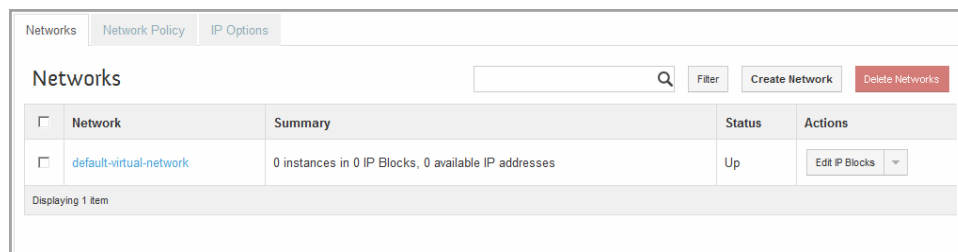
This procedure shows how to associate a network policy with a network when using OpenStack.

Associating a Network Policy to a Network

1. Using the OpenStack Networking module, select the **Project** tab and click **Networking**.

The **Networks** window is displayed; see [Figure 56 on page 256](#).

Figure 56: Networks Screen

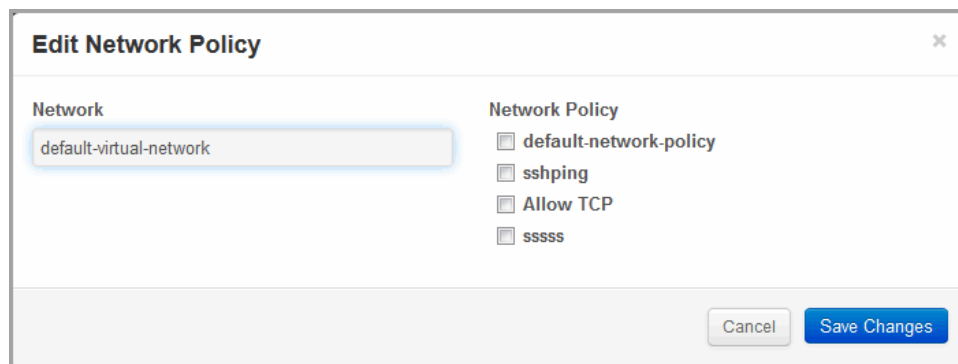


2. Click the check box to select the network you want to associate with a policy, then click the drop-down box in the **Actions** column and select **Edit Policy**.

The **Edit Network Policy** window is displayed; see [Figure 57 on page 256](#).

Available network policies are listed in the **Edit Network Policy** window.

Figure 57: Edit Network Policy



3. Click the check box of any policies to be associated with the selected network.
4. When finished, click **Save Changes**.

- Related Documentation
- [Creating an Image and Launching a Virtual Machine on page 257](#)
 - [Creating a Floating IP Address Pool and Allocating it to a Virtual Machine on page 262](#)

Creating an Image and Launching a Virtual Machine

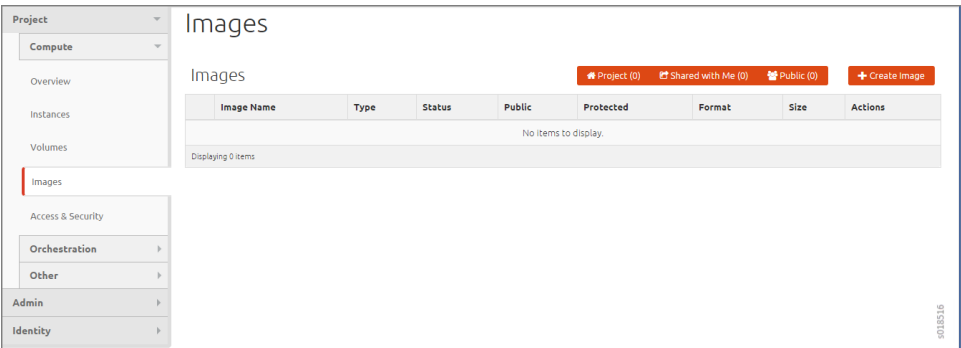
- [Creating an Image on page 257](#)
- [Launching a Virtual Machine \(Instance\) on page 260](#)

Creating an Image

You can use the OpenStack dashboard to specify an image to upload to the Image Service for a project in your system.

1. In OpenStack, select **Project > Compute > Images**. The Images window is displayed. See [Figure 58 on page 257](#).

Figure 58: 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 59 on page 258](#).

Figure 59: 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 18 on page 258](#) describes each of the fields on the screen.



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 18: Create An Image Fields

Field	Description
Name	Required field. Enter a name for this image.

Table 18: Create An Image Fields (continued)

Field	Description
Description	Enter a description for the image.
Image Source	<p>Select Image File or Image Location.</p> <p>If you select Image File, you are prompted to browse to the local location of the file.</p>
Image Location	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.
Format	<p>Required field. Select the format of the image from a list:</p> <p>AKI- Amazon Kernel Image AMI- Amazon Machine Image ARI- Amazon Ramdisk Image ISO- Optical Disk Image QCOW2-QEMU Emulator Raw VDI VHD VMDK</p>
Architecture	Enter the architecture.
Minimum Disk (GB)	Enter the minimum disk size required to boot the image. If you do not specify a size, the default is 0 (no minimum).
Minimum Ram (MB)	Enter the minimum RAM required to boot the image. If you do not specify a size, the default is 0 (no minimum).
Public	Check the box if this is a public image. Leave unchecked for a private image.
Protected	Check the box for a protected image.

- When you are finished, click **Create Image**.

Launching a Virtual Machine (Instance)

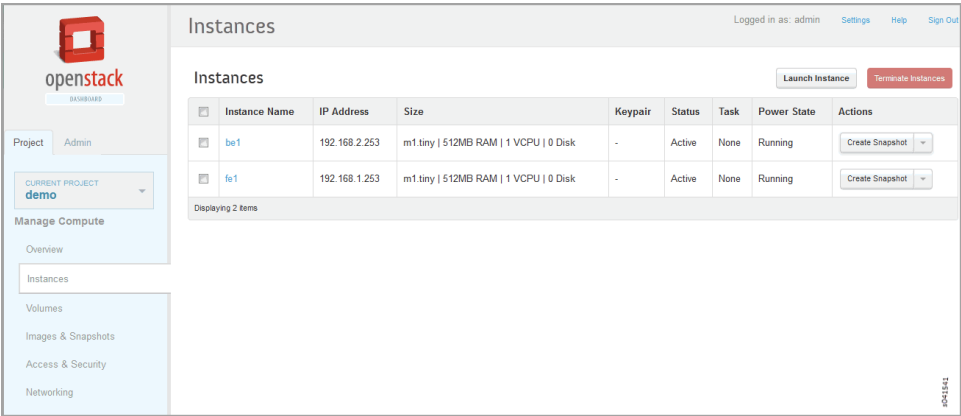
After you have created virtual networks for your project, you can create and launch virtual machines. Virtual networks (VNs) are populated with virtual machines (VMs), also called instances. A VM is a simulation of a physical machine, such as a workstation or a server, that runs on a host that supports virtualization. Many VMs can run on the same host, sharing the host's resources. A VM has its own operating system image that can be different from that of other VMs running on the same host.

To use the OpenStack module to define and launch VMs (instances):

1. On the OpenStack dashboard **Project** tab, make sure your project is selected in the left column in the **Current Project** box, then click **Instances**.

The **Instances** pane is displayed, displaying all instances (VMs) currently in the selected project; see [Figure 60 on page 260](#).

Figure 60: OpenStack Instances



2. To create and launch a new instance, click **Launch Instance** in the upper-right corner.

The **Launch Instance** window is displayed, where you can define and launch a new instance.

Figure 61: Launch Instance, Details Tab

Launch Instance

Details Access & Security Networking Volume Options Post-Creation

Instance Source
Image

Image
redmine-db

Instance Name
redmine-db-10

Flavor
m1.tiny

Instance Count
1

Compute Hostname
compute-nodea33

Specify the details for launching an instance.
The chart below shows the resources used by this project in relation to the project's quotas.

Flavor Details

Name	m1.tiny
VCPUs	1
Root Disk	0 GB
Ephemeral Disk	0 GB
Total Disk	0 GB
RAM	512 MB

Project Quotas

Number of Instances (4) 99,996 Available

Compute Host to launch Instance on 99,994 Available

Total RAM (9,216 MB) 9,990,784 MB Available

Cancel Launch

3. Make sure the **Details** tab is active; see Figure 61 on page 261, then define your instance using the fields shown in Table 19 on page 261

Table 19: Launch Instance Details Tab Fields

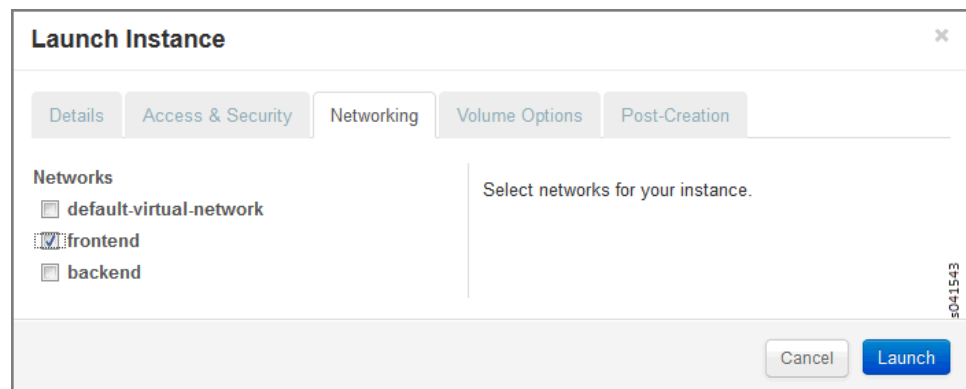
Field	Description
Instance Source	Select from the list the source type: Image or Snapshot .
Image	Select from a list the image to use for this instance. The images represent the operating systems and applications available for this project.
Instance Name	Enter a name for this instance
Flavor	From the list, select the OpenStack Flavor for this instance. Flavors provide general definitions for sizing VMs. The Flavor Details of the Flavor you select are displayed on the right column of this window.
Instance Count	Enter the number of instances you want to launch using the details defined in this window. On the right side column, Project Quotas displays the number of instances currently active and the number still available for this project.

Table 19: Launch Instance Details Tab Fields (continued)

Field	Description
Compute Hostname	To launch a VM on a specific compute node, enter the name of the compute node. This functionality is available only to administrators.

- Click the **Networking** tab in the **Launch Instance** window to identify one or more networks to associate with this instance; see [Figure 62 on page 262](#).

Figure 62: Launch Instance, Networking Tab



- After you finish defining this instance, click **Launch** at the lower right.
Your new VM instance is launched as part of your project.

See Also • [OpenStack documentation](#)

Related Documentation • [Creating Virtual Networks and Policies in Juniper Networks Contrail on page 239](#)
• [Creating Virtual Networks and Policies in OpenStack Contrail on page 247](#)

Creating a Floating IP Address Pool and Allocating it to a Virtual Machine

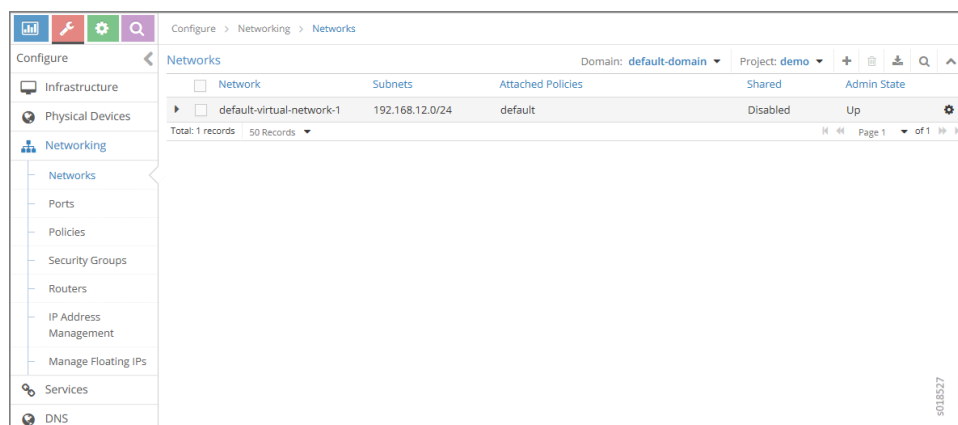
- [Creating a Floating IP Address Pool on page 263](#)
- [Allocating a Floating IP Address to a Virtual Machine on page 264](#)

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. You can configure floating IP address pools in project networks in Contrail, then allocate floating IP addresses from the pool to virtual machine instances in other virtual networks.

1. Select **Configure > Networking > Networks**; see [Figure 63 on page 263](#). Make sure your project is the active project in the upper right.

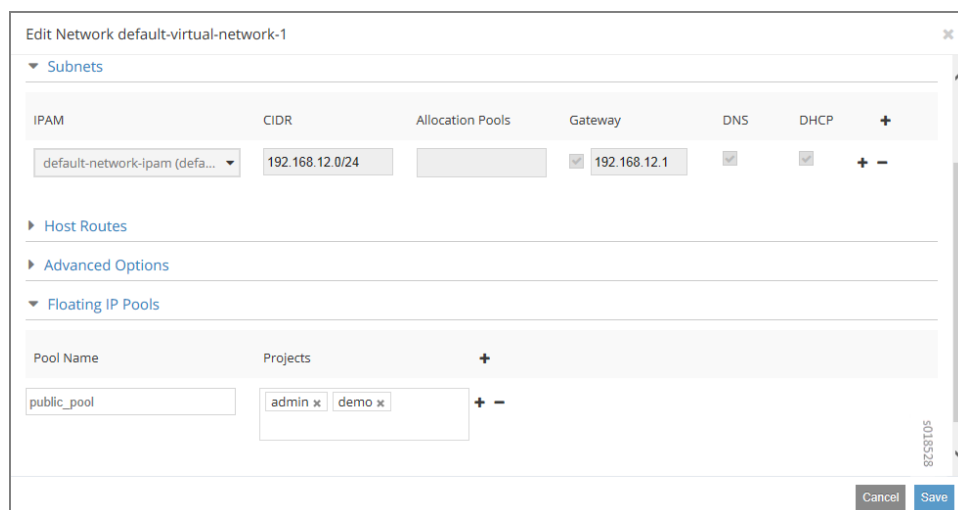
Figure 63: Configure > Networking > Networks



2. Click the network you want to associate with a floating IP pool, then in the **Action** column, click the action icon and select **Edit**.

The **Edit Network** window for the selected network is displayed; see [Figure 64 on page 263](#).

Figure 64: Edit Network



3. In the **Floating IP Pools** section, click the **Pool Name** field, enter a name for your floating IP pool, and click the + (plus sign) to add the IP pool to the table below the field.
 - Multiple floating IP pools can be created at the same time.
 - A floating IP pool can be associated to multiple projects.
4. Click **Save** to create the floating IP address pool, or click **Cancel** to remove your work and start over.

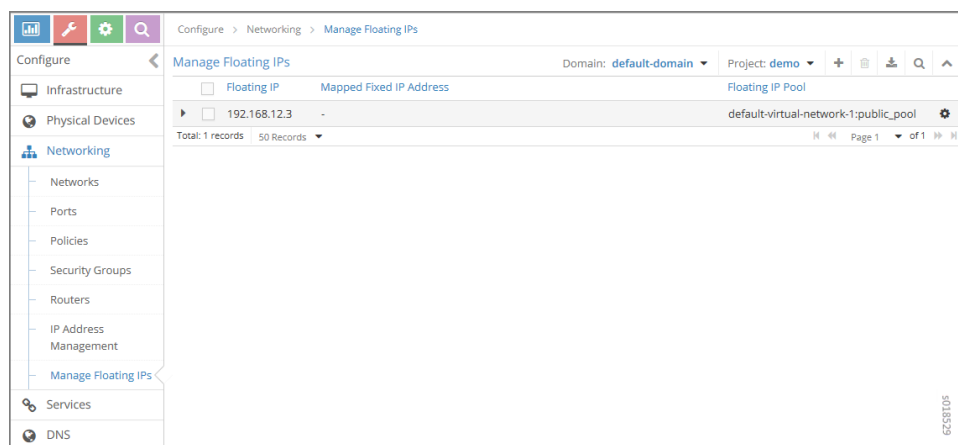
Allocating a Floating IP Address to a Virtual Machine

To allocate a floating IP address pool to a VM instance:

1. In the Contrail Web user interface, select **Configure > Networking > Manage Floating IPs**. The the Manage Floating IPs window is displayed.

Make sure your project is displayed (active) in the upper right. See [Figure 65 on page 264](#).

Figure 65: Manage Floating IPs



2. In the Manage Floating IPs window, click +.

The **Allocate Floating IP** window is displayed; see [Figure 66 on page 265](#).

Figure 66: Allocate Floating IP Window

The figure displays two instances of the 'Allocate Floating IP' window. Both windows have a title bar with a close button (X) and a vertical ID 's018530' on the right side.

Top Window (Dynamic Allocation):

- Floating IP Pool:** demo:default-virtual-network-1:public_pool (192.168.12.0/24)
- Allocation Type:** Dynamic
- Number of IP Addresses:** 1
- Buttons:** Cancel, Save

Bottom Window (Specific IP Allocation):

- Floating IP Pool:** demo:default-virtual-network-1:public_pool (192.168.12.0/24)
- Allocation Type:** Specific IP
- IP Address:** 192.168.12.5
- Buttons:** Cancel, Allocate

3. Select the name of the floating IP pool from the Floating IP Pool list . The floating IP pool is shared among multiple projects.
4. Select either **Dynamic** or **Specific IP** from the Allocation Type list.
5. If Dynamic is selected, type the number of IP addresses. If Specific IP is selected, type the specific IP address.
6. If Dynamic is selected, click **Save**. If Specific IP is selected, click **Allocate**.
7. After the floating IP pool has been allocated, you can associate it to or disassociate it from instance addresses. In the Manage Floating IPs window, select the floating IP pool you want, then click the gear wheel icon and select **Associate Port** or **Disassociate**.
If you select Associate Port, the **Associate Floating IP** window is displayed; see [Figure 67 on page 266](#)

Figure 67: Associate Floating IP

8. In the **Port** field, select the UUID of the VM instance to associate with the selected floating IP pool from the Port list, and click **Save**.

See Also • [Creating a Floating IP Address Pool and Allocating it to a Virtual Machine on page 262](#)

- Related Documentation**
- [Creating Projects in OpenStack for Configuring Tenants in Contrail on page 238](#)
 - [Creating Virtual Networks and Policies in Juniper Networks Contrail on page 239](#)
 - [Creating Virtual Networks and Policies in OpenStack Contrail on page 247](#)
 - [Creating an Image and Launching a Virtual Machine on page 257](#)

Using Security Groups with Virtual Machines (Instances)

- [Security Groups Overview on page 266](#)
- [Creating Security Groups and Adding Rules on page 266](#)

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

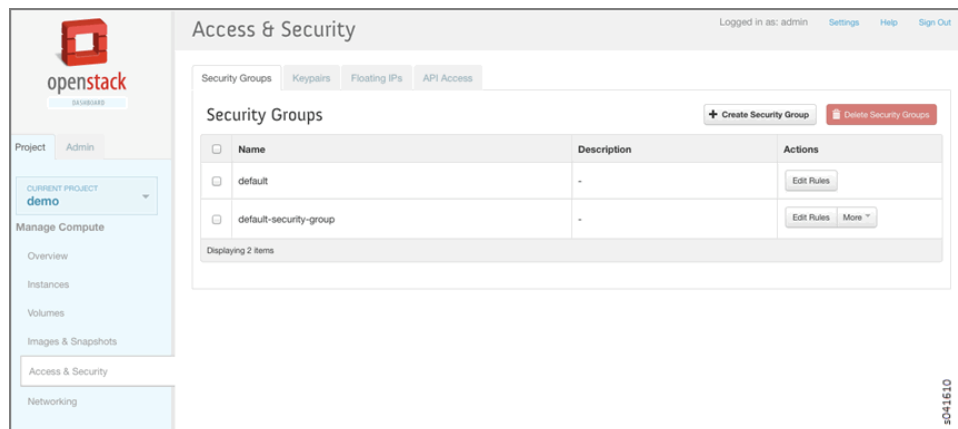
A default security group is created for each project. You can add security rules to the default security group and you can create additional security groups and add rules to them. The security groups are then associated with a VM, when the VM is launched or at a later date.

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 68 on page 267](#).

Figure 68: 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 69 on page 267](#). Any rules already associated with the security group are listed.

Figure 69: Edit Security Group Rules



3. Click **Add Rule** to add a new rule; see [Figure 70 on page 268](#).

Figure 70: Add Rule

Add Rule

IP Protocol

Type

Code

Source

Description:
 Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:
Protocol: You must specify the desired IP protocol to which this rule will apply; the options are TCP, UDP, or ICMP.
Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.
Source: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Table 20: 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 From Port field.
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 71 on page 269](#).

Each new security group has a unique 32-bit security group ID and an ACL is associated with the configured rules.

Figure 71: Create Security Group

5. When an instance is launched, there is an opportunity to associate a security group; see [Figure 72 on page 269](#).

In the **Security Groups** list, select the security group name to associate with the instance.

Figure 72: 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 Contrail Release 2.0 and later, support for IPv6 overlay networks is provided.

- [Overview: IPv6 Networks in Contrail on page 270](#)
- [Creating IPv6 Virtual Networks in Contrail on page 270](#)
- [Adding IPv6 Peers on page 271](#)

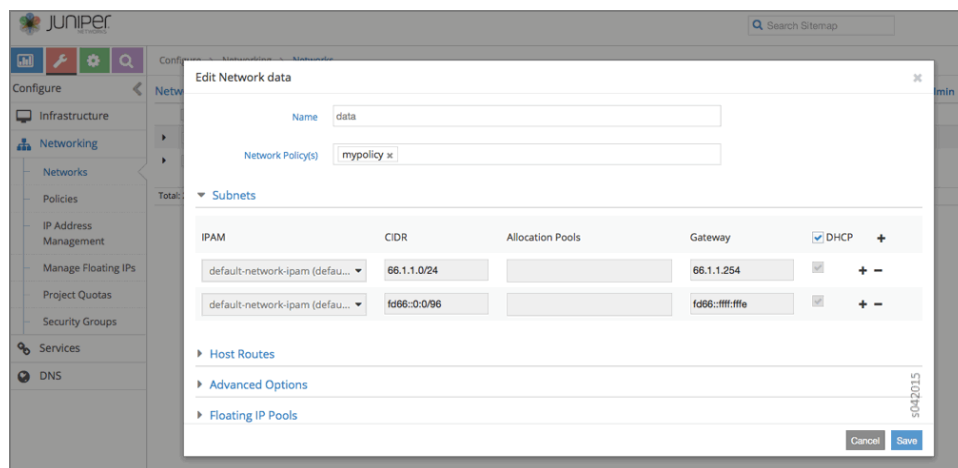
Overview: IPv6 Networks in Contrail

In Contrail Release 2.0 and later, support for IPv6 overlay networks is provided, including:

- Configuring IPv6 subnets from the Contrail user interface or by using Neutron APIs
- IPv6 address assignment to virtual machine interfaces over DHCPv6
- IPv6 forwarding in overlay networks between virtual machines, and between virtual machines and BGP peers
- IPv6 to-VPN peering with other BGP peers
- IPv6 forwarding in Layer 2-only networks
- IPv6 interface static routes

Creating IPv6 Virtual Networks in Contrail

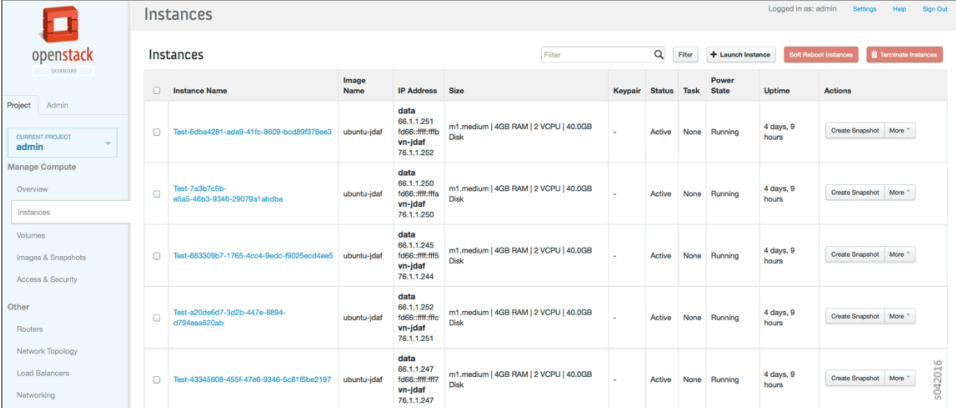
You can create an IPv6 virtual network from the Contrail user interface in the same way you create an IPv4 virtual network. When you create a new virtual network by selecting **Configure > Networking > Networks**, the Edit fields accept IPv6 addresses, as shown in the following image.



Address Assignments

When virtual machines are launched with an IPv6 virtual network created in the Contrail user interface, the virtual machine interfaces get assigned addresses from all the families configured in the virtual network.

The following is a sample of IPv6 instances with address assignments, as listed in the OpenStack Horizon user interface.



Instance Name	Image Name	IP Address	Size	Keypair	Status	Task	Power State	Uptime	Actions
Test-6d8a4261-a5a9-415e-8009-bcd89578ea3	ubuntu-jdaff	data 66.1.1.251 5696:::8f0c vn-jdaff 76.1.1.252	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	Create Snapshot More
Test-7a3b7c5b-e5a9-48a3-9346-29079a7ab0ba	ubuntu-jdaff	data 66.1.1.250 5696:::8f0c vn-jdaff 76.1.1.250	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	Create Snapshot More
Test-663309b7-1785-4cc4-8edc-f9025ac04ee0	ubuntu-jdaff	data 66.1.1.245 5696:::8f0c vn-jdaff 76.1.1.244	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	Create Snapshot More
Test-a205a6a7-3a2b-447e-8894-e794eead50ba	ubuntu-jdaff	data 66.1.1.252 5696:::8f0c vn-jdaff 76.1.1.251	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	Create Snapshot More
Test-43343608-455a-47a6-9346-5a815ba2197	ubuntu-jdaff	data 66.1.1.247 5696:::8f0c vn-jdaff 76.1.1.247	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	Create Snapshot More

Enabling DHCPv6 In Virtual Machines

To allow IPv6 address assignment using DHCPv6, the virtual machine network interface configuration must be updated appropriately.

For example, to enable DHCPv6 for Ubuntu-based virtual machines, add the following line in the `/etc/network/interfaces` file:

```
iface eht0 inet6 dhcp
```

Also, `dhclient -6` can be run from within the virtual machine to get IPv6 addresses using DHCPv6.

Adding IPv6 Peers

The procedure to add an IPv6 BGP peer in Contrail is similar to adding an IPv4 peer. Select **Configure > Infrastructure > BGP Peers**, include `inet6-vpn` in the Address Family list to allow advertisement of IPv6 addresses.

A sample is shown in the following.



NOTE: Additional configuration is required on the peer router to allow inet6-vpn peering.

Configuring EVPN and VXLAN

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 on page 274](#)
- [Configuring Forwarding on page 276](#)
- [Configuring the VXLAN Identifier on page 277](#)
- [Configuring Encapsulation Methods on page 278](#)

Configuring the VXLAN Identifier Mode

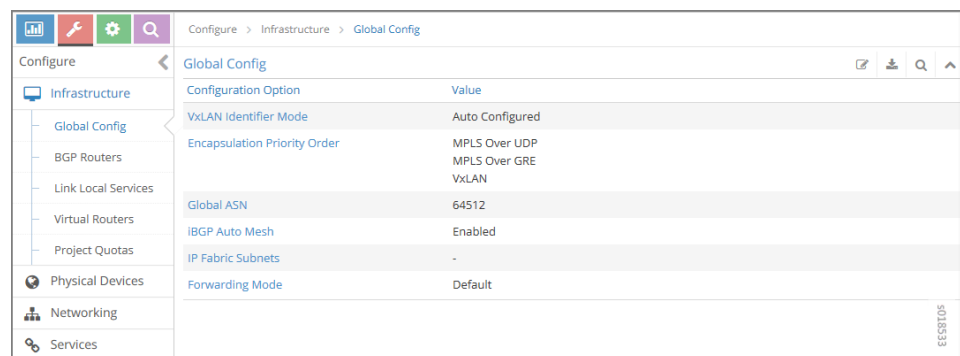
You can configure the global VXLAN identifier mode to select an auto-generated VNID or a user-generated VXLAN ID, either through the Contrail Web UI or by modifying a python file.

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 73: Global Config Window for VXLAN ID



2. Click the edit icon  .

The Edit Global Config window is displayed as shown in [Figure 74 on page 275](#).

Figure 74: Edit Global Config Window for VXLAN Identifier Mode

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

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

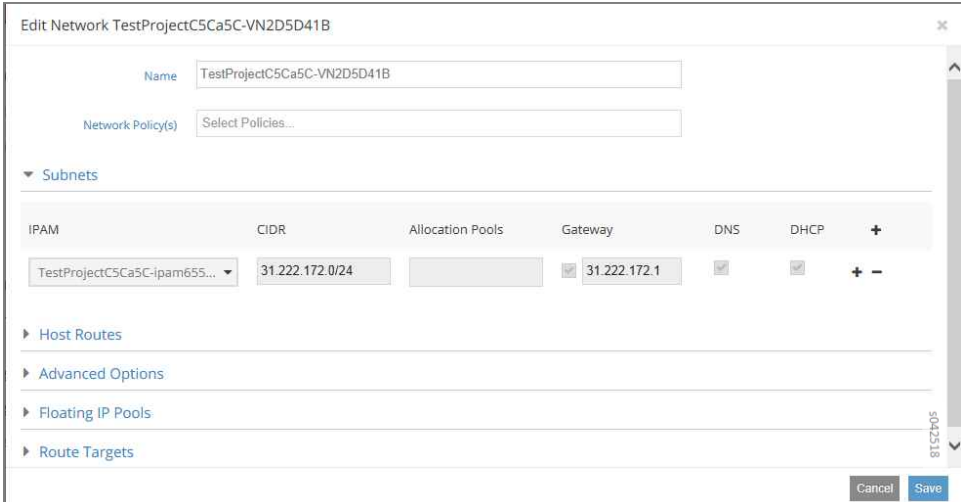
In Contrail, the default forwarding mode is enabled for fallback bridging (IP FIB and MAC FIB). The mode can be changed, either through the Contrail Web UI or by using python provisioning commands.

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 75 on page 276](#).

Figure 75: 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 < l2_l3| l2 >
```

Options:


l2_l3 = Enable IP FIB and MAC FIB (fallback bridging)

l2 = 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 76 on page 278](#).

Figure 76: Edit Network Window for VXLAN Identifier

Figure 76 shows the 'Edit Network default-virtual-network-1' window. The 'Advanced Options' tab is active. The 'Admin State' is set to 'Up'. There are checkboxes for 'Shared' and 'External'. The 'DNS Servers' section has a 'DNS Servers' button with a plus icon. The 'Forwarding Mode' is set to 'L2 and L3'. The 'VxLAN Identifier' is set to '0-1048575'. There are checkboxes for 'Allow Transit', 'Flood unknown unicast', and 'Extend To Physical Router(s)'. At the bottom right are 'Cancel' and 'Save' buttons.

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 77 on page 279](#).

Figure 77: Edit Global Config Window for Encapsulation Priority Order

Under Encapsulation Priority Order select one of the following:

- MPLS over UDP
- MPLS over GRE
- VxLAN

Click the + plus symbol to the right of the first priority to add a second priority or third priority.

Use the following procedure to change the default encapsulation method to VXLAN by editing the `encap.py` file.



NOTE: VXLAN is *only* supported for EVPN unicast. It is not supported for IP traffic or multicast traffic. VXLAN priority and presence in the `encap.py` file or configured in the Web UI is ignored for traffic not supported by VXLAN.

To set the priority of encapsulation methods to VXLAN:

1. Modify the `encap.py` file found in the `/opt/contrail/utils/` directory.

The default encapsulation line is:

```
encap_obj=EncapsulationPrioritiesType(encapsulation=['MPLSoUDP','MPLSoGRE'])
```

Modify the line to:

```
encap_obj=EncapsulationPrioritiesType(encapsulation=['VXLAN',  
'MPLSoUDP','MPLSoGRE'])
```

2. After the status is modified, execute the following script:

```
python encap_set.py <add|update|delete> <username> <password> <tenant_name>  
<config_node_ip>
```

The configuration is applied globally for all virtual networks.

Example of Deploying a Multi-Tier Web Application Using Contrail

- [Example: Deploying a Multi-Tier Web Application on page 281](#)
- [Sample Network Configuration for Devices for Simple Tiered Web Application on page 287](#)

Example: Deploying a Multi-Tier Web Application

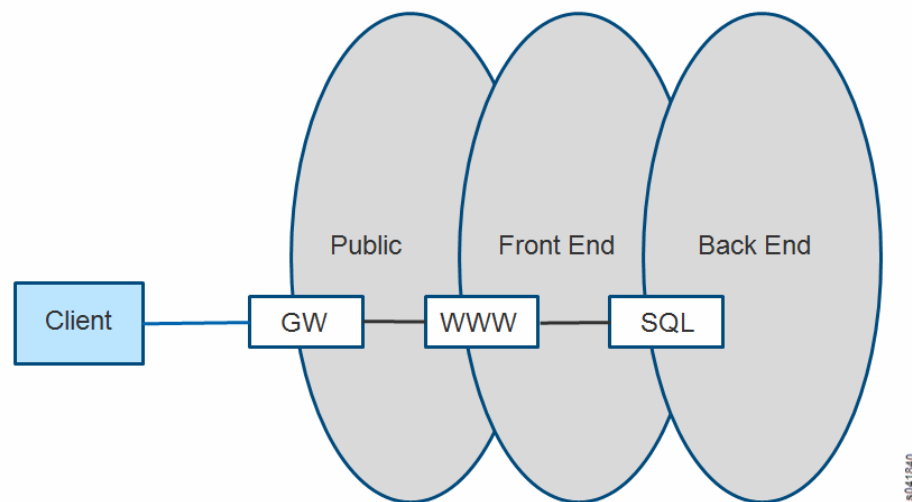
- [Multi-Tier Web Application Overview on page 281](#)
- [Example: Setting Up Virtual Networks for a Simple Tiered Web Application on page 282](#)
- [Verifying the Multi-Tier Web Application on page 284](#)
- [Sample Addressing Scheme for Simple Tiered Web Application on page 285](#)
- [Sample Physical Topology for Simple Tiered Web Application on page 286](#)
- [Sample Physical Topology Addressing on page 286](#)

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 78 on page 282](#).

Figure 78: 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” on page 238](#).

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 Virtual Networks and Policies in OpenStack Contrail” on page 247 or “Creating Virtual Networks and Policies in Juniper Networks Contrail” on page 239.

3. Create a floating IP pool named **public_pool** for the **public** network within the **demo** project; see Figure 79 on page 283.

Figure 79: Create Floating IP Pool

The screenshot shows the 'Edit Network public' dialog box. It contains the following fields and options:

- Network Name:** public
- Network Policy(s):** Select Policies...
- Address Management:** default-network... (dropdown), xxx.xxx.xxx.xxx/xx (text), + - (buttons)
- Floating IP Pools:** public_pool (text), demo x (dropdown), + - (buttons)
- Pool Name:** admin (dropdown)
- Buttons:** Cancel, Save

4. Allocate the floating IP pool **public_pool** to the **demo** project; see Figure 80 on page 283.

Figure 80: Allocate Floating IP

The screenshot shows the 'Allocate Floating IP' dialog box. It contains the following fields and options:

- Floating IP Pool:** public:public_pool (dropdown)
- Buttons:** Cancel, Save

5. Verify that the floating IP pool has been allocated; see **Configure > Networking > Allocate Floating IPs**.

For more information; see [“Creating a Floating IP Address Pool and Allocating it to a Virtual Machine” on page 262](#) and [“Allocating a Floating IP Address to a Virtual Machine” on page 264](#).

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.

For more information; see [“Creating a Network Policy with Juniper Networks Contrail” on page 244](#), [“Associating a Network to a Policy—Juniper Networks Contrail” on page 246](#), or [“Creating a Network Policy—OpenStack Contrail” on page 252](#), and [“Associating a Network to a Policy—OpenStack Contrail” on page 255](#).

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.

For more information; see [“Launching a Virtual Machine \(Instance\)” on page 260](#).

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 21 on page 285](#) as a guide for addressing devices in the simple tiered web example.

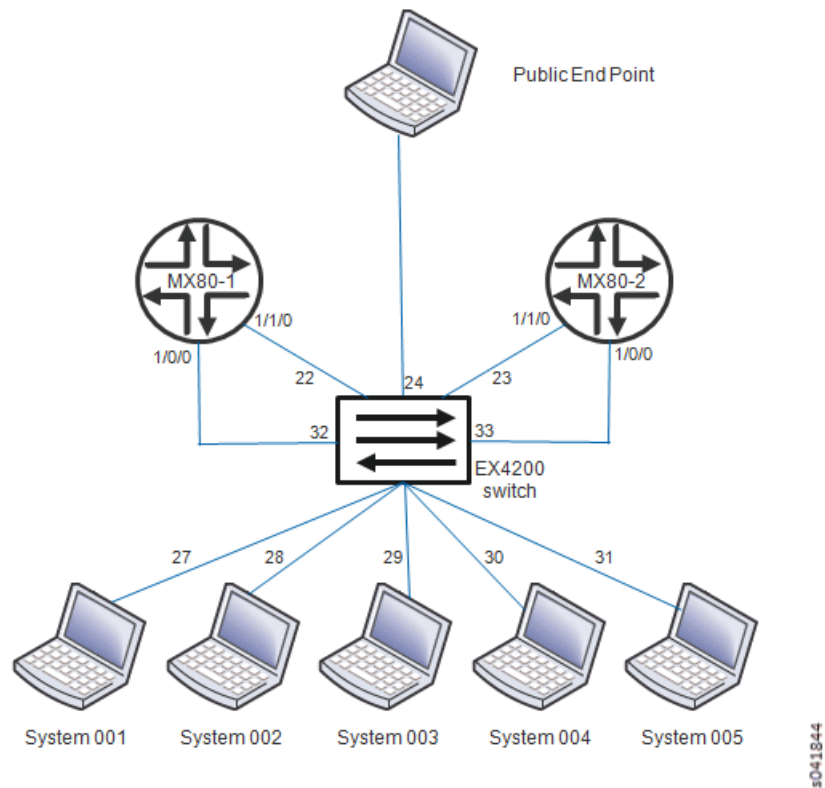
Table 21: Sample Addressing Scheme for Example

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 81 on page 286 provides a guideline diagram for the physical topology for the simple tiered web application example.

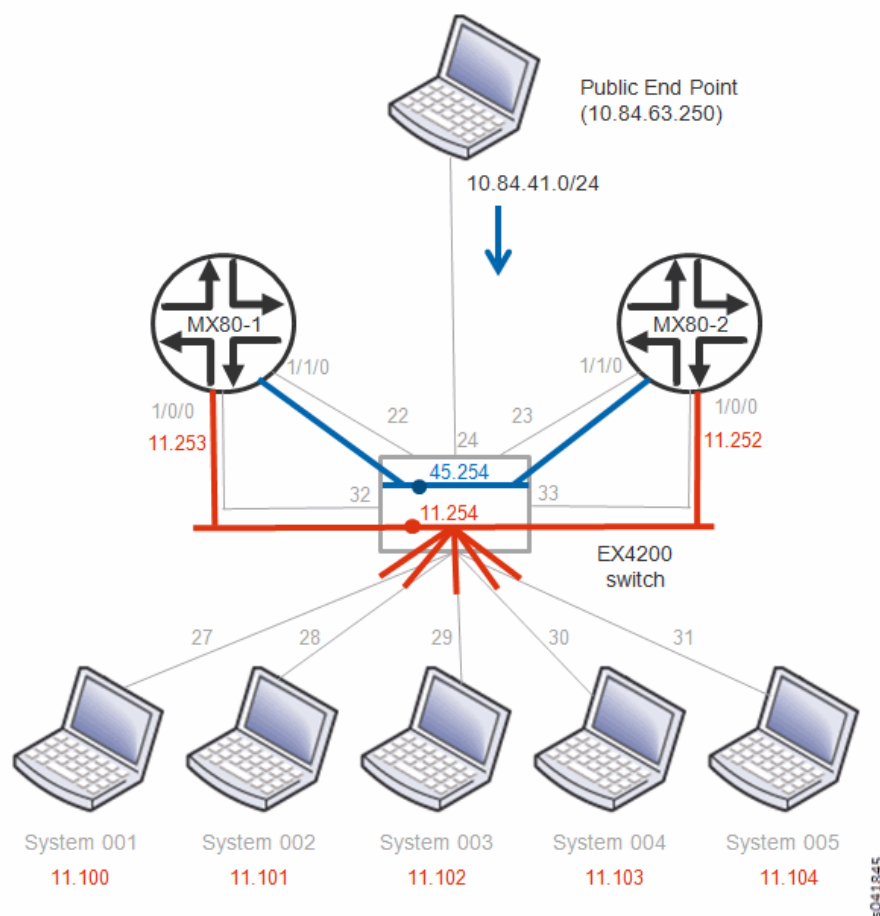
Figure 81: Sample Physical Topology for Simple Tiered Web Application



Sample Physical Topology Addressing

Figure 82 on page 287 provides a guideline diagram for addressing the physical topology for the simple tiered web application example.

Figure 82: Sample Physical Topology Addressing



See Also • [Sample Network Configuration for Devices for Simple Tiered Web Application on page 287](#)

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](#)” on page 281. 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 "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 10.84.11.252/24;
            }
        }
    }
    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 regress {
            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;
    }
  }
}
```

CHAPTER 12

Configuring Services

- [Configuring DNS Servers on page 293](#)
- [Configuring Discovery Service on page 302](#)
- [Support for Multicast on page 306](#)
- [Using Static Routes with Services on page 308](#)
- [Configuring Metadata Service on page 312](#)
- [BGP as a Service on page 313](#)

Configuring DNS Servers

- [DNS Overview on page 293](#)
- [Defining Multiple Virtual Domain Name Servers on page 294](#)
- [IPAM and Virtual DNS on page 294](#)
- [DNS Record Types on page 295](#)
- [Configuring DNS Using the Interface on page 296](#)
- [Configuring DNS Using Scripts on page 301](#)

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.

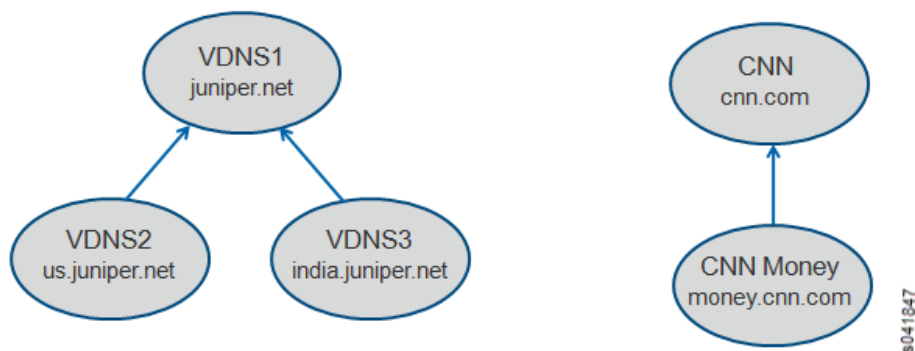
The following are the key attributes of domain name service in a virtual world:

- 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 83 on page 294](#) shows examples of virtual DNS servers defined in **default-domain**, providing the name service for the DNS domains indicated.

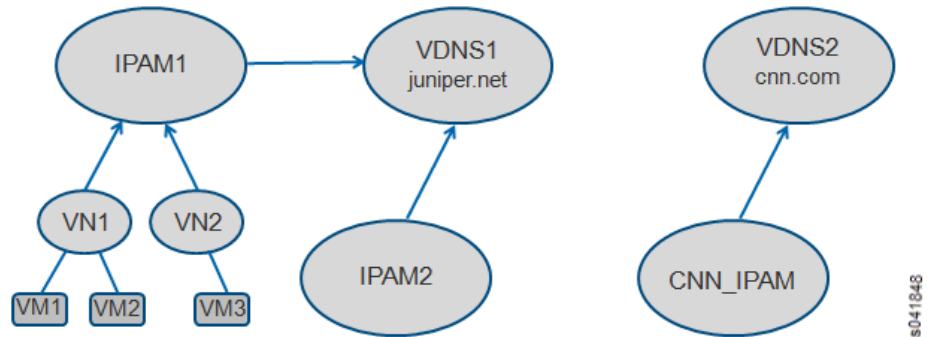
Figure 83: DNS Servers Examples



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 84 on page 295](#)

Figure 84: IPAM and Virtual DNS



DNS Record Types

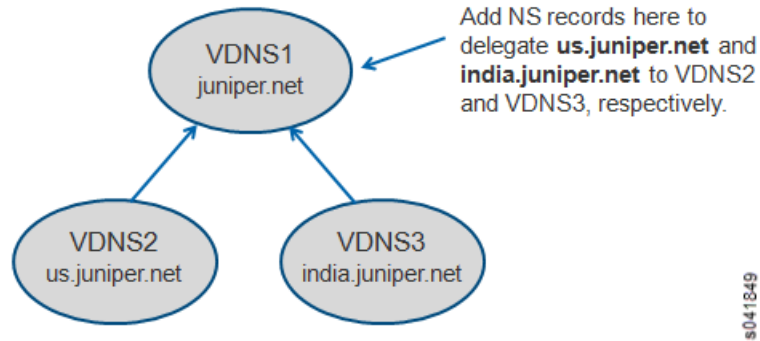
DNS records can be added statically. DNS record types **A**, **CNAME**, **PTR**, and **NS** are currently supported in the system. Each record includes the type, class (IN), name, data, and TTL values. See [Table 22 on page 295](#) for descriptions of the record types.

Table 22: DNS Record Types Supported

DNS Record Type	Description
A	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.
CNAME	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.
PTR	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.
NS	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 85 on page 296](#) shows an example usage for the DNS record type of **NS**.

Figure 85: 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 86 on page 296](#).

Figure 86: Configure DNS Records

Configure > DNS > Servers

Search

Configure DNS Records

default-domain admin

Configure Virtual DNS

Create Delete

Virtual DNS Name	DNS Domain Name	Next DNS Server
No Data Found		

DNS Records Associated IPAMs

DNS Records of {{dnsname}}

Add Record Delete

Name	Type : Data	TTL (secs)	Class
------	-------------	------------	-------

#041850

2. To add a new DNS server, click the **Create** button.

Enter DNS server information in the **Add DNS** window; see [Figure 87 on page 297](#)

Figure 87: 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

Complete the fields for the new server; see [Table 23 on page 297](#).

Table 23: 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 TTL 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 None .
Load Balancing Order	Select the load-balancing order from a list— Random , Fixed , Round Robin . When a name has multiple records matching, the configured record order determines the order in which the records are sent in the response. Select Random to have the records sent in random order. Select Fixed to have records sent in the order of creation. Select Round Robin to have the record order cycled for each request to the record.
OK	Click OK to create the record.
Cancel	Click Cancel 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 88 on page 298](#).

Figure 88: Add DNS Record

4. Complete the fields for the new record; see [Table 24 on page 298](#).

Table 24: Add DNS Record Fields

Field	Description
Record Name	Enter a name for this record.
Type	Select the record type from a list— A , CNAME , PTR , NS .
IP Address	Enter the IP address for the location for this record.
Class	Select the record class from a list— IN is the default.
Time To Live	Enter the TTL in seconds.
OK	Click OK to create the record.
Cancel	Click Cancel 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 89 on page 299](#).

Figure 89: Associate IPAMs to DNS

Complete the IPAM associations, using the field descriptions in [Table 25 on page 299](#).

Table 25: Associate IPAMs to DNS Fields

Field	Description
Associate to All IPAMs	Select this box to associate the selected DNS server to all available IPAMs.
Available IPAMs	This column displays the currently available IPAMs.
Associated IPAMs	This column displays the IPAMs currently associated with the selected DNS server.
>>	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.
<<	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.
OK	Click OK to commit the changes indicated in the window.
Cancel	Click Cancel to clear all entries and start over.

- Use the **IP Address Management** page (**Configure > Networking > IP Address Management**); see [Figure 90 on page 300](#)) 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 90: Configure IP Address Management

- 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 91 on page 300](#).

Figure 91: DNS Server

- In the first field, select the **DNS Method** from a list (**None**, **Default DNS**, **Tenant DNS**, **Virtual DNS**; see [Table 26 on page 300](#).

Table 26: DNS Modes

DNS Mode	Description
None	Select None 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 26: DNS Modes (continued)

DNS Mode	Description
Tenant	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.
Virtual DNS	Configure this mode to support virtual DNS servers (VDNS) to resolve the DNS requests from the VMs. Each IPAM can have a virtual DNS server configured in this mode.

9. Complete the remaining fields on this page, and click **OK** to commit the changes, or click **Cancel** to clear the fields and start over.

Configuring DNS Using Scripts

DNS can be configured via the user interface or by using scripts that are available in the `opt/contrail/utils` directory. The scripts are described in [Table 27 on page 301](#).



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 27: DNS Scripts

Action	Script
Add a virtual DNS server	Script: <code>add_virtual_dns.py</code> 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>
Delete a virtual DNS server	Script: <code>del_virtual_dns_record.py</code> 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>
Add a DNS record	Script: <code>add_virtual_dns_record.py</code> 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>

Table 27: DNS Scripts (continued)

Action	Script
Delete a DNS record	Script: <code>del_virtual_dns_record.py</code> 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>
Associate a virtual DNS server with an IPAM	Script: <code>associate_virtual_dns.py</code> 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>
Disassociate a virtual DNS server with an IPAM	Script: <code>disassociate_virtual_dns.py</code> 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>

Configuring Discovery Service

The Contrail Discovery Service publishes the IP address and port of the multiple components of the configuration node. The system runs multiple instances of each process for high availability and load balancing purposes.

- [Contrail Discovery Service Introduction on page 302](#)
- [Discovery Service Registration and Publishing on page 303](#)
- [Discovery Service Subscription on page 303](#)
- [Discovery Service REST API on page 304](#)
- [Discovery Service Heartbeats on page 306](#)
- [Discovery Service Internal Databases on page 306](#)
- [Discovery Service Client Library on page 306](#)
- [Discovery Service Debugging on page 306](#)

Contrail Discovery Service Introduction

The following ports are used by the discovery service.

- API port : 5998 TCP
- Hearbeat port: 5998 TCP

To display the publishers, connect to the `http://discovery-server-ip:5998/services` URL.

To display the subscribers, connect to the `http://discovery-server-ip:5998/clients` URL.

The Contrail Discovery Service uses the following configuration file and log file:

```
/etc/contrail/discovery.conf
/var/log/contrail/discovery.log
```

Discovery Service Registration and Publishing

The Discovery Service publishers send registration requests to the discovery server using a REST API.

The Discovery Service publishers send periodic heartbeat to the discovery server. The default interval for the heartbeat is 5 seconds.

If three successive heartbeats are missed, the Discovery Service is marked down.

The Discovery Service status is maintained internally. It indicates if the service is up or down based on the received heartbeat messages.

The discovery server currently supports three policies for selecting what information to return. The three policies are:

Load Balance—The service is returned based on the in-use count (how many subscribers are currently using the service).

Round Robin—The service is assigned based on a timestamp. The earliest (oldest) publisher is selected for the next assignment.

Fixed—An ordered list of available servers is always sent. If a service goes offline and comes back again, that service moves to the bottom of the list.

The three policies are configured in the `/etc/contrail/discovery.cfg` file under the service type section.

The response to a publish request is a cookie that must be sent back in the heartbeats.

Discovery Service Subscription

Clients that need service send requests to the discovery server using a REST API.

The client can specify how many instances of a service to be returned. The default is 1. If the requested number of instances is 0, the information about all of the publishers of that service type is returned. Use this to display all the providers of a particular service.

A client is identified by a token (uuid). The token is typically sent as part of a subscription request. The client information is removed from the discovery server database when the time to live (TTL) expires.

A response to a client includes a TTL value. When the TTL expires, the client refreshes the information by sending another subscription request. The TTL sent to the client is a random value, in the range of 5 to 30 minutes.

If a service is overloaded and a new one is started, the new clients are automatically assigned a new service instance. To spray the new servers to the existing subscribers, use the `discovery_cli.py` file to reassign them on demand.

Clients find the discovery service by using the configured IP address and port.

Discovery Service REST API

A REST API is available for registering and publishing the Contrail Discovery Server.

The following values are defined in the Contrail Discovery Server file:

- **POST: /publish or POST /publish/<publisher-id>**
- Content type: application/json or application/xml
- Body: information to be published (service type and data)

The following example shows the REST API for registering and publishing the discovery service

```
JSON simple:
{
  "control-node": {"ip_addr": "192.168.2.0", "port":1682 }
}
```

```
JSON verbose:
{
  "service-type" : "foobar",
  "foobar" : {"ip_addr": "192.168.2.0", "port":1682 }
}
```

```
XML simple:
<foobar2>
  <ip-addr>1.1.1.1</ip-addr>
  <port>4289</port>
</foobar2>
```

```
XML verbose:
<publish>
  <foobar2>
    <ip-addr>1.1.1.1</ip-addr>
    <port>4289</port>
  </foobar2>
  <oper-state>down</oper-state>
  <service-type>foobar2</service-type>
</publish>
```

```
JSON Response: {"cookie": c76716813f4b}
```

```
XML Response: <response><cookie>c76716813f4b</cookie></response>
```

The following fields are allowed in the body of the file:

service-type—Name of the service to publish

admin-state—Up or down state

remote-addr—IP address of the client

remote-version—Version number of the client

remote-name—Hostname of the client

oper-state—Each published service can set the oper-state up or down based on its internal state. You can display the reason the oper-state is up or down using the port 5998 URL.

A REST API is available for subscribing to the Contrail Discovery Server.

The following values are defined in the Contrail Discovery Server **discovery_cli.py** file:

- POST http://discovery-server-ip:5998/subscribe
- Content-Type: application/json or application/xml
- Body: Service type, instance count, client ID

The following example shows the REST API for subscribing to the discovery service.

```
JSON: {
  "service": "control-node",
  "instances": 1,
  "client": "6c3f48bf-1098-46e8-8117-5cc745b45983",
  "remote-addr" : '1.1.1.'
}
XML:
<control-node>
  <instances>1</instances>
  <client>UUID</client>
  <remote-addr>1.1.1.1</remote-addr>
</control-node>

Response: TTL, List of <service type, Blob>
JSON: {
  "Apiservice": [{"ip_addr": "10.84.13.34", "port": "8082"}],
  "ttl": 357
}
XML:
<response>
  <ttl>300</ttl>
  <control-node>
    <ip_addr>192.168.2.0</ip_addr>
    <port>1682</port>
  </control-node>
</response>
```

The following fields are allowed in the body of the file:

Service Type—This is a string denoting what service is being requested (Apiservice). The instance count is the number of servers needed.

Client ID—This is a unique ID for the subscriber. Typically it is constructed from the UUID and the name of the subscriber.



NOTE: The subscription response includes a list of the services.

Discovery Service Heartbeats

A cookie is returned in response to a request to publish API. The cookie is sent in the heartbeat message to the discovery server. If three heartbeat messages are missed, the discovery server marks the service down and it is no longer assigned to the subscribers.

The heartbeat responses from the discovery server are either 200 Ok or 401. The 401 response is sent if the discovery server does not recognize the cookie. This could happen if the discovery server is restarted with the `reset_config` option. In this case, the client should plan on republishing the information.

Discovery Service Internal Databases

The database is maintained in Cassandra. There is a persistent copy so that the discovery service can maintain state across restarts.

Discovery Service Client Library

Python and C++ client libraries are available that allow publishing and subscription of services.

Discovery Service Debugging

To see a list of Discovery Service publishers, connect to the `http://discovery-server-ip:5998/services` URL.

To see list of Discovery Service subscribers, connect to the `http://discovery-server-ip:5998/clients` URL.

To see the log messages for the Discovery Service, display the `/var/log/contrail/discovery.log` file.

Related Documentation

- [Configuring Load Balancing as a Service in Contrail on page 388](#)

Support for Multicast

This section describes how the Contrail Controller supports broadcast and multicast.

- [Subnet Broadcast on page 307](#)
- [All-Broadcast/Limited-Broadcast and Link-Local Multicast on page 307](#)
- [Host Broadcast on page 308](#)

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:

Virtual network name – **vn1**

Unicast routing instance – **vn1.uc.inet**

Subnets (IPAM) allocated – **1.1.1.0/24; 2.2.0.0/16; 3.3.0.0/16**

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**:

1.1.1.255 -> forward to NH1 (composite next hop)

2.2.255.255 -> DROP

3.3.255.255 -> forward to NH2

The following entries are made to the next-hop table:

NH1 – **1.1.1.253; 1.1.1.252; 1.1.1.251**

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:

Virtual network name – **vn1**

Unicast routing instance – **vn1.uc.inet**

Subnets (IPAM) allocated – **1.1.1.0/24; 2.2.0.0/16; 3.3.0.0/16**

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**:

255.255.255.255/* -> NH1

The following entries are made to the next-hop table:

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

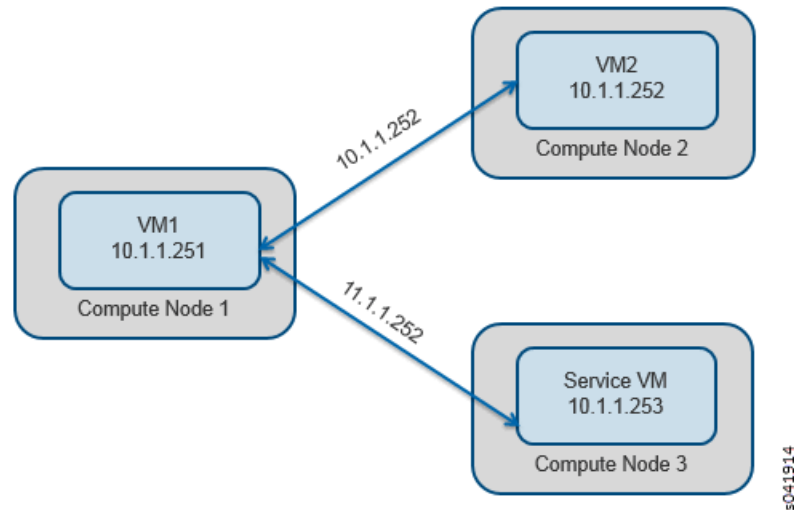
- [Static Routes for Service Instances on page 308](#)
- [Configuring Static Routes on a Service Instance on page 309](#)
- [Configuring Static Routes on Service Instance Interfaces on page 310](#)
- [Configuring Static Routes as Host Routes on page 312](#)

Static Routes for Service Instances

Static routes can be configured in a virtual network to direct traffic to a service virtual machine.

The following figure shows a virtual network with subnet 10.1.1.0/24. All of the traffic from a virtual machine that is directed to subnet 11.1.1.0/24 can be configured to be routed by

means of a service machine, by using the static route 11.1.1.252 configured on the service virtual machine interface.



Configuring Static Routes on a Service Instance

To configure static routes on a service instance, first enable the static route option in the service template to be used for the service instance.

To enable the static route option in a service template:

1. Go to **Configure > Services > Service Templates** and click **Create**.
2. At **Add Service Template**, complete the fields for **Name**, **Service Mode**, and **Image Name**.
3. Select the **Interface Types** to use for the template, then for each interface type that might have a static route configured, click the check box under the **Static Routes** column to enable the static route option for that interface.

The following figure shows a service template in which the left and right interfaces of service instances have the static routes option enabled. Now a user can configure

a static route on a corresponding interface on a service instance that is based on the service template shown.

Add Service Template

Name:

Service Mode:

Image Name:

Interface Types	Shared IP	Static Routes	+
Management	<input type="checkbox"/>	<input type="checkbox"/>	+ -
Left	<input type="checkbox"/>	<input checked="" type="checkbox"/>	+ -
Right	<input type="checkbox"/>	<input checked="" type="checkbox"/>	+ -

► [Advanced options](#)

Cancel Save

5041915

Configuring Static Routes on Service Instance Interfaces

To configure static routes on a service instance interface:

1. Go to **Configure > Services > Service Instances** and click **Create**.
2. At **Create Service Instances**, complete the fields for **Instance Name** and **Services Template**.
3. Select the virtual network for each of the interfaces
4. Click the **Static Routes** dropdown menu under each interface field for which the static routes option is enabled to open the **Static Routes** menu and configure the static routes in the fields provided.



NOTE: If the **Auto Configured** option is selected, traffic destined to the static route subnet is load balanced across service instances.

The following figure shows a configuration to apply a service instance between VN1 (10.1.1.0/24) and VN2 (11.1.1.0/24). The left interface of the service instance is configured with VN1 and the right interface is configured to be VN2 (11.1.1.0/24). The static route 11.1.1.0/24 is configured on the left interface, so that all traffic from VN1 that is destined to VN2 reaches the left interface of the service instance.

The screenshot shows the 'Create Service Instances' window with the following configuration:

- Instance Name:** nat
- Services Template:** nat - [in-network (management, left, right)]
- Interface 1:** Management, Auto Configured
- Interface 2:** Left, vn1
- Static Routes for Interface 2:**

Prefix	Next hop	
11.1.1.0/24	Interface 2	+ -
- Interface 3:** Right, vn2
- Static Routes for Interface 3:** (Empty table)

Buttons at the bottom: Cancel, Save. A vertical label 's041916' is on the right side.

The following figure shows static route 10.1.1.0/24 configured on the right interface, so that all traffic from VN2 that is destined to VN1 reaches the right interface of the service virtual machine.

The screenshot shows the 'Create Service Instances' window with the following configuration:

- Interface 2:** Left, vn1
- Static Routes for Interface 2:**

Prefix	Next hop	
11.1.1.0/24	Interface 2	+ -
- Interface 3:** Right, vn2
- Static Routes for Interface 3:**

Prefix	Next hop	
10.1.1.0/24	Interface 3	+ -

Buttons at the bottom: Cancel, Save. A vertical label 's041917' is on the right side.

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.

The screenshot shows the 'Create Network' dialog box. The 'Address Management' section is expanded, showing a table with columns 'IPAM', 'IP Block', and 'Gateway'. The table contains one row with 'ipam1', '1.2.3.0/24', and '1.2.3.254'. Below this, the 'Route Targets' section is expanded, showing a table with columns 'IPAM' and 'Route Prefix'. The table contains two rows: 'ipam1' with '1.1.1.0/24' and 'ipam1' with '2.2.2.0/24'. The 'Host Routes' section is also expanded, showing a table with columns 'IPAM' and 'Route Prefix'. The table contains two rows: 'ipam1' with '1.1.1.0/24' and 'ipam1' with '2.2.2.0/24'. The 'Cancel' and 'Save' buttons are at the bottom right.

IPAM	IP Block	Gateway
ipam1	1.2.3.0/24	1.2.3.254

IPAM	Route Prefix
ipam1	1.1.1.0/24
ipam1	2.2.2.0/24

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
```

BGP as a Service

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 on page 313](#)
- [BGPaaS Customer Use Cases on page 314](#)
- [Configuring BGPaaS on page 315](#)

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.

BGPaaS Customer Use Cases

This section provides example scenarios for implementing BGPaaS with Contrail.

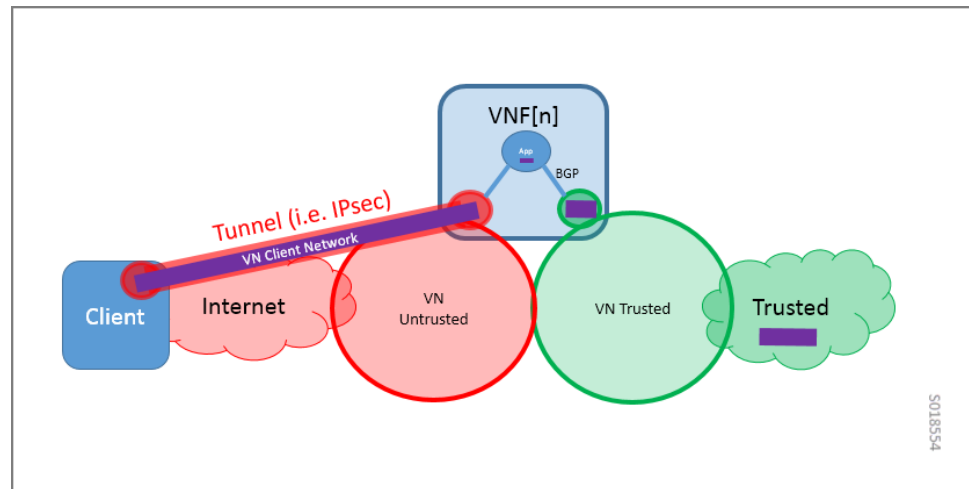
- [Dynamic Tunnel Insertion Within a Tenant Overlay on page 314](#)
- [Dynamic Network Reachability of Applications on page 315](#)
- [Liveness Detection for High Availability on page 315](#)

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 [xref target has no title] is a generic example showing the need to insert a dynamic tunnel into a virtual network.



Dynamic Network Reachability of Applications

The Domain Name System (DNS) is a widespread application that uses BGP as a mechanism to tune reachability of its services, based on metrics such as load, maintenance, availability, and the like. As DNS services are migrated to environments using overlays, a mechanism to preserve the existing application behavior and requirements is needed, including the ability to announce and withdraw reachability to the available application.

This requirement is not limited to DNS. Other applications, such as virtualized evolved packet core (vEPC) and others, use BGP as a mechanism for network reachability based on availability and load.

Liveness Detection for High Availability

Various keepalive mechanisms for tenant reachability have been provided by network components such as BGP, OSPF, PING, VRRP, BFD, or application-specific mechanisms. With BGP on the vRouter agent, BGP can be used to provide a liveness detection mechanism between the tenant on the local compute node and the services that the specific tenant VM is providing.

Configuring BGPaaS

The following are methods for configuring BGPaaS:

- [Configuring BGPaaS Using VNC API on page 316](#)
- [Using the Contrail User Interface to Configure BGPaaS on page 316](#)

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 configuration interface. On the left, the 'Configure' menu is open, and 'BGP as a Service' is selected under the 'Services' section. The main panel displays the 'Create BGP as a Service' dialog box. The dialog has the following fields and values:

- Name:** bgpaas-1
- IP Address:** 11.95.197.1
- Autonomous System:** 50000
- Address Family:** inet x, inet6 x
- Virtual Machine Interface(s):** 7d8fb01c-26af-4128-b030-ab446d3da0e5 (11.95.197.1) x
- Advanced Options:**
 - Hold Time:** 90
 - Admin State:** ☒

At the bottom right of the dialog are 'Cancel' and 'Save' buttons.

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.

CHAPTER 13

Configuring Service Chaining

- [Service Chaining on page 319](#)
- [Service Chaining MX Series Configuration on page 323](#)
- [Example: Creating an In-Network or In-Network-NAT Service Chain on page 324](#)
- [Example: Creating a Transparent Service Chain on page 333](#)
- [Example: Creating a Service Chain With the CLI on page 337](#)
- [ECMP Load Balancing in the Service Chain on page 340](#)
- [Customized Hash Field Selection for ECMP Load Balancing on page 341](#)
- [Using the Contrail Heat Template on page 345](#)
- [Service Chain Route Reorigination on page 349](#)
- [Service Instance Health Check on page 367](#)

Service Chaining

Contrail Controller supports chaining of various Layer 2 through Layer 7 services such as firewall, NAT, IDP, and so on.

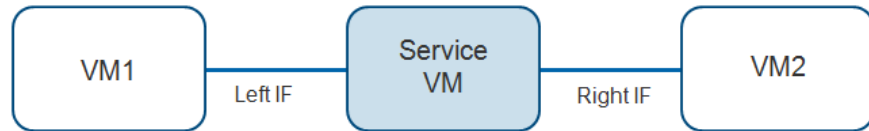
- [Service Chaining Basics on page 319](#)
- [Service Chaining Configuration Elements on page 321](#)

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.

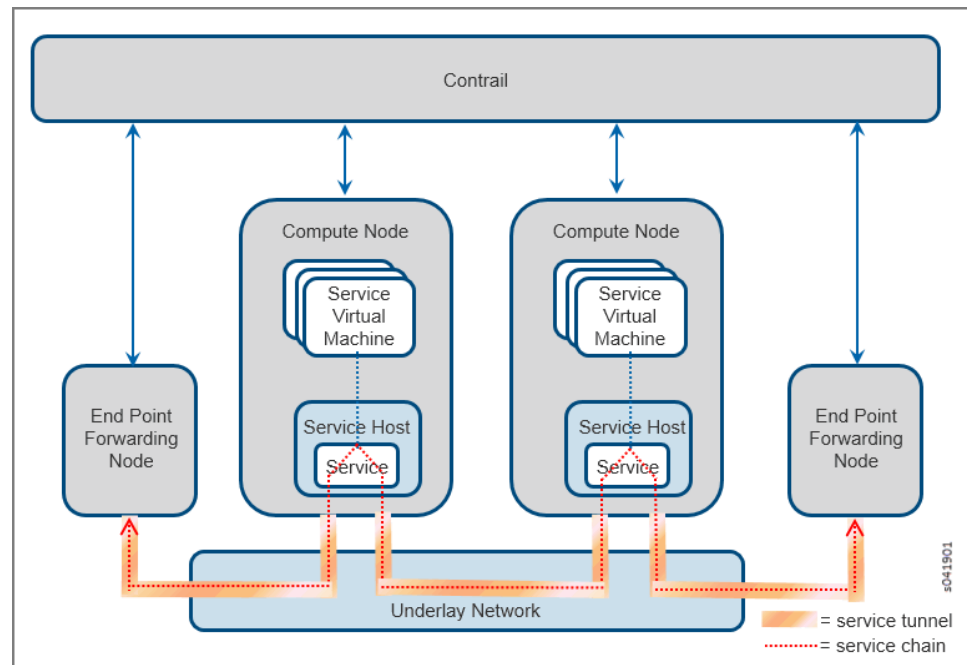
[Figure 92 on page 320](#) shows the basic service chain schema, with a single service. The service VM spawns the service, using the convention of left interface (left IF) and right interface (right IF). Multiple services can also be chained together.

Figure 92: 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 93 on page 320](#) 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 93: Contrail Service Chain



The following are the modes of services that can be configured.

Transparent or bridge mode

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.

In-network or routed mode

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.

In-network-nat mode

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:

Example: 'apply_service': [DomainName:ProjectName:ServiceInstanceName]

**Related
Documentation**

- [Example: Creating an In-Network or In-Network-NAT Service Chain on page 324](#)
- [Example: Creating a Service Chain With the CLI on page 337](#)
- [ECMP Load Balancing in the Service Chain on page 340](#)

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”](#) on page 287.

Example: Creating an In-Network or In-Network-NAT Service Chain

This section provides an example of creating an **in-network** service chain and an **in-network-nat** service chain using the Juniper Networks Contrail user interface. This service chain example also shows scaling of service instances.

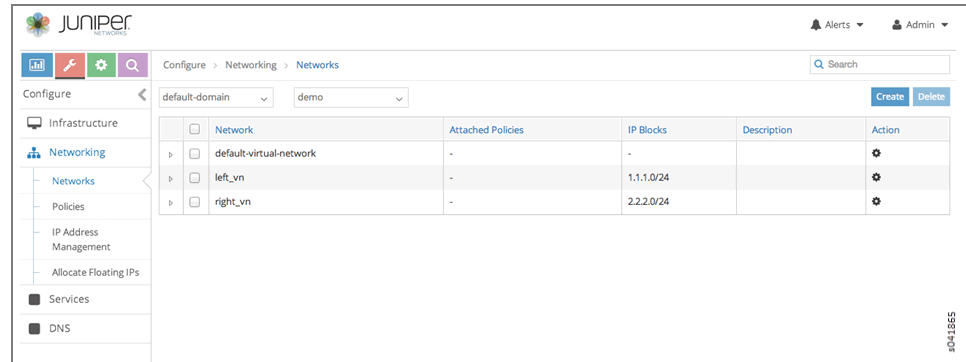
- [Creating an In-Network or In-Network-NAT Service Chain](#) on page 325

Creating an In-Network or In-Network-NAT Service Chain

To create an **in-network** or **in-network-nat** service chain:

1. Create a left and a right virtual network. Select **Configure > Networking > Networks** and create **left_vn** and **right_vn**; see [Figure 94 on page 325](#).

Figure 94: Create Networks



2. Configure a service template for an in-network service template for NAT. Navigate to **Configure > Services > Service Templates** and click the **Create** button on **Service Templates**. The **Add Service Template** window appears; see [Figure 95 on page 326](#).

Figure 95: Add Service Template

Add Service Template

Name: nat-template

Service Mode: In-Network

Image Name: nat-service

Interface Types	Shared IP	Static Routes	+
Management	<input type="checkbox"/>	<input type="checkbox"/>	+ -
Left	<input checked="" type="checkbox"/>	<input type="checkbox"/>	+ -
Right	<input type="checkbox"/>	<input type="checkbox"/>	+ -

▼ Advanced options

Service Scaling: ☒

Instance Flavor: m1.medium(RAM:4096, CPU cores:2, Disk:...) ▼

Cancel Save

Table 28: Add Service Template Fields

Field	Description
Name	Enter a name for the service template.
Service Mode	Select the service mode: In-Network (for firewall service), In-Network-NAT (for NAT service), or Transparent .
Service Scaling	If you will be using multiple virtual machines for a single service instance to scale out the service, select the Service Scaling check box. When scaling is selected, you can choose to use the same IP address for a particular interface on each virtual machine interface or to allocate new addresses for each virtual machine. For a NAT service, the left (inner) interface should have the same IP address, and the right (outer) interface should have a different IP address.
Image Name	Select from a list of available images the image for the service. NOTE: Only images that have been tagged as public in Glance will appear in the drop-down list.

Table 28: Add Service Template Fields (continued)

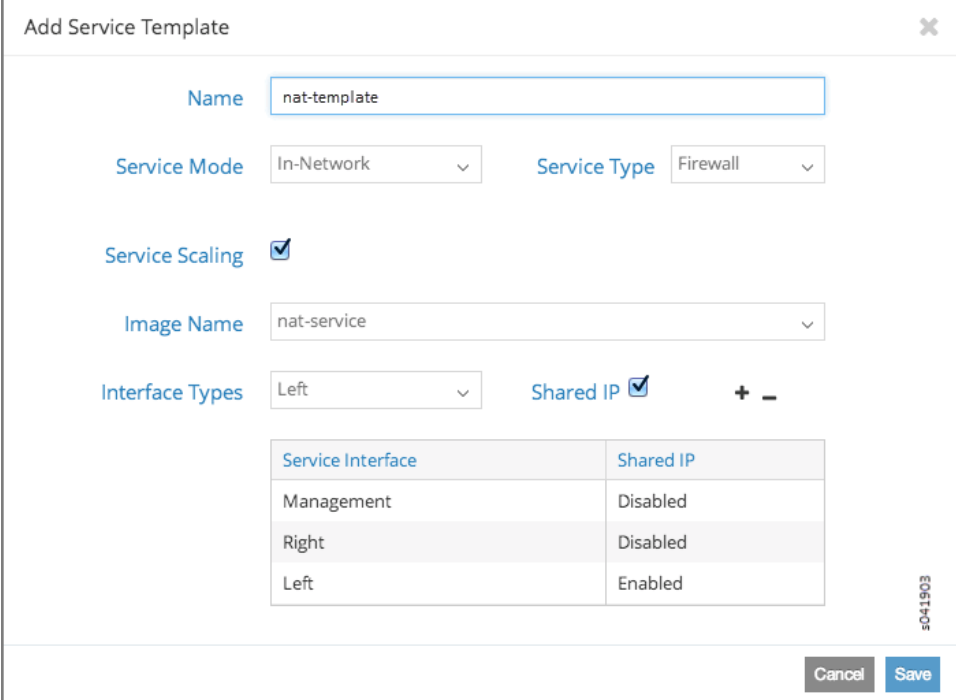
Field	Description
Interface Types	<p>Select the interface type or types for this service:</p> <ul style="list-style-type: none"> For firewall or NAT services, both Left Interface and Right Interface are required. For an analyzer service, only a Left Interface is required. For Juniper Networks virtual images, Management Interface is also required, in addition to any left or right requirement.

- On **Add Service Template**, complete the following for the in-network service template:
 - Name:** nat-template
 - Service Mode:** In-Network
 - Service Scaling:** Select from Advanced
 - Image Name:** nat-service
 - Interface Types:** Select Left Interface and Right Interface. For Juniper Networks virtual images, select Management Interface as the first interface.
 - The Left Interface will be automatically marked for sharing the same IP address
- If multiple instances are to be launched for a particular service instance, select the **Service Scaling** check box, which enables the **Shared IP** feature. [Figure 96 on page 328](#) shows the **Left** interface selected, with the **Shared IP** check box selected, so the left interface will share the IP address.



NOTE: The **Shared IP** for **Service Scaling** is an internal infrastructure feature used only for service scaling, it cannot be used for other features.

Figure 96: Add Service Template Shared IP



Add Service Template

Name

Service Mode **Service Type**

Service Scaling ☒

Image Name

Interface Types **Shared IP** ☒ + -

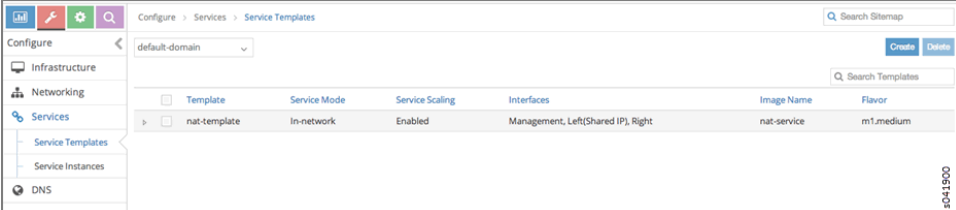
Service Interface	Shared IP
Management	Disabled
Right	Disabled
Left	Enabled

Cancel **Save**

- Click **Save**.

The service template is created and appears on the **Service Templates** screen, see [Figure 97 on page 328](#).

Figure 97: Service Templates



Service Templates

Template	Service Mode	Service Scaling	Interfaces	Image Name	Flavor
nat-template	In-network	Enabled	Management, Left(Shared IP), Right	nat-service	m1.medium

- Create the service instance. Navigate to **Configure > Services > Service Instances**, and click **Create**, then select the template to use and select the corresponding left, right, or management networks; see [Figure 98 on page 329](#).

Figure 98: Create Service Instances

Table 29: Create Service Instances Fields

Field	Description
Instance Name	Enter a name for the service instance.
Services Template	Select from a list of available service templates the service template to use for this instance.
Number of Instances	If scaling is enabled, enter a value in the Number of Instances field to define the number of instances of service virtual machines to launch.
Interface List and Virtual Networks	An ordered list of interfaces as defined in the Service Template. If you are using the Management Interface , select Auto Configured . The software will use an internally-created virtual network. For Left Interface , select left_vn and for Right Interface , select right_vn .

- If static routes are enabled for specific interfaces, open the **Static Routes** field below each enabled interface and enter the static route address details; see [Figure 99 on page 330](#).

Figure 99: Create Service Instances

Create Service Instances

Instance Name:

Services Template: nat-ecmp-template - [in-network (management, left, right)]

Number of instances: 1

Interface 1: Management Auto Configured

Interface 2: Left vn10 (admin)

Static Routes

Prefix	Next hop	
10.204.80.0/28	Interface 2	+ -

Interface 3: Right vn10 (admin)

Cancel Save

- The console for the service instances can be viewed. At **Configure > Services > Service Instances**, click the arrow next to the name of the service instance to reveal the details panel for that instance, then click **View Console** to see the console details; see [Figure 100 on page 330](#) and [Figure 101 on page 331](#).

Figure 100: Service Instance Details

fw-instance firewall-template (Transparent) Active 1 Instances Management Network: Automatic Left Network: Automatic Right Network: Automatic

Instance Name: fw-instance

Template: firewall-template (Transparent)

Number of instances: 1 Instances

Networks: Management Network: Automatic, Left Network: Automatic, Right Network: Automatic

Image: m1.medium

Flavor: vcnbridge

Instance Details

Virtual Machine	Status	Power State	Networks
fw-instance_1	ACTIVE	RUNNING	svc-vn-mgmt250.250.1.252 svc-vn-left250.250.2.253 svc-vn-right250.250.3.253

Static Route

[View Console](#)

Figure 101: Service Instance Console

0.204.216.36:5999/vnc_auto.html?token=9eada783-24e7-4808-9325-4ad257bf3762

Connected (unencrypted) to: QEMU (instance-0000000b)

Interface	Admin	Link	Proto	Local	Remote
ge-0/0/0	up	up			
ge-0/0/0.0	up	up	inet	250.250.1.253/24	
ge-0/0/0	up	up			
ip-0/0/0	up	up			
lsq-0/0/0	up	up			
lt-0/0/0	up	up			
mt-0/0/0	up	up			
sp-0/0/0	up	up			
sp-0/0/0.0	up	up	inet	10.0.0.1	--> 10.0.0.16
sp-0/0/0.16383	up	up	inet	10.0.0.6	--> 0/0
				128.0.0.1	--> 128.0.1.16
				128.0.0.6	--> 0/0
ge-0/0/1	up	up			
ge-0/0/1.0	up	up	inet	1.1.1.253/24	
ge-0/0/2	up	up			
ge-0/0/2.0	up	up	inet	2.2.2.253/24	
dsc	up	up			
gre	up	up			
ipip	up	up			
lo0	up	up			
lo0.16384	up	up	inet	127.0.0.1	--> 0/0
lo0.16385	up	up	inet	10.0.0.1	--> 0/0
---(more)---					

5041919

9. Configure the network policy. Navigate to **Configure > Networking > Policies**.
 - Name the policy and associate it with the networks created earlier: **left_vn** and **right_vn**.
 - Set source network as **left_vn** and destination network as **right_vn**.
 - Select **Apply Service** and select the service (**nat-ecmp**).

Figure 102: Create Policy

Create Policy

Policy Name
fw-policy

Policy Rules

Action	Protocol	Source Network	Source Ports	Direction	Destination Network	Destination Ports	Apply Service	Mirror to
PAS	ANY	left_vn	Source	<>	right_vn	Destin	<input checked="" type="checkbox"/>	<input type="checkbox"/>

fw-instance x

Cancel Save

5041870

10. Associate the policy with both the **left_vn** and the **right_vn**. Navigate to **Configure > Networking > Network**.
 - On the right side of **left_vn**, click the gear icon to enable **Edit Network**.
 - In the **Edit Network** dialog box for **left_vn**, select **nat-policy** in the **Network Policy(s)** field.
 - Repeat the same process for the **right_vn**.

Figure 103: Edit Network

Network Name:

Network Policy(s):

Address Management: IP Block: Gateway:

IPAM	IP Block	Gateway
default-domain:default-project:default-network-ipam	1.1.1.0/24	1.1.1.254

Route Targets

Floating IP Pools

Host Routes

Advanced Options

Cancel Save

11. Launch virtual machines (from OpenStack) and test the traffic through the service chain by doing the following:
 - a. Navigate to **Configure > Networking > Policies**.
 - b. Launch **left_vm** in virtual network **left_vn**.
 - c. Launch **right_vm** in virtual network **right_vn**.
 - d. Ping from **left_vm** to **right_vm** IP address (**2.2.2.252** in Figure 104 on page 332).
 - e. A **TCPDUMP** on the **right_vm** should show that packets are NAT-enabled and have the source IP set to **2.2.2.253**.

Figure 104: Launch Instances

Instances

Instance Name	IP Address	Size	Keypair	Status	Task	Power State	Actions
nat-instance_1	250.250.1.253 left_vn 1.1.1.253 right_vn 2.2.2.253	m1.medium 4GB RAM 2 VCPU 40GB Disk	-	Active	None	Running	Create Snapshot More
right_vm	2.2.2.252	m1.tiny 512MB RAM 1 VCPU 0 Disk	-	Active	None	Running	Create Snapshot More
left_vm	1.1.1.252	m1.tiny 512MB RAM 1 VCPU 0 Disk	-	Active	None	Running	Create Snapshot More

Displaying 3 items

- Related Documentation**
- [Service Chaining on page 319](#)
 - [Example: Creating a Transparent Service Chain on page 333](#)
 - [ECMP Load Balancing in the Service Chain on page 340](#)

Example: Creating a Transparent Service Chain

This section provides an example of creating a transparent mode service chain using the Juniper Networks Contrail user interface. Also called bridge mode, transparent mode is used for services that do not modify the packet, such as Layer 2 firewall, Intrusion Detection and Prevention (IDP), and so on. The following service chain example also shows scaling of service instances.

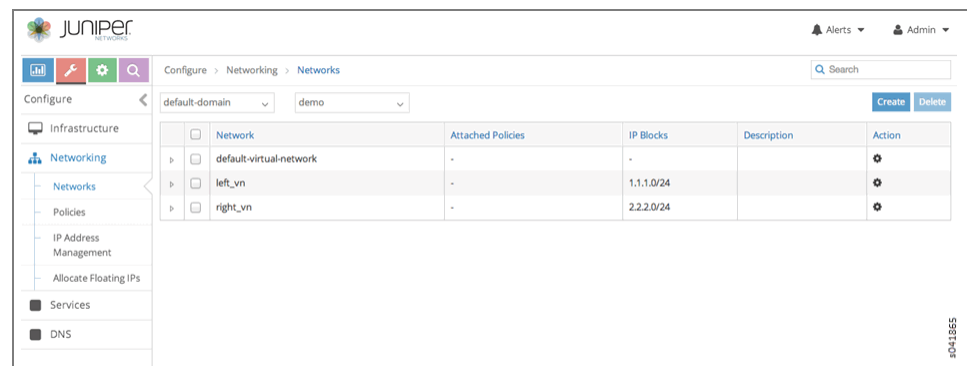
- [Creating a Transparent Mode Service Chain on page 333](#)

Creating a Transparent Mode Service Chain

To create a transparent mode service chain:

1. Create a left and a right virtual network. Select **Configure > Networking > Networks** and create **left_vn** and **right_vn**; see [Figure 105 on page 333](#).

Figure 105: Create Networks



2. Configure a service template for a transparent mode. Navigate to **Configure > Services > Service Templates** and click the **Create** button on **Service Templates**. The **Add Service Template** window appears; see [Figure 106 on page 334](#).

Figure 106: Add Service Template

Add Service Template

Name:

Service Mode:

Image Name:

Interface Types	Shared IP	Static Routes	+
Management	<input type="checkbox"/>	<input type="checkbox"/>	+ -
Left	<input checked="" type="checkbox"/>	<input type="checkbox"/>	+ -
Right	<input checked="" type="checkbox"/>	<input type="checkbox"/>	+ -

Advanced options

Service Scaling: ☒

Instance Flavor:

Cancel Save

Table 30: Add Service Template Fields

Field	Description
Name	Enter a name for the service template.
Service Mode	Select the service mode: In-Network or Transparent .
Service Scaling	If you will be using multiple virtual machines for a single service instance to scale out the service, select the Service Scaling check box. When scaling is selected, you can choose to use the same IP address for a particular interface on each virtual machine interface or to allocate new addresses for each virtual machine. For a NAT service, the left (inner) interface should have the same IP address, and the right (outer) interface should have a different IP address.
Image Name	Select from a list of available images the image for the service.
Interface Types	Select the interface type or types for this service: <ul style="list-style-type: none"> For firewall or NAT services, both Left Interface and Right Interface are required. For an analyzer service, only Left Interface is required. For Juniper Networks virtual images, Management Interface is also required, in addition to any left or right requirement.

3. On **Add Service Template**, complete the following for the transparent mode service template:

- **Name:** firewall-template
- **Service Mode:** Transparent
- **Service Scaling:** Select this.
- **Image Name:** vsrx-bridge
- **Interface Types:** Select Left Interface, Right Interface, and Management Interface.

If multiple instances are to be launched for a particular service instance, select the **Service Scaling** check box, which enables the **Shared IP** feature.

4. Click **Save**.

5. Create the service instance. Navigate to **Configure > Services > Service Instances**, and click **Create**, then select the template to use and select the corresponding left, right, or management networks; see [Figure 107 on page 335](#).

Figure 107: Create Service Instances

Table 31: Create Service Instances Fields

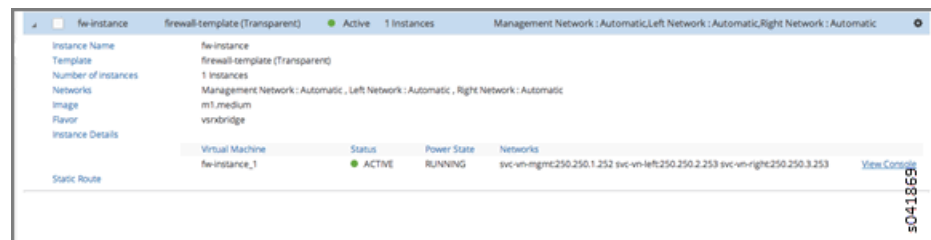
Field	Description
Instance Name	Enter a name for the service instance.
Services Template	Select from a list of available service templates the service template to use for this instance.
Left Network	Select from a list of available virtual networks the network to use for the left interface. For transparent mode, select Auto Configured .

Table 31: Create Service Instances Fields (continued)

Field	Description
Right Network	Select from a list of available virtual networks the network to use for the right interface. For transparent mode, select Auto Configured
Management Network	If you are using the Management Interface , select Auto Configured . The software will use an internally-created virtual network. For transparent mode, select Auto Configured

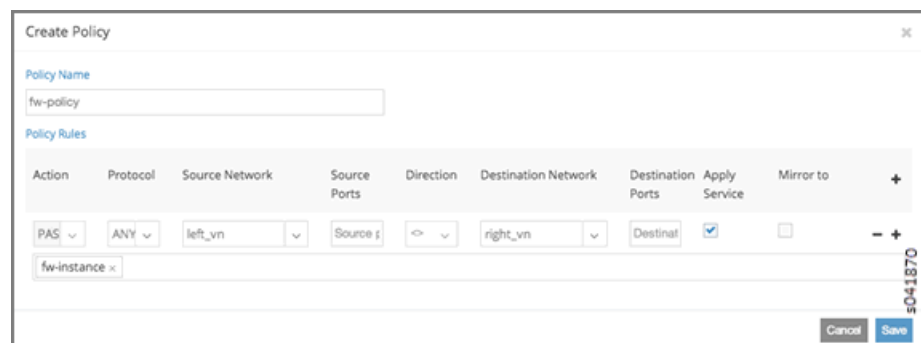
6. If scaling is enabled, enter a value in the **Number of Instances** field to define the number of instances of service virtual machines to launch; see [Figure 108 on page 336](#).

Figure 108: Service Instance Details



7. Next, configure the network policy. Navigate to **Configure > Networking > Policies**.
- Name the policy **fw-policy**.
 - Set source network as **left_vn** and destination network as **right_vn**.
 - Check **Apply Service** and select the service (**fw-instance**).

Figure 109: Create Policy



8. Next, associate it to the networks created earlier – **left_vn** and **right_vn**. Navigate to **Configure > Networking > Policies**.
- On the right side of **left_vn**, click the gear icon to enable **Edit Network**.
 - In the **Edit Network** dialog box for **left_vn**, select **nat-policy** in the **Network Policy(s)** field.

- Repeat the process for the **right_vn**.
- 9. Next, launch virtual machines (from OpenStack) and test the traffic through the service chain by doing the following:
 - a. Navigate to **Configure > Networking > Policies**.
 - b. Launch **left_vm** in virtual network **left_vn**.
 - c. Launch **right_vm** in virtual network **right_vn**.
 - d. Ping from **left_vm** to **right_vm** IP address (2.2.2.252 in [Figure 110 on page 337](#)).
 - e. A **TCPDUMP** on the **right_vm** should show that packets have the source IP set to 2.2.2.253.

Figure 110: Launch Instances

Instance Name	IP Address	Size	Keypair	Status	Task	Power State	Actions
net-instance_1	250.250.1.253	m1.medium 4GB RAM 2 VCPU 40GB Disk	-	Active	None	Running	Create Snapshot More
right_vm	2.2.2.252	m1.tiny 512MB RAM 1 VCPU 0 Disk	-	Active	None	Running	Create Snapshot More
left_vm	1.1.1.252	m1.tiny 512MB RAM 1 VCPU 0 Disk	-	Active	None	Running	Create Snapshot More

Related Documentation

- [Service Chaining on page 319](#)

Example: Creating a Service Chain With the CLI

This section provides syntax and examples for creating service chaining objects for Contrail Controller.

- [CLI for Creating a Service Chain on page 337](#)
- [CLI for Creating a Service Template on page 338](#)
- [CLI for Creating a Service Instance on page 338](#)
- [CLI for Creating a Service Policy on page 338](#)
- [Example: Creating a Service Chain with VSRX and In-Network or Routed Mode on page 339](#)

CLI for Creating a Service Chain

All of the commands needed to create service chaining objects are located in **/opt/contrail/utls**.

CLI for Creating a Service Template

The following commands are used to create a service template:

```
./service-template.py add    [--svc_type {firewall, analyzer}]  
  
                             [--image_name IMAGE_NAME]  
  
                             template_name  
  
./service-template.py del    template_name
```

CLI for Creating a Service Instance

The following commands are used to create a service instance:

```
./service-instance.py add    [--proj_name PROJ_NAME]  
  
                             [--mgmt_vn MGMT_VN]  
  
                             [--left_vn LEFT_VN]  
  
                             [--right_vn RIGHT_VN]  
  
                             instance_name  
  
                             template_name  
  
./service-instance.py del    [--proj_name PROJ_NAME]  
  
                             instance_name  
  
                             template_name
```

CLI for Creating a Service Policy

The following commands are used to create a service policy:

```
./service-policy.py add      --svc_list SVC_LIST [SVC_LIST ...]  
  
                             --vn_list VN_LIST [VN_LIST ...]  
  
                             [--proj_name PROJ_NAME]  
  
                             policy_name  
  
./service-policy.py del      [--proj_name PROJ_NAME]  
  
                             policy_name
```

Example: Creating a Service Chain with VSRX and In-Network or Routed Mode

The following example creates a VSRX firewall service in a virtual network named **test**, using a project named **demo** and a template, an instance, and a policy, all named **test**.

1. Add images to Glance (OpenStack image service).

- a. Download the following images:

```
precise-server-cloudimg-amd64-disk1.img
```

```
junos-vsrx-12.1-nat.img
```

- b. Add the images to Glance, using the names **ubuntu** and **vsrx**.

```
(source /etc/contrail/openstackrc; glance add name='ubuntu' is_public=true
container_format=ovf disk_format=qcow2 <
precise-server-cloudimg-amd64-disk1.img)
```

```
(source /etc/contrail/openstackrc; glance add name='vsrx' is_public=true
container_format=ovf disk_format=qcow2 < junos-vsrx-12.1-dhcp.img)
```

2. Create a service template of type **firewall** and named **vsrx**.

```
./service-template.py add test_template --svc_type firewall --image_name vsrx
```

3. Create virtual networks.

```
VN1
```

```
VN2
```

4. Create a service template.

```
./service-template.py add --svc_scaling ecmp-template
```

5. Create a service instance.

```
./service-instance.py add --proj_name admin --left_vn VN1 --right_vn VN2
--max_instances 3 ecmp-instance ecmp-template
```

6. Create a service policy.

```
./service-policy.py add proj_name admin --svc_list ecmp-instance --vn_list VN1 VN2
ecmp-policy
```

7. Create virtual machines and attach them to virtual networks.

```
VM1 (attached to VN1)—use ubuntu image
```

```
VM2 (attached to VN2)—use ubuntu image
```

8. Launch the instances **VM1** and **VM2**.

9. Send ping traffic from **VM1** to **VM2**.

10. Send traffic from **VM1** in **VN1** to **VM2** in **VN2**.
11. You can use the Contrail Juniper Networks interface to monitor the ping traffic flows. Select **Monitor > Infrastructure > Virtual Routers** and select an individual vRouter. Click through to view the vRouter details, where you can click the **Flows** tab to view the flows.

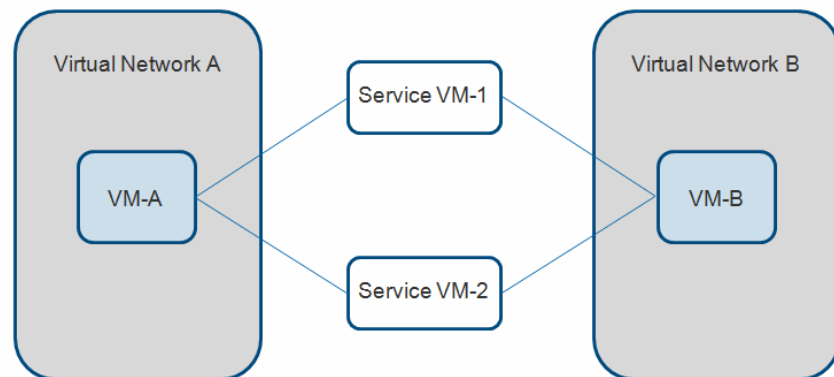
Related Documentation

- [Service Chaining on page 319](#)

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 111 on page 340](#), 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 111: 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 on page 319](#)
 - [Example: Creating a Service Chain With the CLI on page 337](#)
 - [Customized Hash Field Selection for ECMP Load Balancing on page 341](#)

Customized Hash Field Selection for ECMP Load Balancing

Overview: Custom Hash Feature

Starting with Contrail Release 3.0, it is possible to configure the set of fields used to hash upon during equal-cost multipath (ECMP) load balancing.

Earlier versions of Contrail had this set of fields fixed to the standard 5-tuple set of: source L3 address, destination L3 address, L4 protocol, L4 SourcePort, and L4 DestinationPort.

With the custom hash feature, users can configure an exact subset of fields to hash upon when choosing the forwarding path among a set of eligible ECMP candidates.

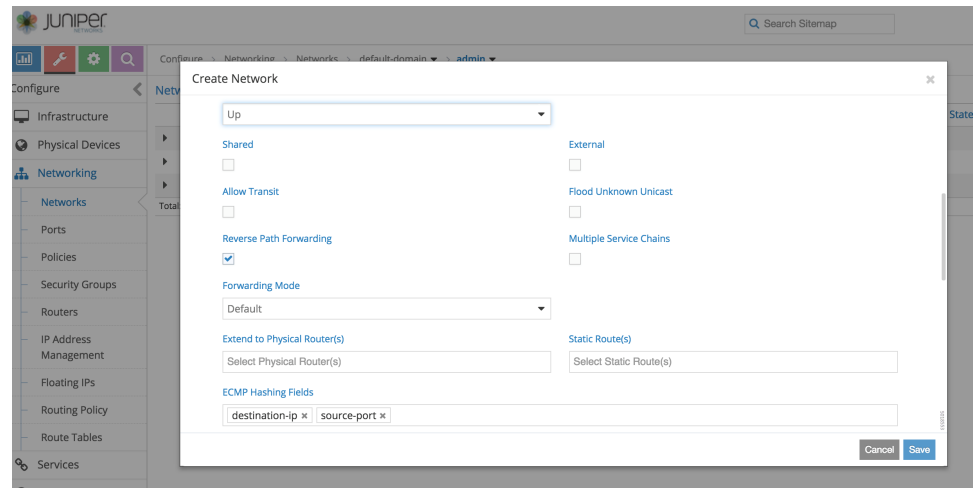
The custom hash configuration can be applied in the following ways:

- globally
- per virtual network (VN)
- per virtual network interface (VNI)

VNI configurations take precedence over VN configurations, and VN configurations take precedence over global level configuration (if present).

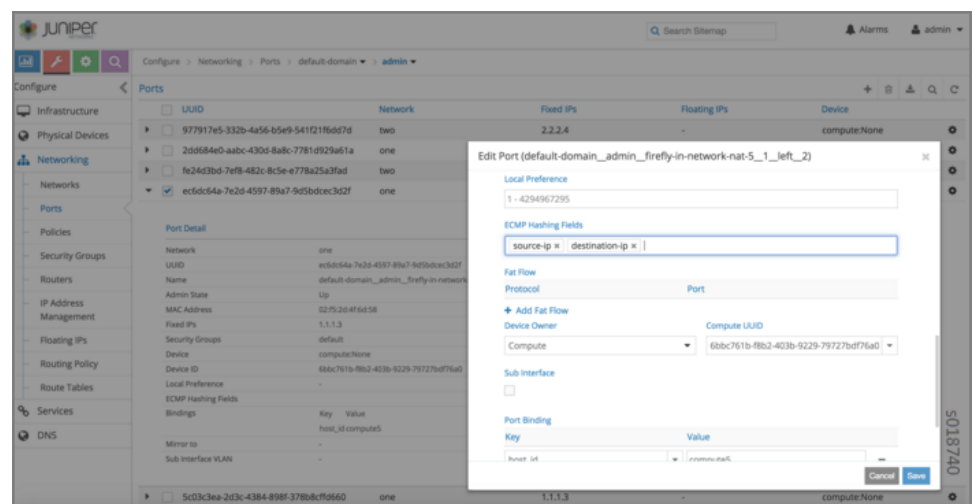
Custom hash is useful whenever packets originating from a particular source and addressed to a particular destination must go through the same set of service instances during transit. This might be required if source, destination, or transit nodes maintain a certain state based on the flow, and the state behavior could also be used for subsequent new flows, between the same pair of source and destination addresses. In such cases, subsequent flows must follow the same set of service nodes followed by the initial flow.

You can use the Contrail UI to identify specific fields in the network upon which to hash at the **Configure > Networking > Network, Create Network** window, in the **ECMP Hashing Fields** section as shown in the following figure.



If the hashing fields are configured for a virtual network, all traffic destined to that VN will be subject to the customized hash field selection during forwarding over ECMP paths by vRouters. This may not be desirable in all cases, as it could potentially skew all traffic to the destination network over a smaller set of paths across the IP fabric.

A more practical scenario is one in which flows between a source and destination must go through the same service instance in between, where one could configure customized ECMP fields for the virtual machine interface (VMI) of the service instance. Then, each service chain route originating from that VMI would get the desired ECMP field selection applied as its path attribute, and eventually get propagated to the ingress vRouter node. See the following example.



Using ECMP Hash Fields Selection

Custom hash fields selection is most useful in scenarios where multiple ECMP paths exist for a destination. Typically, the multiple ECMP paths point to ingress service instance nodes, which could be running anywhere in the Contrail cloud.

Configuring ECMP Hash Fields Over Service Chains

Use the following steps to create customized hash fields with ECMP over service chains.

1. Create the virtual networks needed to interconnect using service chaining, with ECMP load-balancing.
2. Create a service template and enable scaling.
3. Create a service instance, and using the service template, configure by selecting:
 - the desired number of instances for scale-out
 - the left and right virtual network to connect
 - the shared address space, to make sure that instantiated services come up with the same IP address for left and right, respectively

This configuration enables ECMP among all those service instances during forwarding.

4. Create a policy, then select the service instance previously created and apply the policy to to the desired 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

```

Using the Contrail Heat Template

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 on page 345](#)
- [Heat Architecture on page 346](#)
- [Heat Version 2 Resources on page 346](#)
- [Heat Version 2 with Service Chaining and Port Tuple Sample Workflow on page 347](#)
- [Example: Creating a Service Template Using Heat on page 347](#)

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.

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

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. Existing Version 1 templates will continue to work; however, there won't be any new features or fixes for them. Version 1 resources will be available only to aid the transition to the new version.

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: 2013-09-23
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. Define an environment file to give input to the Heat template.

```
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 using the following command:

```
heat stack- create stack1 -f service_template.yaml -e service_template.env
```

Related Documentation

- *Service Chain Version 2 with Port Tuple*

Service Chain Route Reorigination

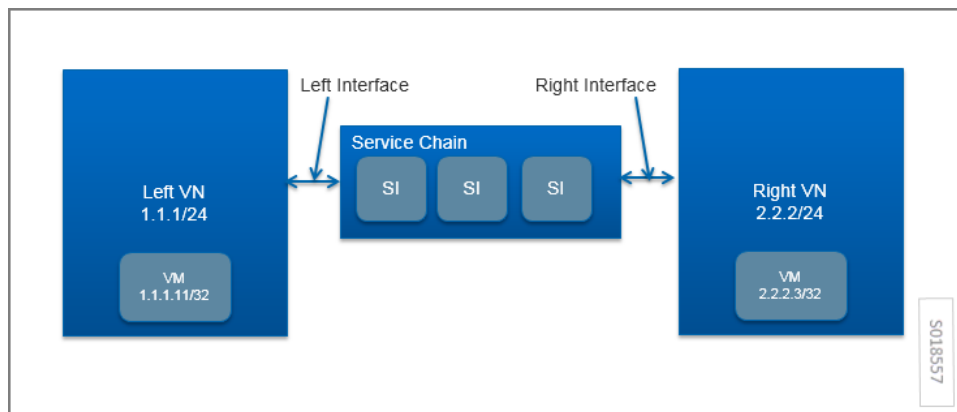
- [Overview: Service Chains in Contrail on page 349](#)
- [Route Aggregation on page 350](#)
- [Routing Policy on page 357](#)
- [Control for Route Reorigination on page 365](#)

Overview: Service Chains 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 the following example figure, 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.



To use reorigination of routes for service chaining (for example, putting the route for the right network in the left routing table) requires the following features.

- **Route aggregation**

For scaling purposes, it is useful to publish an aggregated route as the service chain route, rather than publishing every route of each VM (/32). This reduces the memory footprint for the route table in the gateway router and also reduces route exchanges between control nodes and the gateway router. The route can be aggregated to the default route (0/0), to the VN subnet prefix, or to any arbitrary route prefix.

- **Path attribute modification for reoriginated routes**

There are cases where the **BgpPath** attribute for the service chain route needs to be modified. An example is the case of service chain failover, in which there are two service chains with identical services that are connected between the same two VNs. The operator needs to control which service chain is used for traffic between two networks, in addition to ensuring redundancy and high availability by providing failover support. Path attribute modification for reoriginated routes is implemented by means of routing policy, by providing an option to alter the MED (multi-exit discriminator) or **local-pref** of the reoriginated service chain route.

- **Control to enable and disable reorigination of the route**

In some scenarios, the operator needs a control to stop reorigination of the route as the service chain route, for example, when static routes are configured on service VM interfaces. Control to enable or disable reorigination of the route is implemented by tagging the routes with the **no-reoriginate** community. Routes with the **no-reoriginate** community tag are skipped for route reorigination.

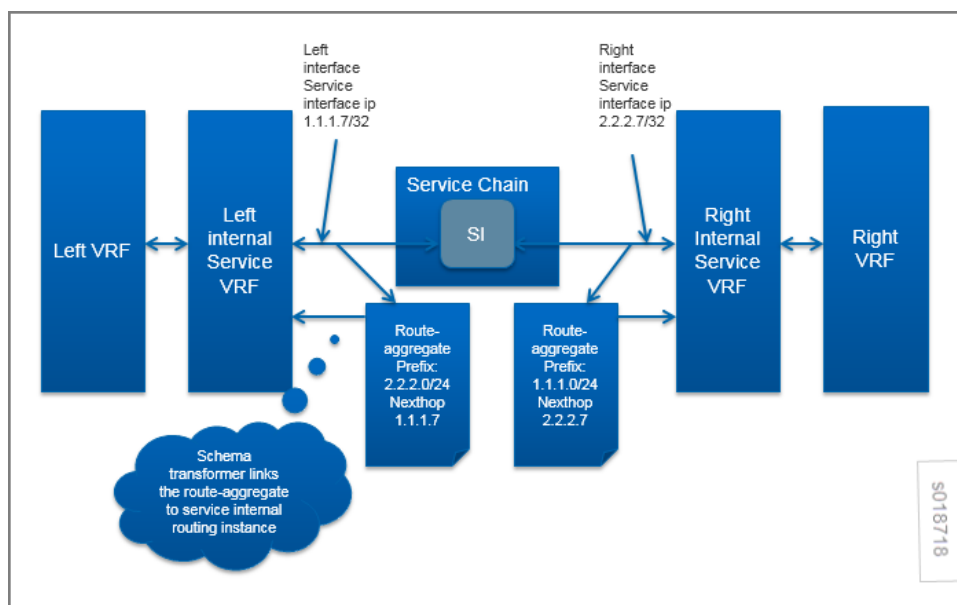
Route Aggregation

The route aggregation configuration object contains a list of prefixes to aggregate. The next-hop field in the route aggregate object contains the address of the route whose next hop is stitched as a next hop of the aggregate route.

Route aggregation is configured on the service instance. The operator can attach multiple route aggregation objects to a service instance. For example, if routes from the right VN need to be aggregated and reoriginated in the route table of the left VN, the route aggregate object is created with a prefix of the right VN's subnet prefix and attached to the left interface of the service instance.

If the service chain has multiple service instances, the route aggregate object is attached to the left interface of the left most service instance and to the right interface of the right most service instance.

The relationships are shown in the following illustration.



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 on page 351](#)
- [Service Instance Link to Route Aggregate Object on page 352](#)
- [Routing Instance Link to Route Aggregate Object on page 352](#)

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 on page 353](#)
- [Configure Service Instance on page 354](#)
- [Create a Virtual Network and Network Policy on page 354](#)

- [Validate the Route Aggregate Object in the API Server on page 355](#)
- [Validate the Route Aggregate Object in the Control Node on page 356](#)

Configure Route Aggregate Object

You can use the Contrail UI **Create Route Aggregate** screen to name the route aggregate object and identify the routes to aggregate. See the following example.

Sample VNC Script to Create a Route Aggregate Object

You can use a VNC script to create a route aggregate object, as in the following sample.

```
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)
```

Configure 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 following example.

Create Service Instance

si-aggregate st-with-aggregate - (transparent (left, right)...

▼ Interface Details

Interface Type Virtual Network

left Auto Configured

Interface Type Virtual Network

right Auto Configured

▼ Advanced Options

Routing Policy

▼ Route Aggregate

Interface Type Route Aggregate

right left-to-right

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	right	right	ANY	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Service Instance

si-aggregate

+ Add Rule

Cancel Save

Attach the network policy to create the service chain between the left and right VNs. See the following example.

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-7fe7708d97f-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"
  }
}
```

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

To check the state of the route aggregate object on the control node, point your browser to:

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 able. For example:

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 05:00:29.215876	paths							
protocol	last_modified	local_preference	local_as	peer_as	peer_router_id	source_as	path	next_hop	label
Aggregate	2016-Feb-18 05:00:29.215876	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.

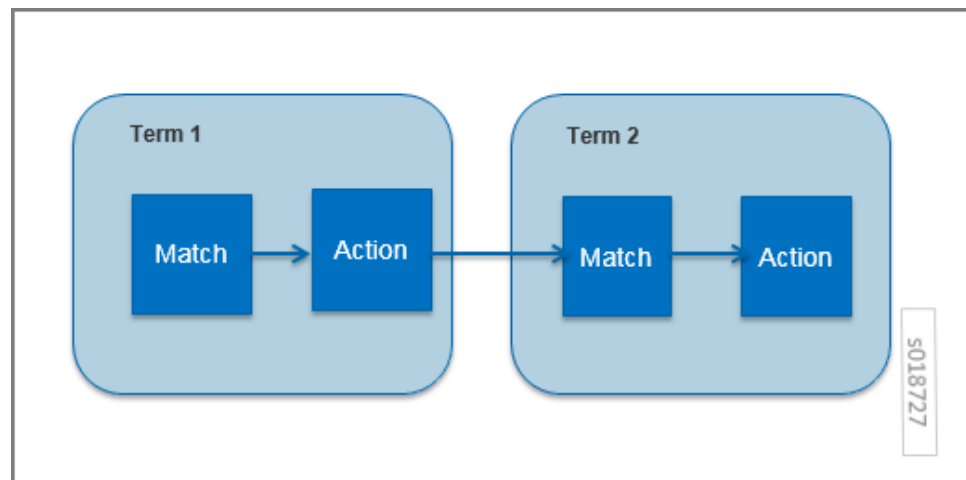
The routing policy contains list terms. A term can be a terminal rule, meaning that upon a match on the specified term, no further terms are evaluated and the route is dropped or accepted, based on the action in that term.

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"> • exact • orlonger • longer <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.
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.

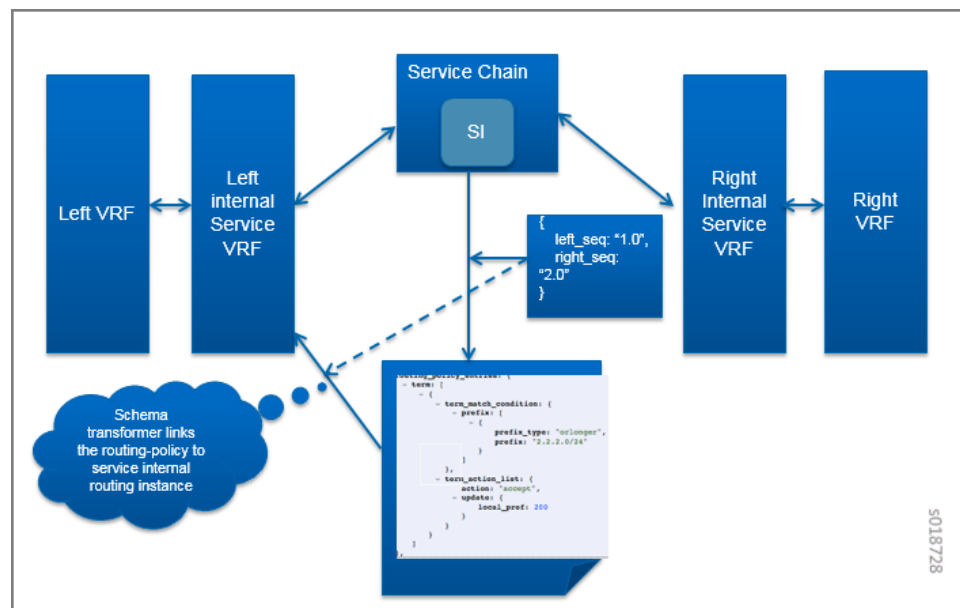
Update Action	User Input	Description
community	List of community	As part of the policy update, the following actions can be taken for community: <ul style="list-style-type: none"> Add a list of community to the existing community. Set a list of community. Remove a list of community (if present) from the existing community.
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 on page 361](#)
- [Configure Service Instance on page 362](#)
- [Configure the Network Policy for the Service Chain on page 362](#)

Create Routing Policy

First, create the routing policy. See the following example.

Create Routing Policy

Name
failover

Term(s)
from: { prefix 2.2.2.0/24 orlonger } then: { local-preference 200 }

From
prefix 2.2.2.0/24 orlonger

Then
local-preference 200

Cancel Save

s018729

Configure Service Instance

Create a service instance and attach the routing policy to both the left and right interfaces. The order of the policy is calculated by the UI, based on the order of the policy specified in the list.

Configure the Network Policy for the Service Chain

At **Edit Policy**, create a policy for the service chain, see the following example.

Using a VNC Script to Create Routing Policy

The following example shows use of a VNC API script to create a routing policy.

```
from vnc_api.vnc_api import *
vnc_lib = VncApi("admin", "<password>", "admin")
project=vnc_lib.project_read(fq_name=["default-domain", "admin"])
routing_policy=RoutingPolicy(name="vnc_3", parent_obj=project)
policy_term=PolicyTermType()
policy_statement=PolicyStatementType()

match_condition=TermMatchConditionType(protocol=["bgp"], community="22:33")
prefix_match=PrefixMatchType(prefix="1.1.1.0/24", prefix_type="orlonger")
match_condition.set_prefix([prefix_match])

term_action=TermActionListType(action="accept")
```

```

action_update=ActionUpdateType(local_pref=101, med=10)
add_community=ActionCommunityType()
comm_list=CommunityListType(["11:22"])
add_community.set_add(comm_list)
action_update.set_community(add_community)
term_action.set_update(action_update)

policy_term.set_term_action_list(term_action)
policy_term.set_term_match_condition(match_condition)

policy_statement.add_term(policy_term)
routing_policy.set_routing_policy_entries(policy_statement)
vnc_lib.routing_policy_create(routing_policy)

```

Verify Routing Policy in API Server

You can verify the service instance references and the routing instance references for the routing policy by looking in the API server configuration database. See the following example.

```

-----
- routing_policy_entries: {
  - term: {
    - {
      - term_match_condition: {
        - prefix: {
          - {
            prefix_type: "orlonger",
            prefix: "2.2.2.0/24"
          }
        }
      },
      - term_action_list: {
        action: "accept",
        - update: {
          local_pref: 200
        }
      }
    }
  }
},
+ id_perms: {...},
- routing_instance_refs: [
  - {
    - to: [
      "default-domain",
      "admin",
      "right",
      "service-ace7ae00-56e3-42d1-96ec-7fe77088d97f-default-domain_admin_ha-chain"
    ],
    href: "http://nodea27.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://nodea27.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://nodea27.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

See the following example.

routing_policies						
name	generation	ref_count	terms			deleted
default-domain:admin:failover	0	2	<div><div>terminal</div><div>matches</div><div>actions</div></div> <div><div>true</div><div><div>matches</div><div>prefix [2.2.2.0/24 or longer]</div></div><div><div>actions</div><div>accept</div><div>local-pref 200</div></div></div>			false
default-domain:default-project:default-routing-policy	0	0	terms			false

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

See the following example.

ShowBgpRoutingPolicyConfigResp		
routing_policies		
name	terms	
default-domain:admin:failover	terms	
	match	action
	from { prefix 2.2.2.0/24 orlonger }	then { local-preference 200 accept }

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>

See the following example.

service_chain_info					static_routes aggregate_routes routing_policies			
service_chain_info					static_routes	aggregate_routes	routing_policies	
family	routing_instance	chain_address	prefix	service_instance			policy_name	sequence
inet	default-domain:admin:right:right	1.1.1.6	prefix 2.2.2.0/24	default-domain:admin:he-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>

See the following example.

routing_policies	
routing_policies	
policy_name	generation
default-domain:admin:failover	0

Control for Route Reorigination

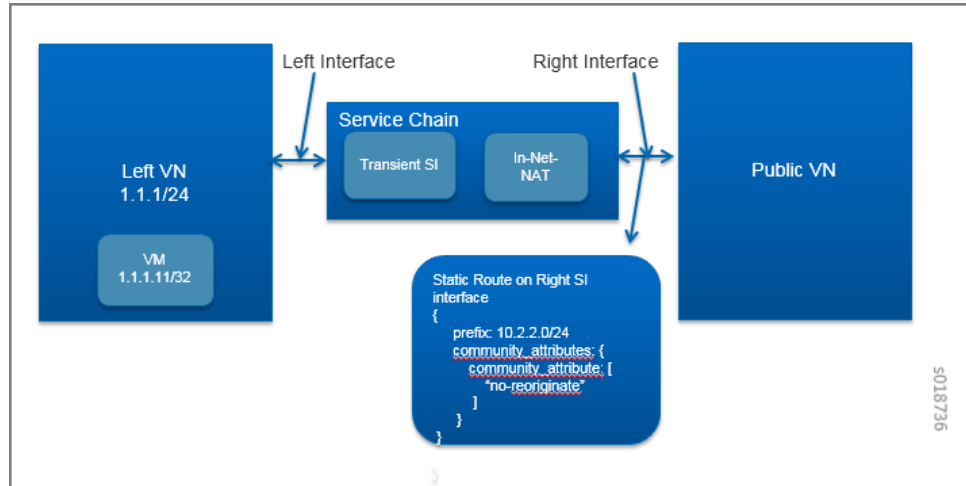
The ability to prevent reorigination of interface static routes is typically required when routes are configured on an interface that belongs to a service VM.

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.

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: "a6e1e71f-f828-43de-a493-b193bdb73ded",
    parent_type: "project",
    parent_uuid: "634f90d9-da62-4c2f-a238-7cc1c1a055a5",
    parent_bref: "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
      }
    },
    ...
  },
  ...
}
```

Service Instance Health Check

In Contrail Release 3.0 and greater, a service instance health check can be used to determine the liveness of a service provided by a virtual machine (VM).

- [Health Check Overview on page 367](#)
- [Health Check Object Configuration on page 367](#)
- [Using the Health Check on page 368](#)
- [Health Check Process on page 369](#)

Health Check Overview

The service instance health check is used to determine the liveness of a service provided by a VM, checking whether the service is operationally up or down. The vRouter agent uses ping and an HTTP URL to the link-local address to check the liveness of the interface.

If the health check determines that a service is no longer operational, it removes the routes for the VM, thereby disabling packet forwarding to the VM.

The service instance health check is used with service template Version 2.

Health Check Object Configuration

Table 32 on page 368 shows the configurable properties of the health check object.

Table 32: Health Check Configurable Fields

Field	Description
- enabled	Indicates that health check is enabled. The default is False .
- health-check-type	Indicates the health check type: link-local or end-to-end . The default is link-local .
- monitor-type	The protocol type to be used: PING or HTTP .
- delay	The delay, in seconds, to repeat the health check.
- timeout	The 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.

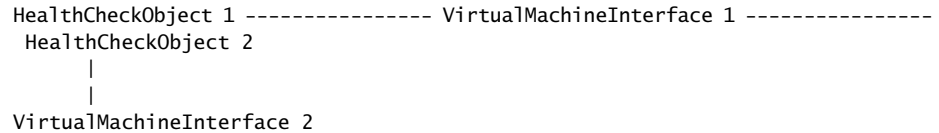
Restrictions include:

- 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.

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:



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.

Adding Physical Network Functions in Service Chains

- [Using Physical Network Functions in Contrail Service Chains on page 371](#)
- [Example: Adding a Physical Network Function Device to a Service Chain on page 372](#)

Using Physical Network Functions in Contrail Service Chains

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 on page 371](#)
- [Prerequisites and Assumptions on page 372](#)

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 on page 372](#)

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

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 on page 373](#)

Prerequisites for Adding a PNF to a Service Chain

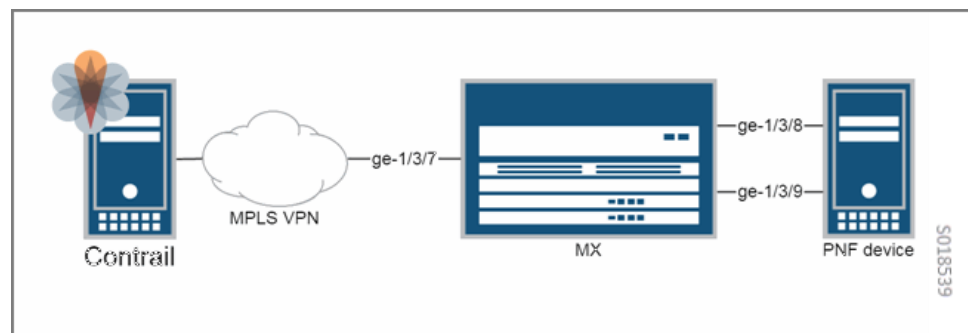
Prerequisites The following are the minimum requirements needed before you can add a PNF to a service chain using the procedure shown in the example included in this topic:

- 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;
      }
    }
  }
}

```

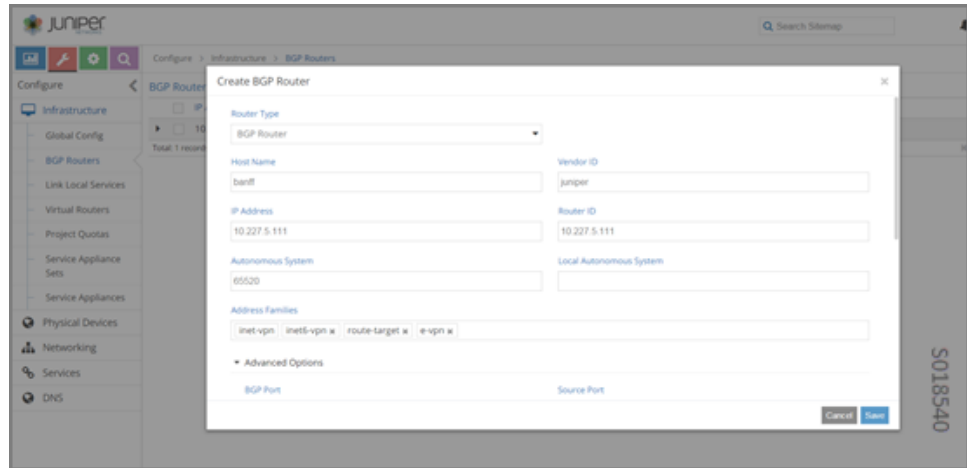
```
    }  
  }
```

If the MX is a service control gateway (SCG), the following configuration must also be present to support the service subscriptions:

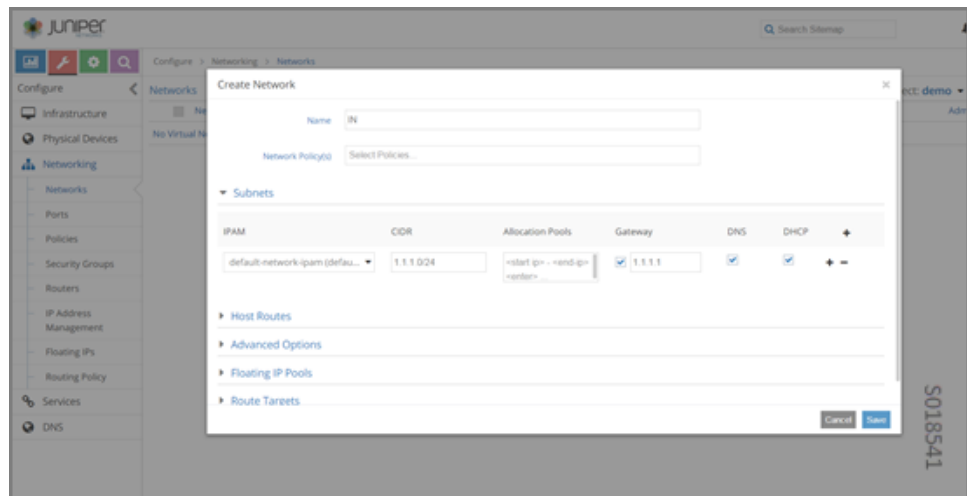
```
  firewall {  
    family inet {  
      filter skip_tdf_service {  
        term term1 {  
          then {  
            skip-services;  
            accept;  
          }  
        }  
      }  
    }  
  }  
}  
routing-instances {  
  <*-sc-entry-point> {  
    forwarding-options {  
      family 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.

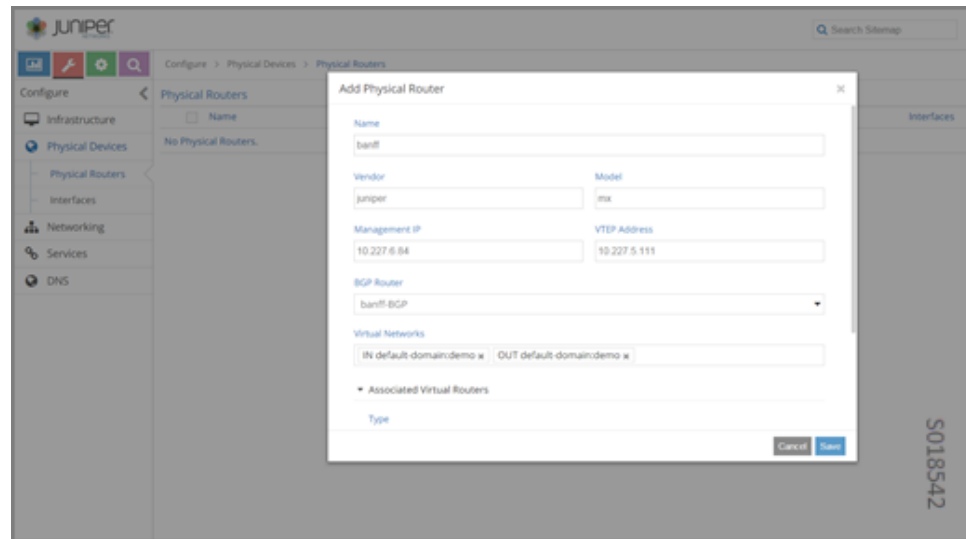


2. Create two virtual networks. Select **Configure > Networking > Networks** and create a network named **IN** and a network named **OUT**. The following figure provides an example.

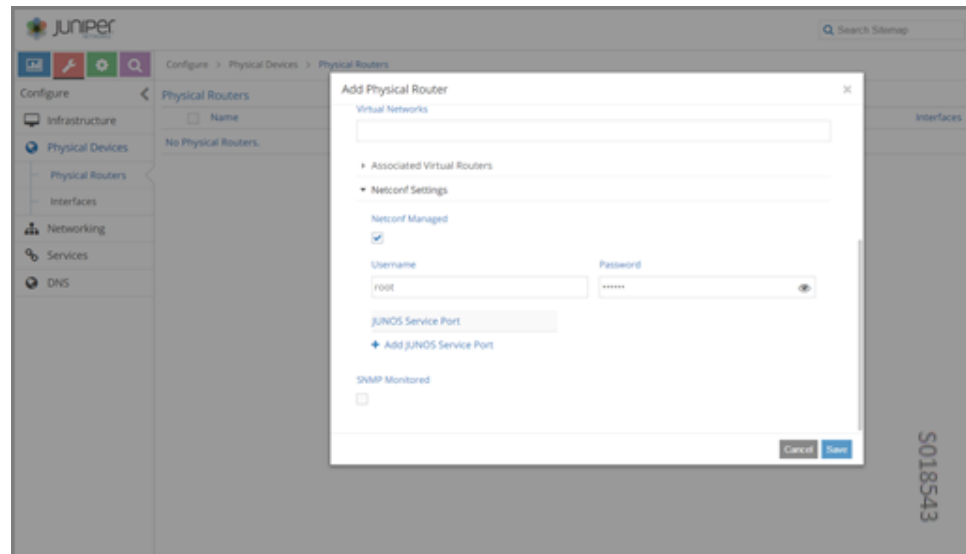


3. Create a physical router associated with the BGP router. Select **Configure > Physical Devices > Physical Routers** and create a physical router. The VTEP address of the physical router should be same as the BGP router's IP address. Associate the physical router with the BGP router created previously, and select for Virtual Networks the

networks created for this example (**IN** and **OUT**). The following figure provides an example.

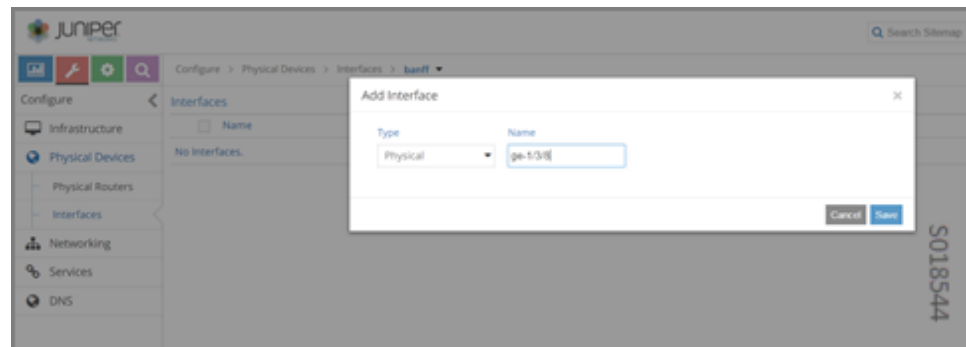


4. While still on the **Add Physical Router** window, use the slider to scroll down to the **Netconf Settings** section and add the appropriate NETCONF information for your system. The following figure provides an example.

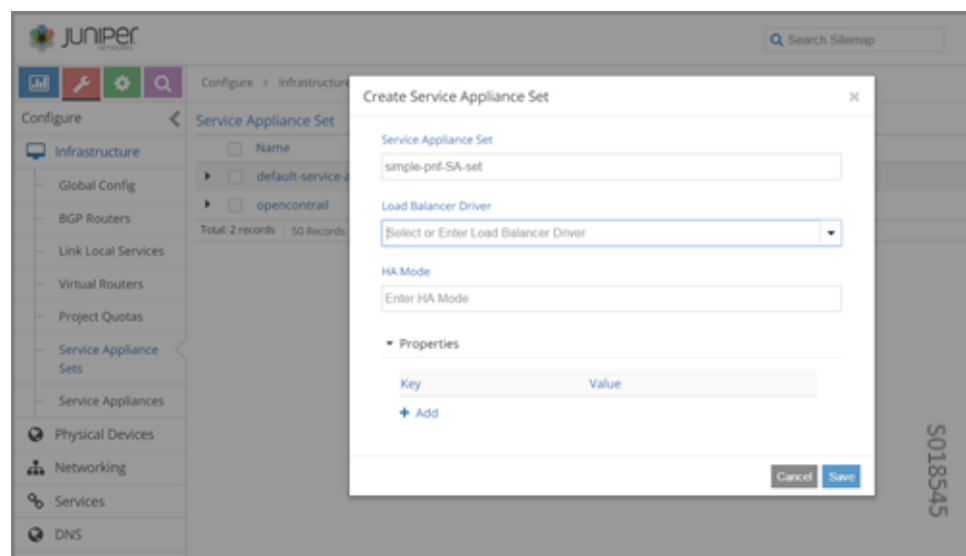


5. Add the physical interfaces that connect to the PNF device. Go to **Configure > Physical Devices > Interfaces** and select the PNF to get to the **Add Interfaces** window, where

you enter the name and type for each interface. The following figure provides an example.

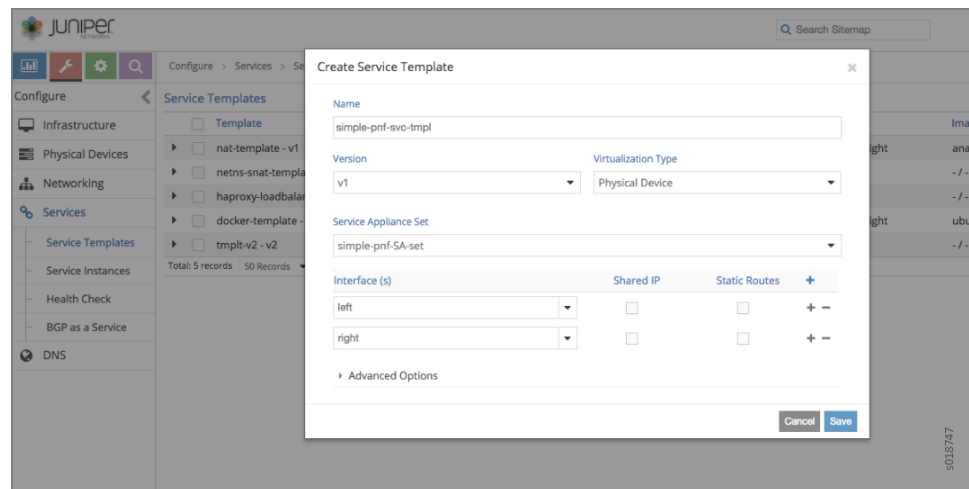


6. Add a service appliance set. Go to **Configure > Infrastructure > Service Appliance Sets** to get to the **Create Service Appliance Set** window, where you enter the name of the service appliance set. The following figure provides an example.

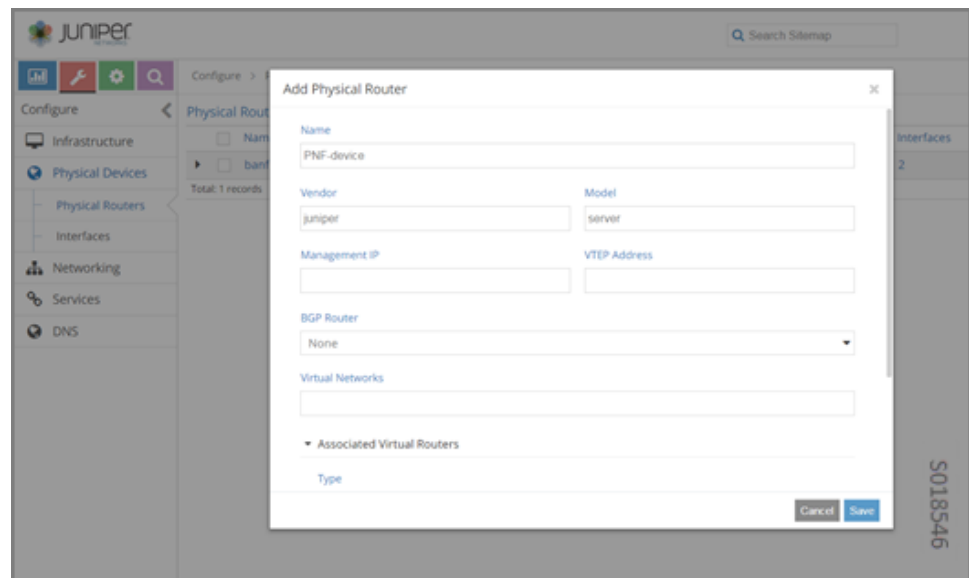


7. Configure a service template, **Configure > Services > Service Templates** and click the **Create** button on **Service Templates** to get to **Add Service Template**. Ensure that the

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.

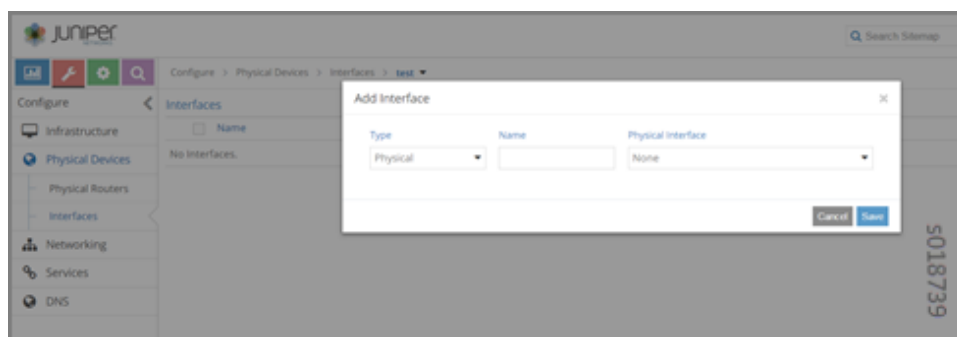


8. Add a physical router that represents the PNF device. Go to **Configure > Physical Devices > Physical Routers** to get to the **Add Physical Router** window, where you enter a name for the physical router. The following figure provides an example.

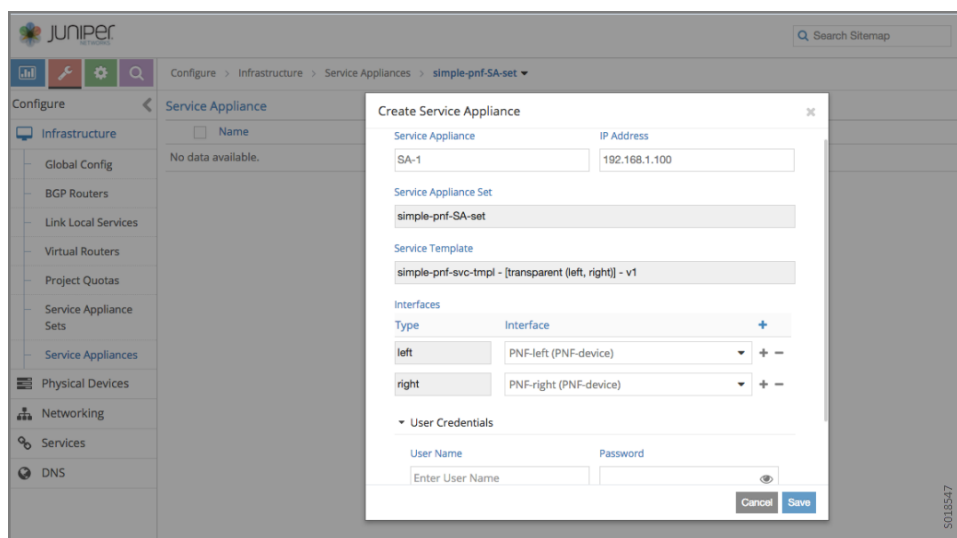


9. Create two interfaces for the PNF. The interfaces should connect to the interfaces already created in this example, and should connect in the manner illustrated in the

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 on page 319](#)
- [Example: Creating a Transparent Service Chain on page 333](#)

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 on page 319](#)
- [Example: Creating a Transparent Service Chain on page 333](#)
- [Using Physical Network Functions in Contrail Service Chains on page 371](#)

CHAPTER 15

Load Balancers

- [Using Load Balancers in Contrail 3.0 and Greater on page 381](#)
- [Configuring Load Balancing as a Service in Contrail on page 388](#)

Using Load Balancers in Contrail 3.0 and Greater

As of Contrail Release 3.00, new LBaaS features are available in the following:

- [Invoking LBaaS Drivers on page 381](#)
- [Using a Service Appliance Set as the LBaaS Provider on page 383](#)
- [Understanding the Load Balancer Agent on page 384](#)
- [F5 Networks Load Balancer Integration in Contrail on page 384](#)
- [F5 Load Balancer Global Routed Mode on page 385](#)
- [Traffic Flow in Global Routed Mode on page 385](#)
- [Routing Traffic to Pool Members on page 386](#)
- [Routing Reverse Traffic from Pool Members to the F5 Device on page 386](#)
- [Initial Configuration on an F5 Device on page 387](#)
- [Initial Configuration on an MX Series Device Used as DC Gateway on page 387](#)
- [Configuration on MX Device for Each Pool Member on page 387](#)
- [Example: Creating a Load Balancer on page 387](#)

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.

Current supported load balancer drivers include:

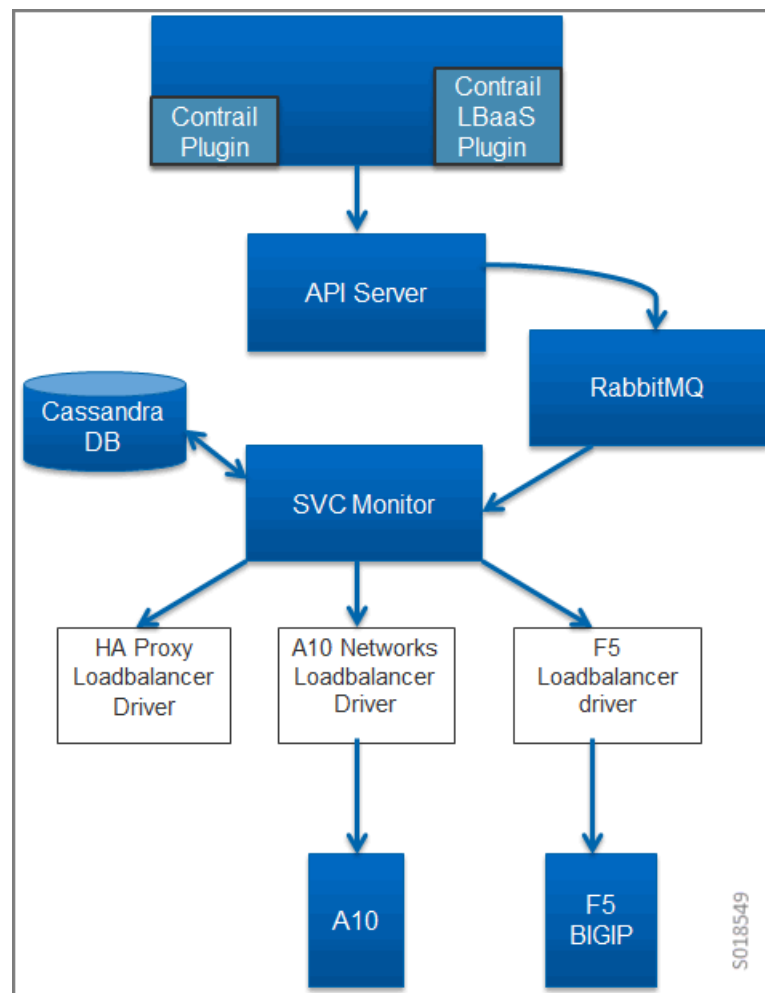
- HAProxy
- A10 Networks
- F5 Networks

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.

The following figure provides an overview of the LBaaS components as of Contrail 3.0.



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 F5 Networks 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

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.

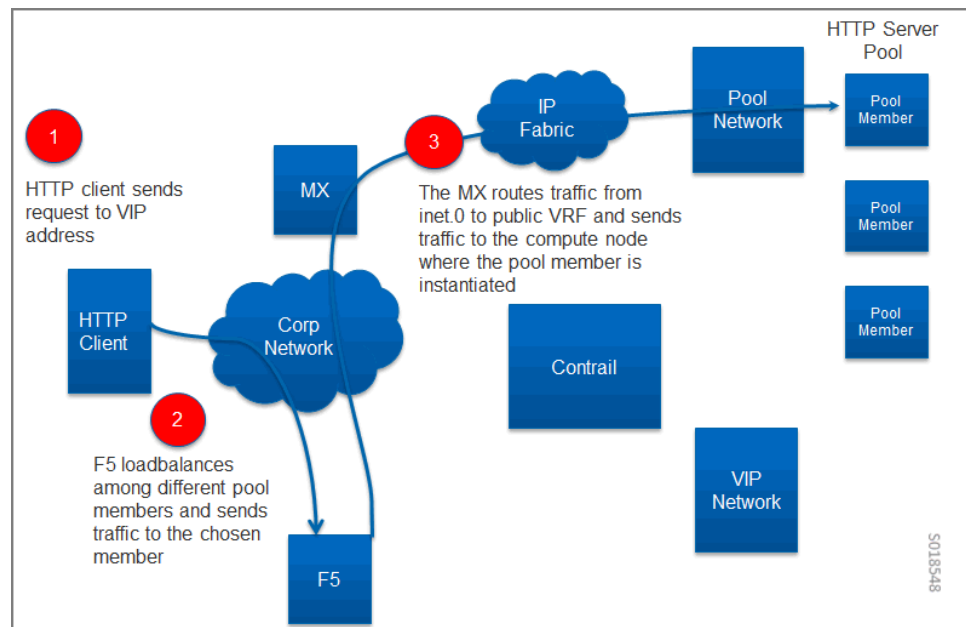
The F5 device is preprovisioned with following:

- 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.

The global routed traffic flow is shown in the following figure.



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": "vlan2"}'
```

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
```

Related Documentation

- [Configuring Load Balancing as a Service in Contrail on page 388](#)
- [Support for OpenStack LBaaS Version 2.0 APIs](#)

Configuring Load Balancing as a Service in Contrail

- [Overview: Load Balancing as a Service on page 388](#)
- [Contrail LBaaS Implementation on page 389](#)

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.

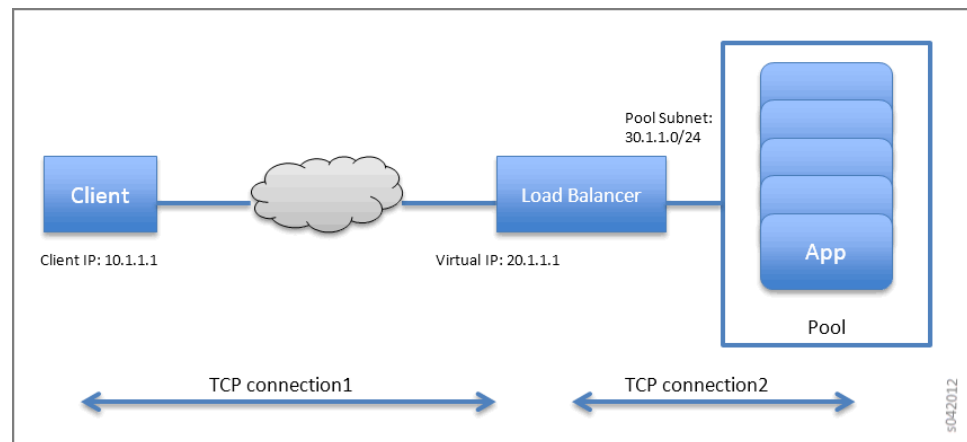
The LBaaS load balancer enables the creation of a pool of virtual machines serving applications, all front-ended by a virtual-ip. The LBaaS implementation has the following features:

- 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 vrouters.

Example: Configuring LBaaS This feature is enabled on Contrail through Neutron API calls. The following is an example of creating a pool network and a VIP network. 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.

- Create a healthmonitor.

```
neutron lb-healthmonitor-create --delay 20 --timeout 10 --max-retries 3 --type HTTP
```

- Associate a healthmonitor to a pool.

```
neutron lb-healthmonitor-associate <healthmonitor-uuid> mypool
```

- 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
```

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.

- Related Documentation**
- [Using Load Balancers in Contrail 3.0 and Greater on page 381](#)
 - *Support for OpenStack LBaaS Version 2.0 APIs*

CHAPTER 16

Configuring High Availability

- [Juniper OpenStack High Availability on page 395](#)
- [High Availability Support on page 404](#)
- [Example: Adding New OpenStack or Contrail Roles to an Existing High Availability Cluster on page 407](#)

Juniper OpenStack High Availability

- [Introduction on page 395](#)
- [Contrail High Availability on page 396](#)
- [OpenStack High Availability on page 396](#)
- [Supported Platforms on page 396](#)
- [Juniper OpenStack High Availability Architecture on page 396](#)
- [Juniper OpenStack Objectives on page 397](#)
- [Limitations on page 397](#)
- [Solution Components on page 398](#)
- [Virtual IP with Load Balancing on page 398](#)
- [Failure Handling on page 398](#)
- [Deployment on page 399](#)
- [Minimum Hardware Requirement on page 399](#)
- [Compute on page 399](#)
- [Network on page 400](#)
- [Installation on page 400](#)
- [Testbed File for Fab on page 401](#)

Introduction

The Juniper Networks software-defined network (SDN) controller has two major components: OpenStack and Contrail. High availability (HA) of the controller requires that both OpenStack and Contrail are resistant to failures. Failures can range from a service instance failure, node failure, link failure, to all nodes down due to a power outage. The basic expectation from a highly available SDN controller is that when failures occur, already provisioned workloads continue to work as expected without any traffic drop,

and the controller is available to perform operations on the cluster. Juniper Networks OpenStack is a distribution from Juniper Networks that combines OpenStack and Contrail into one product.

Contrail High Availability

Contrail has high availability already built into various components, including support for the Active-Active model of high availability, which works by deploying the Contrail node component with an appropriate required level of redundancy.

The Contrail control node runs BGP and maintains adjacency with the vRouter module in the compute nodes. Additionally, every vRouter maintains a connection with all available control nodes.

Contrail uses Cassandra as the database. Cassandra inherently supports fault tolerance and replicates data across the nodes participating in the cluster.

A highly available deployment of Contrail, at minimum, requires at least:

- **two** control nodes
- **three** config nodes (including analytics and webui)
- **three** database nodes

OpenStack High Availability

High availability of OpenStack is supported by deploying the OpenStack controller nodes in a redundant manner on multiple nodes. Previous releases of Contrail supported only a single instance of the OpenStack controller, and multiple instances of OpenStack posed new problems that needed to be solved, including:

- State synchronization of stateful services (e.g. MySQL) across multiple instances.
- Load-balancing of requests across the multiple instances of services.

Supported Platforms

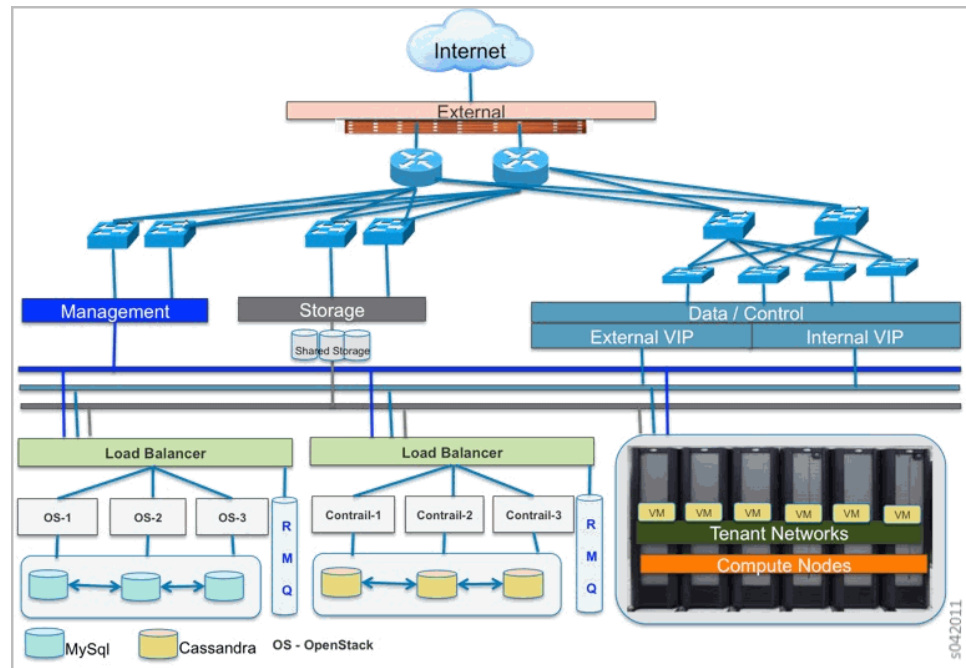
Juniper OpenStack Controller has tested high availability on the following platforms:

- Linux — Ubuntu 12.04 with kernel version 3.13.0-34
- OpenStack Havana

Juniper OpenStack High Availability Architecture

A typical cloud infrastructure deployment consists of a pool of resources of compute, storage, and networking infrastructure, all managed by a cluster of controller nodes.

The following figure illustrates a high-level reference architecture of a high availability deployment using Juniper OpenStack deployed as a cluster of controller nodes.



Juniper OpenStack Objectives

The main objectives and requirements for Juniper OpenStack high availability are:

- 99.999% availability for tenant traffic.
- Anytime availability for cloud operations.
- Provide VIP-based access to the API and UI services.
- Load balance network operations across the cluster.
- Management and orchestration elasticity.
- Failure detection and recovery.

Limitations

The following are limitations of Juniper OpenStack high availability:

- Only one failure is supported.
- During failover, a REST API call may fail. The application or user must reattempt the call.
- Although zero packet drop is the objective, in a distributed system such as Contrail, a few packets may drop during ungraceful failures.
- Juniper OpenStack high availability is not tested with any third party load balancing solution other than HAProxy.

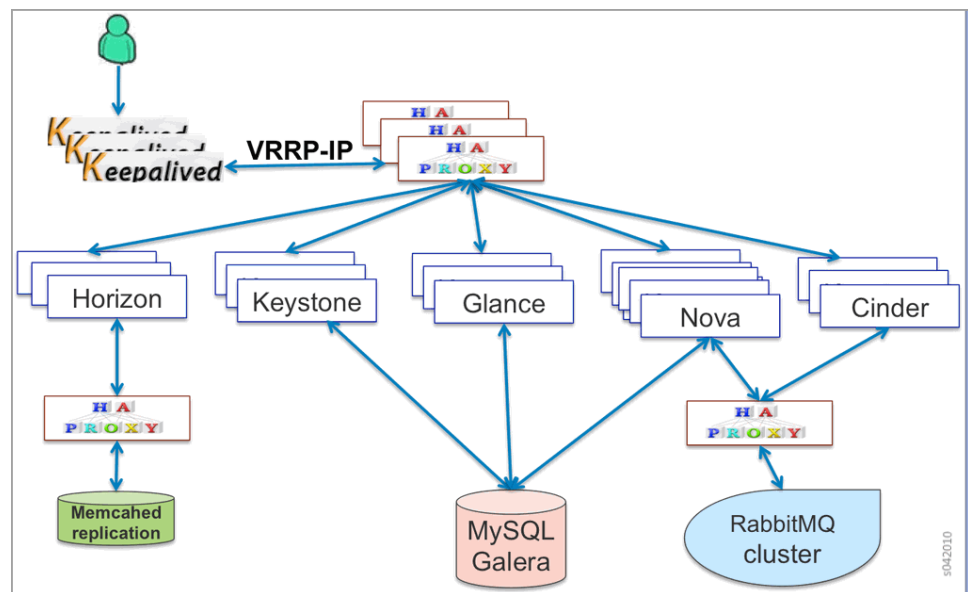
Solution Components

Juniper Openstack's high availability active-active model provides scale out of the infrastructure and orchestration services. The model makes it very easy to introduce new services in the controller and in the orchestration layer.

Virtual IP with Load Balancing

HAProxy is run on all nodes to load balance the connections across multiple instances of the services. To provide a Virtual IP (VIP), Keepalived (open source health check framework and hot standby protocol) runs and elects a master based on VRRP protocol. The VRRP master owns the VIP. If the master node fails, the VIP moves to a new master elected by VRRP.

The following figure shows OpenStack services provisioned to work with HAProxy and Keepalived, with HAProxy at the front of OpenStack services in a multiple operating system node deployment. The OpenStack database is deployed in clustered mode and uses Galera for replicating data across the cluster. RabbitMQ has clustering enabled as part of a multinode Contrail deployment. The RabbitMQ configuration is further tuned to support high availability.



Failure Handling

This section describes how various types of failures are handled, including:

- Service failures
- Node failures
- Networking failures

Service Failures

When an instance of a service fails, HAProxy detects the failure and load balances any subsequent requests across other active instances of the service. The supervisor process monitors for service failures and brings up the failed instances. As long as there is one instance of a service operational, the Juniper OpenStack controller continues to operate. This is true for both stateful and stateless services across Contrail and OpenStack.

Node Failures

The Juniper OpenStack controller supports single node failures involving both graceful shutdown or reboots and ungraceful power failures. When a node that is the VIP master fails, the VIP moves to the next active node, as it is elected to be the VRRP master. HAProxy on the new VIP master sprays the connections over to the active service instances as before, while the failed down node is brought back online. Stateful services (MySQL, Galera, Zookeeper, and so on) require a quorum to be maintained when a node fails. As long as a quorum is maintained, the controller cluster continues to work without problems. Data integrity is also inherently preserved by Galera, Rabbit, and other stateful components in use.

Network Failures

A connectivity break, especially in the control data network causes the controller cluster to partition into two. As long as the caveat of minimum number of nodes is maintained for one of the partitions, the controller cluster continues to work. Stateful services detect the partitioning and reorganize their cluster around the reachable nodes. Existing workloads continue to function and pass traffic and new workloads can be provisioned. When the connectivity is restored, the joining node becomes part of the working cluster and the system gets restored to its original state.

Deployment

Minimum Hardware Requirement

A minimum of 3 servers (physical or virtual machines) are required to deploy a highly available Juniper OpenStack Controller. In Active-Active mode, the controller cluster uses Quorum-based consistency management for guaranteeing transaction integrity across its distributed nodes. This translates to the requirement of deploying $2n+1$ nodes to tolerate n failures.

Juniper OpenStack Controller offers variety of deployment choices. Depending on the use case, the roles can be deployed either independently or in some combined manner. The type of deployment determines the sizing of the infrastructure. The numbers below present minimum requirements across compute, storage, and network.

Compute

- Quad core Intel(R) Xeon 2.5 Gz or higher
- 32 GB or higher RAM for the controller hosts (increases with number of hypervisors being supported)
- Minimum 1 TB disk, SSD, HDD

Network

A typical deployment separates control data traffic from the management traffic.

- Dual 10 GE that is bonded (using LAG 802.3ad) for redundant control data connection.
- Dual 1 GE bonded (using LAG 802.3 ad) for redundant management connection.
- Single 10G and 1G can be used if link redundancy is not desired.

The deployment needs virtual IP (VIP) addresses from the networks in which the NICs participate, external VIP on the management network and internal VIP on the control data network. External facing services are load balanced using the external VIP and the internal VIP is used for communication between other services.

Packaging

High availability support requires new components in the Contrail OpenStack deployment, which are packaged in **contrail-openstack-ha**, including HAProxy, Keepalived, Galera, and their requisite dependencies.

Installation

Installation is supported through fabric (fab) scripts. The testbed.py has new sections to incorporate external and internal VIPs. If OpenStack and Contrail roles are co-located on the nodes, only one set of external and internal VIPs is needed.

Install also supports separating OpenStack and Contrail roles on physically different servers. In this case, the external and internal VIPs specified are used for the OpenStack controller, and a separate set of VIPs, **contrail_external_vip** and **contrail_internal_vip**, are used for the Contrail controller nodes. It is also possible to specify separate RabbitMQs for OpenStack and Contrail controllers.

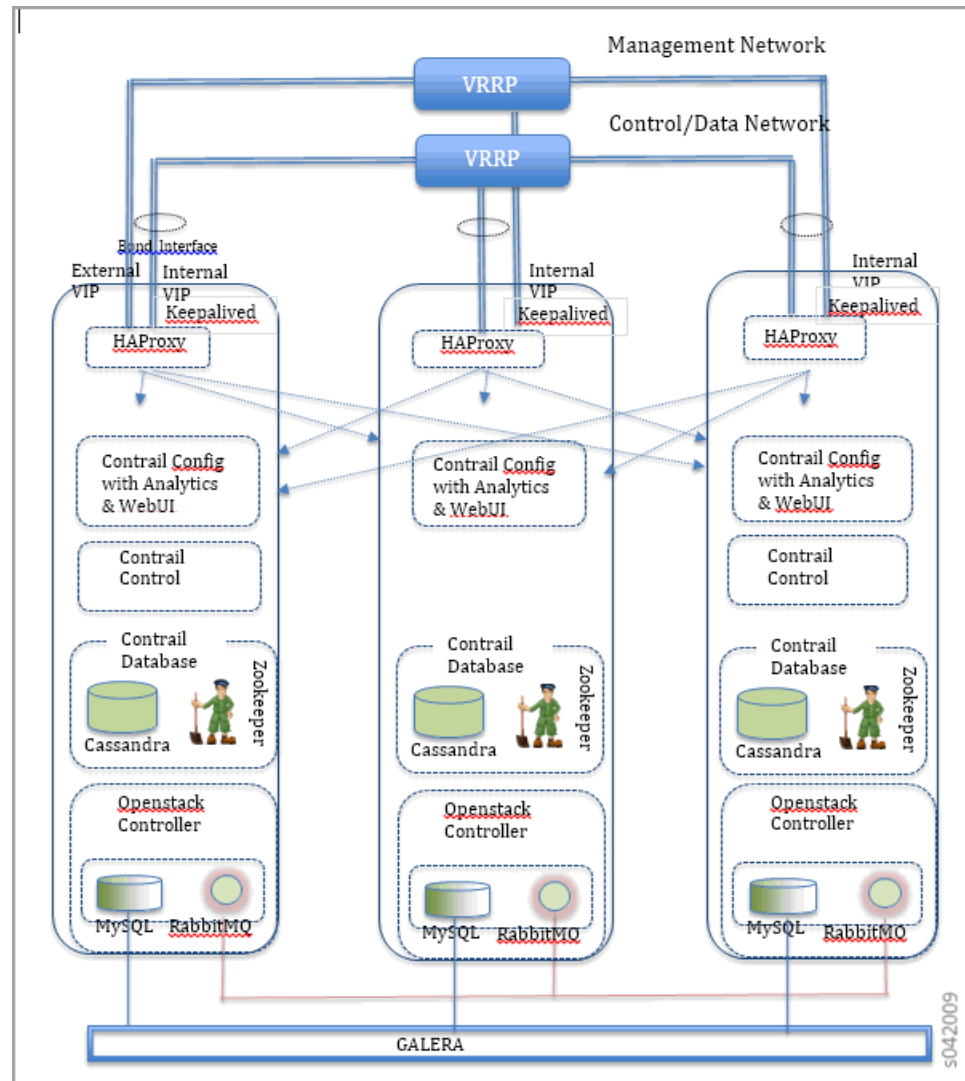
When multiple OpenStack roles are specified along with VIPs, the install-contrail target treats the installation as a high availability install and adds the **contrail-openstack-ha** package.

Similarly, **setup_all** treats the setup as a Contrail high availability setup and provisions the following services using the listed fab tasks.

- Keepalived — **fab setup_keepalived**
- high availability proxy — **fab fixup_restart_haproxy_in_openstack**
- Galera — **fab setup_galera_cluster**, **fab fix_wsrep_cluster_address**
- Glance — **fab setup_glance_images_loc**
- Keystone — **fab sync_keystone_ssl_certs**

Additionally, the provisioning scripts are changed to use VIPs instead of the physical IP of the node in all OpenStack and Contrail configuration files. The following figure shows

a typical three-node deployment, where Openstack and Contrail roles are co-located on three servers.



Testbed File for Fab

A sample testbed.py file is available at:

https://github.com/Juniper/contrail-fabric-utils/blob/R1.10/fabfile/testbeds/testbed_multibox_example.py

You can use the sample file by uncommenting and changing the high availability section to match your deployment.

The content of the sample testbed.py file for the minimum high availability configuration is the following.

```
from fabric.api import env

#Management ip addresses of hosts in the cluster
```

```
host1 = 'user@<ip address>'
host2 = 'user@<ip address>'
host3 = 'user@<ip address>'
host4 = 'user@<ip address>'
host5 = 'user@<ip address>'

#External routers if any
#for eg.
#ext_routers = [('mx1', <ip address>3')]
ext_routers = [('mx1', 'n.n.n.1')]

public_vn_rtgt = 20000
public_vn_subnet = <subnet address>8"

#Autonomous system number
router_asn = <asn number>2

#Host from which the fab commands are triggered to install and provision
host_build = 'user@<ip address>'

#Role definition of the hosts.
env.roldefs = {
    'all': [host1, host2, host3, host4, host5],
    'cfgm': [host1, host2, host3],
    'openstack': [host1, host2, host3],
    'control': [host2, host3 ],
    'compute': [host4, host5 ],
    'collector': [host1, host2, host3],
    'webui': [host1, host2, host3],
    'database': [host1, host2, host3],
    'build': [host_build],
}

env.hostnames = {
    'all': ['vse2100-2', 'vse2100-3', 'vse2100-4', 'vse2100-5', 'vse2100-6']
}

#Openstack admin password
env.openstack_admin_password = '<password>'

env.password = '<password>'
#Passwords of each host
env.passwords = {
    host1: '<password>',
    host2: '<password>',
    host3: '<password>',
    host4: '<password>',
    host5: '<password>',
    host_build: '<password>',
}

#For reimage purpose
env.ostypes = {
    host1: 'ubuntu',
    host2: 'ubuntu',
    host3: 'ubuntu',
    host4: 'ubuntu',
    host5: 'ubuntu',
}

#OPTIONAL BONDING CONFIGURATION
```

```

#=====
#Interface Bonding
#OPTIONAL BONDING CONFIGURATION
#=====
#Interface Bonding
bond= {
    host1 : { 'name': 'bond0', 'member': ['eth1','eth2'], 'mode':'802.3ad' },
    host2 : { 'name': 'bond0', 'member': ['eth1','eth2'], 'mode':'802.3ad' },
    host3 : { 'name': 'bond0', 'member': ['eth1','eth2'], 'mode':'802.3ad' },
    host4 : { 'name': 'bond0', 'member': ['eth1','eth2'], 'mode':'802.3ad' },
    host5 : { 'name': 'bond0', 'member': ['eth1','eth2'], 'mode':'802.3ad' },
}

#OPTIONAL SEPARATION OF MANAGEMENT AND CONTROL + DATA
#=====
#Control Interface
control_data = {
    host1 : { 'ip': '<ip address>', 'gw' : '<ip address>', 'device':'bond0'
},
    host2 : { 'ip': '<ip address>', 'gw' : '<ip address>', 'device':'bond0' },
    host3 : { 'ip': '<ip address>', 'gw' : '<ip address>', 'device':'bond0' },
    host4 : { 'ip': '<ip address>', 'gw' : '<ip address>', 'device':'bond0'
},
    host5 : { 'ip': '<ip address>', 'gw' : '<ip address>', 'device':'bond0'
},
}

# VIP
env.ha = {
    'internal_vip' : '<ip address>',
    'external_vip' : '<ip address>'
}

#To disable installing contrail interface rename package
env.interface_rename = False

#To enable multi-tenancy feature
#multi_tenancy = True

#To Enable parallel execution of task in multiple nodes
do_parallel = True

# To configure the encapsulation priority. Default: MPLSoGRE
#env.encap_priority = "'MPLSoUDP','MPLSoGRE','VXLAN'"

```



NOTE: The management interface configuration occurs outside of fab, so if the user needs a bond interface, the user needs to create a bond and assign the management NICs to it.

The management network must be a routable network.

- Related Documentation**
- [High Availability Support on page 404](#)
 - [Example: Adding New OpenStack or Contrail Roles to an Existing High Availability Cluster on page 407](#)

High Availability Support

- In Ubuntu setups, OpenStack high availability and Contrail high availability are both supported, for Contrail Release 1.10 and greater.
- In CentOS setups, only Contrail high availability is supported, and only for Contrail Release 1.20 and greater.

This section describes how to set up Contrail options for high availability support.

- [Contrail High Availability Features on page 404](#)
- [Configuration Options for Enabling Contrail High Availability on page 405](#)
- [Supported Cluster Topologies for High Availability on page 405](#)
- [Deploying OpenStack and Contrail on the Same High Available Nodes on page 405](#)
- [Deploying OpenStack and Contrail on Different High Available Nodes on page 406](#)
- [Deploying Contrail Only on High Available Nodes on page 406](#)

Contrail High Availability Features

The Contrail OpenStack high availability design and implementation provides:

- A high availability active-active implementation for scale-out of the cloud operation and for flexibility to expand the controller nodes to service the compute fabric.
- Anytime availability of the cloud for operations, monitoring, and workload monitoring and management.
- Self-healing of the service and states.
- VIP-based access to the cloud operations API provides an easy way to introduce new controllers and an API to the cluster with zero downtime. Improved capital efficiencies compared with dedicated hardware implementations, by using nodes assigned to controllers and making them a federated node in the cluster.
- Operational load distribution across the nodes in the cluster.

For more details about high availability implementation in Contrail, see [“High Availability Support” on page 404](#).

Configuration Options for Enabling Contrail High Availability

The following are options available to configure high availability within the Contrail configuration file (**testbed.py**).

Option	Description
internal_vip	The virtual IP of the OpenStack high availability nodes in the control data network. In a single interface setup, the internal_vip will be in the management data control network.
external_vip	The virtual IP of the OpenStack high availability nodes in the management network. In a single interface setup, the external_vip is not required.
contrail_internal_vip	The virtual IP of the Contrail high availability nodes in the control data network. In a single interface setup, the contrail_internal_vip will be in the management data control network.
contrail_external_vip	The virtual IP of the Contrail high availability nodes in the management network. In a single interface setup, the contrail_external_vip is not required.
nfs_server	The IP address of the NFS server that will be mounted to /var/lib/glance/images for the openstack node. The default is to env.roledefs['compute'][0] .
nfs_glance_path	The NFS server path to save images. The default is to /var/tmp/glance-images/ .
manage_amqp	A flag to tell the setup_all task to provision separate rabbitmq setups for openstack services in openstack nodes.

Supported Cluster Topologies for High Availability

This section describes configurations for the cluster topologies supported, including:

- OpenStack and Contrail on the same high available nodes
- OpenStack and Contrail on different high available nodes
- Contrail only on high available nodes

Deploying OpenStack and Contrail on the Same High Available Nodes

OpenStack and Contrail services can be deployed in the same set of high available nodes by setting the **internal_vip** parameter in the **env.ha** dictionary of the **testbed.py**.

Because the high available nodes are shared by both OpenStack and Contrail services, it is sufficient to specify only **internal_vip**. However, if the nodes have multiple interfaces with management and data control traffic separated by provisioning multiple interfaces, then the **external_vip** also needs to be set in the **testbed.py**.

Example

```
env.ha = {
    'internal_vip' : 'an-ip-in-control-data-network',
```

```
        'external_vip' : 'an-ip-in-management-network',  
    }  
}
```

Deploying OpenStack and Contrail on Different High Available Nodes

OpenStack and Contrail services can be deployed on different high available nodes by setting the **internal_vip** and the **contrail_internal_vip** parameter in the **env.ha** dictionary of the **testbed.py**.

Because the OpenStack and Contrail services use different high available nodes, it is required to separately specify **internal_vip** for OpenStack high available nodes and **contrail_internal_vip** for Contrail high available nodes. If the nodes have multiple interfaces, with management and data control traffic separated by provisioning multiple interfaces, then the **external_vip** and **contrail_external_vip** options also must be set in the **testbed.py**.

Example

```
env.ha = {  
    'internal_vip' : 'an-ip-in-control-data-network',  
    'external_vip' : 'an-ip-in-management-network',  
    'contrail_internal_vip' : 'another-ip-in-control-data-network',  
    'contrail_external_vip' : 'another-ip-in-management-network',  
}
```

To manage separate **rabbitmq** clusters in the OpenStack high available nodes for OpenStack services to communicate, specify **manage_amqp** in the **env.openstack** dictionary of **testbed.py**. If **manage_amqp** is not specified, the default is for the OpenStack services to use the **rabbitmq** cluster available in the Contrail high available nodes for communication.

Example:

```
env.openstack = {  
    'manage_amqp' : 'yes'  
}
```

Deploying Contrail Only on High Available Nodes

Contrail services can be deployed only on a set of high available nodes by setting the **contrail_internal_vip** parameter in the **env.ha** dictionary of the **testbed.py**.

Because the high available nodes are used by only Contrail services, it is sufficient to specify only **contrail_internal_vip**. If the nodes have multiple interfaces with management and data control traffic are separated by provisioning multiple interfaces, the **contrail_external_vip** also needs to be set in the **testbed.py**.

Example

```
env.ha = {  
    'contrail_internal_vip' : 'an-ip-in-control-data-network',  
    'contrail_external_vip' : 'an-ip-in-management-network',  
}
```

```
}
```

To manage separate **rabbitmq** clusters in the OpenStack node for the OpenStack services to communicate, specify **manage_amqp** in the **env.openstack** dictionary of the **testbed.py**. If the **manage_amqp** is not specified, the default is the OpenStack services will use the cluster available in the Contrail high available nodes for communication.

Example:

```
env.openstack = {
    'manage_amqp' : 'yes'
}
```

- Related Documentation**
- [Juniper OpenStack High Availability on page 395](#)
 - [Example: Adding New OpenStack or Contrail Roles to an Existing High Availability Cluster on page 407](#)

Example: Adding New OpenStack or Contrail Roles to an Existing High Availability Cluster

This section provides an example of adding new nodes or roles to an existing cluster with high availability enabled. It is organized in the following:

- [Adding New OpenStack or Contrail Roles to an Existing High Availability Cluster on page 407](#)
- [Purging a Controller From an Existing Cluster on page 408](#)
- [Replacing a Node With a Node That Has the Same IP Address on page 409](#)
- [Known Limitations and Configuration Guidelines on page 409](#)
- [Understanding How the System Adds a New Node to an Existing Cluster on page 410](#)

Adding New OpenStack or Contrail Roles to an Existing High Availability Cluster

To add new nodes or roles to an existing cluster with high availability enabled:

1. Install the new server with the new base OS image (currently Ubuntu 14.04 is supported)
2. Download the latest Contrail installer package to the new server.
3. Run the **dpkg -I <contrail_install_deb.deb>** command on the new server.
4. Run the **/opt/contrail/contrail_packages/setup.sh** script in the new server to install the Contrail repository.

5. Modify the **testbed.py** file in the build server as follows:
 - a. Add the new host in the list of hosts as `host<x>`
 - b. In the **env.roledefs** list, add the new node as appropriate. For example, add the node in the **openstack** role list, so that the new node is configured as an OpenStack server. Every role can be added mutually exclusive of each other.
 - c. Add the hostname of the new server in the **env.hostnames** field.
 - d. Add the password for the new server in the **env.passwords** field.
 - e. Add the control data interface of the new server if applicable.
6. In the build server, go to the `/opt/contrail/utils` directory and use the **fab install_new_contrail:new_ctrl='root@<host_ip>'** command to install all the required packages in the new server based on the roles configured in the **testbed.py** file.
7. If you have two interfaces, the control and data interface and the management interface, then the control and data interface can be setup using the **fab setup_interface_node'root@<host_ip>'** command.
8. In the build server, go to the `/opt/contrail/utils` directory and use the **fab join_cluster:'root@<host_ip>'** command. This adds the new server to the corresponding cluster based on the role that is configured in the **testbed.py** file.

The new node is added to the existing cluster. See *Understanding How a New Node is Added to an Existing Cluster*.

Purging a Controller From an Existing Cluster

To purge a controller node from an existing cluster with high availability enabled:

1. Open the **testbed.py** file which contains the topology information of the cluster.
2. In the **env.roledefs** list, remove the role as appropriate. For example, if you need the node to cease existing as an OpenStack controller, remove the node from the **openstack** role list.



NOTE: Every role can be added mutually exclusive to each other. However, there are certain minimum node requirements for OpenStack and Database roles (at least 3 nodes) and if the size of the rest of the cluster does not meet these minimum requirements after purging the node, then deleting the node is not allowed.



NOTE: The node should not be deleted from the host list (or the passwords) but only from the **env.roledefs** list. The node can be removed from the host list after the purge operation is completed.

3. In the build server, go to the `/opt/contrail/utils` directory and use the **fab** `purge_node:'root@<ip_address>'` command. This removes all the configuration and stops the relevant services from the node that you need to purge.
4. Remove the rest of the configuration related to the node from the `testbed.py` file after the previous command completes.



CAUTION: In the event that the node that needs to be deleted is already down (non-operational), it should not be brought up again since it would join the cluster again with unknown consequences.

Replacing a Node With a Node That Has the Same IP Address

To replace a node in an existing cluster with high availability enabled with a node that has the same IP address:

1. Make sure that the cluster continues to operate when the node being replaced is taken off the cluster. (For example, it meets the minimum failure requirements)
2. Reimage the node that is being replaced and make sure it gets the same IP address.
3. Follow the steps to add a node to an existing cluster.

Known Limitations and Configuration Guidelines

The following lists some known limitations and some configuration guidelines for when you add new nodes or roles to an existing cluster with high availability enabled:

- Adding a new node to an existing high availability cluster is only supported in Contrail Release 2.21 and later.
- Converting a single node or cluster that does not have high availability enabled to a cluster that does have high availability enabled is not supported.
- New nodes must be appended to the existing list of nodes at the end of the `testbed.py` lists.
- We recommend you maintain a cluster with an odd number of controllers since high availability is based on a quorum and to support n failures, $(2n + 1)$ nodes are required.
- The new node is required to be running the same release as of the other nodes in the cluster.
- You need to use the **nodetool cleanup** command after a new node joins the Cassandra cluster. You can safely schedule this for low usage hours to prevent disruption of the cluster operation.
- While deleting a node from an existing cluster, the remaining cluster should be operational and meet the high availability cluster requirements, if not, purging the node is not allowed.

Understanding How the System Adds a New Node to an Existing Cluster

The following lists the actions the system takes when adding a new node to an existing cluster with high availability enabled:

- If a new OpenStack server is configured, the system:
 - Adds the new node as a participant in the VRRP mastership election.
 - Generates a keepalived configuration similar to other nodes in the existing cluster.
 - Restarts the keepalived process in all nodes so the configuration takes effect.
 - Modifies the haproxy configuration in all the existing nodes to add the new node as another backend for roles like keystone, nova, glance, and cinder.
 - Restarts the haproxy process in all the nodes so that the new configuration takes effect.
 - Modifies the mysql configuration in the `/etc/mysql/conf.d/wsrep.conf` file to add the new node into the Galera cluster in all the existing controllers.
 - Restarts MySQL in every node sequentially and waits until they come back up.
 - Generates a MySQL configuration similar to other existing controllers and waits until the new OpenStack server syncs all the data from the existing MySQL donors.
 - Installs CMON and generates the CMON configuration.
 - Adds a CMON user in all the MySQL databases so that CMON is able to monitor the MySQL server and regenerates the CMON configuration in all the nodes so that the CMON in the other nodes can also monitor the new MySQL database.
 - Generates `cmon_param` files based on the new configuration and invokes monitor scripts.
 - Instructs the new node to join the RabbitMQ cluster. (If RabbitMQ is not externally managed.)
 - Modifies the `rabbitmq.config` file in all the existing nodes to include the new node and restarts the RabbitMQ server sequentially so that the new node joins the cluster.
- If a new database node is configured, the system:
 - Generates the database configuration (`cassandra.yaml` file) for the new node and uses the `nodetool` command to make the new node join the existing cluster. (Uses the `fab setup_database_node` command)
 - Generates a new zookeeper configuration in the existing node, adds the new node into the existing zookeeper configuration, and restarts the zookeeper nodes sequentially.
 - Starts the Cassandra database in the new node.

- If a new Configuration Manager node is configured, the system:
 - Generates all the configuration required for a new Configuration Manager (cfgm) node and start the server (Uses the **fab setup_config_node** command)
 - Modifies the HAProxy configuration to add the new node into the existing list of backends.
 - If required, modifies the zookeeper and Cassandra server list in the **/etc/contrail/contrail-api.conf** file in the existing config nodes.
 - Restarts the config nodes so that the new configuration takes effect.
- If a new controller node is configured, the system:
 - Generates all the configuration required for a new control node and starts the control node. (Uses the **fab setup_control_node** command)
 - Instructs the control node to peer with the existing control nodes.

If a new collector, analytics, or WebUI node is configured, the system:

- Generates all the configuration required for a new collector node and starts the control node (Uses the **fab setup_collector_node** and **fab setup_webui_node** commands)
- Updates the existing configuration in all the new nodes to add a new database node if required.
- Starts the collector process in the new node.
- Starts the webui process in the new node.

**Related
Documentation**

- [High Availability Support on page 404](#)
- [Juniper OpenStack High Availability on page 395](#)

Configuring Multitenancy Support

- [Configuring Multitenancy Support on page 413](#)
- [Configuring Network QoS Parameters on page 415](#)

Configuring Multitenancy Support

The following sections describe enabling and viewing multitenancy support.

- [Multitenancy Permissions on page 413](#)
- [API Server on page 414](#)
- [API Library Keystone Integration on page 414](#)
- [Supporting Utilities on page 414](#)

Multitenancy Permissions

The multi tenancy feature of the API server enables multiple tenants to coexist on the system without interfering with each other. This is achieved by encoding ownership information and permissions with each resource, allowing fine-grained control over create, read, update, and delete (CRUD) operations on those resources.

The Contrail **api-server** enforces resources permissions in a manner similar to Unix files. Each resource has an owner and group. Permissions associated with owner, group, and "others" are:

R - reading resource

W - create/update resource

X - link (refer to) object

CRUD permission requirements for resources managed by **api-server** are as follows:

C - write on parent object

For example, to create a virtual network requires write permission on the project.

R - read on object (parent if a collection)

U - write on object

D - write on parent

ref(link) - execute on object

For example, on a virtual network using **network-ipam**, **network-ipam** should have X permissions for owner, group, or "others".

API Server

If multitenancy is enabled, **api-server** deploys keystone middleware in its pipeline. The keystone middleware architecture supports a common authentication protocol in use between OpenStack projects.

The keystone middleware works in conjunction with **api-server** to derive the user name and role for each incoming request. Once obtained, the user name and role are matched against resource ownership and permissions. If the ownership matches or the permissions allow access, access is granted.

For example, assume Tenant A has the following attributes:

- owner = Bob
- group = Staff
- permissions = 750

In this example, only Bob can create a virtual network in Tenant A. Other staff members can view the virtual networks in Tenant A. No others can create or view any virtual networks in Tenant A.

Clients can obtain an **auth_token** by posting credentials to the keystone admin API (**/v2.0/tokens**). The **VncApi** client library does this automatically. If an **auth_token** is present in an incoming request, **api-server** validates credentials derived from the token against object permissions. If an incoming request has an invalid or missing **auth_token**, a 401 error is returned.

Notes:

- Multitenancy is enabled by the flag **multi_tenancy** in **/etc/contrail/api-server.conf**
- If multitenancy is enabled, **memcaching** is automatically enabled, to improve token validation response time.

API Library Keystone Integration

VncApi has been updated to check for any 401 error that **api-server** returns as a result of a missing or invalid token. This forces **VncApi** to connect with the keystone middleware and fetch an **auth_token**. All subsequent requests to **api-server** include the **auth_token**.

Supporting Utilities

- **/opt/contrail/utlis/chmod.py**— To change permissions and ownership (user or group membership) of a resource. Requires the resource type (for example, **virtual-network**)

and the resource FQN (for example, **default-domain:default-project:default-virtual-network**).

Invoke **python /opt/contrail/utils/chmod.py -h** to see usage information

Example 1 - See current permissions:

```
[root@host]# python /opt/contrail/utils/chmod.py <ip address> project
default-domain:default-project
Type = project
Name = default-domain:default-project
API Server = <ip address>
Keystone credentials admin/<password>/admin
Obj uuid = 6765f112-938f-4251-b3a9-fbbdcc09db18
Obj perms = cloud-admin/cloud-admin-group 777

[root@host]# python /opt/contrail/utils/chmod.py <ip address> --owner foo
--group bar --perms 555 project default-domain:default-project
Type = project Name = default-domain:default-project
API Server = <ip address>
Owner = foo
Group = bar
Perms = 555
Keystone credentials admin/<password>/admin
Obj uuid = 6765f112-938f-4251-b3a9-fbbdcc09db18
Obj perms = cloud-admin/cloud-admin-group 777
New perms = foo/bar 555
```

- **/opt/contrail/utils/multi_tenancy.py** — Show if multitenancy is enabled or disabled. Also used to turn multitenancy on or off. Requires admin credentials.

Invoke **python /opt/contrail/utils/multi_tenancy.py -h** to see usage information

Example 1: View multitenancy status:

```
[root@host]# python /opt/contrail/utils/multi_tenancy.py <ip address>
API Server = <ip address>
Keystone credentials admin/<password>/admin

Multi Tenancy is enabled
```

Example 2: Turn multitenancy off:

```
[root@host]# python /opt/contrail/utils/multi_tenancy.py <ip address>--off
API Server = <ip address>
Keystone credentials admin/<password>/admin

Multi Tenancy is disabled
```

Configuring Network QoS Parameters

- [Overview on page 416](#)
- [QoS Configuration Examples on page 416](#)
- [Limitations on page 417](#)

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.

CHAPTER 18

Optimizing Contrail

- [vRouter Command Line Utilities on page 419](#)
- [Route Target Filtering on page 435](#)
- [Source Network Address Translation \(SNAT\) on page 437](#)

vRouter Command Line Utilities

- [Overview on page 419](#)
- [vif Command on page 420](#)
- [flow Command on page 422](#)
- [vrfstats Command on page 424](#)
- [rt Command on page 424](#)
- [dropstats Command on page 425](#)
- [mpls Command on page 429](#)
- [mirror Command on page 431](#)
- [vxlan Command on page 432](#)
- [nh Command on page 433](#)

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.

Command	Description
dropstats	Inspect packet drop counters in the router.
mpls	Display the input label map programmed into the router.
mirror	Display the mirror table entries.
vxlan	Display the vxlan table entries.
nh	Display the next hops that the router knows.
--help	Display all command options available for the current command.

The following sections describe each of the router utilities in detail.

vif Command

The router requires router interfaces (**vif**) to forward traffic. Use the **vif** command to see the interfaces that are known by the router.



NOTE: Having interfaces only in the OS (Linux) is not sufficient for forwarding. The relevant interfaces must be added to router. Typically, the set up of interfaces is handled by components like nova-compute or router 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 33: 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"> • Virtual – Interface of a virtual machine (VM). • Physical – Physical interface (NIC) in the system. • Host – An interface toward the host. • Agent – An interface used to trap packets to the vrouter agent when decisions need to be made for the forwarding path.
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"> • P - Policy • L3 - Layer 3 forwarding • L2 - Layer 2 bridging • 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. • Mt - Mirroring transmit direction • Mr - Mirroring receive direction • Tc - Checksum offload on the transmit side. Valid only on the physical interface.
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.
--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 vrouter 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 **vroute** 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 **Invalid Source** counter tracks the number of packets that came from an invalid or unexpected source and thus were dropped.

The remaining counters are of value only to developers.

mpls Command

The **mpls** utility command displays the input label map that has been programmed in the vrouter.

Example: mpls --dump The **--dump** command dumps the complete label map. The output is divided into two columns. The first field is the label and the second is the next hop corresponding to the label. When an MPLS packet with the specified label arrives in the vrouter, it uses the next hop corresponding to the label to forward the packet.

```
# mpls --dump
```

```
MPLS Input Label Map
```

Label	NextHop
16	9
17	11

You can inspect the operation on **nh 9** as follows:

```
# nh --get 9

Id:009  Type:Encap      Fmly: AF_INET  Flags:Valid, Policy,  Rid:0  Ref_cnt:4

      EncapFmly:0806 Oif:3 Len:14 Data:02 d0 60 aa 50 57 00 25 90 c3 08 69
      08 00
```

The nh output shows that the next hop directs the packet to go out on the interface with index 3 (**Oif:3**) with the given rewrite data.

To check the index of 3, use the following:

```
# vif -get 3

vif0/3  OS: tapd060aa50-57

      Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:0

      Vrf:1  Flags:PL3L2 MTU:9160 Ref:6

      RX packets:1056  bytes:103471 errors:0

      TX packets:1041  bytes:102372 errors:0
```

The **-get 3** output shows that the index of 3 corresponds to a tap interface that goes to a VM.

You can also dump individual entries in the map using the **--get** option, as follows:

```
# mpls -get 16

MPLS Input Label Map

      Label      NextHop
      -----
      16          9
```

Example: mpls -help

```
# mpls -help

Usage: mpls --dump

      mpls --get <label>

      mpls --help

--dump  Dumps the mpls incoming label map

--get    Dumps the entry corresponding to label <label>
         in the label map

--help   Prints this help message
```

mirror Command

Use the **mirror** command to dump the mirror table entries.

Example: Inspect Mirroring

The following example inspects a mirror configuration where traffic is mirrored from network **vn1 (1.1.1.0/24)** to network **vn2 (2.2.2.0/24)**. A ping is run from 1.1.1.253 to 2.2.2.253, where both IPs are valid VM IPs, then the flow table is listed:

```
# flow -l

Flow table

Index                Source:Port          Destination:Port      Proto(V)
-----
135024                2.2.2.253:1208       1.1.1.253:0          1 (1)
                        (Action:F, S(nh):17, Statistics:208/17472 Mirror Index : 0)

387324                1.1.1.253:1208       2.2.2.253:0          1 (1)
                        (Action:F, S(nh):8, Statistics:208/17472 Mirror Index : 0)
```

In the example output, **Mirror Index:0** is listed, it is the index to the mirror table. The mirror table can be dumped with the **--dump** option, as follows:

```
# mirror --dump

Mirror Table

Index  NextHop  Flags  References
-----
0      18       3
```

The mirror table entries point to next hops. In the example, the index 0 points to next hop 18. The **References** indicate the number of flow entries that point to this entry.

A next hop get operation on ID 18 is performed as follows:

```
# nh --get 18

Id:018  Type:Tunnel  Fmly: AF_INET  Flags:Valid, Udp,  Rid:0  Ref_cnt:2

  Oif:0 Len:14 Flags Valid, Udp,  Data:00 00 00 00 00 00 00 25 90 c3 08
  69 08 00

  Vrf:-1  Sip:192.168.1.10  Dip:250.250.2.253

  Sport:58818  Dport:8099
```

The **nh --get** output shows that mirrored packets go to a system with IP 250.250.2.253. The packets are tunneled as a UDP datagram and sent to the destination. **Vrf:-1** indicates that a lookup has to be done in the source **Vrf** for the destination.

You can also get an individual mirror table entry using the **--get** option, as follows:

```
# mirror --get 10

Mirror Table

Index      NextHop    Flags    References
-----
10         1          1        1
```

Example: mirror --help

```
# mirror --help

Usage: mirror --dump

        mirror --get <index>

        mirror --help

--dump  Dumps the mirror table

--get   Dumps the mirror entry corresponding to index <index>

--help  Prints this help message
```

vxlan Command

The vxlan command can be used to dump the vxlan table. The vxlan table maps a network ID to a next hop, similar to an MPLS table.

If a packet comes with a vxlan header and if the VNID is one of those in the table, the vrouter will use the next hop identified to forward the packet.

Example: vxlan --dump

```
# vxlan --dump

VXLAN Table

VNID      NextHop
-----
4         16
5         16
```

Example: vxlan --get You can use the **--get** option to dump a specific entry, as follows:

```
# vxlan --get 4

VXLAN Table

VNID      NextHop
-----
4         16
```



```

Example: vxlan --help      # vxlan --help

Usage: vxlan --dump

      vxlan --get <vnid>

      vxlan --help

--dump  Dumps the vxlan table

--get   Dumps the entry corresponding to <vnid>

--help  Prints this help message

```

nh Command

The **nh** command enables you to inspect the next hops that are known by the vrouter. Next hops tell the vrouter the next location to send a packet in the path to its final destination. The processing of the packet differs based on the type of the next hop. The next hop types are described in the following table.

Next Hop Type	Description
Receive	Indicates that the packet is destined for itself and the vrouter should perform Layer 4 protocol processing. As an example, all packets destined to the host IP will hit the receive next hop in the default VRF. Similarly, all traffic destined to the VMs hosted by the server and tunneled inside a GRE will hit the receive next hop in the default VRF first, because the outer packet that carries the traffic to the VM is that of the server.
Encap (Interface)	Used only to determine the outgoing interface and the Layer 2 information. As an example, when two VMs on the same server communicate with each other, the routes for each of them point to an encap next hop, because the only information needed is the Layer 2 information to send the packet to the tap interface of the destination VM. A packet destined to a VM hosted on one server from a VM on a different server will also hit an encap next hop, after tunnel processing.
Tunnel	Encapsulates VM traffic in a tunnel and sends it to the server that hosts the destination VM. There are different types of tunnel next hops, based on the type of tunnels used. Vrouter supports two main tunnel types for Layer 3 traffic: MPLSoGRE and MPLSoUDP. For Layer 2 traffic, a VXLAN tunnel is used. A typical tunnel next hop indicates the kind of tunnel, the rewrite information, the outgoing interface, and the source and destination server IPs.
Discard	A catch-all next hop. If there is no route for a destination, the packet hits the discard next hop, which drops the packet.
Resolve	Used by the agent to lazy install Layer 2 rewrite information.
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.

Next Hop Type	Description
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
```

Route Target Filtering

- [Introduction on page 435](#)
- [Debugging and Troubleshooting Route Target Filtering on page 436](#)
- [RTF Limitations in Contrail 1.10 on page 437](#)

Introduction

BGP route target filtering (RTF) is a method for limiting the distribution of VPN routes to only those systems in the network for which the routes are necessary. If RTF is not active, the Contrail control node advertises all VPN routes to all of its VPN peers, which are either other control nodes or gateway routers such as an MX Series router. On the receiving side, the control node stores all VPN routes it receives from peers in the VPN table (for example, `bgp.l3vpn.0`). Any routes that do not include a route target extended community that is referenced by the local `vrf-import` policies are discarded by Junos.

The control node must send all route updates to its peers, even for unnecessary routes that are discarded. Continuous route updates are both CPU- and memory-intensive. The only routes that are necessary to advertise to gateway routers are those that belong to the virtual networks that are configured for public access. It is not necessary to advertise VM routes belonging to other virtual networks to gateway routers.

If a datacenter has more than two control nodes, the `vrouter-agent` only subscribes to two of the control nodes, indicated by the discovery service. When a VM is initially launched

in a virtual network, it sends an XMPP subscribe request for the virtual network VRF and publishes the VM route to the connected control node. It is not necessary to advertise routes belonging to this type of VRF to control nodes that don't have the **vrouter-agent** subscribed in that VRF.

RTF is used to optimize the route distribution among control nodes and to the gateway routers to avoid unwanted route updates. If the BGP peer has not advertised or configured with RTF address family, then all routes belonging to the VPN table will be advertised.

RTF implementation in the control node does not support advertising and receiving of default route targets.

Constrained route distribution using route target reachability information is defined in RFC 4684, “*Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)*”.

Debugging and Troubleshooting Route Target Filtering

Use the tips in this section to troubleshoot issues with RTF. Use various http introspect commands to reveal details about BGP neighbors for RTF. The following is a sample portion of an http introspect page.

When you access an introspect page, only the first panel of detail columns appears. Use a scroll bar or arrow keys to reveal more columns to the right, and vice versa.

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)

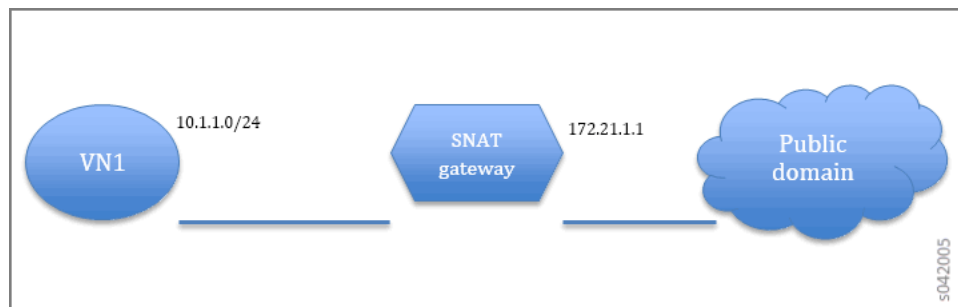
- [Overview on page 437](#)
- [Neutron APIs for Routers on page 438](#)
- [Network Namespace on page 439](#)
- [Using Web UI to Configure Routers with SNAT on page 439](#)

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 112: 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
```

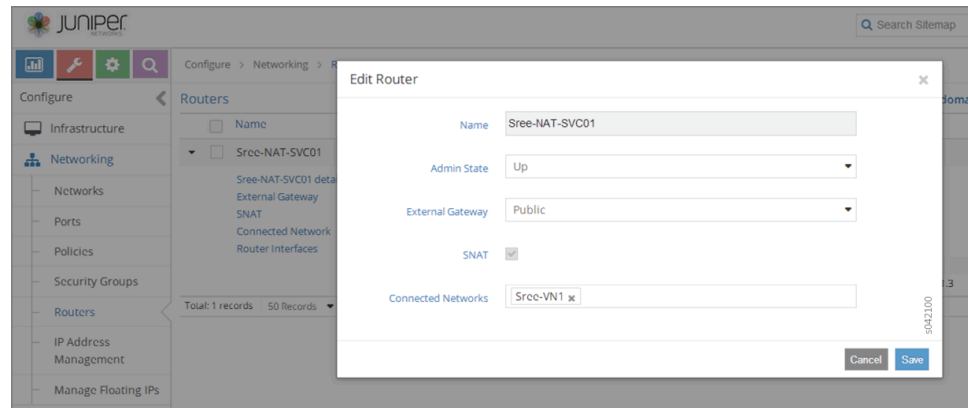
Using Web UI to Configure Routers with SNAT

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 113: 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 114: Router Status for SNAT

Name	External Gateway	Connected Network	Admin State
Sree-NAT-SVC01	Public	Sree-VN1	Up

Sree-NAT-SVC01 details	SNAT	External Gateway	Connected Network	Router Interfaces	UUID	Network	IP
Enabled	Enabled	Sree-VN1			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 115: 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-80888953)
Send CtrlAltDel

inet addr:100.1.1.2 Bcast:100.1.1.255 Mask:255.255.255.0
inets addr: fe80::19c:9aff:fe20:64 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 txqueue:1000
RX bytes:551725 (551.7 KB) TX bytes:564998 (564.9 KB)

10
Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inets addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:1636 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 txqueue:1000
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
10
--- 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:~$

```


PART 4

Monitoring and Troubleshooting Contrail

- [Configuring Traffic Mirroring to Monitor Network Traffic on page 443](#)
- [Understanding Contrail Analytics on page 459](#)
- [Configuring Contrail Analytics on page 481](#)
- [Using Contrail Analytics to Monitor and Troubleshoot the Network on page 491](#)
- [Common Support Answers on page 555](#)

Configuring Traffic Mirroring to Monitor Network Traffic

- [Configuring Traffic Analyzers and Packet Capture for Mirroring on page 443](#)
- [Configuring Interface Monitoring and Mirroring on page 453](#)
- [Analyzer Service Virtual Machine on page 454](#)

Configuring Traffic Analyzers and Packet Capture for Mirroring

Contrail provides traffic mirroring so you can mirror specified traffic to a traffic analyzer where you can perform deep traffic inspection. Traffic mirroring enables you to designate certain traffic flows to be mirrored to a traffic analyzer, where you can view traffic flows in great detail.

Use **Monitor > Debug > Packet Capture** to configure packets to be captured and “mirrored” to a virtual machine configured as a traffic analyzer. The packet activity can then be inspected for monitoring and troubleshooting purposes. This section demonstrates how to set up packet capture to mirror traffic packets to an analyzer.

- [Traffic Analyzer Images on page 443](#)
- [Configuring Traffic Analyzers on page 444](#)
- [Setting Up Traffic Mirroring Using Monitor > Debug > Packet Capture on page 444](#)
- [Setting Up Traffic Mirroring Using Configure > Networking > Services on page 448](#)

Traffic Analyzer Images

Before using the Contrail interface to configure traffic analyzers and packet capture for mirroring, make sure that the following analyzer images are available in the VM image list for your system. The traffic analyzer images are enhanced for viewing details of captured packets in Wireshark. When creating a policy for the traffic analyzer, the traffic analyzer instance should always have the **Mirror to** field selected in the policy, do not select the **Apply Service** field for a traffic analyzer.

- **analyzer-vm-console-qcow2**—Standard traffic analyzer; should be named **analyzer** in the image list. This type of traffic analyzer is always configured with a single interface, and the interface should be a **Left** interface.

- **analyzer-vm-console-two-if qcow2**—This type of traffic analyzer has two interfaces, **Left** and **Management**. This traffic analyzer can have any name except the name **analyzer**, which is reserved for the single interface analyzer.



NOTE: The analyzer-vm images are valid for all versions of Contrail. Download the images from the Contrail 1.0 software download page:
<https://www.juniper.net/support/downloads/?p=contrail#sw>.

Configuring Traffic Analyzers

In Contrail Controller, you use a two-part configuration to mirror captured packet traffic to a traffic analyzer, where the traffic details can be inspected. The configuration has the following steps:

1. Configure analyzer(s) on the host.
2. Set up rules for packet capture.

Additionally, there are two ways to configure the packet capture for the analyzers from within the Contrail interface:

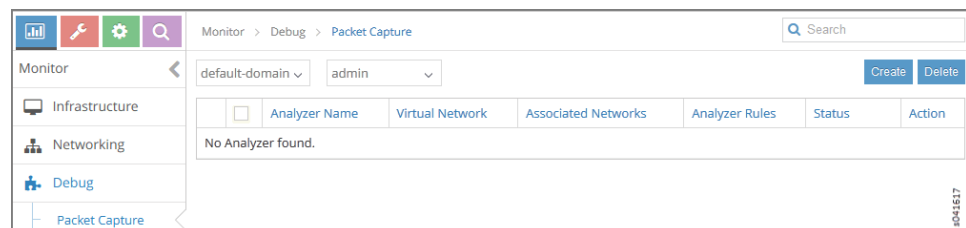
- Configure from **Monitor > Debug > Packet Capture**
- Configure from **Configure > Networking > Services**

Setting Up Traffic Mirroring Using Monitor > Debug > Packet Capture

The following are the steps needed to set up packet capture in order to “mirror” the traffic to an analyzer VM for the purpose of reviewing various aspects of packet traffic moving through the system.

1. Select **Monitor > Debug > Packet Capture**. The **Packet Capture** screen appears; see [Figure 116 on page 444](#).

Figure 116: Packet Capture



2. Click **Create** to add an analyzer; see [Figure 117 on page 445](#).

Figure 117: Create Analyzer

Create Analyzer

Analyzer Name:

Virtual Network:

Associate Networks:

Analyzer Rules

Protocol	Source Network	Source Ports	Direction	Destination Network	Destination Ports
----------	----------------	--------------	-----------	---------------------	-------------------

Cancel Save

3. In the **Analyzer Name** field, enter a name for the analyzer and in the **Virtual Network** field, select **Automatic** or select a specific virtual network from the drop-down list of available networks; click **Save** when finished.
4. To create rules for the analyzer, in the lower portion of the **Create Analyzer** screen, click the **+** button to add a rule.

The **Analyzer Rules** fields appear; see [Figure 118 on page 445](#).

Figure 118: Analyzer Rules

Create Analyzer

Analyzer Name:

Virtual Network:

Associate Networks:

Analyzer Rules

Protocol	Source Network	Source Ports	Direction	Destination Network	Destination Ports
<div>Any</div> <div>Any</div> <div>TCP</div> <div>UDP</div> <div>ICMP</div>	<div>any</div>	<input type="text" value="Source ports"/>	<div><></div>	<div>any</div>	<input type="text" value="Destination ports"/>

Cancel Save

5. Select the rules to apply to determine which packets should be “mirrored”—sent to the analyzer for monitoring.
- See [Table 34 on page 446](#) for guidelines for completing the rule fields.

Table 34: Analyzer Rule Fields

Field	Description
IP Protocol	<p>Select from a list to define from which protocol packets are to be captured:</p> <ul style="list-style-type: none"> • ANY • TCP • UDP • ICMP
Source Network	<p>Select from a list the source network from which packets are to be captured:</p> <ul style="list-style-type: none"> • any • local • <i>domain:network 1</i> • <i>domain:network 2</i> • <i>domain:network</i>
Source Ports	<p>If you want to capture only those packets that originate from a specific port number, enter the port number.</p>
Direction	<p>Select the direction of flow for the packets to be captured:</p> <ul style="list-style-type: none"> • Bidirectional • Unidirectional
Destination Network	<p>Select from a list the destination network for the packets to be captured:</p> <ul style="list-style-type: none"> • any • local • <i>domain:network 1</i> • <i>domain:network 2</i> • <i>domain:network</i>
Destination Ports	<p>If you want to capture only those packets that are destined to a specific port number, enter the port number.</p>
Cancel, Save	<p>When finished, click Save to commit your selections, or click Cancel to clear the entries and start over.</p>

- To associate virtual networks with the analyzer, click the **Associate Networks** field in the center portion of the screen. Select from a drop-down list of available networks the networks to associate with this analyzer; see [Figure 119 on page 447](#).

Figure 119: Create Analyzer Associate Networks

Create Analyzer

Analyzer Name

demo

Virtual Network

Automatic

Associate Networks

frontend

default-virtual-network

backend

Protocol	Source Network	Source Ports	Direction	Destination Network	Destination Ports
Any	any	Source ports	<=>	any	Destination ports

Cancel

Save



NOTE: If there is already a network policy attached to the virtual network selected, any conflicting rules configured for the analyzer will not take effect.

7. View the analyzer activity from **Monitor > Debug > Packet Capture**. For the selected analyzer, click in the **Action** column and select **View Analyzer**; see [Figure 120 on page 447](#).

Figure 120: Launch Analyzer VM

Monitor > Debug > Packet Capture

default-domain

admin

Create

Delete

	Analyzer Name	Virtual Network	Associated Networks	Analyzer Rules	Status	Action
<input checked="" type="checkbox"/>	demo	Automatic	backend frontend	protocol any network default-domainadmin:frontend port any <=> network default-domainadmin:backend port any mirror_to default-domainadmin:demo	Active	<div><div>Edit</div><div>View Analyzer</div><div>Delete</div></div>

Associated Networks:

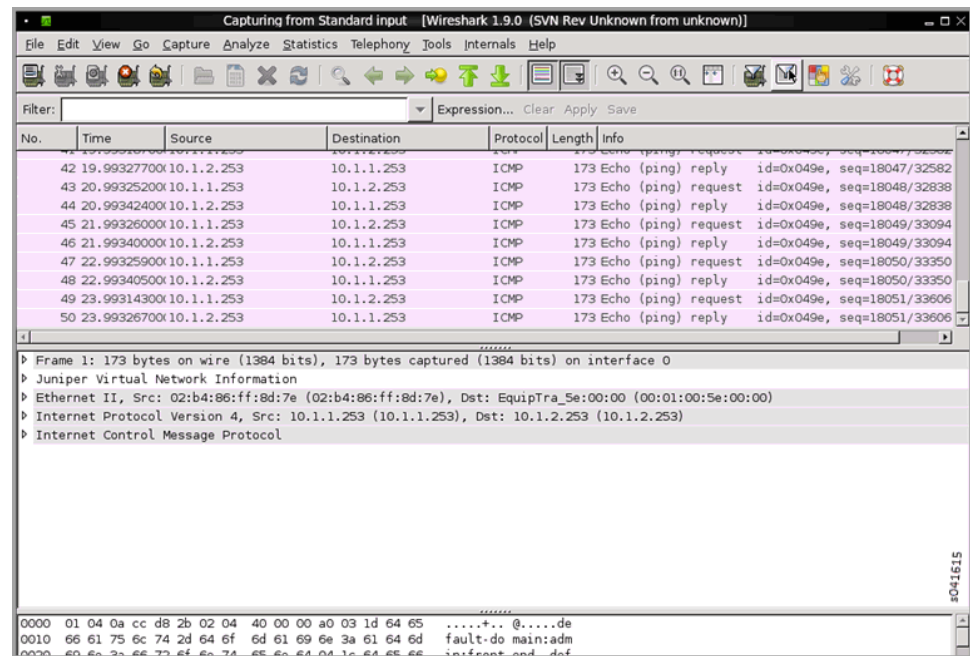
backend frontend

Analyzer Rules:

protocol any network default-domainadmin:frontend port any <=> network default-domainadmin:backend port any mirror_to default-domainadmin:demo

8. The Wireshark **Packet Capture Display** appears; see [Figure 121 on page 448](#).

Figure 121: Packet Capture Display



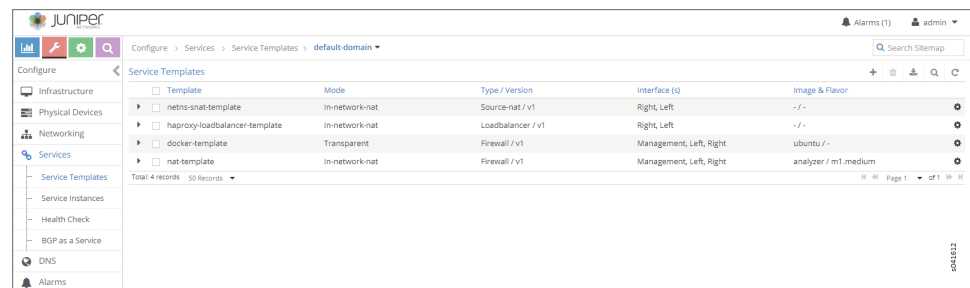
Setting Up Traffic Mirroring Using Configure > Networking > Services

You can set up packet capture for mirroring to an analyzer within a service chain utilizing more than one interface by starting with a service template. The following procedure provides the steps needed.

1. Access **Configure > Networking > Services > Service Templates**.

The **Service Templates** screen appears; see [Figure 122 on page 448](#).

Figure 122: Service Templates



2. To create a new service template, click the **Create** button.

The **Add Service Template** window appears; see [Figure 123 on page 449](#).

Figure 123: Add Service Template

Create

Service Template

Permissions

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 35 on page 449](#).

Table 35: Add Service Template Fields

Field	Description
Name	Enter a descriptive text name for this service template.
Service Mode	Select Transparent from the drop-down list to indicate that this service template is for purposes of mirroring.
Service	Select Analyzer from the drop-down list to indicate that this service template is for a traffic analyzer.
Image Name	Select from a drop-down list of available images the analyzer image to use for this analyzer service template. You should select the analyzer named analyzer two interfaces if you used the recommended naming for the image analyzer-vm-console-two-if qcow2 in the image list.
Interface Types	From the drop-down list, click the check boxes to indicate which two interface types are used for this analyzer service template: <ul style="list-style-type: none">• Left• Right• Management
Save	When finished, click OK to commit the changes
Cancel	Click Cancel to clear the fields and start over.

4. Create a service instance by clicking the **Service Instances** link and clicking the **Create** button.

The **Create Service Instances** window appears; see [Figure 124 on page 450](#).

Figure 124: Create Service Instances

5. Complete the fields by using the guidelines in [Table 36 on page 450](#).

Table 36: Create Service Instances Fields

Field	Description
Services Template	Select from a drop-down list of available service templates the template to use for this service instance (e.g. AnalyzerTemplate).
Instance Name	Enter a text name for this service instance.
Left Virtual Network	Select from a drop-down list of available networks the network for the left interface, or select Automatic .
Right Virtual Network	Select from a drop-down list of available networks the network for the right interface, or select Automatic .
Management Virtual Network	Select from a drop-down list of available networks the network for the management interface, or select Automatic .
Save	Click Save to commit your changes.
Cancel	Click Cancel 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.

7. Click **Create** to get to the **Create Policy** window; see [Figure 125 on page 451](#).

Figure 125: Create Policy

8. Click the **+** button in the lower portion of the screen to open the **Policy Rules** fields; see [Figure 126 on page 451](#).

Figure 126: Policy Rules

9. To add policy rules, complete the fields, using the guidelines in [Table 37 on page 451](#).



NOTE: When there is a network policy attached to the virtual network, any conflicting rules configured for the analyzer will not take effect.

Table 37: Add Rule Fields

Field	Description
Action	Enter a text name for this service instance.
Protocol	Select from a drop-down list of available networks the network for the left interface, or select Automatic .

Table 37: Add Rule Fields (continued)

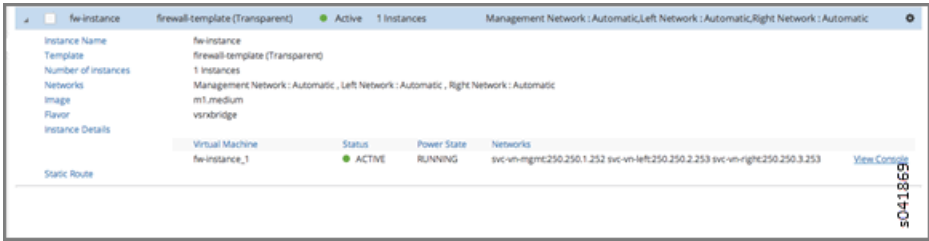
Field	Description
Source Network	Select from a drop-down list of available networks the network for the right interface, or select Automatic .
Source Ports	Select from a drop-down list of available networks the network for the management interface, or select Automatic .
Direction	Select the direction of flow for the packets to be captured: <ul style="list-style-type: none"> • Bidirectional • Unidirectional
Destination Network	Select from a list the destination network for the packets to be captured: <ul style="list-style-type: none"> • any • local • <i>domain:network 1</i> • <i>domain:network 2</i> • <i>domain:network</i>
Destination Ports	Select from a list the destination network for the packets to be captured: <ul style="list-style-type: none"> • any • local • <i>domain:network 1</i> • <i>domain:network 2</i> • <i>domain:network</i>
Apply Service	Check this box to open a field where you can select a service to apply.
Mirror to	Check this box to open a field where you can select a service to accept the mirrored packets.
Save	Click Save to commit your changes.
Cancel	Click Cancel to clear your changes and start over.

10. Click the **Mirror to** box and select the available analyzer service instance, then 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 127 on page 453](#). The analyzer service VM launches the Contrail enhanced Wireshark as it starts and captures the mirrored packets destined to this service.

Figure 127: Service Instances View Console



NOTE: When using the Firefox web browser, you may have difficulty viewing the mixed content presented by the View Console enhanced Wireshark option. To fix this, please enable mixed content in Firefox. Alternatively, you can select [Click here to show only console to view the console information in a separate window.](#)

- See Also**
- [Configuring Interface Monitoring and Mirroring on page 453](#)
 - [Analyzer Service Virtual Machine on page 454](#)

Configuring Interface Monitoring and Mirroring

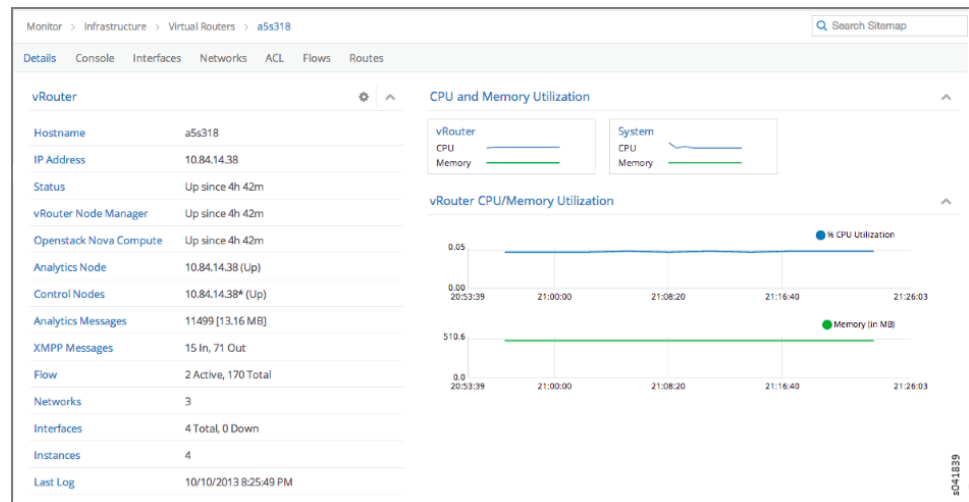
Contrail supports user monitoring of traffic on any guest virtual machine interface when using the Juniper Contrail user interface.

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 128 on page 454](#).

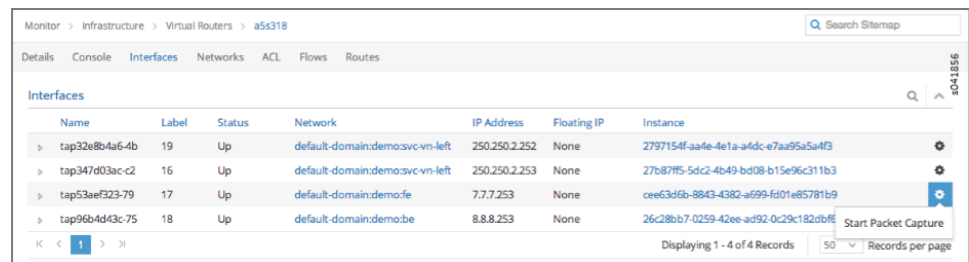
Figure 128: Individual vRouter



A list of interfaces for that vRouter appears.

- For the interface to mirror, click the Action icon in the last column and select the option Packet Capture; see Figure 129 on page 454.

Figure 129: Interfaces



The mirror packet capture starts and displays at this screen.

The mirror packet capture stops when you exit this screen.

Related Documentation

Analyzer Service Virtual Machine

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 on page 455
- Metadata Format on page 455
- Wireshark Changes on page 456
- Troubleshooting Packet Display on page 456

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.

Type: 1 Byte

Length: 1 Byte

Value: up to length

Type

- 1 – Captured host IPv4 address
- 2 – Action field
- 3 – Source VN
- 4 – Destination VN
- 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

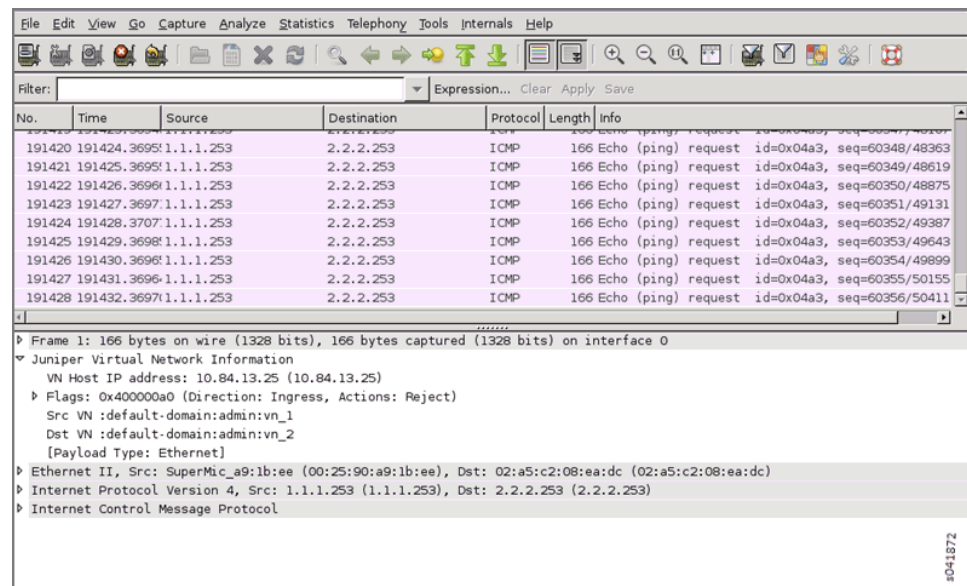
TLV end

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 130 on page 456](#).

Figure 130: 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 on page 443](#)
 - [Configuring Interface Monitoring and Mirroring on page 453](#)

Understanding Contrail Analytics

- [Contrail Analytics Overview on page 459](#)
- [Contrail Alerts on page 460](#)
- [Underlay Overlay Mapping in Contrail on page 463](#)

Contrail Analytics Overview

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 file `testbed.py`.

**Related
Documentation**

- [Contrail Alerts on page 460](#)
- [Analytics Scalability on page 481](#)
- [High Availability for Analytics on page 482](#)
- [Ceilometer Support in a Contrail Cloud on page 484](#)
- [Underlay Overlay Mapping in Contrail on page 463](#)
- [Monitoring the System on page 492](#)
- [Debugging Processes Using the Contrail Introspect Feature on page 494](#)
- [Monitor > Infrastructure > Dashboard on page 498](#)
- [Monitor > Infrastructure > Control Nodes on page 500](#)
- [Monitor > Infrastructure > Virtual Routers on page 507](#)
- [Monitor > Infrastructure > Analytics Nodes on page 518](#)
- [Monitor > Infrastructure > Config Nodes on page 523](#)
- [Monitor > Networking on page 526](#)
- [Understanding Flow Sampling on page 546](#)
- [Query > Flows on page 534](#)
- [Query > Logs on page 541](#)
- [System Log Receiver in Contrail Analytics on page 483](#)
- [Example: Debugging Connectivity Using Monitoring for Troubleshooting on page 549](#)

Contrail Alerts

Starting with Contrail 3.0 and greater, Contrail alerts are provided on a per-user visible entity (UVE) basis.

Contrail analytics raise or clear alerts using Python-coded rules that examine the contents of the UVE and the configuration of the object. Some rules are built in. Others can be added using Python *stevedore* plugins.

This topic describes Contrail alerts capabilities.

Alert API Format

The Contrail alert analytics API provides the following:

- Read access to the alerts as part of the UVE GET APIs.
- Alert acknowledgement using POST requests.

- UVE and alert streaming using server-sent events (SSEs).

For example:

GET `http://<analytics-ip>:8081/analytics/uves/control-node/a6s40?flat`

```
{
  NodeStatus: {...},
  ControlCpuState: {...},
  UVEAlarms: {
    alarms: [
      {
        description: [
          {
            value: "0 != 2",
            rule: "BgpRouterState.num_up_bgp_peer !=
BgpRouterState.num_bgp_peer"
          }
        ],
        ack: false,
        timestamp: 1442995349253178,
        token: "eyJ0aW1lc3RhbXAiOiAxNDQyOTk1MzQ5MTUzMTc4LCAiaHR0cF9wb3J0Ijog
NTk5NSwgImhvc3RfaXAiOiA0MTAuODQuMTMuNDUifQ==",
        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&filter=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/alarms-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"
},
```

**Related
Documentation**

- [Monitoring the System on page 492](#)
- [Debugging Processes Using the Contrail Introspect Feature on page 494](#)
- [Monitor > Infrastructure > Dashboard on page 498](#)
- [Monitor > Infrastructure > Control Nodes on page 500](#)
- [Monitor > Infrastructure > Virtual Routers on page 507](#)
- [Monitor > Infrastructure > Analytics Nodes on page 518](#)
- [Monitor > Infrastructure > Config Nodes on page 523](#)
- [Monitor > Networking on page 526](#)
- [Understanding Flow Sampling on page 546](#)
- [Query > Flows on page 534](#)
- [Query > Logs on page 541](#)
- [Example: Debugging Connectivity Using Monitoring for Troubleshooting on page 549](#)

Underlay Overlay Mapping in Contrail

- [Overview: Underlay Overlay Mapping using Contrail Analytics on page 464](#)
- [Underlay Overlay Analytics Available in Contrail on page 464](#)
- [Architecture and Data Collection on page 465](#)
- [New Processes/Services for Underlay Overlay Mapping on page 465](#)
- [External Interfaces Configuration for Underlay Overlay Mapping on page 466](#)
- [Physical Topology on page 466](#)

- [SNMP Configuration on page 467](#)
- [Link Layer Discovery Protocol \(LLDP\) Configuration on page 467](#)
- [IPFIX and sFlow Configuration on page 467](#)
- [Sending pRouter Information to the SNMP Collector in Contrail on page 468](#)
- [pRouter UVEs on page 469](#)
- [Contrail User Interface for Underlay Overlay Analytics on page 470](#)
- [Viewing Topology to the Virtual Machine Level on page 470](#)
- [Viewing the Traffic of any Link on page 471](#)
- [Trace Flows on page 471](#)
- [Search Flows and Map Flows on page 472](#)
- [Overlay to Underlay Flow Map Schemas on page 473](#)
- [Module Operations for Overlay Underlay Mapping on page 476](#)
- [SNMP Collector Operation on page 476](#)
- [Topology Module Operation on page 477](#)
- [IPFIX and sFlow Collector Operation on page 477](#)
- [Troubleshooting Underlay Overlay Mapping on page 478](#)
- [Script to add pRouter Objects on page 479](#)

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: 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.

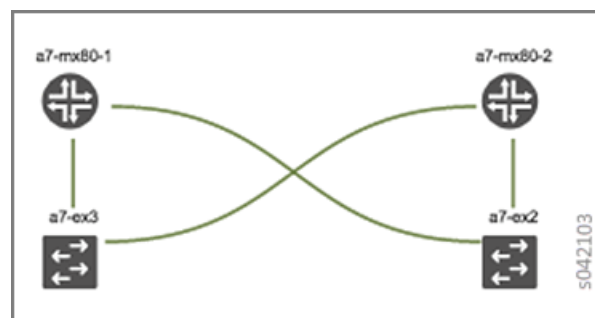
Physical Topology

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 131: 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's 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 all-ethernet


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, but the information can also be sent to the SNMP collector by means of a **device.ini** 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 132: 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 133: 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"
  }
]
```

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 134: 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: [...],
    + ifXTable: [...],
    + arpTable: [...],
    + lldpTable: {...},
    + ifStats: [...]
  }
}
```

s042435

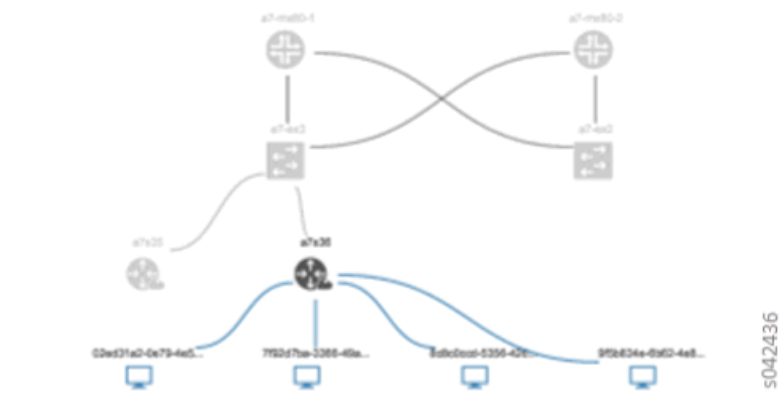
Contrail User Interface for Underlay Overlay Analytics

The topology view and related functionality is accessed from the Contrail Web user interface, **Monitor > Physical Topology**.

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.

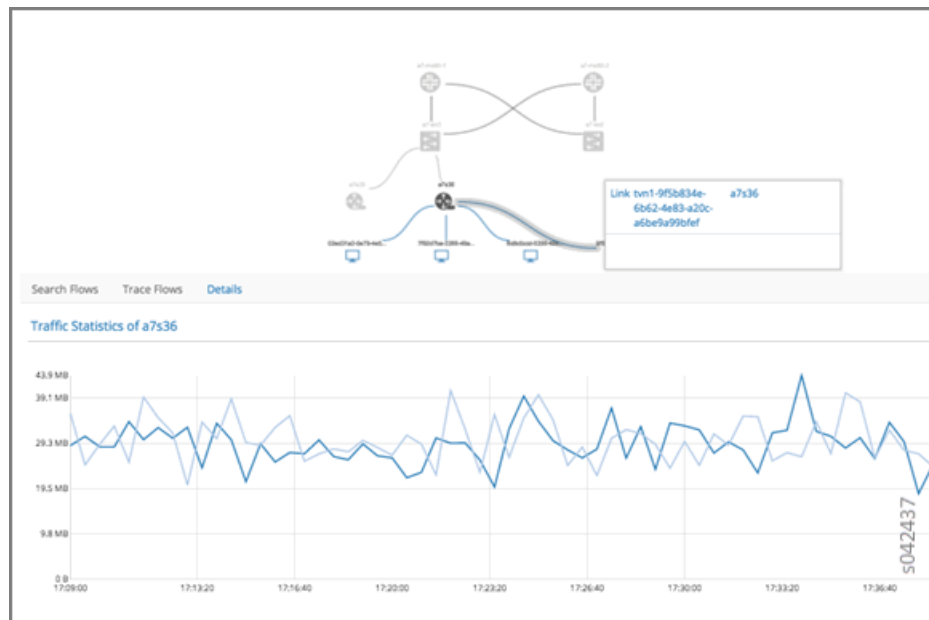
Figure 135: Physical Topology Related to a vRouter



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.

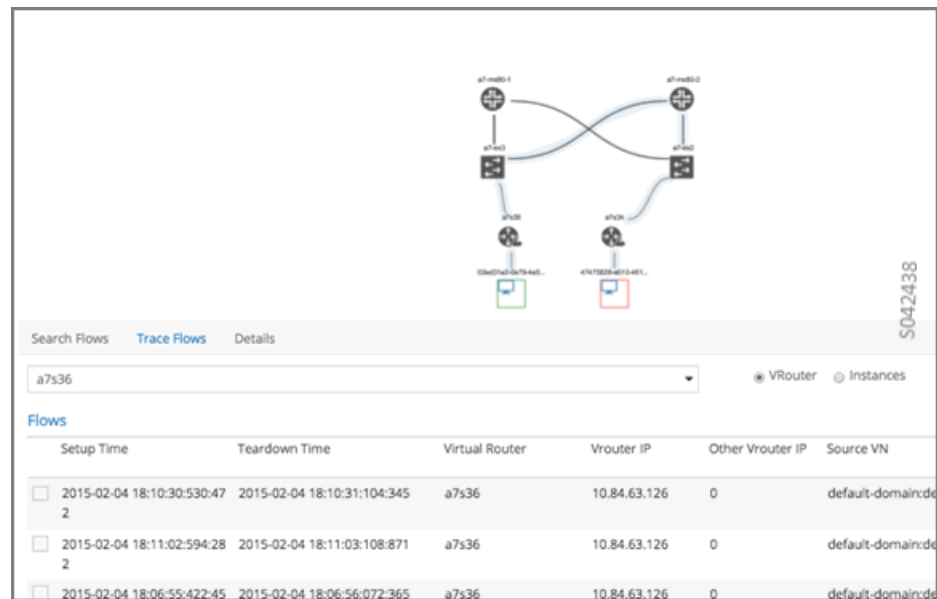
Figure 136: Traffic Statistics Graph



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 137: List of Active Flows



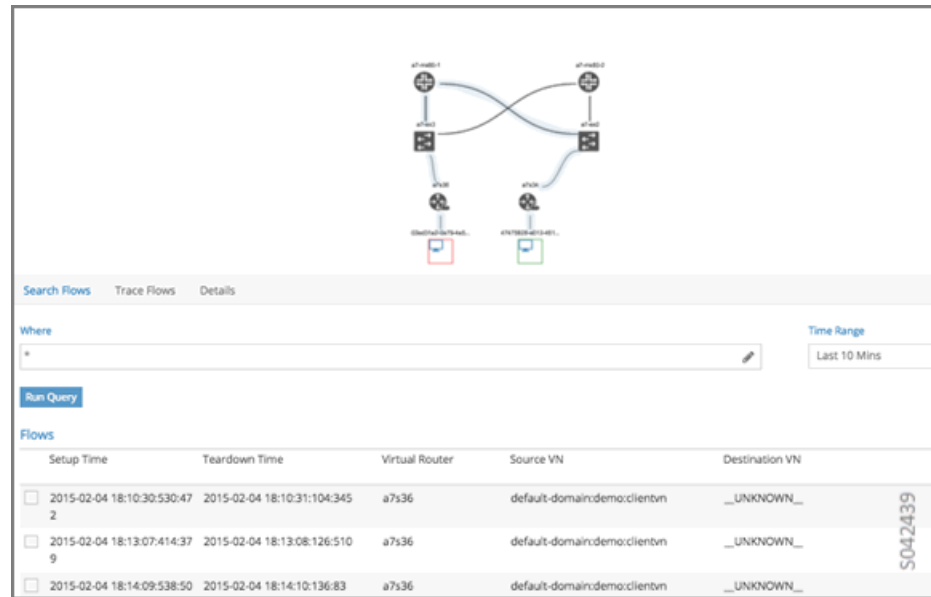
Limitations of Trace Flow Feature

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 138: 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**
- [Contrail Analytics Overview on page 459](#)
 - [Contrail Alerts on page 460](#)

Configuring Contrail Analytics

- [Analytics Scalability on page 481](#)
- [High Availability for Analytics on page 482](#)
- [System Log Receiver in Contrail Analytics on page 483](#)
- [Ceilometer Support in a Contrail Cloud on page 484](#)

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).

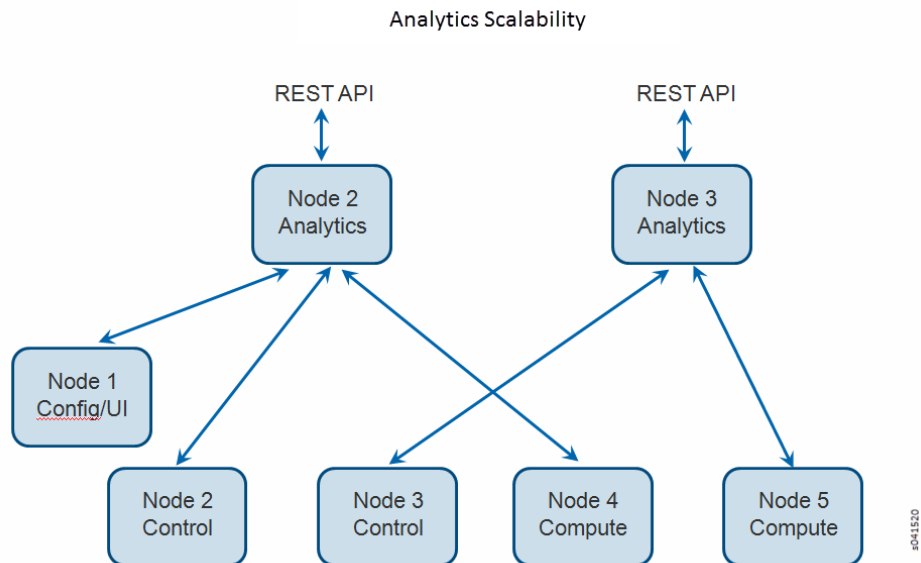
This section provides a brief description of the recommended configuration of analytics in Contrail to achieve horizontal scalability.

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 139 on page 482](#) 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 139: 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.
- Subscribe to the Contrail Discovery Service to get a list of functioning endpoints. If there are any failures, it can take 5-30 minutes for the Contrail Discovery Service to send an update.

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, using the Contrail Discovery Service.

System Log Receiver in Contrail Analytics

- [Overview on page 483](#)
- [Redirecting System Logs to Contrail Collector on page 483](#)
- [Exporting Logs from Contrail Analytics on page 483](#)

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.

Ceilometer Support in a Contrail Cloud

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 on page 484](#)
- [Ceilometer Details on page 484](#)
- [Verification of Ceilometer Operation on page 485](#)
- [Contrail Ceilometer Plugin on page 487](#)
- [Ceilometer Installation and Provisioning on page 489](#)

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 Fabric with `enable_ceilometer = True` in the `testbed.py` file.
- Alternately, provisioned using Server Manager with `enable_ceilometer = True` in the `cluster.json` or `cluster` configuration.



.....
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:

Polling agent—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.

Notification agent—Agent designed to listen to notifications on message queue and convert them to events and samples.

Collector—Gathers and records event and metering data created by the notification and polling agents.

API server—Provides a REST API to query and view data recorded by the collector service.

Alarms—Daemons to evaluate and notify based on defined alarming rules.

Database—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:

Polling agent—`ceilometer-agent-central` and `ceilometer-agent-compute`

Notification agent—`ceilometer-agent-notification`

Collector —`ceilometer-collector`

API Server—`ceilometer-api`

Alarms—`ceilometer-alarm-evaluator` and `ceilometer-alarm-notifier`

On RHEL Server 7.0, the service names are:

Polling agent—`openstack-ceilometer-central` and `openstack-ceilometer-compute`

Notification agent—`openstack-ceilometer-notification`

Collector —`openstack-ceilometer-collector`

API server—`openstack-ceilometer-api`

Alarms—`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	User ID	Type	Unit	Resource ID	Project ID
ip.floating.receive.bytes	a726f93a-65fa-4cad-828b-54dbfcf4a119	cumulative	B		None
ip.floating.receive.packets	a726f93a-65fa-4cad-828b-54dbfcf4a119	cumulative	packet		None
ip.floating.transmit.bytes	a726f93a-65fa-4cad-828b-54dbfcf4a119	cumulative	B		None
ip.floating.transmit.packets	a726f93a-65fa-4cad-828b-54dbfcf4a119	cumulative	packet		None
network	7fa6796b-756e-4320-9e73-87d4c52ecc83	gauge	network	15c0240142084d16b3127d6f844adb9	
network	9408e287-d3e7-41e2-89f0-5c691c9ca450	gauge	network	15c0240142084d16b3127d6f844adb9	
network	b3b72b98-f61e-4e1f-9a9b-84f4f3ddec0b	gauge	network	15c0240142084d16b3127d6f844adb9	
network	cb829abd-e6a3-42e9-a82f-0742db55d329	gauge	network	15c0240142084d16b3127d6f844adb9	
network.create	7fa6796b-756e-4320-9e73-87d4c52ecc83	delta	network	15c0240142084d16b3127d6f844adb9	
network.create	9408e287-d3e7-41e2-89f0-5c691c9ca450	delta	network	15c0240142084d16b3127d6f844adb9	
network.create	b3b72b98-f61e-4e1f-9a9b-84f4f3ddec0b	delta	network	15c0240142084d16b3127d6f844adb9	
network.create	cb829abd-e6a3-42e9-a82f-0742db55d329	delta	network	15c0240142084d16b3127d6f844adb9	
port	0d401d96-c2bf-4672-abf2-880eecf25ceb	gauge	port	01edcedd989f43b3a2d6121d424b254d	
port	211b94a4-581d-45d0-8710-c6c69df15709	gauge	port	01edcedd989f43b3a2d6121d424b254d	
port	2287ce25-4eef-4212-b77f-3cf590943d36	gauge	port	01edcedd989f43b3a2d6121d424b254d	
port.create	f62f3732-222e-4c40-8783-5bcbcf1fd6a1c	delta	port	01edcedd989f43b3a2d6121d424b254d	
port.create	f8c89218-3cad-48e2-8bd8-46c1bc33e752	delta	port	01edcedd989f43b3a2d6121d424b254d	

```

| 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
Project ID			User 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	{u'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 | | |
|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 325.0 | 1066.0 | 0.368603042877 | 439069.674 | 2015-02-13T19:50:40.795000 | 2892 |
| 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 automatically as part of the OpenStack controller node and the compute agent service is installed as part of the compute node.

The following fabric tasks are added to facilitate the installation and provisioning:

fab install_ceilometer—Installs the Ceilometer packages on the OpenStack controller node.

fab install_ceilometer_compute—Installs the Ceilometer packages on the compute node.

fab setup_ceilometer—Provisions the Ceilometer controller services on the OpenStack controller node.

fab setup_ceilometer_compute—Provisions the Contrail Ceilometer plugin package on the OpenStack controller node.

fab install_contrail_ceilometer_plugin—Installs the Contrail Ceilometer plugin package on the OpenStack controller node.

fab setup_contrail_ceilometer_plugin—Provisions the Contrail Ceilometer plugin package on the OpenStack controller node.



NOTE: The fabric tasks are automatically called as part of the **fab install_openstack** and **fab setup_openstack** commands for the OpenStack controller node, and as part of the **fab install_vrouter**, **fab setup_vrouter** commands for the compute node

CHAPTER 22

Using Contrail Analytics to Monitor and Troubleshoot the Network

- [Monitoring the System on page 492](#)
- [Debugging Processes Using the Contrail Introspect Feature on page 494](#)
- [Monitor > Infrastructure > Dashboard on page 498](#)
- [Monitor > Infrastructure > Control Nodes on page 500](#)
- [Monitor > Infrastructure > Virtual Routers on page 507](#)
- [Monitor > Infrastructure > Analytics Nodes on page 518](#)
- [Monitor > Infrastructure > Config Nodes on page 523](#)
- [Monitor > Networking on page 526](#)
- [Query > Flows on page 534](#)
- [Query > Logs on page 541](#)
- [Understanding Flow Sampling on page 546](#)
- [Example: Debugging Connectivity Using Monitoring for Troubleshooting on page 549](#)

Monitoring the System

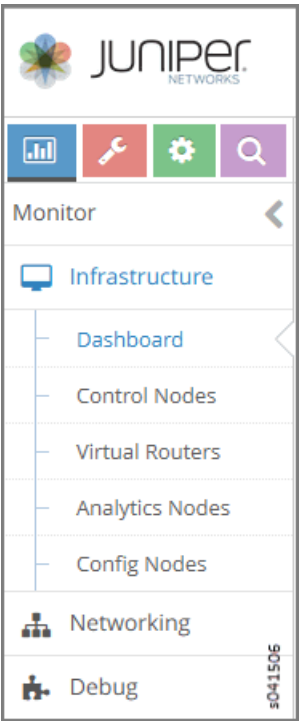
The **Monitor** icon on the Contrail Controller provides numerous options so you can view and analyze usage and other activity associated with all nodes of the system, through the use of reports, charts, and detailed lists of configurations and system activities.

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 140 on page 492](#)

Figure 140: Monitor Menu



See [Table 38 on page 492](#) for descriptions of the items available under each of the menu options from the **Monitor** icon.

Table 38: Monitor Menu Options

Option	Description
Infrastructure > Dashboard	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 " Monitor > Infrastructure > Dashboard " on page 498.

Table 38: Monitor Menu Options (continued)

Option	Description
Infrastructure > Control Nodes	<p>View a summary for all control nodes in the system, and for each control node, view:</p> <ul style="list-style-type: none"> Graphical reports of memory usage and average CPU load. Console information for a specified time period. A list of all peers with details about type, ASN, and the like. A list of all routes, including next hop, source, local preference, and the like. <p>See "Monitor > Infrastructure > Control Nodes" on page 500.</p>
Infrastructure > Virtual Routers	<p>View a summary of all vRouters in the system, and for each vRouter, view:</p> <ul style="list-style-type: none"> Graphical reports of memory usage and average CPU load. Console information for a specified time period. A list of all interfaces with details such as label, status, associated network, IP address, and the like. A list of all associated networks with their ACLs and VRFs. A list of all active flows with source and destination details, size, and time. <p>See "Monitor > Infrastructure > Virtual Routers" on page 507.</p>
Infrastructure > Analytics Nodes	<p>View activity for the analytics nodes, including memory and CPU usage, analytics host names, IP address, status, and more. See "Monitor > Infrastructure > Analytics Nodes" on page 518.</p>
Infrastructure > Config Nodes	<p>View activity for the config nodes, including memory and CPU usage, config host names, IP address, status, and more. See "Monitor > Infrastructure > Config Nodes" on page 523.</p>
Networking > Networks	<p>For all virtual networks for all projects in the system, view graphical traffic statistics, including:</p> <ul style="list-style-type: none"> Total traffic in and out. Inter VN traffic in and out. The most active ports, peers, and flows for a specified duration. All traffic ingress and egress from connected networks, including their attached policies. <p>See "Monitor > Networking" on page 526.</p>
Networking > Dashboard	<p>For all virtual networks for all projects in the system, view graphical traffic statistics, including:</p> <ul style="list-style-type: none"> Total traffic in and out. Inter VN traffic in and out. <p>You can view the statistics in varying levels of granularity, for example, for a whole project, or for a single network. See "Monitor > Networking" on page 526.</p>
Networking > Projects	<p>View essential information about projects in the system including name, associated networks, and traffic in and out.</p>

Table 38: Monitor Menu Options (continued)

Option	Description
Networking > Networks	View essential information about networks in the system including name and traffic in and out.
Networking > Instances	View essential information about instances in the system including name, associated networks, interfaces, vRouters, and traffic in and out.
Debug > Packet Capture	<ul style="list-style-type: none"> • Add and manage packet analyzers. • Attach packet captures and configure their details. • 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.

- Related Documentation**
- [Monitor > Infrastructure > Dashboard on page 498](#)
 - [Monitor > Infrastructure > Control Nodes on page 500](#)
 - [Monitor > Infrastructure > Virtual Routers on page 507](#)
 - [Monitor > Networking on page 526](#)
 - [Query > Logs on page 541](#)
 - [Query > Flows on page 534](#)

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.

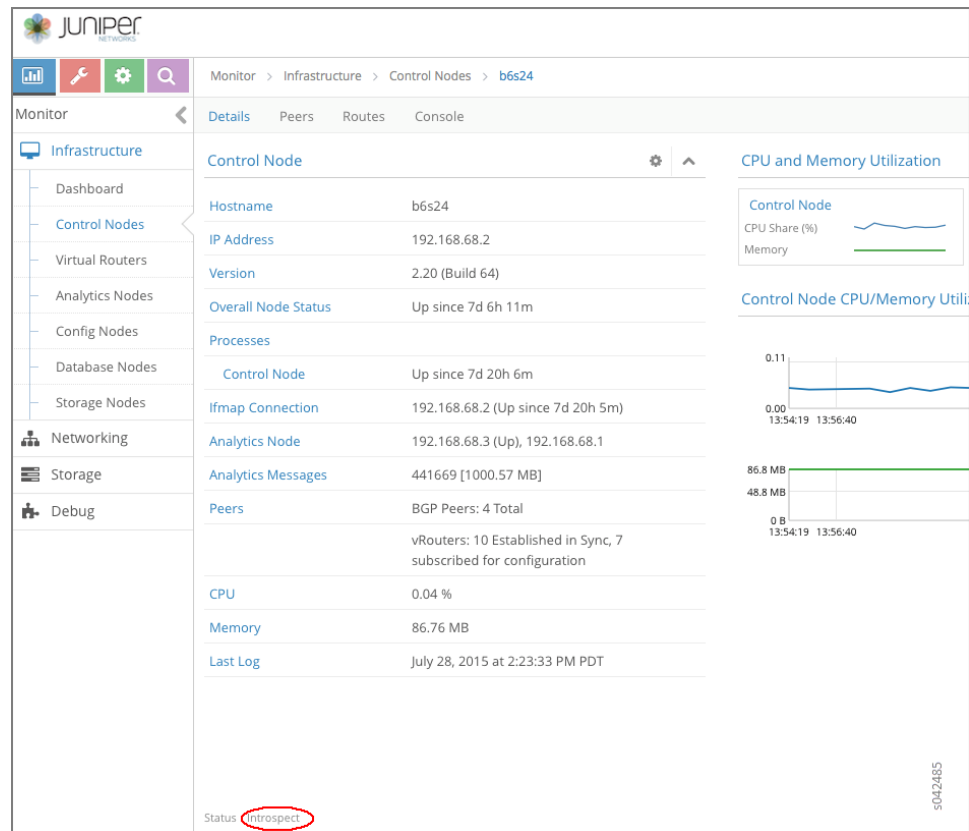
You can display the HTTP introspect of a Contrail daemon directly by accessing the following Introspect ports:

- `<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 141 on page 495 shows the *contrail-control* infrastructure page. Notice the Introspect link at the bottom of the Control Nodes Details tab window.

Figure 141: 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 142 on page 496 shows the Controller Introspect window.

Figure 142: Controller Introspect Window

The screenshot shows the 'Controller Introspect' window in the Contrail interface. The title bar includes 'Contrail' and navigation buttons: 'Collapse', 'Expand', 'Wrap', and 'NoWrap'. The main content area has a title 'Controller Introspect'. On the left, there is a text input field containing 'AgentXmppConnectionStatusReq'. Below this field is a 'Send' button. The right side of the window is mostly empty, with a small vertical text '0/02388' on the far right edge.

Figure 143 on page 496 shows an example of the BGP Peer (bgp_peer.xml) Introspect page.

Figure 143: BGP Peer Introspect Page

The screenshot shows the 'Bgp_peer Introspect' page in the Contrail interface. The title bar includes 'Contrail' and navigation buttons: 'Collapse', 'Expand', 'Wrap', and 'NoWrap'. The main content area has a title 'Bgp_peer Introspect'. On the left, there is a list of introspect actions: 'BgpNeighborReq', 'ShowBgpNeighborSummaryReq', 'ClearBgpNeighborReq', 'ShowRouteReq', 'ShowRouteSummaryReq', 'ShowRoutingInstanceReq', 'ShowRoutingInstanceSummaryReq', 'ShowMulticastManagerReq', 'ShowBgpInstanceConfigReq', 'ShowBgpNeighborConfigReq', and 'ShowBgpServerReq'. The right side of the page contains three sections: 1. 'BgpNeighborReq' with input fields for 'neighbor(string)' and 'domain(string)', and a 'Send' button. 2. 'ShowBgpNeighborSummaryReq' with a 'Send' button. 3. 'ClearBgpNeighborReq' with an input field for 'name(string)' and a 'Send' button. The right edge of the page has a small vertical text '9927405'.

Figure 144 on page 497 shows an example of the BGP Neighbor Summary Introspect page.

Figure 144: BGP Neighbor Summary Introspect Page

Contrail									
ShowBgpNeighborSummaryResp									
neighbors									
peer	deleted	deleted_at	peer_address	peer_id	peer_asn	encoding	peer_type	state	local_address
b6s23	false	-	192.168.68.1	192.168.68.1	64512	BGP	internal	Established	192.168.68.2
b6s25	false	-	192.168.68.3	192.168.68.3	64512	BGP	internal	Established	192.168.68.2
mx1	false	-	192.168.100.1	192.168.100.1	64512	BGP	internal	Established	192.168.68.2
mx2	false	-	192.168.100.2	192.168.100.2	64512	BGP	internal	Established	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 145 on page 498 shows an example of the Agent (agent.xml) Introspect page.

Figure 145: 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 os ke up ch
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 os ke up ch

5042489

Monitor > Infrastructure > Dashboard

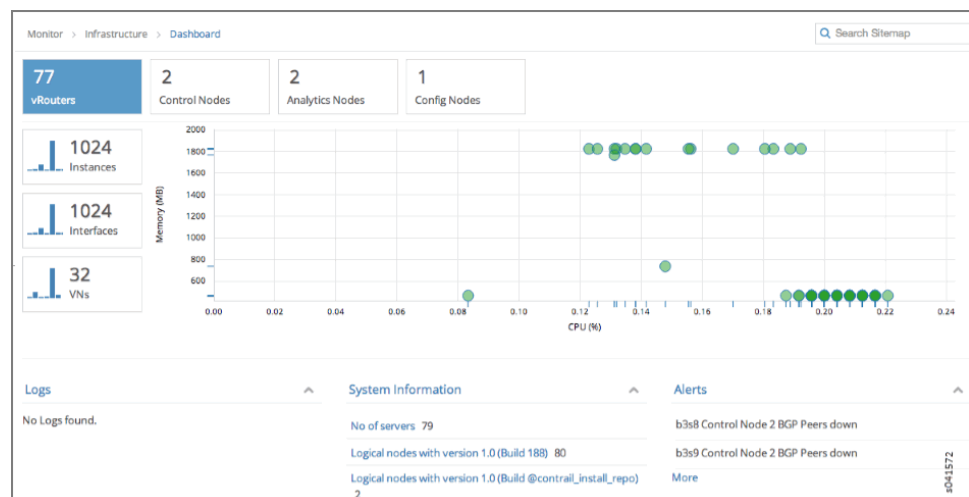
Use **Monitor > Infrastructure > Dashboard** to get an “at-a-glance” view of the system infrastructure components, including the numbers of virtual routers, control nodes, analytics nodes, and config nodes currently operational, a bubble chart of virtual routers showing the CPU and memory utilization, log messages, system information, and alerts.

- [Monitor Dashboard on page 498](#)
- [Monitor Individual Details from the Dashboard on page 499](#)
- [Using Bubble Charts on page 499](#)
- [Color-Coding of Bubble Charts on page 500](#)

Monitor Dashboard

Click **Monitor > Infrastructure > Dashboard** on the left to view the **Dashboard**. See [Figure 146 on page 498](#).

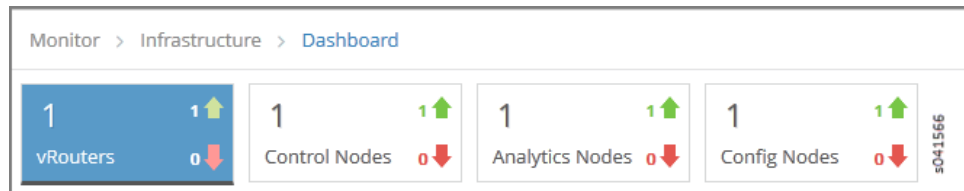
Figure 146: Monitor > Infrastructure > Dashboard



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 147 on page 499](#). 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 147: Dashboard Summary Boxes



Detailed information about monitoring each of the areas represented by the boxes is provided in the links in [Table 39 on page 499](#).

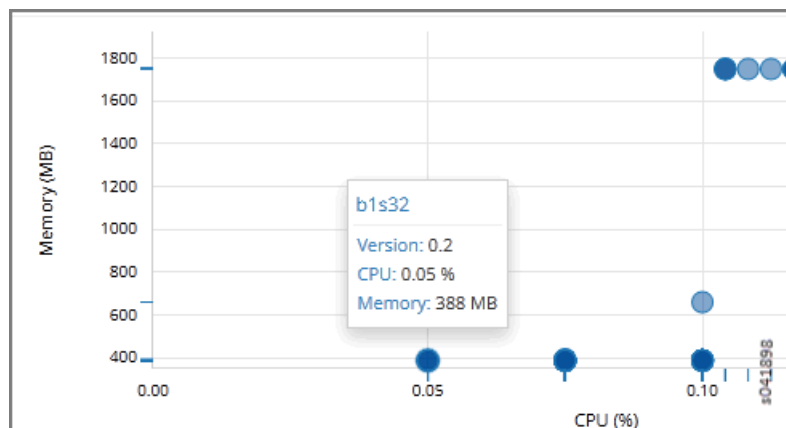
Table 39: Dashboard Summary Boxes

Box	For More Information
vRouters	"Monitor > Infrastructure > Virtual Routers" on page 507
Control Nodes	"Monitor > Infrastructure > Control Nodes" on page 500
Analytics Nodes	"Monitor > Infrastructure > Analytics Nodes" on page 518
Config Nodes	"Monitor > Infrastructure > Config Nodes" on page 523

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 148 on page 500](#). You can click through the summary information to get more details about the component.

Figure 148: 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 on page 507](#)
- [Monitor > Infrastructure > Control Nodes on page 500](#)
- [Monitor > Infrastructure > Analytics Nodes on page 518](#)
- [Monitor > Infrastructure > Config Nodes on page 523](#)

Monitor > Infrastructure > Control Nodes

Use **Monitor > Infrastructure > Control Nodes** to gain insight into usage statistics for control nodes.

- [Monitor Control Nodes Summary on page 501](#)
- [Monitor Individual Control Node Details on page 501](#)
- [Monitor Individual Control Node Console on page 503](#)
- [Monitor Individual Control Node Peers on page 505](#)
- [Monitor Individual Control Node Routes on page 506](#)

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 149 on page 501](#). See [Table 40 on page 501](#) for descriptions of the fields on this screen.

Figure 149: Control Nodes Summary



Table 40: Control Nodes Summary Fields

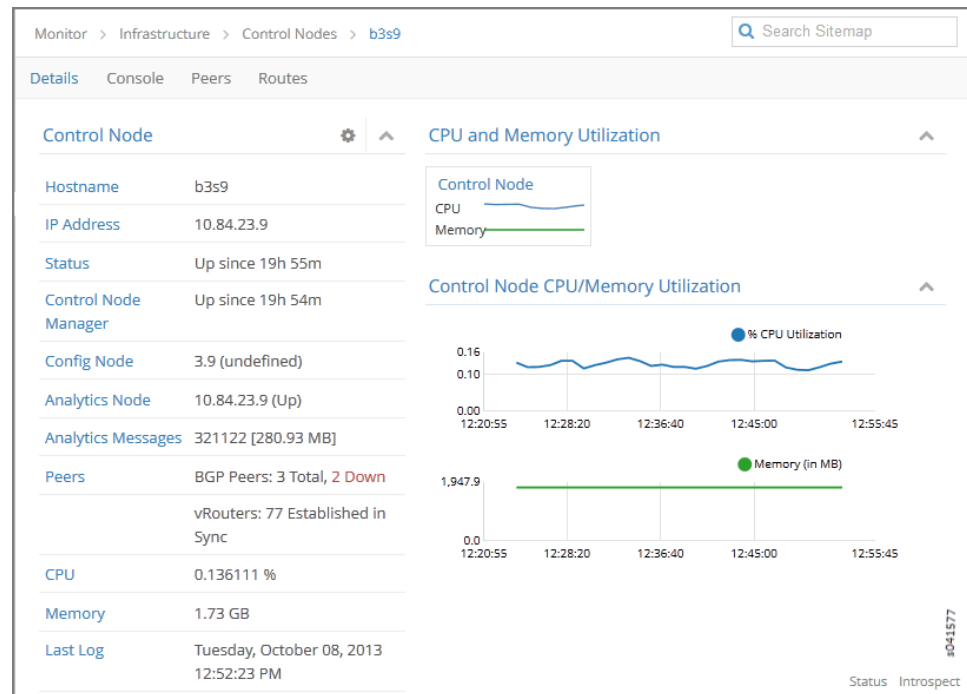
Field	Description
Host name	The name of the control node.
IP Address	The IP address of the control node.
Version	The software version number that is installed on the control node.
Status	The current operational status of the control node — Up or Down.
CPU (%)	The CPU percentage currently in use by the selected control node.
Memory	The memory in MB currently in use and the total memory available for this control node.
Total Peers	The total number of peers for this control node.
Established in Sync Peers	The total number of peers in sync for this control node.
Established in Sync vRouters	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 150 on page 502](#).

Figure 150: 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 41 on page 502](#) for descriptions of the fields on this tab.

Table 41: 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.
Config Node	The IP address of the configuration node associated with this control node.
Analytics Node	The IP address of the node from which analytics (monitor) information is derived.
Analytics Messages	The total number of analytics messages in and out from this node.
Peers	The total number of peers established for this control node and how many are in sync and of what type.

Table 41: Individual Control Node—Details Tab Fields (continued)

Field	Description
CPU	The average percent of CPU load incurred by this control node.
Memory	The average memory usage incurred by this control node.
Last Log	The date and time of the last log message issued about this control node.
Control Node CPU/Memory Utilization	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 151 on page 503](#).

Figure 151: 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 42 on page 503](#) for descriptions of the fields on the **Console** tab screen.

Table 42: 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 11 options, ranging from the Last 5 mins through to the Last 24 hrs . The default display is for the Last 5 mins .

Table 42: Control Node: Console Tab Fields (continued)

Field	Description
Log Category	Select a log category to display: All _default_ XMPP IFMap TCP
Log Type	Select a log type to display.
Log Level	Select a log severity level to display: SYS_EMERG SYS_ALERT SYS_CRIT SYS_ERR SYS_WARN SYS_NOTICE SYS_INFO SYS_DEBUG
Search	Enter any text string to search and display logs containing that string.
Limit	Select from a list an amount to limit the number of messages displayed: No Limit Limit 10 messages Limit 50 messages Limit 100 messages Limit 200 messages Limit 500 messages
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.
Time	This column lists the time received for each log message displayed.
Category	This column lists the log category for each log message displayed.
Log Type	This column lists the log type for each log message displayed.
Log	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 152 on page 505](#).

Figure 152: 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: - {

```

s041579

See [Table 43 on page 505](#) for descriptions of the fields on the **Peers** tab screen.

Table 43: Control Node: Peers Tab Fields

Field	Description
Peer	The hostname of the peer.
Peer Type	The type of peer.
Peer ASN	The autonomous system number of the peer.
Status	The current status of the peer.
Last flap	The last flap detected for this peer.
Messages (Recv/Sent)	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 153 on page 506](#).

Figure 153: Individual Control Node—Routes Tab

Routing Table	Prefix	Protocol	Source	Next hop	Label	Secur...	Origin VN
bgp.03vpn.0	10.84.21.1:13:192.168.30.240/32	XMPP	b1s1	10.84.21.1	28	3	default-domaindemo.v n30
		BGP	10.84.23.9	10.84.21.1	28	3	default-domaindemo.v n30
	10.84.21.1:14:192.168.31.242/32	XMPP	b1s1	10.84.21.1	29	3	default-domaindemo.v n31
		BGP	10.84.23.9	10.84.21.1	29	3	default-domaindemo.v n31
	10.84.21.1:1:192.168.2.231/32	XMPP	b1s1	10.84.21.1	16	3	default-domaindemo.v n2

See [Table 44 on page 506](#) for descriptions of the fields on the **Routes** tab screen.

Table 44: Control Node: Routes Tab Fields

Field	Description
Routing Instance	You can select a single routing instance from a list of all instances for which to display the active routes.
Address Family	Select an address family for which to display the active routes: <ul style="list-style-type: none"> All (default) l3vpn inet inetmcast
(Limit Field)	Select to limit the display of active routes: <ul style="list-style-type: none"> Limit 10 Routes Limit 50 Routes Limit 100 Routes Limit 200 Routes
Peer Source	Select from a list of available peers the peer for which to display the active routes, or select All.

Table 44: Control Node: Routes Tab Fields (continued)

Field	Description
Prefix	Enter a route prefix to limit the display of active routes to only those with the designated prefix.
Protocol	Select a protocol for which to display the active routes: All (default) XMPP BGP ServiceChain Static
Display Routes	Click this button to refresh the display of routes after selecting different display criteria.
Reset	Click this button to clear any selected criteria and return the display to default values.
<i>Column</i>	<i>Description</i>
Routing Table	The name of the routing table that stores this route.
Prefix	The route prefix for each active route displayed.
Protocol	The protocol used by the route.
Source	The host source for each active route displayed.
Next hop	The IP address of the next hop for each active route displayed.
Label	The label for each active route displayed.
Security	The security value for each active route displayed.
Origin VN	The virtual network from which the route originates.
AS Path	The AS path for each active route displayed.

Monitor > Infrastructure > Virtual Routers

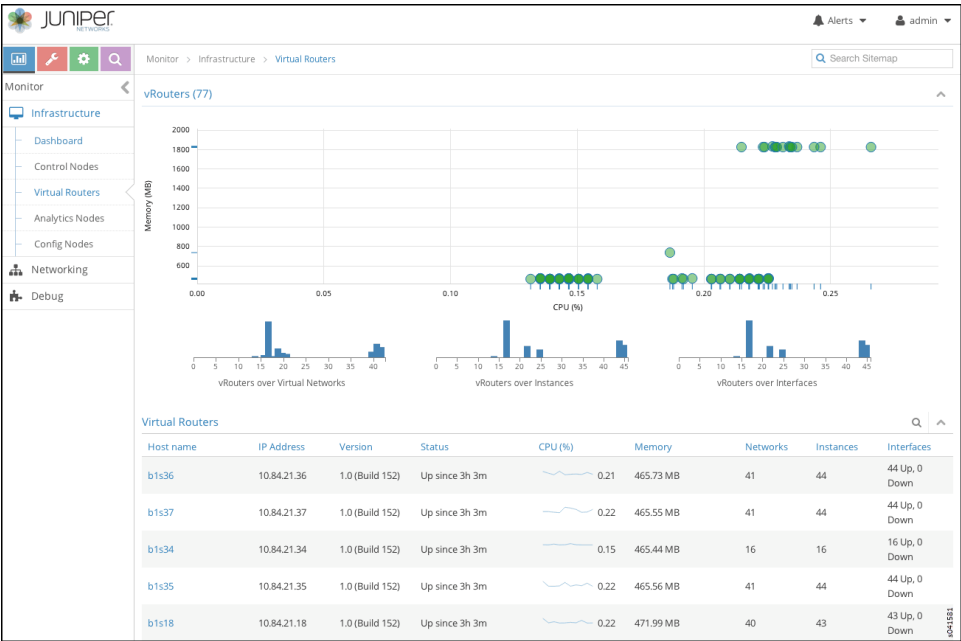
- [Monitor vRouters Summary on page 508](#)
- [Monitor Individual vRouters Tabs on page 509](#)
- [Monitor Individual vRouter Details Tab on page 509](#)
- [Monitor Individual vRouters Interfaces Tab on page 510](#)
- [Configuring Interface Monitoring and Mirroring on page 511](#)
- [Monitor Individual vRouters Networks Tab on page 512](#)
- [Monitor Individual vRouters ACL Tab on page 513](#)

- [Monitor Individual vRouters Flows Tab on page 514](#)
- [Monitor Individual vRouters Routes Tab on page 515](#)
- [Monitor Individual vRouter Console Tab on page 516](#)

Monitor vRouters Summary

Click **Monitor > Infrastructure > Virtual Routers** to view the **vRouters** summary screen.
See [Figure 154 on page 508](#).

Figure 154: vRouters Summary



See [Table 45 on page 508](#) for descriptions of the fields on the **vRouters Summary** screen.

Table 45: 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.
Memory (MB)	The memory currently in use and the total memory available for this vRouter.
Networks	The total number of networks for this vRouter.

Table 45: vRouters Summary Fields (continued)

Field	Description
Instances	The total number of instances for this vRouter.
Interfaces	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 155 on page 509](#).
- **Console**—similar display of information as on individual control nodes **Console** tab. See [Figure 163 on page 517](#).
- **Interfaces**—details about associated interfaces. See [Figure 156 on page 511](#).
- **Networks**—details about associated networks. See [Figure 159 on page 513](#).
- **ACL**—details about access control lists. See [Figure 160 on page 514](#).
- **Flows**—details about associated traffic flows. See [Figure 161 on page 515](#).
- **Routes**—details about associated routes. See [Figure 162 on page 516](#).

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 155 on page 509](#). See [Table 46 on page 510](#) for descriptions of the fields on this tab.

Figure 155: Individual vRouters—Details Tab

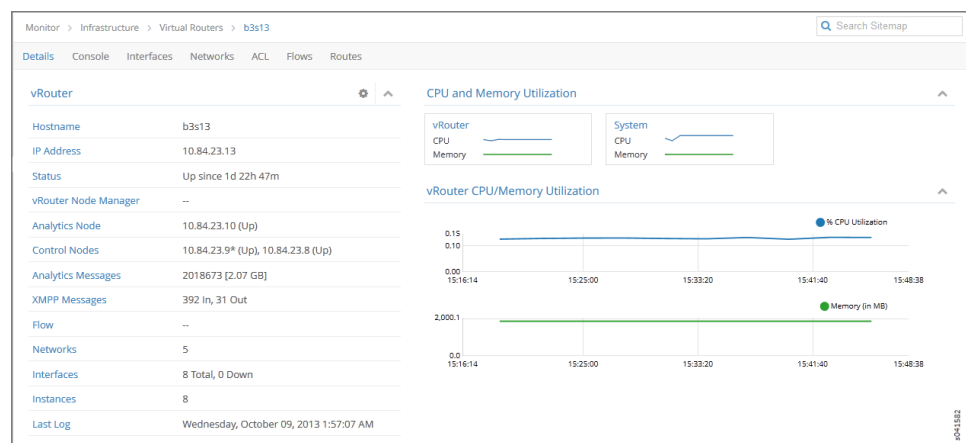


Table 46: vRouters Details Tab Fields

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.
Networks	The number of networks associated with this vRouter.
Interfaces	The number of interfaces associated with this vRouter.
Instances	The number of instances associated with this vRouter.
Last Log	The date and time of the last log message issued about this vRouter.
vRouter CPU/Memory Utilization	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 156 on page 511](#). See [Table 47 on page 511](#) for descriptions of the fields on the **Interfaces** tab screen.

Figure 156: Individual vRouters—Interfaces Tab

Name	Label	Status	Network	IP Address	Floating IP	Instance
tap25e5cee3-07	18	Up	default-domain:demo:vn30	192.168.30.247	None	005132fd-0d83-4db7-88c8-bd49d68e9480
tap4d91aab1-f1	25	Up	default-domain:demo:vn26	192.168.26.247	None	65d6c6e9-7a82-43d8-a706-f74d81715920
tap5a8cd9dd-5b	27	Up	default-domain:demo:vn23	192.168.23.249	None	a159c518-4fb6-402a-ae0d-eb5b4457b551
tap603a5e0b-8b	16	Up	default-domain:demo:vn19	192.168.19.247	None	fe622580-b0cf-4c6d-89e5-d2065e7e87e4
tap68ad232c-76	19	Up	default-domain:demo:vn28	192.168.28.247	None	91089d89-76b5-46c2-abc9-b9693bcb37ac

Details :

```
{
  index: "6",
  name: "tap68ad232c-76",
  uuid: "68ad232c-76d1-4fe2-a200-42182497545e",
  vrf_name: "default-domain:demo:vn28:vn28",
  active: "Active",
  dhcp_service: "Enable",
}
```

Table 47: 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.
Instance	The name of any instance associated with the interface.

Configuring Interface Monitoring and Mirroring

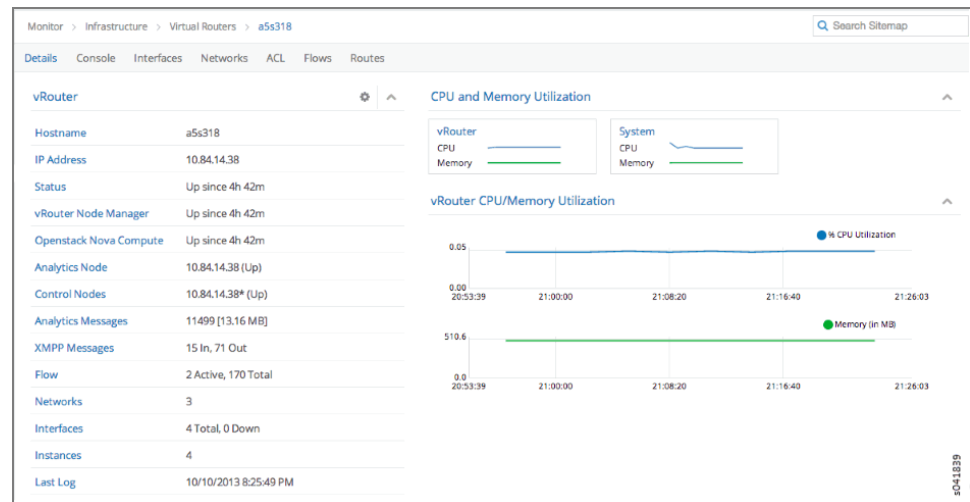
Contrail supports user monitoring of traffic on any guest virtual machine interface when using the Juniper Contrail user interface.

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 128 on page 454](#).

Figure 157: 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 129 on page 454](#).

Figure 158: Interfaces

Monitor > Infrastructure > Virtual Routers > a5s318						
Details Console Interfaces Networks ACL Flows Routes						
Interfaces						
Name	Label	Status	Network	IP Address	Floating IP	Instance
tap32e8b4a6-4b	19	Up	default-domain:demo:svc-vn-left	250.250.2.252	None	2797154f-a4fe-4e1a-a4dc-e7aa95a5a4f3
tap347d03ac-c2	16	Up	default-domain:demo:svc-vn-left	250.250.2.253	None	27b87ff5-5dc2-4b49-bd08-b15e96c311b3
tap53aef323-79	17	Up	default-domain:demo:fe	7.7.7.253	None	cee63d6b-8843-4382-a699-fd01e85781b9
tap96b4d43c-75	18	Up	default-domain:demo:be	8.8.8.253	None	26c28bb7-0259-42ee-ad92-0c29c182dbfe

The mirror packet capture starts and displays at this screen.

The mirror packet capture stops when you exit this screen.

See Also •

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 159 on page 513](#). See [Table 48 on page 513](#) for descriptions of the fields on the **Networks** tab screen.

Figure 159: Individual vRouters—Networks Tab

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

```

- {
  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: - {

```

Table 48: 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 160 on page 514](#). See [Table 49 on page 514](#) for descriptions of the fields on the **ACL** tab screen.

Figure 160: 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
195af177-0a28-49a1-9cf0-2ce-ac22af5a1	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 49: 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.
Destination Network or Prefix	The name or prefix of the destination network associated with the listed ACL.
Destination Port	The destination port associated with the listed ACL.
ACE Id	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 161 on page 515. See Table 50 on page 515 for descriptions of the fields on the **Flows** tab screen.

Figure 161: Individual vRouters—Flows Tab



Table 50: vRouters: Flows Tab Fields

Field	Description
ACL UUID	The default is to show All flows, however, you can select from a drop down list any single flow to view its details.
ACL / SG UUID	The universal unique identifier (UUID) associated with the listed ACL or SG.
Protocol	The protocol associated with the listed flow.
Src Network	The name of the source network associated with the listed flow.
Src IP	The source IP address associated with the listed flow.
Src Port	The source port of the listed flow.
Dest Network	The name of the destination network associated with the listed flow.
Dest IP	The destination IP address associated with the listed flow.
Dest Port	The destination port associated with the listed flow.
Bytes/Pkts	The number of bytes and packets associated with the listed flow.
Setup Time	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 162 on page 516. See Table 51 on page 516 for descriptions of the fields on the **Routes** tab screen.

Figure 162: Individual vRouters—Routes Tab

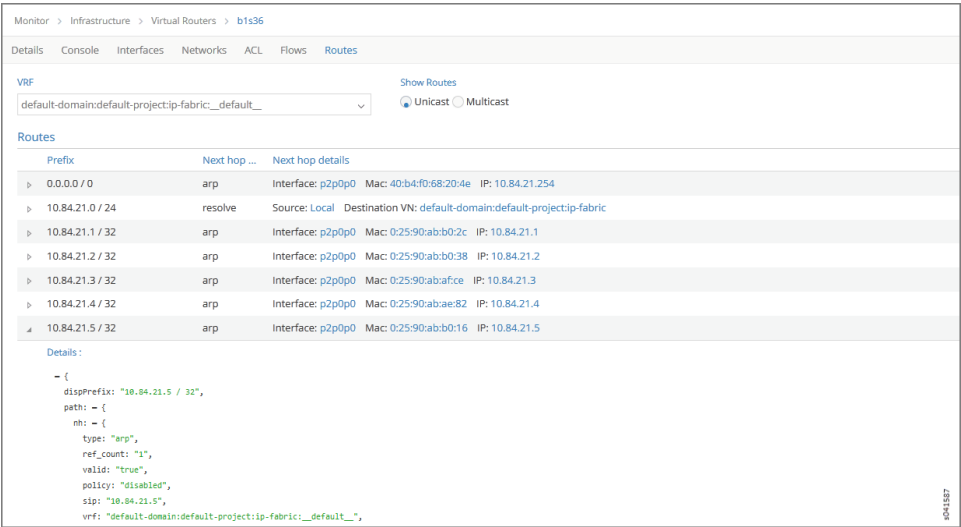


Table 51: vRouters: Routes Tab Fields

Field	Description
VRF	Select from a drop down list the virtual routing and forwarding (VRF) to view.
Show Routes	Select to show the route type: Unicast or Multicast.
Prefix	The IP address prefix of a route.
Next hop	The next hop method for this route.
Next hop details	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 163 on page 517](#). See [Table 52 on page 517](#) for descriptions of the fields on the **Console** tab screen.

Figure 163: Individual vRouter—Console Tab

Monitor > Infrastructure > Virtual Routers > b1s36

Details Console Interfaces Networks ACL Flows Routes

Console Logs

Time Range: Custom
 From Time: Oct 02, 2013 05:00:39 AM
 To Time: Oct 02, 2013 05:05:39 AM

Log Category: All
 Log Type: any
 Log Level: SYS_INFO
 Limit: Limit 10 messages
 Auto Refresh: ☒

Display Logs Reset

Time	Category	Log Type	Log
2013-10-02 05:05:39:572:199	Agent	AgentRouteLog	Added route 192.168.31.222/32 in VRF default-domain:demo:vn31:vn31 10.84.23.9
2013-10-02 05:05:34:761:107	Agent	AgentRouteLog	Added route 192.168.31.224/32 in VRF default-domain:demo:vn31:vn31 10.84.23.9
2013-10-02 05:05:34:731:318	Agent	AgentRouteLog	Added route 192.168.31.223/32 in VRF default-domain:demo:vn31:vn31 10.84.23.9
2013-10-02 05:05:32:283:326	Agent	AgentRouteLog	Added route 192.168.31.225/32 in VRF default-domain:demo:vn31:vn31 10.84.23.8
2013-10-02 05:05:31:282:424	Agent	AgentRouteLog	Added route 192.168.31.227/32 in VRF default-domain:demo:vn31:vn31 10.84.23.8
2013-10-02 05:05:29:319:521	Agent	AgentRouteLog	Added route 192.168.31.229/32 in VRF default-domain:demo:vn31:vn31 10.84.23.9

Table 52: 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 Last 5 mins through to the Last 24 hrs , plus a Custom time range.
From Time	If you select Custom in Time Range , enter the start time.
To Time	If you select Custom in Time Range , enter the end time.
Log Category	Select a log category to display: <ul style="list-style-type: none"> • All • _default_ • XMPP • IFMap • TCP
Log Type	Select a log type to display.
Log Level	Select a log severity level to display: <ul style="list-style-type: none"> • SYS_EMERG • SYS_ALERT • SYS_CRIT • SYS_ERR • SYS_WARN • SYS_NOTICE • SYS_INFO • SYS_DEBUG

Table 52: Control Node: Console Tab Fields (continued)

Field	Description
Limit	Select from a list an amount to limit the number of messages displayed: <ul style="list-style-type: none"> • No Limit • Limit 10 messages • Limit 50 messages • Limit 100 messages • Limit 200 messages • Limit 500 messages
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.
<i>Columns</i>	
Time	This column lists the time received for each log message displayed.
Category	This column lists the log category for each log message displayed.
Log Type	This column lists the log type for each log message displayed.
Log	This column lists the log message for each log displayed.

Monitor > Infrastructure > Analytics Nodes

Select **Monitor > Infrastructure > Analytics Nodes** to view the console logs, generators, and query expansion (QE) queries of the analytics nodes.

- [Monitor Analytics Nodes on page 518](#)
- [Monitor Analytics Individual Node Details Tab on page 519](#)
- [Monitor Analytics Individual Node Generators Tab on page 520](#)
- [Monitor Analytics Individual Node QE Queries Tab on page 521](#)
- [Monitor Analytics Individual Node Console Tab on page 522](#)

Monitor Analytics Nodes

Select **Monitor > Infrastructure > Analytics Nodes** to view a summary of activities for the analytics nodes; see [Figure 164 on page 519](#). See [Table 53 on page 519](#) for descriptions of the fields on the analytics summary.

Figure 164: Analytics Nodes Summary



Table 53: 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 165 on page 520](#).

See [Table 54 on page 520](#) for descriptions of the fields on this screen.

Figure 165: Monitor Analytics Individual Node Details Tab

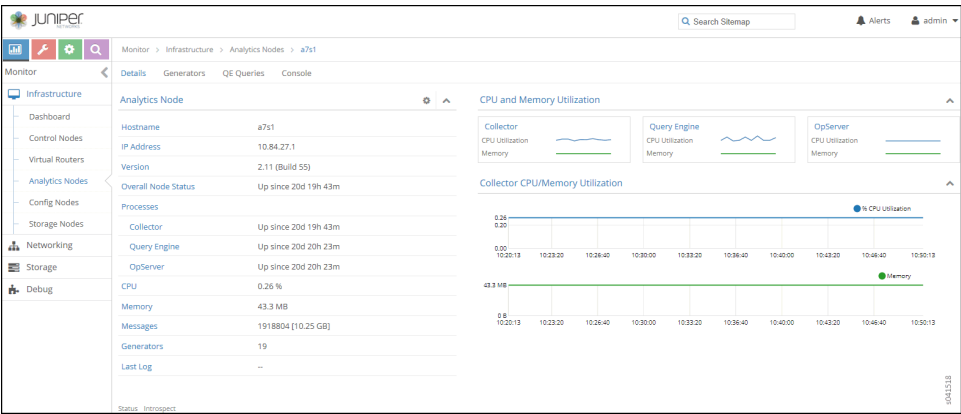


Table 54: Monitor Analytics Individual Node Details Tab Fields

Field	Description
Hostname	The name of this node.
IP Address	The IP address of this node.
Version	The installed version of the software.
Overall Node Status	The current operational status of the node — Up or Down — and the length of time in this state.
Processes	The current status of each analytics process, including Collector, Query Engine, and OpServer.
CPU (%)	The average CPU percentage usage for this node.
Memory	The average memory usage of this node.
Messages	The total number of messages for this node.
Generators	The total number of generators associated with this node.
Last Log	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 166 on page 521](#). Click the expansion arrow next to any generator name to reveal more details. See [Table 55 on page 521](#) for descriptions of the fields on the **Peers** tab screen.

Figure 166: Individual Analytics Node—Generators Tab

Name	Status	Messages	Bytes
a7s1:Analytics:contrail-analytics-api0	Up since 20d 23h 57m, Connected since 20d 23h 16m	476046	1.25 GB
a7s1:Analytics:contrail-analytics-node-mgr0	Up since 20d 23h 56m, Connected since 20d 23h 16m	5	14.32 KB
a7s1:Analytics:contrail-collector0	Up since 20d 23h 16m, Connected since 20d 23h 16m	1932437	10.25 GB
a7s1:Analytics:contrail-query-engine0	Up since 20d 23h 57m, Connected since 20d 23h 16m	928348	1.62 GB
a7s1:Analytics:contrail-snmp-collector0	Up since 20d 23h 57m, Connected since 20d 23h 16m	3	4.5 KB
a7s1:Analytics:contrail-topology0	Up since 20d 23h 57m, Connected since 20d 23h 16m	3	4.46 KB
a7s1:Compute:Storage-Stats-mgr0	Up since 20d 23h 15m, Connected since 20d 23h 15m	947488	1.22 GB
a7s1:Compute:contrail-vrouter-agent0	Up since 20d 23h 57m, Connected since 20d 23h 16m	314603	1.03 GB

Table 55: 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 167 on page 521](#).

See [Table 56 on page 521](#) for descriptions of the fields on the **QE Queries** tab screen.

Figure 167: Individual Analytics Node—QE Queries Tab

Enqueue Time	Query	Progress
No QE Queries to display		

Table 56: 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.
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 168 on page 522](#). See [Table 57 on page 522](#) for descriptions of the fields on the **Console** tab screen.

Figure 168: Analytics Individual Node—Console Tab

Table 57: 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 Last 5 mins through to the Last 24 hrs . The default display is for the Last 5 mins .
Log Category	Select a log category to display: <ul style="list-style-type: none"> All _default_ XMPP IFMap TCP
Log Type	Select a log type to display.
Log Level	Select a log severity level to display: <ul style="list-style-type: none"> SYS_EMERG SYS_ALERT SYS_CRIT SYS_ERR SYS_WARN SYS_NOTICE SYS_INFO SYS_DEBUG
Keywords	Enter any text string to search for and display logs containing that string.

Table 57: Monitor Analytics Individual Node Console Tab Fields (continued)

Field	Description
(Limit field)	Select the number of messages to display: No Limit Limit 10 messages Limit 50 messages Limit 100 messages Limit 200 messages Limit 500 messages
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.
Time	This column lists the time received for each log message displayed.
Category	This column lists the log category for each log message displayed.
Log Type	This column lists the log type for each log message displayed.
Log	This column lists the log message for each log displayed.

Monitor > Infrastructure > Config Nodes

Select **Monitor > Infrastructure > Config Nodes** to view the information about the system config nodes.

- [Monitor Config Nodes on page 523](#)
- [Monitor Individual Config Node Details on page 524](#)
- [Monitor Individual Config Node Console on page 525](#)

Monitor Config Nodes

Select **Monitor > Infrastructure > Config Nodes** to view a summary of activities for the analytics nodes. See [Figure 169 on page 524](#).

Figure 169: Config Nodes Summary

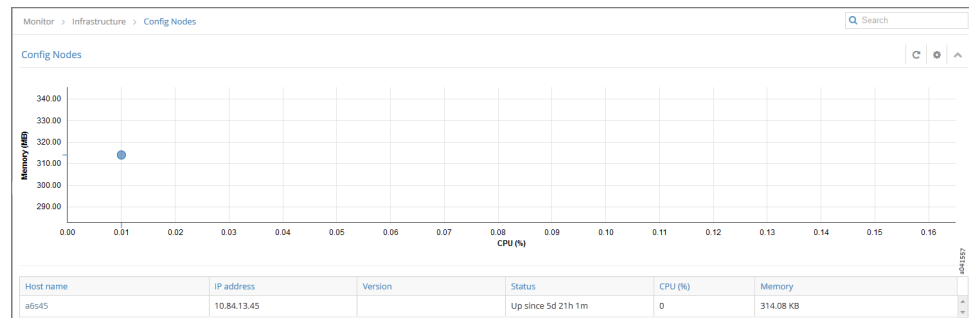


Table 58 on page 524 describes the fields in the Config Nodes summary.

Table 58: Config Nodes Summary Fields

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.

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 170 on page 524.

Figure 170: Individual Config Nodes— Details Tab

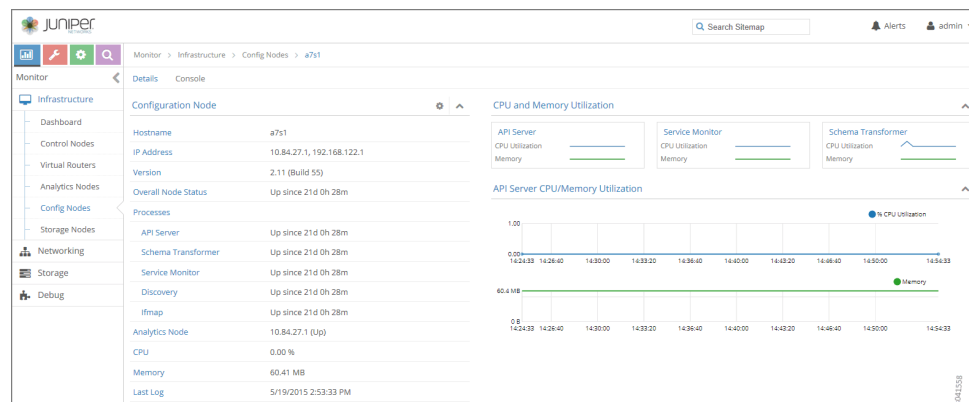


Table 59 on page 525 describes the fields on the Details screen.

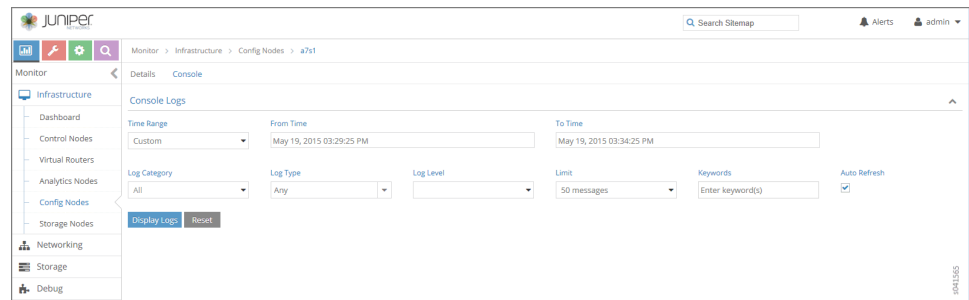
Table 59: Individual Config Nodes— Details Tab Fields

Field	Description
Hostname	The name of the config node.
IP Address	The IP address of this node.
Version	The installed version of the software.
Overall Node Status	The current operational status of the node — Up or Down — and the length of time it is in this state.
Processes	The current operational status of the processes associated with the config node, including AI Server, Schema Transformer, Service Monitor, Discovery, and Ifmap.
Analytics Node	The analytics node associated with this node.
CPU (%)	The average CPU percentage usage for this node.
Memory	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 171 on page 525](#).

Figure 171: Individual Config Node—Console Tab



See [Table 60 on page 525](#) for descriptions of the fields on the **Console** tab screen.

Table 60: Individual Config Node-Console Tab Fields

Field	Description
Time Range	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.
Log Category	Select from the drop down menu a log category to display. The option to view All is also available.
Log Type	Select a log type to display.

Table 60: Individual Config Node-Console Tab Fields (continued)

Field	Description
Log Level	Select a log severity level to display:
Limit	Select from a list an amount to limit the number of messages displayed: All Limit 10 messages Limit 50 messages Limit 100 messages Limit 200 messages Limit 500 messages
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

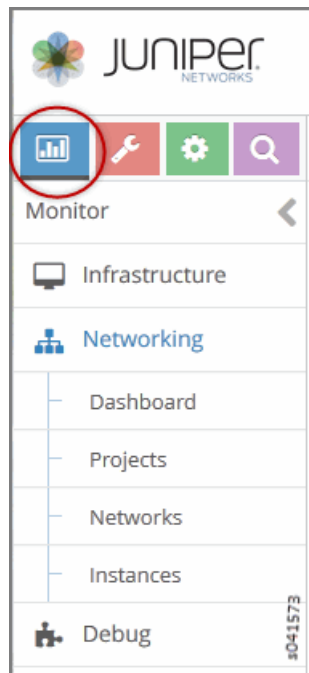
The **Monitor -> Networking** pages give an overview of the networking traffic statistics and health of domains, projects within domains, virtual networks within projects, and virtual machines within virtual networks.

- [Monitor > Networking Menu Options on page 526](#)
- [Monitor -> Networking -> Dashboard on page 527](#)
- [Monitor > Networking > Projects on page 528](#)
- [Monitor Projects Detail on page 529](#)
- [Monitor > Networking > Networks on page 531](#)

Monitor > Networking Menu Options

Figure 172 on page 527 shows the menu options available under **Monitor > Networking**.

Figure 172: 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 173 on page 527](#).

Figure 173: Traffic Statistics for Domain Window

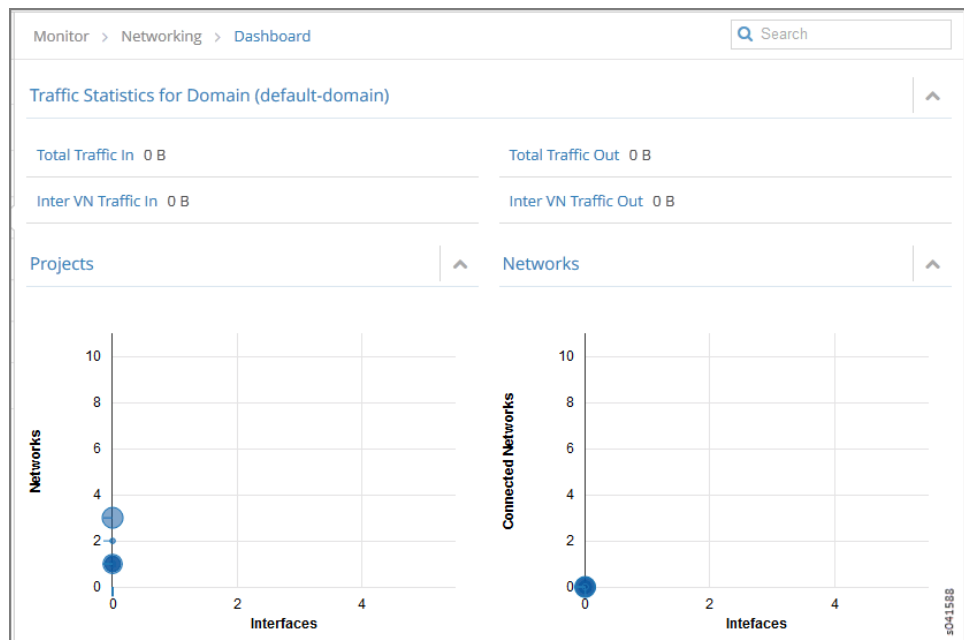


Table 61 on page 528 describes the fields in the Traffic Statistics for Domain window.

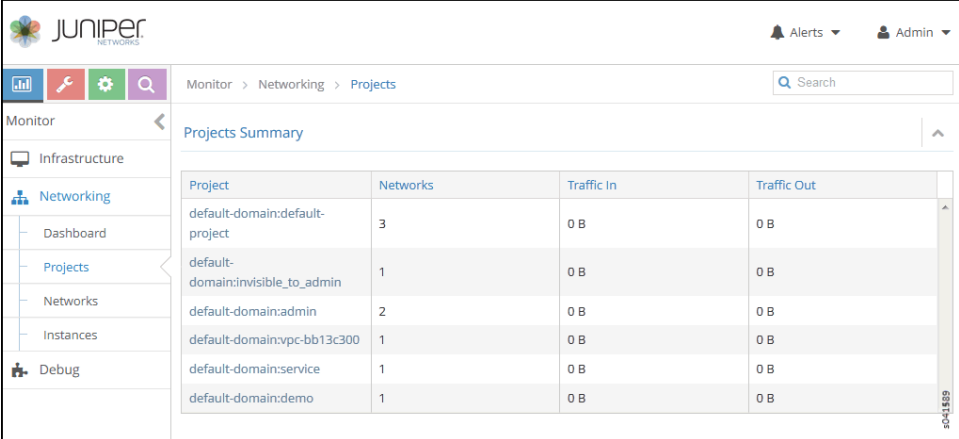
Table 61: Projects Summary Fields

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.
Projects	This chart displays the networks and interfaces for projects with the most throughput over the past 30 minutes. Click Projects then select Monitor > Networking > Projects , to display more detailed statistics.
Networks	This chart displays the networks for projects with the most throughput over the past 30 minutes. Click Networks then select Monitor > Networking > Networks , to display more detailed statistics.

Monitor > Networking > Projects

Select **Monitor > Networking > Projects** to see information about projects in the system. See Figure 174 on page 528.

Figure 174: Monitor > Networking > 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 62 on page 528 for descriptions of the fields on this screen.

Table 62: Projects Summary Fields

Field	Description
Projects	The name of the project. You can click the name to access details about connectivity for this project.
Networks	The volume of inter-virtual network traffic out of this domain.

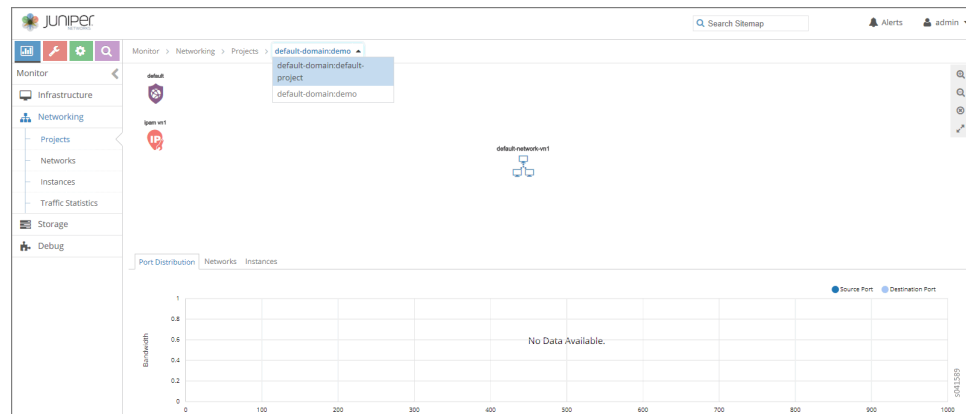
Table 62: Projects Summary Fields (continued)

Field	Description
Traffic In	The volume of traffic into this domain.
Traffic Out	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 175 on page 529](#). Hover over any of the connections to get more details.

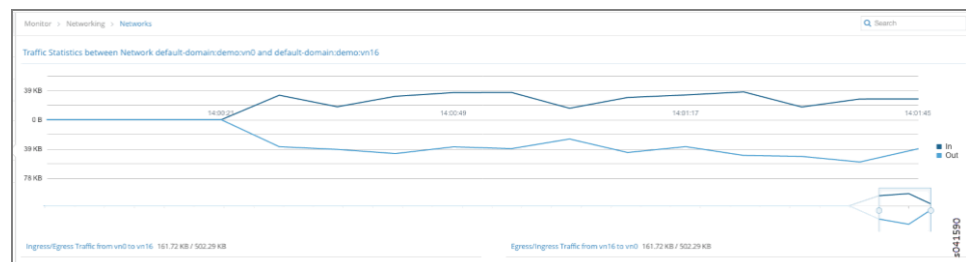
Figure 175: Monitor Projects Connectivity Details



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 176 on page 529](#).

Figure 176: 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 177: Projects Instances Summary

Instance	Virtual Network	Interfaces	vRouter	IP Address	Floating IP	Traffic (In/Out)
out	default-domain:admin:right	1	hp1	2.2.2.252		129.87 KB / 119.83 KB
NAT1_1	default-domain:admin:right	1	hp1	2.2.2.253 250.250.1.253 (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

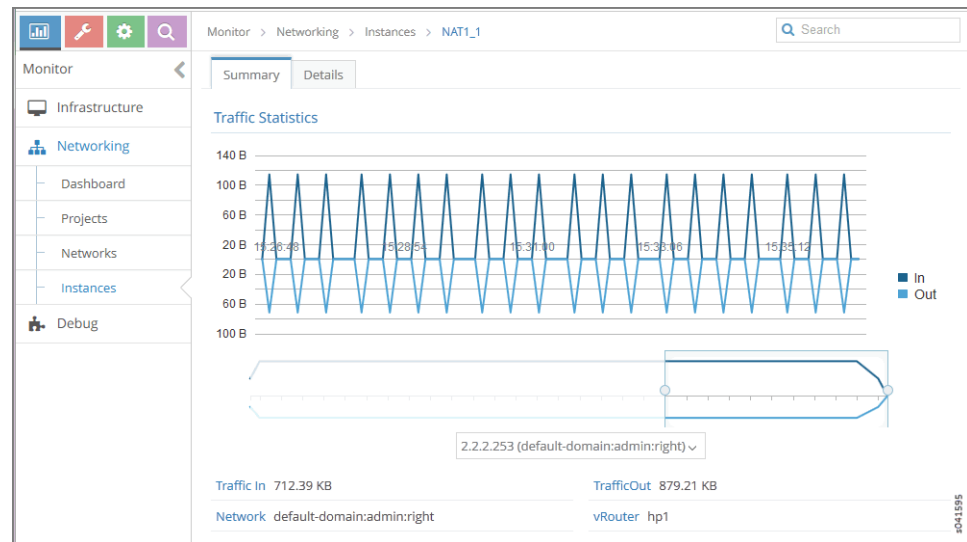
See Table 3 for a description of the fields on this screen.

Table 63: Projects Instances Summary Fields

Field	Description
Instance	The name of the instance. Click the name then select Monitor > Networking > Instances to display details about the traffic statistics for this instance.
Virtual Network	The virtual network associated with this instance.
Interfaces	The number of interfaces associated with this instance.
vRouter	The name of the vRouter associated with this instance.
IP Address	Any IP addresses associated with this instance.
Floating IP	Any floating IP addresses associated with this instance.
Traffic (In/Out)	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 178 on page 531](#).

Figure 178: Instance Traffic Statistics



Monitor > Networking > Networks

Select **Monitor > Networking > Networks** to view a summary of the virtual networks in your system. See [Figure 179 on page 531](#).

Figure 179: 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 64: Network Summary Fields

Field	Description
Network	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.
Instances	The number of instances launched in this network.
Traffic (In/Out)	The volume of inter-virtual network traffic in and out of this network.
Throughput (In/Out)	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 180 on page 532 shows the **Summary** tab for an individual network, which displays connectivity details and traffic statistics for the selected network.

Figure 180: Individual Network Connectivity Details—Summary Tab

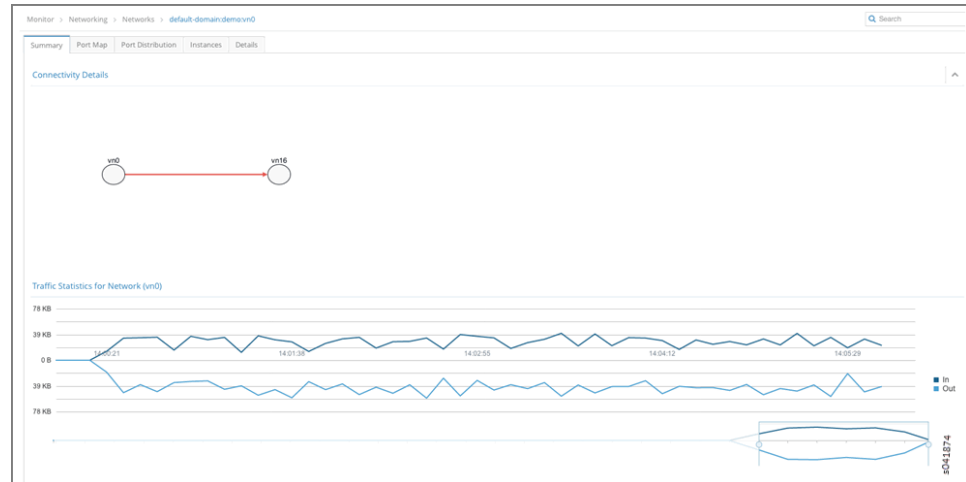


Figure 181 on page 532 shows the **Port Map** tab for an individual network, which displays the relative distribution of traffic for this network by protocol, by port.

Figure 181: Individual Network— Port Map Tab

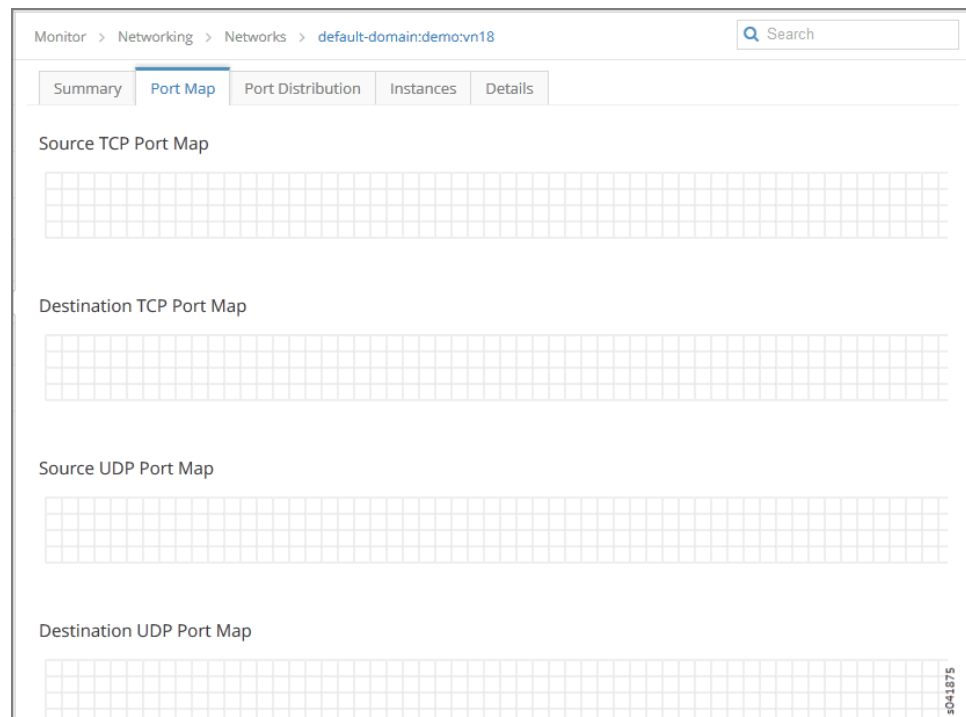


Figure 182 on page 533 shows the **Port Distribution** tab for an individual network, which displays the relative distribution of traffic in and out by source port and destination port.

Figure 182: Individual Network—Port Distribution Tab

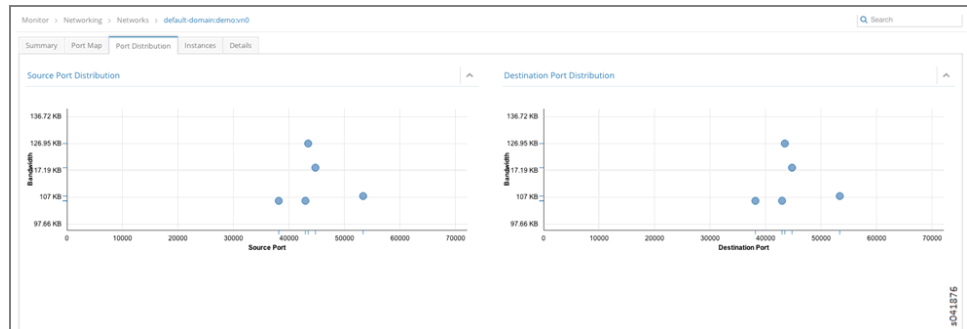


Figure 183 on page 533 shows the **Instances** tab for an individual network, which displays details for each instance associated with this network, including the number of interfaces, the associated vRouter, the instance IP address, and the volume of traffic in and out.

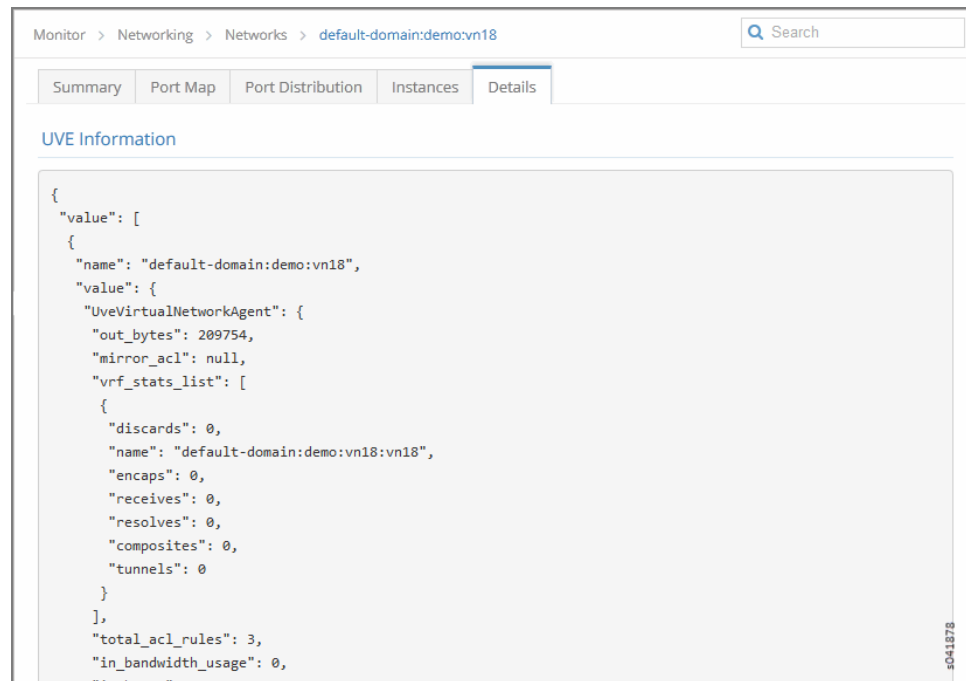
Additionally, you can click the arrow near the instance name to reveal even more details about the instance—the interfaces and their addresses, UUID, CPU (usage), and memory used of the total amount available.

Figure 183: Individual Network Instances Tab

Monitor > Networking > Networks > default-domain:demo:vn18						
Search						
Summary Port Map Port Distribution Instances Details						
Instances Summary						
	Instance	Interfaces	vRouter	IP Address	Floating IP	Traffic (In/Out)
▶	vn18_vm-b342ca93-9acd-4275-acb8-df7b5843884c	1	b1s29	192.168.18.225		1.13 KB / 712.00 B
▲	vn18_vm-22a42bf6-fccc-4db3-b5ac-80082bbefbef	1	b1s42	192.168.18.236		1.13 KB / 712.00 B
	Interfaces IP Address: 192.168.18.236 Label: 17 Mac Address: 02:e9:94:e7:0e:56 Network: default-domain:demo:vn18 Traffic (In/Out): 1.13 KB/712.00 B UUID 22a42bf6-fccc-4db3-b5ac-80082bbefbef CPU 0.01 Memory (Used/Total) 1.23 GB / 15.63 GB					
▶	vn18_vm-f676567a-826f-4e9d-9a81-b4649b7fcde2	1	b1s15	192.168.18.235		1.13 KB / 712.00 B

Figure 184 on page 534 shows the **Details** tab for an individual network, which displays the code used to define this network --the User Virtual Environment (UVE) code.

Figure 184: Individual Network Details Tab



Query > Flows

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 on page 534](#)
- [Example: Query Flow Series on page 537](#)
- [Query > Flow Records on page 538](#)
- [Query > Flows > Query Queue on page 540](#)

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 185 on page 535](#)

Figure 185: Query Flow Series Window

The query fields available on the screen for the **Flow Series** tab are described in [Table 65 on page 535](#). Enter query data into the fields to create a SQL-like query to display and analyze flows.

Table 65: Query Flow Series Fields

Field	Description
Time Range	<p>Select a range of time to display the flow series:</p> <ul style="list-style-type: none"> • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom <p>Click Custom to enter a specific custom time range in two fields: From Time and To Time.</p>
Select	Click the edit button (pencil icon) to open a Select window (Figure 186 on page 536), where you can click one or more boxes to select the fields to display from the flow series, such as Source VN , Dest VN , Bytes , Packets , and more.
Where	Click the edit button (pencil icon) to open a query-writing window, where you can specify query values for variables such as sourcevn , sourceip , destvn , destip , protocol , sport , dport .
Direction	Select the desired flow direction: INGRESS or EGRESS .
Filter	Click the edit button (pencil icon) to open a Filter window (Figure 187 on page 537), where you can select filter items to sort by, the sort order, and limits to the number of results returned.
Run Query	Click Run Query 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 Time Granularity is selected, you have the option to view results in graph or flowchart form. Graph buttons appear on the screen above the Export button. Click a graph button to transform the tabular results into a graphical chart display.

Table 65: Query Flow Series Fields (continued)

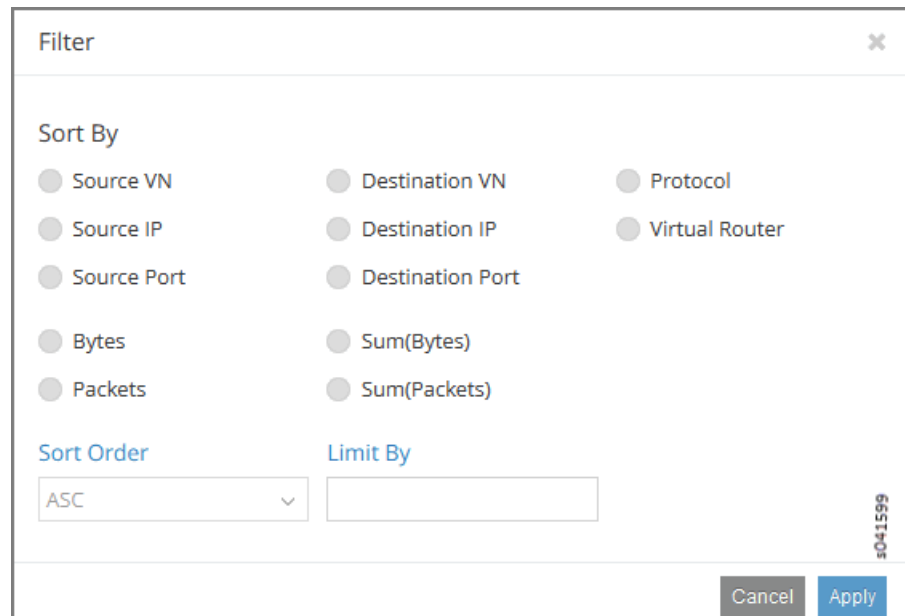
Field	Description
Export	The Export button is displayed after you click Run Query . This allows you to export the list of flows to a text .csv file.

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 186 on page 536](#). 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 186: Flow Series Select

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 187 on page 537](#).

Figure 187: Flow Series Filter



The 'Filter' dialog box contains the following elements:

- Sort By:** A grid of radio buttons for selecting the sort criteria: Source VN, Destination VN, Protocol, Source IP, Destination IP, Virtual Router, Source Port, Destination Port, Bytes, Sum(Bytes), Packets, and Sum(Packets).
- Sort Order:** A dropdown menu currently set to 'ASC'.
- Limit By:** An empty text input field.
- Buttons:** 'Cancel' and 'Apply' buttons at the bottom right.
- Footer:** A small vertical text '© 2018 Juniper Networks, Inc.' on the right side.

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 188 on page 537](#).

Figure 188: Example: Query Flow Series



The 'Query Flow Series' interface includes the following fields and controls:

- Time Range:** A dropdown menu set to 'Last 10 Mins'.
- Select:** A text input field containing the query 'sourcevn, destvn, time-granularity, sum(bytes)'.
- Where:** An empty text input field.
- Filter:** An empty text input field.
- Time Granularity:** A dropdown menu set to '10' with a unit of 'secs'.
- Direction:** A dropdown menu set to 'INGRESS'.
- Buttons:** 'Run Query' at the bottom left and a small '© 2018 Juniper Networks, Inc.' footer on the right.

The query returns tabular time series data, see [Figure 189 on page 538](#), for the following combinations of Source VN and Dest VN:

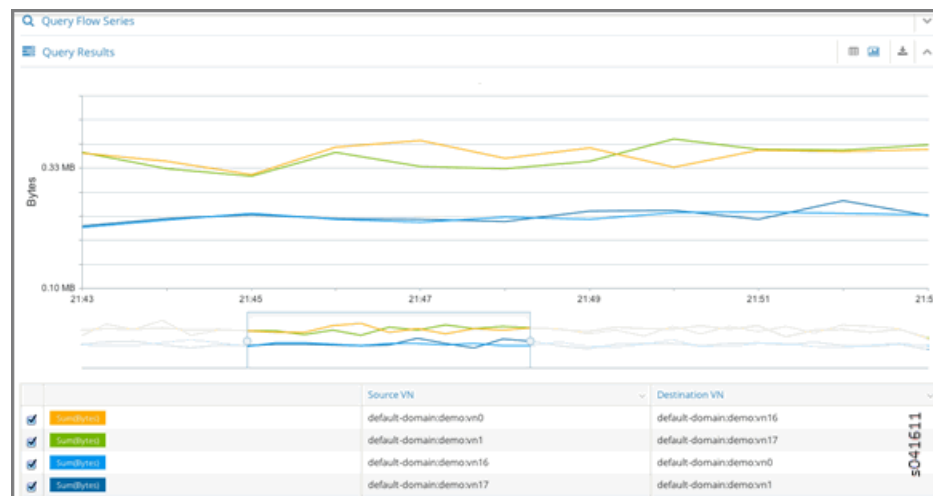
1. Flow Class 1: Source VN = default-domain:demo:front-end, Dest VN= __UNKNOWN__
2. Flow Class 2: Source VN = default-domain:demo:front-end, Dest VN=default-domain:demo:back-end

Figure 189: 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	default-domain:demo:vn0	default-domain:demo:vn16	INGRESS	421,128
2013-08-05 18:59:40:0	default-domain:demo:vn0	default-domain:demo:vn16	INGRESS	227,000
2013-08-05 18:59:50:0	default-domain:demo:vn0	default-domain:demo:vn16	INGRESS	216,816
2013-08-05 19:00:00:0	default-domain:demo:vn0	default-domain:demo:vn16	INGRESS	387,036
2013-08-05 18:59:30:0	default-domain:demo:vn1	default-domain:demo:vn17	INGRESS	52,944
2013-08-05 18:59:40:0	default-domain:demo:vn1	default-domain:demo:vn17	INGRESS	52,692
2013-08-05 18:59:50:0	default-domain:demo:vn1	default-domain:demo:vn17	INGRESS	58,040
2013-08-05 19:00:00:0	default-domain:demo:vn1	default-domain:demo:vn17	INGRESS	42,480
2013-08-05 18:59:30:0	default-domain:demo:vn16	default-domain:demo:vn0	INGRESS	17,832
2013-08-05 18:59:40:0	default-domain:demo:vn16	default-domain:demo:vn0	INGRESS	27,320
2013-08-05 18:59:50:0	default-domain:demo:vn16	default-domain:demo:vn0	INGRESS	20,792
2013-08-05 19:00:00:0	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 190 on page 538](#).

Figure 190: 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 191: Flow Records

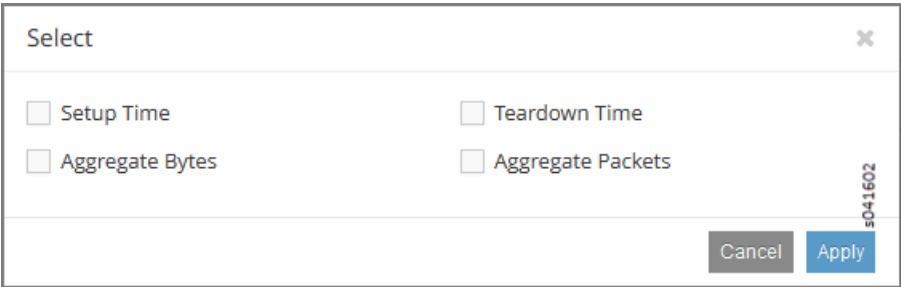
The query fields available on the screen for the **Flow Records** tab are described in [Table 66 on page 539](#). Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 66: Query Flow Records Fields

Field	Description
Time Range	<p>Select a range of time for the flow records:</p> <ul style="list-style-type: none"> • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom <p>Click Custom to enter a specified custom time range in two fields: From Time and To Time.</p>
Select	Click the edit button (pencil icon) to open a Select window (Figure 192 on page 540), where you can click one or more boxes to select attributes to display for the flow records, including Setup Time , Teardown Time , Aggregate Bytes , and Aggregate Packets .
Where	Click the edit button (pencil icon) to open a query-writing window where you can specify query values for sourcevn , sourceip , destvn , destip , protocol , sport , dport . .
Direction	Select the desired flow direction: INGRESS or EGRESS .
Run Query	Click Run Query 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.
Export	The Export button is displayed after you click Run Query , allowing you to export the list of flows to a text .csv file.

The **Select** window allows you to select one or more attributes to display for the flow records selected, see [Figure 192 on page 540](#).

Figure 192: Flow Records Select Window



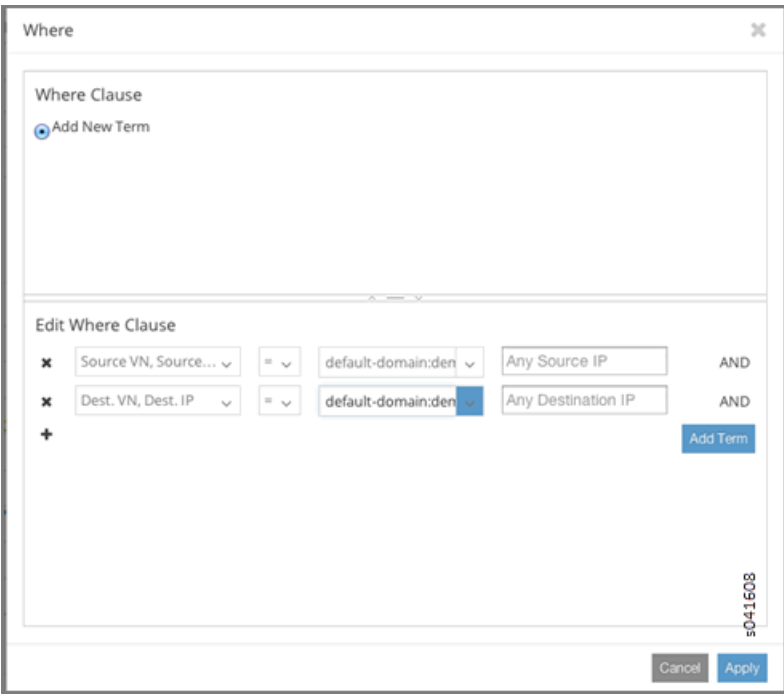
The 'Select' window contains four checkboxes for selecting flow record fields: 'Setup Time', 'Teardown Time', 'Aggregate Bytes', and 'Aggregate Packets'. At the bottom right, there are 'Cancel' and 'Apply' buttons. A vertical label 's041602' is positioned on the right side of the window.

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 193: Where Clause Window



The 'Where' window is used to define a 'Where Clause'. It features an 'Add New Term' button. Below, the 'Edit Where Clause' section shows two terms being built. The first term is 'Source VN, Source...' followed by an equals sign, a dropdown menu showing 'default-domain:den', and a text box containing 'Any Source IP', with 'AND' to its right. The second term is 'Dest. VN, Dest. IP' followed by an equals sign, a dropdown menu showing 'default-domain:den', and a text box containing 'Any Destination IP', with 'AND' to its right. An 'Add Term' button is located at the bottom right of the clause editor. At the bottom of the window are 'Cancel' and 'Apply' buttons. A vertical label 's041608' is positioned on the right side of the 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 194 on page 541](#).

Figure 194: Flows Query Queue

Date	Query	Progress	Records	Status	Time Taken	
2013-10-09 18:07:06	{ "table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": { "flow_class_id", "direction_ing", "sum(bytes)", "T=60", "dir": 1 } }	100%	180	completed	150 secs	⚙️
2013-10-09 17:55:48	{ "table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": { "flow_class_id", "direction_ing", "sum(bytes)", "T=60", "dir": 1 } }	100%	180	completed	145 secs	⚙️
2013-10-09 17:29:39	{ "table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": { "flow_class_id", "direction_ing", "sum(bytes)", "T=60", "dir": 1 } }	100%	180	completed	170 secs	⚙️
2013-10-09 16:57:10	{ "table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": { "flow_class_id", "direction_ing", "sum(bytes)", "T=60", "dir": 1 } }	100%	180	completed	270 secs	⚙️
2013-10-09 16:39:48	{ "table": "FlowSeriesTable", "start_time": 1381360140000000, "end_time": 1381361940000000, "select_fields": { "flow_class_id", "direction_ing", "T=60", "sum(bytes)", "dir": 1 } }	100%	30	completed	60 secs	⚙️
2013-10-09 11:07:29	{ "table": "FlowSeriesTable", "start_time": 1381338420000000, "end_time": 1381342020000000, "select_fields": { "flow_class_id", "direction_ing", "sum(bytes)", "T=60", "dir": 1 } }	100%	7	completed	15 secs	⚙️

Displaying 1 - 6 of 31 Records

The query fields available on the screen for the **Flow Records** tab are described in [Table 67 on page 541](#). Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 67: Query Flow Records Fields

Field	Description
Date	The date and time the query was started.
Query	A display of the parameters set for the query.
Progress	The percentage completion of the query to date.
Records	The number of records matching the query to date.
Status	The status of the query, such as completed .
Time Taken	The amount of time in seconds it has taken the query to return the matching records.
(Action icon)	Click the Action icon and select View Results to view a list of the records that match the query, or click Delete to remove the query from the queue.

See Also • [Understanding Flow Sampling on page 546](#)

Query > Logs

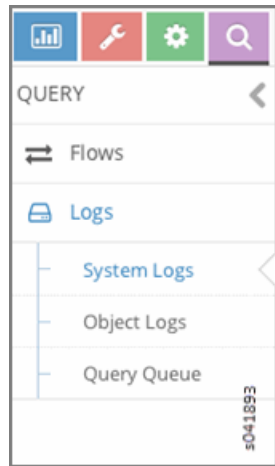
The **Query > Logs** option allows you to access the system log and object log activity of any Contrail Controller component from one central location.

- [Query > Logs Menu Options on page 542](#)
- [Query > Logs > System Logs on page 542](#)
- [Sample Query for System Logs on page 543](#)
- [Query > Logs > Object Logs on page 544](#)

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 195 on page 542](#).

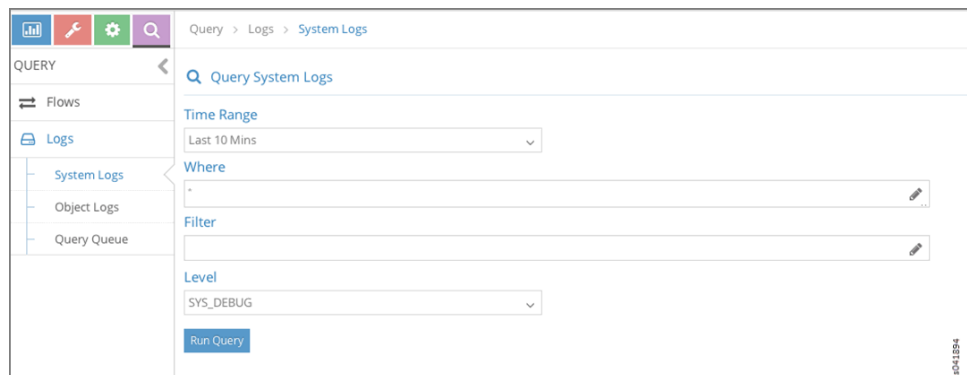
Figure 195: Query > Logs



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 196 on page 542](#).

Figure 196: Query > Logs > System Logs



The query fields available on the **Query System Logs** screen are described in [Table 68 on page 543](#).

Table 68: Query System Logs Fields

Field	Description
Time Range	<p>Select a range of time for which to see the system logs:</p> <ul style="list-style-type: none"> • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom <p>If you click Custom, enter a desired time range in two new fields: From Time and To Time.</p>
Where	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.
Level	<p>Select the message severity level to view:</p> <ul style="list-style-type: none"> • SYS_NOTICE • SYS_EMERG • SYS_ALERT • SYS_CRIT • SYS_ERR • SYS_WARN • SYS_INFO • SYS_DEBUG
Run Query	Click this button to retrieve the system logs that match the query. The logs are listed in a box with columns showing the Time , Source , Module Id , Category , Log Type , and Log message.
Export	This button appears after you click Run Query , 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 197 on page 544](#).

Figure 197: Edit Where Clause

The 'Where' dialog box is shown with a close button (X) in the top right corner. It contains two main sections: 'Where Clause' and 'Edit Where Clause'.

Where Clause: This section has a radio button labeled 'Add New Term' which is currently selected.

Edit Where Clause: This section contains a list of terms to be included in the where clause. Each term is represented by a row with a delete icon (X), a field for the attribute name, a comparison operator, a field for the value, and a logical connector.

Attribute	Operator	Value	Connector
ModuleId	=	VRouterAgent	AND
Source	=	b1s16	AND

Below the list is a plus icon (+) and an 'Add Term' button. At the bottom of the dialog are 'OK' and 'Cancel' buttons.

- The information you defined at the Where screen displays on the **Query System Logs**. Enter any more defining information needed; see [Figure 198 on page 544](#). When finished, click **Run Query** to display the results.

Figure 198: Sample Query System Logs

The 'Query System Logs' interface is shown. It includes a search bar at the top with the text 'Query System Logs'. Below the search bar are several configuration options:

- Time Range:** A dropdown menu currently set to 'Last 10 Mins'.
- Where:** A text field containing the query '(ModuleId = VRouterAgent AND Source = b1s16)' with a delete icon (X) on the right.
- Filter:** An empty text field with a delete icon (X) on the right.
- Level:** A dropdown menu currently set to 'SYS_DEBUG'.

At the bottom left is a 'Run Query' button. A small identifier 's041896' is visible in the bottom right corner.

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 199 on page 545](#).

Figure 199: Query > Logs > Object Logs

The query fields available on the **Object Logs** screen are described in [Table 69 on page 545](#).

Table 69: Object Logs Query Fields

Field	Description
Time Range	<p>Select a range of time for which to see the logs:</p> <ul style="list-style-type: none"> • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom <p>If you click Custom, enter a desired time range in two new fields: From Time and To Time.</p>
Object Type	<p>Select the object type for which to show logs:</p> <ul style="list-style-type: none"> • Virtual Network • Virtual Machine • Virtual Router • BGP Peer • Routing Instance • XMPP Connection
Object Id	Select from a list of available identifiers the name of the object you wish to use.
Select	<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"> • ObjectLog • SystemLog

Table 69: Object Logs Query Fields (continued)

Field	Description
Where	Click the edit button (pencil icon) to open the query-writing window, where you can specify query values for variables such as Source , ModuleId , and MessageType , in order to retrieve information as specific as you wish.
Run Query	Click this button to retrieve the system logs that match the query. The logs are listed in a box with columns showing the Time , Source , Module Id , Category , Log Type , and Log message.
Export	This button appears after you click Run Query , allowing you to export the list of system messages to a text/csv file.

Understanding Flow Sampling

This topic describes how flow records are sampled and exported to the Contrail Collector, flow handling, and flow aging.

- [Flow Sampling on page 546](#)
- [Flow Handling on page 547](#)
- [Flow Aging on page 547](#)
- [TCP State-Based Flow Handling and Aging on page 548](#)

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.



NOTE: In releases earlier than Contrail Release 2.22, all flow records were exported from the agent. Depending on the scale of flows, some of these exported flows might be dropped due to queue overflow.

In Contrail Release 2.22 and later, flow records are sampled and exported to the Contrail Collector based on the 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 setup. 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 on page 548](#)
- [Protocol Based Flow Aging on page 548](#)
- [Fat Flow on page 548](#)

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 tear down, 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, (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. Thus 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

In Contrail Release 2.22 and later, Contrail supports optimization to reduce the number of flows setup by reusing a flow. That is, a single flow pair can be used for any number of sessions between two endpoints for the same application protocol.

Any number of DNS sessions from a client to the server can use a single flow pair. The effect is that the flow hash key is reduced from five-tuples to four-tuples consisting of source and destination IP addresses, the server port, and the IP protocol. The client port is not used in the flow key.

This feature can be configured by specifying the list of *fat-flow* protocols on a virtual machine interface. For each such application protocol, the list contains the protocol and port pairs. In the example, the server **virtual-machine-interface**, protocol **udp** and port **53** can be configured as a fat-flow-protocol.

If you want to enable the fat-flow feature on the client side, the configuration must be applied on the client virtual machine interface as well.

Related Documentation

- [Query > Flows on page 534](#)

Example: Debugging Connectivity Using Monitoring for Troubleshooting


- [Using Monitoring to Debug Connectivity on page 549](#)

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 200 on page 549](#)

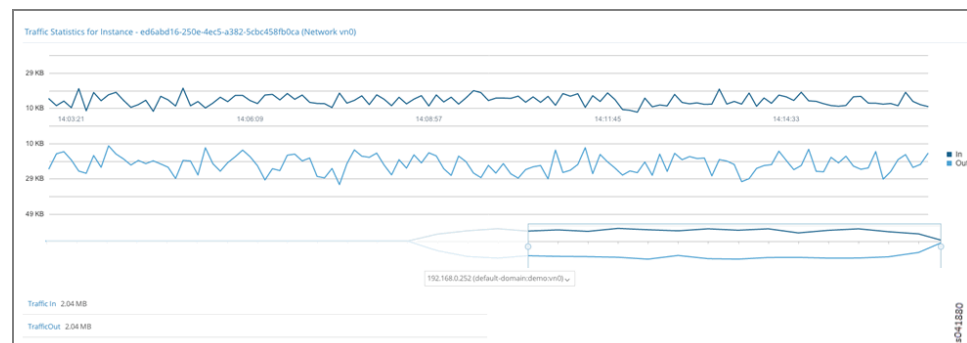
Figure 200: Navigate to Instance



Instance	Traffic In	Traffic Out
ed6abd16-250e-4ec5-a382-5cbc458fb0ca	1.73 MB	1.74 MB
682b7414-c4ba-45ee-9f8c-9c22c0b0c0d	1.72 MB	1.72 MB

2. Click the instance to view **Traffic Statistics for Instance**. see [Figure 201 on page 549](#).

Figure 201: Traffic Statistics for Instance



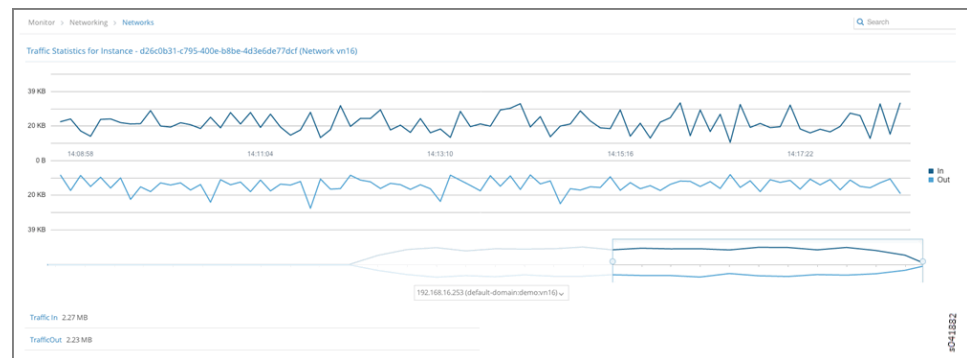
3. Instance **d26c0b31-c795-400e-b8be-4d3e6de77dcf** with IP address **192.168.0.253** in the virtual network **vn16**. see [Figure 202 on page 550](#) and [Figure 203 on page 550](#).

Figure 202: Navigate to Instance



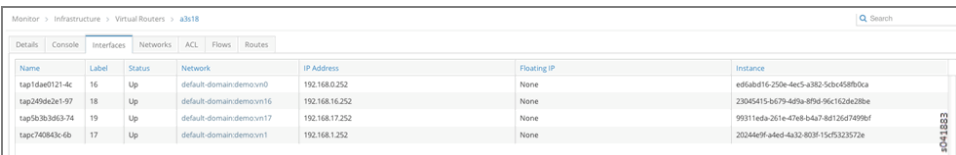
Instance	Traffic In	Traffic Out
d26c0b31-c795-400e-b8be-4d3e6de77dcf	2.18 MB	2.13 MB
23045415-b679-4d8a-8f9d-96c1626c28be	2.11 MB	2.16 MB

Figure 203: 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 204 on page 550](#).

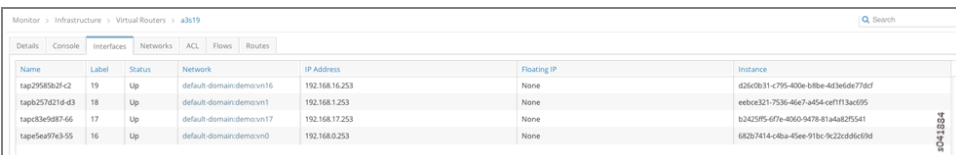
Figure 204: Navigate to a3s18 Interfaces



Name	Label	Status	Network	IP Address	Floating IP	Instance
tap1d4e0121-4c	16	Up	default-domain:demo:vn0	192.168.0.252	None	ed6abd16-250e-4ec5-a382-5cbc458fb0ca
tap349d8c2e1-97	18	Up	default-domain:demo:vn16	192.168.16.252	None	23045415-b679-4d8a-8f9d-96c1626c28be
tap5b7b3a63-74	19	Up	default-domain:demo:vn17	192.168.17.252	None	99311eda-261e-47e8-b4a7-8d13d67499bf
tapc740843c-6b	17	Up	default-domain:demo:vn1	192.168.1.252	None	20244ef9-afed-4a32-8038-15c75323572e

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 205 on page 550](#).

Figure 205: Navigate to a3s19 Interfaces



Name	Label	Status	Network	IP Address	Floating IP	Instance
tap29595b2f-c2	19	Up	default-domain:demo:vn16	192.168.16.253	None	d26c0b31-c795-400e-b8be-4d3e6de77dcf
tapb257621d-d3	18	Up	default-domain:demo:vn1	192.168.1.253	None	eebce321-7536-4ee7-a454-eef1f13ac095
tapc3e3e87-66	17	Up	default-domain:demo:vn17	192.168.17.253	None	b242595-677e-4060-9478-81a4a8275d41
tapc5ea73e3-55	16	Up	default-domain:demo:vn0	192.168.0.253	None	682b7414-c4ba-45ee-91bc-9c22cd86c9d

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 206 on page 551](#) and [Figure 207 on page 551](#).

Figure 206: ACL Connectivity a3s18

Monitor > Infrastructure > Virtual Routers > a3s18									
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
a7249326-3f50-477a-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
b32143a3-0e80-4ae2-9c...	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
b8c9810-e9fc-419b-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
d1b47291-7a21-469e-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 207: ACL Connectivity a3s19

Monitor > Infrastructure > Virtual Routers > a3s19									
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
a7249326-3f50-477a-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
b32143a3-0e80-4ae2-9c...	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
b8c9810-e9fc-419b-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
d1b47291-7a21-469e-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 208 on page 551 and Figure 209 on page 552.

Figure 208: Routes default-domain:demo:vn0:vn0

Monitor > Infrastructure > Control Nodes > a3s15							
Details Console Peers Routes							
Routing Instance		default-domain:demo:vn0	Address Family		All	Limit 50 Routes	
Peer Source		All	Prefix		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	-
10.84.17.4:1:255.255.255.255,0.0.0.0	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	-
10.84.17.5:1:255.255.255.255,0.0.0.0	inetmcast	XMPP	a3s19	10.84.17.5	0	100	-

Figure 209: Routes default-domain:demo:vn16:vn16

Monitor > Infrastructure > Control Nodes > a3s15							
<div> <div>Details Console Peers Routes</div> <div> <div>Routing Instance</div> <div>default-domain:demo</div> </div> <div> <div>Address Family</div> <div>All</div> </div> <div> <div>Limit 50 Routes</div> <div></div> </div> </div>							
<div> <div>Peer Source</div> <div>All</div> </div> <div> <div>Prefix</div> <div>Prefix</div> </div> <div> <div>Display Routes</div> <div>Reset</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	-
192.168.0.252/32	inet	BGP	10.84.17.3	10.84.17.4	16	100	AS_PATH: 0
192.168.0.252/32	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	-
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 210 on page 552](#).

Figure 210: Verify Route and Next Hop a3s18

Monitor > Infrastructure > Virtual Routers > a3s18							
<div> <div>Details Console Interfaces Networks ACL Flows Routes</div> <div> <div>VRF</div> <div>default-domain:demo:vn0:vn0</div> </div> <div> <div>Show Routes</div> <div>Unicast Multicast</div> </div> </div>							
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					

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 211 on page 553](#).

Figure 211: 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 212 on page 553 and Figure 213 on page 553.

Figure 212: Flows for a3s18

Monitor > Infrastructure > Virtual Routers > a3s18

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
TCP	vn0	192.168.0.252	43434	vn16	192.168.16.253	9100	1884588/5417	21:00:22.131180 2013-Aug-06
TCP	vn16	192.168.16.252	9100	vn0	192.168.0.252	43434	196968/5891	21:00:22.131192 2013-Aug-06
TCP	vn16	192.168.16.253	9101	vn0	192.168.0.252	53369	190500/5805	21:00:22.206222 2013-Aug-06
TCP	vn0	192.168.0.252	53369	vn16	192.168.16.253	9101	189008/5302	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	1785788/3107	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	1891368/5486	21:00:22.395695 2013-Aug-06
TCP	vn0	192.168.0.252	34236	vn16	192.168.16.252	9100	0/0	21:00:22.400371 2013-Aug-06

Figure 213: Flows for a3s19

Monitor > Infrastructure > Virtual Routers > a3s19

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	1063980/1975	21:00:24.111374 2013-Aug-06
UDP	vn16	192.168.16.253	9201	vn0	192.168.0.252	44794	1106040/1963	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	192.168.0.252	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	21:00:23.440564 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	53390	1088602/1953	21:00:25.440166 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	1704773/8094	21:00:23.514261 2013-Aug-06

CHAPTER 23

Common Support Answers

- [Debugging Ping Failures for Policy-Connected Networks on page 555](#)
- [Debugging BGP Peering and Route Exchange in Contrail on page 561](#)
- [Troubleshooting the Floating IP Address Pool in Contrail on page 577](#)
- [Removing Stale Virtual Machines and Virtual Machine Interfaces on page 604](#)
- [Troubleshooting Link-Local Services in Contrail on page 608](#)

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 214: Virtual Machine Status Window

Name	Label	Status	Network	IP Address	Floating IP	Instance
tap8085f9c6-67	16	Up	vn1 (admin)	31.1.1.253	10.204.219.108	549533ef-403e-403e-bc47-6c748e6f5c8 / vn1

Check the tap interface status in the http agent introspect, for example:

http://nodef1.englab.juniper.net:8085/Snh_ItfReq?name=

Figure 215: Tap Interface Status Window

Index	name	uuid	vrf_name	active
4	tap8085f9c6-67	8889c6-672e-4c1e-9e83-e05386c911ab	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 VNI-VN2 ACL information from the compute node:

Figure 216: 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-a72e-30f4dbc2b5ee	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 217: 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 218: Virtual Network Route Information Window

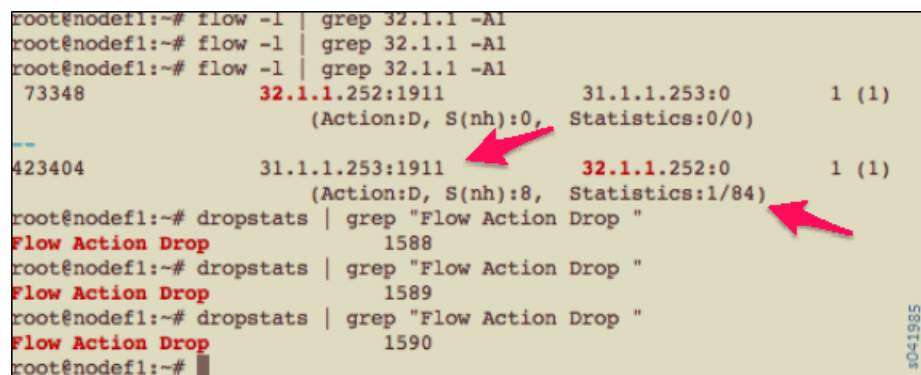
Prefix	Next hop Type	Next hop details
32.1.1.251 / 32	tunnel	Source IP: 192.168.40.11 Destination IP: 192.168.40.11 disabled Valid: true
32.1.1.252 / 32	interface	Interface: tap2e0a5c4d-1f Destination VN: default-domain:admin:vn1:vn2
32.1.1.253 / 32	tunnel	Source IP: 192.168.40.11 Destination IP: 192.168.40.11 disabled Valid: true

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 219: Flow and Dropstats Command List



```

root@nodefl:~# flow -l | grep 32.1.1 -Al
root@nodefl:~# flow -l | grep 32.1.1 -Al
root@nodefl:~# flow -l | grep 32.1.1 -Al
73348          32.1.1.252:1911          31.1.1.253:0          1 (1)
                (Action:D, S(nh):0, Statistics:0/0)
--
423404          31.1.1.253:1911          32.1.1.252:0          1 (1)
                (Action:D, S(nh):8, Statistics:1/84)
root@nodefl:~# dropstats | grep "Flow Action Drop "
Flow Action Drop          1588
root@nodefl:~# dropstats | grep "Flow Action Drop "
Flow Action Drop          1589
root@nodefl:~# dropstats | grep "Flow Action Drop "
Flow Action Drop          1590
root@nodefl:~#

```

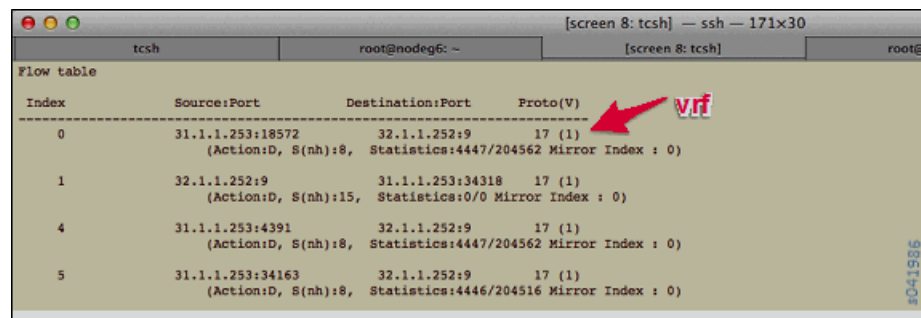
To help in debugging flows, you can use the detailed flow query from the agent introspect page for the compute node.

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 220: Flow Command Output Window



Index	Source:Port	Destination:Port	Proto(V)
0	31.1.1.253:18572 (Action:D, S(nh):8, Statistics:4447/204562 Mirror Index : 0)	32.1.1.252:9	17 (1) vrf
1	32.1.1.252:9 (Action:D, S(nh):15, Statistics:0/0 Mirror Index : 0)	31.1.1.253:34318	17 (1)
4	31.1.1.253:4391 (Action:D, S(nh):8, Statistics:4447/204562 Mirror Index : 0)	32.1.1.252:9	17 (1)
5	31.1.1.253:34163 (Action:D, S(nh):8, Statistics:4446/204516 Mirror Index : 0)	32.1.1.252:9	17 (1)

Fetching details of a single flow:

Figure 221: Fetch Flow Record Window

Output from **FetchFlowRecord** shows unresolved IP addresses:

Figure 222: Unresolved IP Address Window

implicit_deny	no
short_flow	yes
setup_time_utc	1394959054698162
local_flow	no
src_vn	__UNKNOWN__
dst_vn	__UNKNOWN__
reverse_flow	no

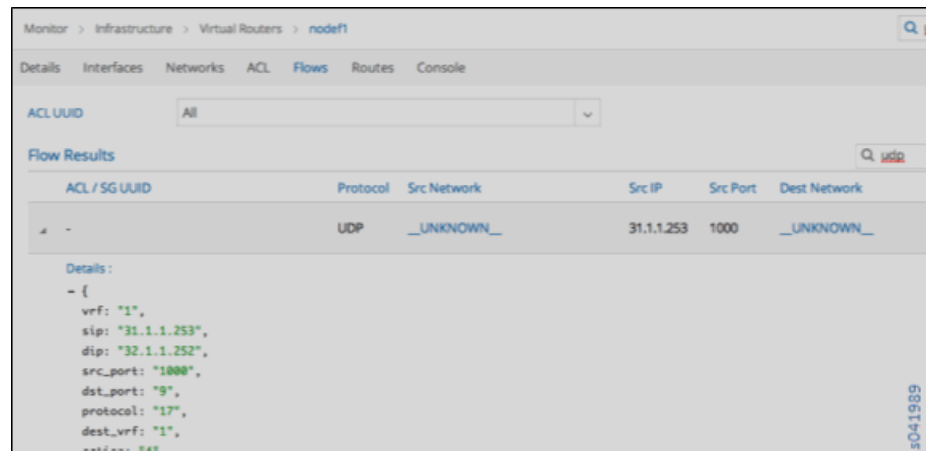
```

    },
    action_str: - {
      list: - {
        ActionStr: - {
          action: "drop"
        }
      }
    }
  }

```

You can also retrieve information about unresolved flows from the Contrail UI, as shown in the following:

Figure 223: Unresolved Flow Details Window

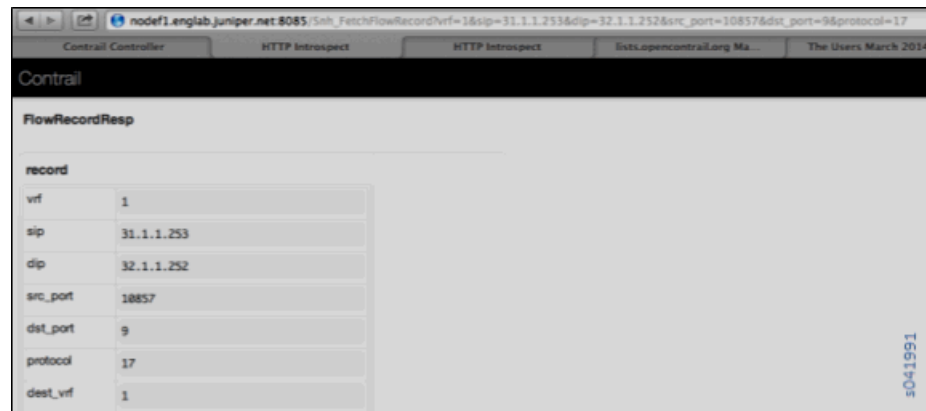


3. Check for protocol-specific network policy action.

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 224: 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 225: 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> <div>uuid</div> <div>8b0329d7-ad9e-41ac-af2e-30f4dbc2b50e</div> </div> </div> <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

Use the troubleshooting steps and guidelines in this topic when you have errors with Contrail BGP peering and route exchange.

- [Example Cluster on page 562](#)
- [Verifying the BGP Routers on page 562](#)
- [Verifying the Route Exchange on page 565](#)
- [Debugging Route Exchange with Policies on page 567](#)
- [Debugging Peering with an MX Series Router on page 569](#)

- [Debugging a BGP Peer Down Error with Incorrect Family on page 571](#)
- [Configuring MX Peering \(iBGP\) on page 573](#)
- [Checking Route Exchange with an MX Series Peer on page 575](#)
- [Checking the Route in the MX Series Router on page 576](#)

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']

WebUI : 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 226: 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"
    }
  ]
}
```

5041945

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 227: 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?

Figure 228: Sample Output, BGP Neighbor Config:

ShowBgpNeighborConfigResp						
neighbors						
instance_name	name	vendor	autonomous_system	identifier	address	address_families
default-domain:default-project:ip-fabric:___default___	default-domain:default-project:ip-fabric:___default___:nodea20	contrail	64512	10.204.216.16	10.204.216.16	address_families
						inet-vpn
						e-vpn

http://<host ip address>:8083/Snh_ShowBgpPeeringConfigReq?

Figure 229: Sample Output, BGP Peering Config:

ShowBgpPeeringConfigResp

peerings		neighbor_count	sessions
instance_name	name		
default-domain:default-project:ip-fabric:default...	site(default-domain:default-project:ip-fabric:default...mode20, default-domain:default-project:ip-fabric:default...mode21)	1	<div> <div>session</div> <div> <div>valid attributes</div> <div> <div>attributes</div> <div> <div>bgp.router.address.families</div> <div> <div>address_families</div> <div> <div>inet-vpn</div> <div>e-vpn</div> </div> </div> </div> </div> </div> </div>

504193

5. Check the BGP neighbor states on the sample setup.

`http://<host ip address>:8083/Snh_BgpNeighborReq?ip_address=&domain=`

Figure 230: Sample Output, BGP Neighbor States:

BgpNeighborListResp

neighbors												
peer	peer_address	peer_asn	local_address	local_asn	encoding	peer_type	state	send_state	last_event	last_state	last_state_at	last_error
10.204.216.16	10.204.216.16	64512	10.204.216.18	64512	BGP	Internal	Established	In sync	Fac::EvBgpKeepalive	OpenConfire	2014-Feb-10 07:08:21.177692	Unknown
10.204.216.253	10.204.216.253	64512	10.204.216.18	64512	BGP	Internal	Established	In sync	Fac::EvBgpUpdate	OpenConfire	2014-Feb-10 11:24:45.851632	Cause:Administrator reset the peer

5041924

If the peer is not in an established state, check the `last_error` and the `flap_count`. Debug the BGP state machine by using information displayed in the output, such as `last_state` and `last_event`.



NOTE: The image displayed is truncated to fit this page. On the console screen you can scroll horizontally to see more columns and data.

Verifying the Route Exchange

The following two virtual networks are used in the sample debugging session for route exchange.

vn1 -> 1.1.1.0/24
vn2 -> 2.2.2.0/24

Example Procedure for Verifying Route Exchange

- 1. Validate the presence of the routing instance for each virtual network in the sample system.

`http://<host ip address>:8083/Snh_ShowRoutingInstanceReq?name=`



NOTE: Throughout this example, replace `<host ip address>` with the correct location for the control node on your system.

Figure 231: Sample Output, Show Routing Instance:

default-domain:demo:vn1:vn1	default-domain:demo:vn1	4	import_target	export_target	names	peers
			target:64512:1	target:64512:1	name	nodes28
					default-domain:demo:vn1:vn1.inet.0	nodes28
					default-domain:demo:vn1:vn1.inet.0	nodes28
					default-domain:demo:vn1:vn1.inetcast.0	nodes28
default-domain:demo:vn2:vn2	default-domain:demo:vn2	5	import_target	export_target	names	peers
			target:64512:2	target:64512:2	name	nodes22
					default-domain:demo:vn2:vn2.inet.0	nodes22
					default-domain:demo:vn2:vn2.inet.0	nodes22
					default-domain:demo:vn2:vn2.inetcast.0	nodes22

In the sample output, you can see the **import_target** and the **export_target** configured on the routing instance. Also shown are the **xmpp peers (vroutes)** registered to the table.

The user can click on the **inet** table of the required routing instance to display the routes that belong to the instance.

Use the information in Step 2 to validate a route.

2. Validate a route in a given routing instance in the sample setup:

http <://>host ip

address>:8083/Snh_ShowRouteReq?x=default-domain:demo:vn1:vn1.inet.0

In the following sample output (truncated), the user can validate the BGP paths for the protocol and for the source of the route to verify which XMPP agent or vRouter has pushed the route. If the path source is BGP, the route is imported to the VRF table from a BGP peer, either another control-node or an external bgp router such as an MX Series router. BGP paths are displayed in the order of path selection.

Figure 232: Sample Output, Validate Route:

ShowRouteResp									
tables									
routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	infeasible_paths	routes		
default-domain:demo:vn1:vn1	default-domain:demo:vn1:vn1.inet.0	1	2	1	1	0	route	test_modified	paths
							1.1.1.253/32	2014-Feb-10 11:34:12.227288	protocol
									XMPP
									BGP

3. Validate the **l3vpn** table.

http <://>host ip address>:8083/Snh_ShowRouteReq?x=bgp.l3vpn.0

ShowRouteMap																	
tables																	
routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	inaccessible_paths	routes	path									
default-domain:default-project:ip-fabric::default::	bgp-1.bspn-8	2	4	2	2	8	<table border="1"> <thead> <tr> <th>prefix</th><th>last_modified</th></tr> </thead> <tbody> <tr> <td>18.204.216.16/1:1.1.1.253/32</td><td>2014-Feb-18 11:34:12.227283</td></tr> <tr> <td>18.204.216.16/1:2.2.2.253/32</td><td>2014-Feb-18 11:34:35.463954</td></tr> </tbody> </table>	prefix	last_modified	18.204.216.16/1:1.1.1.253/32	2014-Feb-18 11:34:12.227283	18.204.216.16/1:2.2.2.253/32	2014-Feb-18 11:34:35.463954	<table border="1"> <thead> <tr> <th>protocol</th></tr> </thead> <tbody> <tr><td>SMTP</td></tr> <tr><td>SCP</td></tr> </tbody> </table>	protocol	SMTP	SCP
prefix	last_modified																
18.204.216.16/1:1.1.1.253/32	2014-Feb-18 11:34:12.227283																
18.204.216.16/1:2.2.2.253/32	2014-Feb-18 11:34:35.463954																
protocol																	
SMTP																	
SCP																	

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

Figure 234: 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:vn1.net.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:vn2.net.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

1. Create a network policy to allow vn1 and vn2 traffic and associate the policy to the virtual networks.

Figure 235: Create Policy Window

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"/>

2. Validate that the routing instances have the correct import_target configuration.

http://<host ip address>:8083/Snh_ShowRoutingInstanceReq?name=

Figure 236: 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 237: Sample Output, Route Import:

ShowRouteResp

titles									
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			
routes									
prefix		last_modified		paths					
1.1.1.253/32		2014-Feb-10 12:02:47.261344		path					
				protocol					
				XMP					
				BGP					

Figure 239: Create BGP Peer Window

Configuring the MX Series BGP peer with the Python provision utility:

```
python ./provision_mx.py --router_name mx --router_ip <ip address>
--router_asn 12345 --api_server_ip <ip address> --api_server_port 8082
--oper add --admin_user admin --admin_password <password>
--admin_tenant_name admin
```

3. Configure a control-node peer on the MX Series router, using Junos CLI:

```
set protocols bgp group contrail-control-nodes type external
set protocols bgp group contrail-control-nodes local-address <ip address>
set protocols bgp group contrail-control-nodes keep all
set protocols bgp group contrail-control-nodes peer-as 54321
set protocols bgp group contrail-control-nodes local-as 12345
set protocols bgp group contrail-control-nodes neighbor <ip address>
```

Debugging a BGP Peer Down Error with Incorrect Family

Use this procedure to identify and resolve errors that arise from *families* mismatched configurations.



NOTE: This example uses locations at `http://<host ip address>:`. Be sure to replace `<host ip address>` with the correct address for your environment.

1. Check the BGP peer UVE.

`http://<host ip address>:8081/analytics/uves/bgp-peers`

2. Search for the MX Series BGP peer by name in the list.

In the sample output, **families** is the family advertised by the peer and **configured_families** is what is provisioned. In the sample output, the families configured on the peer has a mismatch, thus the peer doesn't move to an established state. You can verify it in the peer UVE.

Figure 240: 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 241: 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 242: Edit Global ASN Window

2. Configure the MX Series iBGP peer, using Contrail WebUI or Python provisioning.

Figure 243: 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 244: 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 245: 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.l3vpn.0 table.

Figure 246: 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.l3vpn.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 247: 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.l3vpn.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 248: Routing Instance Public IPv4 Route Table

tables	routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	infeasible_paths	routes
	default-domain:admin:public:public	default-domain:admin:public:public.inet.0	2	2	0	2	0	routes 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

Figure 249: 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	routes	
							prefix	last_modified
							0.0.0.0/0	2014-Mar-14 10:05:05.729525
							paths	protocol
								SCP
							10.204.216.0/24	2014-Mar-14 10:05:05.729817
							paths	protocol
								SCP
							11.2.3.253/32	2014-Mar-14 10:18:48.797958
							paths	protocol
								XXPP

5. Verify the route in the bgp.l3vpn.0 table.

http://<host ip address>:8083/Snh_ShowRouteReq?x=bgp.l3vpn.0

Figure 250: BGP Routing Instance Route Table

tables								
routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	infeasible_paths	routes	
default-domain:default-project:ip-fabric:default...	bgp.l3vpn.0	3	3	2	1	0	<div>routes</div> <div>prefix</div> <div>10.204.216.253:5:0.0.0.0/0</div> <div>10.204.216.253:5:10.204.216.0/24</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

This document provides troubleshooting methods to use when you have errors with the floating IP address pool when using Contrail.

- [Example Cluster on page 578](#)
- [Example on page 579](#)
- [Example: MX80 Configuration for the Gateway on page 580](#)
- [Ping the Floating IP from the Public Network on page 582](#)
- [Troubleshooting Details on page 582](#)
- [Get the UUID of the Virtual Network on page 582](#)
- [View the Floating IP Object in the API Server on page 583](#)

- [View floating-ips in floating-ip-pools in the API Server on page 586](#)
- [Check Floating IP Objects in the Virtual Machine Interface on page 588](#)
- [View Floating IP Objects in the IFMAP Server View on page 591](#)
- [View the BGP Peer Status on the Control Node on page 596](#)
- [Querying Routes in the Public Virtual Network on page 596](#)
- [Verification from the MX80 Gateway on page 597](#)
- [Viewing the Compute Node Vnsw Agent on page 599](#)
- [Advanced Troubleshooting on page 602](#)

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:

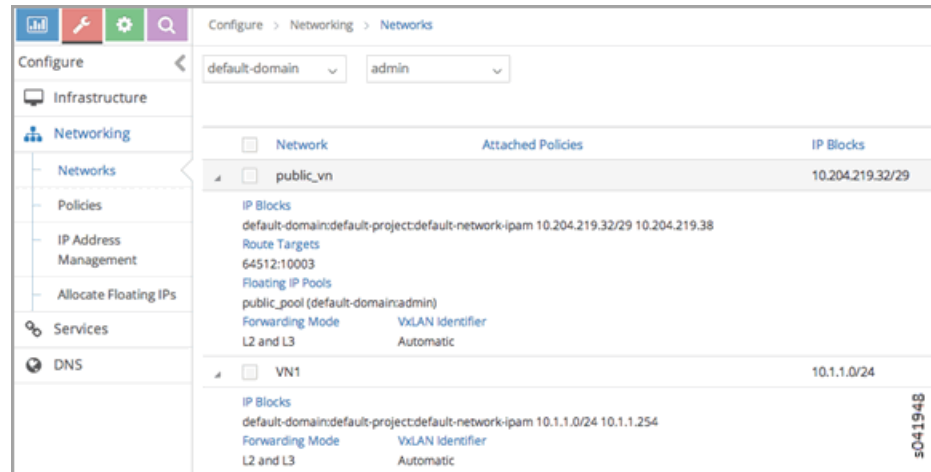
Public virtual network:

- 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**

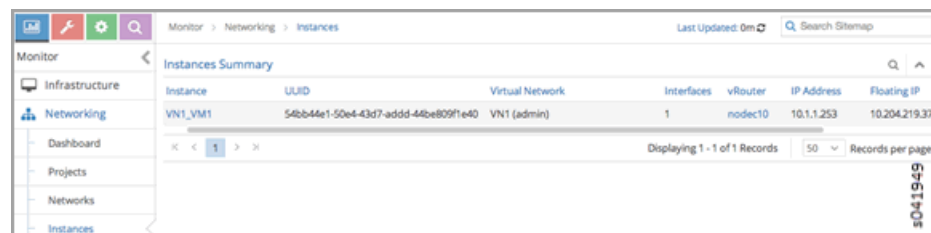
Private virtual network:

- Virtual network name: **vn1**
- Subnet: **10.1.1.0/24**

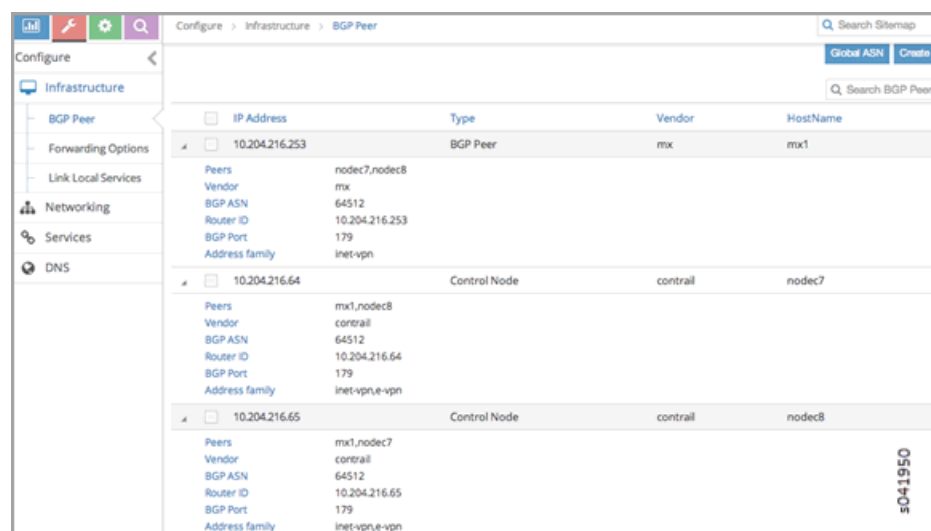
Example



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.



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 ~]#
curlhttp://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.

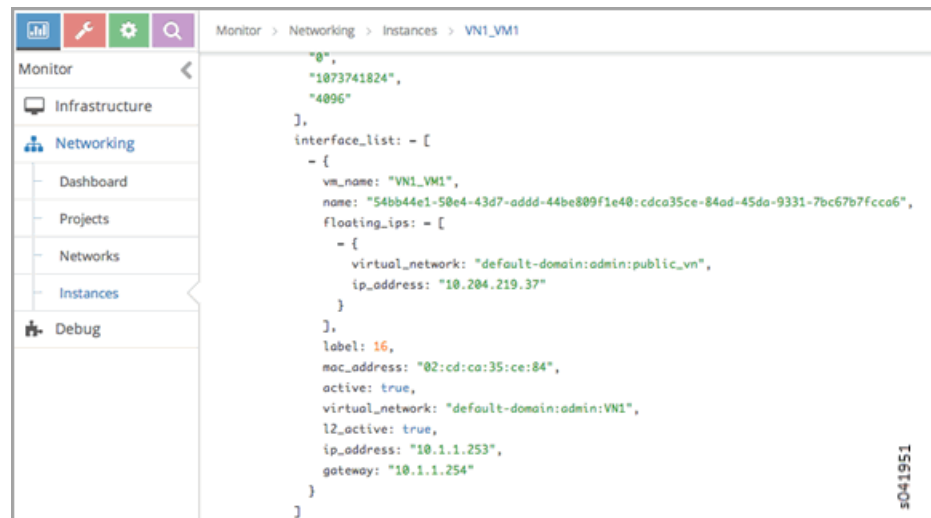
- Using **quantum port-list** to get the virtual machine interface:

Example

```
[root@nodec6 ~]# quantum port-list -F id -F fixed_ips
```

id		fixed_ips	
cdca35ce-84ad-45da-9331-7bc67b7fcc66		{ "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 Floating IP Objects in the IFMAP Server View

Use the following to view the output of `/usr/bin/ifmap_view.py` on the config-nodes. The IFMAP server example output shows the BGP peering configurations and the configurations of the virtual networks `VN1` and `public_vn`.

```
[root@nodec6 ~]# (source /opt/contrail/api-venv/bin/activate ; python
/usr/bin/ifmap_view.py nodec6 8443 test3 test3 -v 2 )
```

. . . .

. . . .

. . . .

```
project = admin

floating-ip = f3eec4d6-889e-46a3-a8f0-879dfaff6ca0

    project = admin
floating-ip-pool = public_pool
security-group = default

    access-control-list = default-access-control-list
virtual-network = VN1

    network-ipam = default-network-ipam
    {
        "ipam_subnets": [
            {
                "subnet": {
                    "ip_prefix": "10.1.1.0",
                    "ip_prefix_len": 24
                },
                "default_gateway": "10.1.1.254"
            }
        ],
        "host_routes": null
    }
routing-instance = VN1

    route-target = 2
    {
        "import_export": null
    }
virtual-network = public_vn

    floating-ip-pool = public_pool
```



```

floating-ip = f3eec4d6-889e-46a3-a8f0-879dfaff6ca0

virtual-machine-interface =
cdca35ce-84ad-45da-9331-7bc67b7fcca6

network-ipam = default-network-ipam
{
  "ipam_subnets": [
    {
      "subnet": {
        "ip_prefix": "10.204.219.32",
        "ip_prefix_len": 29
      },
      "default_gateway": "10.204.219.38"
    }
  ],
  "host_routes": null
}

routing-instance = public_vn
route-target = 10003
{
  "import_export": null
}

route-target = 1
{
  "import_export": null
}

. . . .
. . . .

project = default-project
virtual-network = ip-fabric
routing-instance = __default__
bgp-router = nodec8

```

```
bgp-router = nodec7
{
  "session": [
    {
      "attributes": [
        {
          "bgp_router": null,
          "address_families": {
            "family": [
              "inet-vpn",
              "e-vpn"
            ]
          }
        }
      ],
      "uuid": null
    }
  ]
}

bgp-router = nodec7
bgp-router = mx1
bgp-router = nodec7
{
  "session": [
    {
      "attributes": [
        {
          "bgp_router": null,
          "address_families": {
            "family": [
```

```

        "inet-vpn"
      ]
    }
  }
],
"uuid": null
}
]
}
bgp-router = nodec8
{
  "session": [
    {
      "attributes": [
        {
          "bgp_router": null,
          "address_families": {
            "family": [
              "inet-vpn"
            ]
          }
        }
      ],
      "uuid": null
    }
  ]
}

```

```

. . . .
. . . .
. . . .

```

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

- 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	Security	Origin VN
default-domainadminpublic_vnpublic_vn	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
	10.204.217.32/32	XMPP	nodec10	10.204.216.67	16	1	default-domainadminpublic_vn

- Using the http Introspect:

Following is the format for using an introspect query.

`http://<nodename/ip>:8083/Snh_ShowRouteReq?x=<RoutingInstance of public VN>.inet.0`

Example

`http://nodec8:8083/Snh_BgpNeighborReq?ip_address=&domain=`

nodec7.8083/Snh_ShowRouteReq?v=default-domain:admin:public_vn:public_vn.inet.0

Contrail

Collapse
Expand
Wrap
Fullpage

ShowRouteReq

tables

routing_instance	routing_table_name	prefixes	paths	primary_path	secondary_path	infailable_path	routes																														
default-domain:admin:public_vn:public_vn.inet.0	default-domain:admin:public_vn:public_vn.inet.0	3	4	3	3	0	<div> <div>routes</div> <table> <tr> <th>prefix</th> <th>last_modified</th> <th>paths</th> </tr> <tr> <td>8.8.8.8/8</td> <td>2014-Feb-07 00:10:41.298405</td> <td>paths</td> </tr> <tr> <td></td> <td></td> <td>protocol</td> </tr> <tr> <td></td> <td></td> <td>BGP</td> </tr> <tr> <td>10.204.216.8/24</td> <td>2014-Feb-07 00:10:41.298405</td> <td>paths</td> </tr> <tr> <td></td> <td></td> <td>protocol</td> </tr> <tr> <td></td> <td></td> <td>BGP</td> </tr> <tr> <td>10.204.216.37/32</td> <td>2014-Feb-07 10:10:16.903038</td> <td>paths</td> </tr> <tr> <td></td> <td></td> <td>protocol</td> </tr> <tr> <td></td> <td></td> <td>BGP</td> </tr> </table> </div>	prefix	last_modified	paths	8.8.8.8/8	2014-Feb-07 00:10:41.298405	paths			protocol			BGP	10.204.216.8/24	2014-Feb-07 00:10:41.298405	paths			protocol			BGP	10.204.216.37/32	2014-Feb-07 10:10:16.903038	paths			protocol			BGP
prefix	last_modified	paths																																			
8.8.8.8/8	2014-Feb-07 00:10:41.298405	paths																																			
		protocol																																			
		BGP																																			
10.204.216.8/24	2014-Feb-07 00:10:41.298405	paths																																			
		protocol																																			
		BGP																																			
10.204.216.37/32	2014-Feb-07 10:10:16.903038	paths																																			
		protocol																																			
		BGP																																			

View Corresponding BGP LL3VPN Routes

Use the Contrail UI or the http introspect to view the public route's corresponding BGP LL3VPN routes, as in the following.

Example

- Using the Contrail UI:

Routing Table	Prefix	Protocol	Source	Next Hop	Label	Secur...	Origin/VN
bgp.l3vpn.0	10.204.216.253/5.0.0.0/0	BGP	10.204.216.253	10.204.216.253	16	-	-
	10.204.216.253/5.10.204.216.0/4	BGP	10.204.216.253	10.204.216.253	16	-	-
	10.204.216.67/1.10.1.1.253/32	XMPP	nodec10	10.204.216.67	16	1	default-domain:admin/vn.1
		BGP	10.204.216.64	10.204.216.67	16	1	default-domain:admin/vn.1
	10.204.216.67/2.10.204.219.37/32	XMPP	nodec10	10.204.216.67	16	1	default-domain:admin/vn.1
		BGP	10.204.216.64	10.204.216.67	16	1	default-domain:admin/vn.1

- Using the control-node Introspect:

`http://nodec7:8083/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

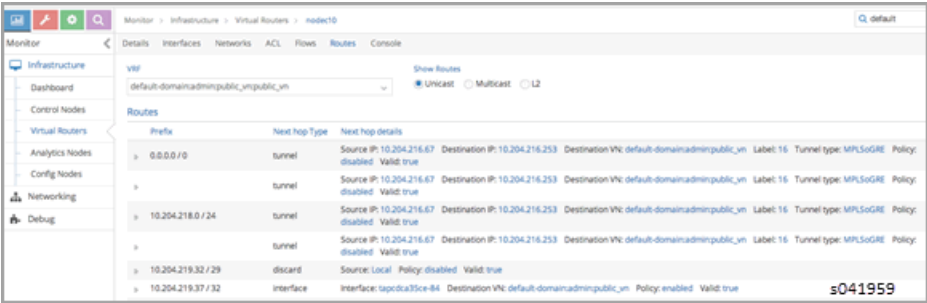
The compute node introspect can be accessed from port 8085. In the following examples, the compute nodes are nodec9 and nodec10.

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

Using the Contrail UI:



Prefix	Next hop Type	Next hop details
0.0.0.0/0	tunnel	Source IP: 10.204.216.67 Destination IP: 10.204.216.253 Destination VN: default-domainadminpublic_vn Label: 16 Tunnel type: MPLSoGRE Policy: disabled Valid: true
10.204.218.0/24	tunnel	Source IP: 10.204.216.67 Destination IP: 10.204.216.253 Destination VN: default-domainadminpublic_vn Label: 16 Tunnel type: MPLSoGRE Policy: disabled Valid: true
10.204.219.32/29	discard	Source: Local Policy: disabled Valid: true
10.204.219.37/32	interface	Interface: tap0ca35ce-84 Destination VN: default-domainadminpublic_vn Policy: enabled Valid: true

Using the Unicast
Route Table Index to
View Next Hops

Alternatively, from the agent introspect, you can view the next hops at the unicast route table.

First, use the following to get the unicast route table index (ucindex) for the routing instance **default-domain:admin:public_vn:public_vn**.

http://nodec10:8085/Snh_VrfListReq?x=default-domain:admin:public_vn:public_vn

Example

In the following example, the unicast route table index is 2.

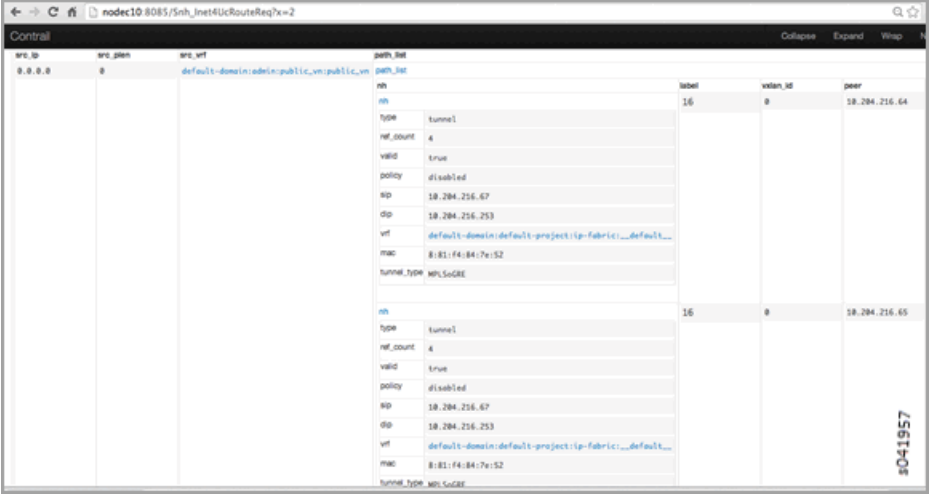


name	ucindex	mcindex	l2index
default-domain:admin:public_vn:public_vn	2	2	2

more false

Next, perform a route request query on ucindex 2, as shown in the following. The tunnel detail indicates the source and destination endpoints of the tunnel and the MPLS label 16 (the label of the virtual machine).

The query should also show a route for 10.204.219.37 with an interface next hop of tap-interface. **http://nodec10:8085/Snh_Inet4UcRouteReq?x=2**



src_ip	src_port	src_vrf	path_list	label	vrf_id	peer
0.0.0.0	0	default-domain:admin:public_vn:public_vn	nh type tunnel ref_count 4 valid true policy disabled ip 10.204.216.67 dip 10.204.216.253 vrf default-domain:default-project:ip-fabric:...default... mac 8:81:F4:B4:7e:52 tunnel_type vni_SoGRE	16	0	10.204.216.64
			nh type tunnel ref_count 4 valid true policy disabled ip 10.204.216.67 dip 10.204.216.253 vrf default-domain:default-project:ip-fabric:...default... mac 8:81:F4:B4:7e:52 tunnel_type vni_SoGRE	16	0	10.204.216.65

10.204.219.37	32	default-domain:admin:public_vn:public_vn	path_list			
			nh	label	vrf_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

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 on page 604](#)
- [Show Virtual Machines on page 605](#)
- [Show Virtual Machines Using Python API on page 607](#)
- [Delete Methods on page 608](#)

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

>>>
```

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

Use the troubleshooting steps and guidelines in this topic when you have errors with Contrail link-local services.

- [Overview of Link-Local Services on page 608](#)
- [Troubleshooting Procedure for Link-Local Services on page 608](#)
- [Metadata Service on page 611](#)
- [Troubleshooting Procedure for Link-Local Metadata Service on page 611](#)

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**.

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

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-0e41067fec53 / 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		s041995

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	s041996

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=

Check that all flow entries display NAT action programmed and display flags for the fields (source or destination IP and ports) that have NAT programmed. Also shown are the number of packets and bytes transmitted in the respective flows.

flow_list									
index	rflow	sip	sport	dip	dport	proto	vrf_id	action	flags
467472	234436	1.2.3.247	123	169.254.169.100	123	17	1	NAT	ACTIVE VRFT SNAT SPAT DNAT SPAT
234436	467472	172.17.28.5	123	10.284.216.72	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?`

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 router 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 router agent will update and proxy the HTTP request. The router 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 router 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 router 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`

PART 5

Contrail Commands and APIs

- [Contrail Commands on page 615](#)
- [Contrail Application Programming Interfaces \(APIs\) on page 635](#)

Contrail Commands

- [contrail-logs \(Accessing Log File Messages\)](#) on page 615
- [contrail-status \(Viewing Node Status\)](#)
- [contrail-version \(Viewing Version Information\)](#)
- [service \(Managing Services\)](#)
- [Backing Up and Restoring Configurations](#) on page 622

contrail-logs (Accessing Log File Messages)

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](#) on page 615
- [Option Descriptions](#) on page 616
- [Example Uses](#) on page 616

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}  
                             Logs from module (default: None)  
--category CATEGORY          Logs of category (default: None)  
--level LEVEL                Logs of level (default: None)  
--message-type MESSAGE_TYPE  Logs of message type (default: None)  
--reverse                    Show logs in reverse chronological order (default:  
False)  
--verbose                    Show internal information (default: True)  
--all                         Show all logs (default: False)  
--object {ObjectVNTTable, ObjectVMTable, ObjectSITable, ObjectVRouter,  
ObjectBgpPeer, ObjectRoutingInstance, ObjectBgpRouter, ObjectXmppConnection,  
ObjectCollectorInfo, ObjectGeneratorInfo, ObjectConfigNode}  
                             Logs of object type (default: None)  
--object-id OBJECT_ID        Logs of object name (default: None)  
--object-select-field {ObjectLog,SystemLog}  
                             Select field to filter the log (default: None)  
--trace TRACE                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)

Syntax [root@host ~]# contrail-status

Release Information Command introduced in Contrail Release 1.0.

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 **analytics**, **configuration**, **web-ui**, and **database**. It is not configured with the roles **control** or **vrouter**.

Sample Output

```
root@host> contrail-status
VRouter is NOT PRESENT
Agent is NOT PRESENT
== Control node ==
supervisor-control:      active
contrail-control         active
supervisor-dns:          active
contrail-dns             active
contrail-named           active
== Analytics ==
supervisor-analytics:    active
contrail-analytics-nodemgr active
contrail-collector       active
contrail-opserver        active
contrail-qe              active
redis-query              active
redis-sentinel           active
redis-uve                active
== Contrail API server ==
supervisor-config:       active
contrail-api             active
contrail-discovery       active
contrail-schema          active
contrail-svc-monitor     active
ifmap                    active
== Contrail quantum ==
== Contrail Web UI ==
supervisor-webui:        active
contrail-webui           active
contrail-webui-middleware active
== Contrail Database ==
supervisord-contrail-database:active
contrail-database        active
```

contrail-version (Viewing Version Information)

Syntax [root@host]# contrail-version

Release Information Command introduced in Contrail Release 1.0.

Description Display a list of all installed components with their version and build numbers.

Required Privilege Level admin

Sample Output

The following example shows version and build information for all installed components.

Sample Output

```
root@host> contrail-version
Package                               Version                               Build-ID | Repo |
RPM Name
-----
contrail-analytics                   1-1309090026.e16                    141
contrail-analytics-venv              0.1-1309062310.e16                  141
contrail-api                         0.1-1309090026.e16                  141
contrail-api-lib                     0.1-1309090026.e16                  141
contrail-api-venv                    0.1-1309080539.e16                  141
contrail-control                     2012.0-1309090026.e16               141
contrail-database                    0.1-1309050028                      141
contrail-dns                         1-1309090026.e16                    141
contrail-fabric-utils                1-1309090026                        141
contrail-libs                        1-1309090026.e16                    141
contrail-nodejs                      0.8.15-1309090026.e16               141
contrail-openstack-analytics         0.1-1309090026.e16                  141
contrail-openstack-cfgm              0.1-1309090026.e16                  141
contrail-openstack-control           0.1-1309090026.e16                  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.e16                    141
contrail-analytics-venv              0.1-1309062310.e16                  141
contrail-api                         0.1-1309090026.e16                  141
```

contrail-api-lib	0.1-1309090026.e16	141
contrail-api-venv	0.1-1309080539.e16	141
contrail-control	2012.0-1309090026.e16	141
contrail-database	0.1-1309050028	141
contrail-dns	1-1309090026.e16	141
contrail-fabric-utils	1-1309090026	141
contrail-libs	1-1309090026.e16	141
contrail-nodejs	0.8.15-1309090026.e16	141
contrail-openstack-analytics	0.1-1309090026.e16	141
contrail-openstack-cfgm	0.1-1309090026.e16	141
contrail-openstack-control	0.1-1309090026.e16	141
contrail-openstack-database	0.1-1309090026.e16	141
contrail-openstack-webui	0.1-1309090026.e16	141
contrail-setup	1-1309090026.e16	141
contrail-webui	1-1309090026	141
openstack-quantum-contrail	2013.2-1309090026	141

service (Managing Services)

Syntax	<code>service contrail-service (start stop restart status)</code>
Release Information	Standard Linux command used for managing and viewing services in Contrail Controller Release 1.0.
Description	<p>Start, stop, or restart a Contrail service. Display the status of a Contrail service.</p> <p>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.</p>
Options	<ul style="list-style-type: none"> • 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
```

Backing Up and Restoring Configurations

- [Back up Procedure on page 622](#)
- [Restore Procedure on page 623](#)
- [Restore Steps Continued on page 632](#)
- [Finishing on page 633](#)

Back up Procedure

Configuration Backup and Restore

1. Take a snapshot of the Cassandra database on all database nodes. Copy it to a different host if you intend to reimage or reset the same servers.

```
user@host:~# nodetool -h localhost -p 7199 snapshot
```

```
Requested creating snapshot for: all keyspaces
```

```
Snapshot directory: 1403160262349
```



.....

NOTE: The snapshot could be in /home/cassandra/ or /var/lib/cassandra. Zip the cassandra directory and store it in a remote host.

.....

2. Get a back up of the MySQL database in the OpenStack node.

```
user@host:~# cat /etc/contrail/mysql.token
```

```
$ABC123
```

```
user@host:~# mysqldump -u root --password=<password> --all-databases >  
openstack.sql
```

3. For testing purposes only: Bring servers to a clean state

```
fab reset_config
```



.....

NOTE: This step is for testing purposes ONLY. This is not necessary if you are bringing up another node.

.....

Restore Procedure

1. Stop Nova services.

```
user@host:~# service nova-api stop
nova-api stop/waiting
user@host:~# service nova-compute stop
nova-compute stop/waiting
user@host:~# service nova-scheduler stop
nova-scheduler stop/waiting
user@host:~# service nova-conductor stop
nova-conductor stop/waiting
```

2. Stop Glance service.

```
user@host:~# service glance-api stop
glance-api stop/waiting
user@host:~# service glance-registry stop
glance-registry stop/waiting
```

3. Stop Keystone service.

```
user@host:~# service keystone stop

keystone stop/waiting
```

4. Stop Config service.

```
fab stop_cfgm
```

5. Stop Collector service.

```
fab stop_collector
```

6. Stop Database service.

```
fab stop_database
```

7. Restore All OpenStack databases.

```
user@host:~# cat /etc/contrail/mysql.token

$ABC123
```

```
user@host:~# mysql -u root --password=<password> < openstack.sql
```

8. Restore the Cassandra database using the script **cass-db-restore-v4.sh**.



NOTE: Please copy the backed up Cassandra database to this server.

```
user@host:~# ./cass-db-restore-v4.sh
```

NAME

Script to restore Cassandra database from snapshot

SYNOPSIS

```
cass-db-restore-v4.sh [--help|-h] [--base_db_dir|-b] [--snapshot_dir|-s]
[--snapshot_name|-n]
```

MUST OPTIONS: base_db_dir, snapshot_dir, snapshot_name

DESCRIPTION

--base_db_dir, -b

Location of running Cassandra database

--snapshot_dir, -s

Snapshot location of Cassandra database

--snapshot_name, -n

Snapshot name

Restore Example

```
cass-db-restore-v4.sh -b /var/lib/cassandra/data -s /root/data.ss -n
1403068337967
```

```
user@host:~# ./cass-db-restore-v4.sh -b /home/cassandra/data -s
/root/data.ss/cassandra/data -n 1403160262349
```

Snapshot available...continuing..

-----dirs to be restored-----

to_bgp_keyspace/route_target_table/

ContrailAnalytics/MessageTableSource/

ContrailAnalytics/MessageTableMessageType/

ContrailAnalytics/StatsTableByU64StrTag/

ContrailAnalytics/MessageTableModuleId/

ContrailAnalytics/ObjectValueTable/

ContrailAnalytics/MessageTable/


```

ContrailAnalytics/StatsTableByStrStrTag/
ContrailAnalytics/MessageTableTimestamp/
ContrailAnalytics/MessageTableCategory/
ContrailAnalytics/SystemObjectTable/
ContrailAnalytics/ObjectTable/
config_db_uuid/obj_fq_name_table/
config_db_uuid/obj_uuid_table/
system/schema_columns/
system/local/
system/schema_columnfamilies/
system/schema_keyspaces/
-----db files in snapshots-----
=====check /home/cassandra/data/to_bgp_keyspace/route_target_table//
=====

to_bgp_keyspace-route_target_table-ic-1-CompressionInfo.db
to_bgp_keyspace-route_target_table-ic-2-CompressionInfo.db

to_bgp_keyspace-route_target_table-ic-1-Data.db
to_bgp_keyspace-route_target_table-ic-2-Data.db

to_bgp_keyspace-route_target_table-ic-1-Filter.db
to_bgp_keyspace-route_target_table-ic-2-Filter.db

to_bgp_keyspace-route_target_table-ic-1-Index.db
to_bgp_keyspace-route_target_table-ic-2-Index.db

to_bgp_keyspace-route_target_table-ic-1-Statistics.db
to_bgp_keyspace-route_target_table-ic-2-Statistics.db

to_bgp_keyspace-route_target_table-ic-1-Summary.db
to_bgp_keyspace-route_target_table-ic-2-Summary.db

to_bgp_keyspace-route_target_table-ic-1-T0C.txt

=====check /home/cassandra/data/ContrailAnalytics/MessageTableSource//
=====

ContrailAnalytics-MessageTableSource-ic-1-CompressionInfo.db
ContrailAnalytics-MessageTableSource-ic-2-CompressionInfo.db

ContrailAnalytics-MessageTableSource-ic-1-Data.db
ContrailAnalytics-MessageTableSource-ic-2-Data.db

ContrailAnalytics-MessageTableSource-ic-1-Filter.db
ContrailAnalytics-MessageTableSource-ic-2-Filter.db

ContrailAnalytics-MessageTableSource-ic-1-Index.db

```

```

ContrailAnalytics-MessageTableSource-ic-2-Index.db

ContrailAnalytics-MessageTableSource-ic-1-Statistics.db
ContrailAnalytics-MessageTableSource-ic-2-Statistics.db

ContrailAnalytics-MessageTableSource-ic-1-Summary.db
ContrailAnalytics-MessageTableSource-ic-2-Summary.db

ContrailAnalytics-MessageTableSource-ic-1-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/MessageTableMessageType//
=====

ContrailAnalytics-MessageTableMessageType-ic-1-CompressionInfo.db
ContrailAnalytics-MessageTableMessageType-ic-2-CompressionInfo.db

ContrailAnalytics-MessageTableMessageType-ic-1-Data.db
ContrailAnalytics-MessageTableMessageType-ic-2-Data.db

ContrailAnalytics-MessageTableMessageType-ic-1-Filter.db
ContrailAnalytics-MessageTableMessageType-ic-2-Filter.db

ContrailAnalytics-MessageTableMessageType-ic-1-Index.db
ContrailAnalytics-MessageTableMessageType-ic-2-Index.db

ContrailAnalytics-MessageTableMessageType-ic-1-Statistics.db
ContrailAnalytics-MessageTableMessageType-ic-2-Statistics.db

ContrailAnalytics-MessageTableMessageType-ic-1-Summary.db
ContrailAnalytics-MessageTableMessageType-ic-2-Summary.db

ContrailAnalytics-MessageTableMessageType-ic-1-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/StatsTableByU64StrTag//
=====

ContrailAnalytics-StatsTableByU64StrTag-ic-1-CompressionInfo.db
ContrailAnalytics-StatsTableByU64StrTag-ic-1-Index.db

ContrailAnalytics-StatsTableByU64StrTag-ic-1-Data.db
ContrailAnalytics-StatsTableByU64StrTag-ic-1-Statistics.db

ContrailAnalytics-StatsTableByU64StrTag-ic-1-Filter.db
ContrailAnalytics-StatsTableByU64StrTag-ic-1-Summary.db

=====check /home/cassandra/data/ContrailAnalytics/MessageTableModuleId//
=====

ContrailAnalytics-MessageTableModuleId-ic-1-CompressionInfo.db
ContrailAnalytics-MessageTableModuleId-ic-2-CompressionInfo.db

ContrailAnalytics-MessageTableModuleId-ic-1-Data.db
ContrailAnalytics-MessageTableModuleId-ic-2-Data.db

ContrailAnalytics-MessageTableModuleId-ic-1-Filter.db
ContrailAnalytics-MessageTableModuleId-ic-2-Filter.db

ContrailAnalytics-MessageTableModuleId-ic-1-Index.db
ContrailAnalytics-MessageTableModuleId-ic-2-Index.db

ContrailAnalytics-MessageTableModuleId-ic-1-Statistics.db

```

```

ContrailAnalytics-MessageTableModuleId-ic-2-Statistics.db

ContrailAnalytics-MessageTableModuleId-ic-1-Summary.db
ContrailAnalytics-MessageTableModuleId-ic-2-Summary.db

ContrailAnalytics-MessageTableModuleId-ic-1-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/ObjectValueTable//
=====

ContrailAnalytics-ObjectValueTable-ic-1-CompressionInfo.db
ContrailAnalytics-ObjectValueTable-ic-2-CompressionInfo.db

ContrailAnalytics-ObjectValueTable-ic-1-Data.db
ContrailAnalytics-ObjectValueTable-ic-2-Data.db

ContrailAnalytics-ObjectValueTable-ic-1-Filter.db
ContrailAnalytics-ObjectValueTable-ic-2-Filter.db

ContrailAnalytics-ObjectValueTable-ic-1-Index.db
ContrailAnalytics-ObjectValueTable-ic-2-Index.db

ContrailAnalytics-ObjectValueTable-ic-1-Statistics.db
ContrailAnalytics-ObjectValueTable-ic-2-Statistics.db

ContrailAnalytics-ObjectValueTable-ic-1-Summary.db
ContrailAnalytics-ObjectValueTable-ic-2-Summary.db

ContrailAnalytics-ObjectValueTable-ic-1-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/MessageTable//
=====

ContrailAnalytics-MessageTable-ic-1-CompressionInfo.db
ContrailAnalytics-MessageTable-ic-2-CompressionInfo.db

ContrailAnalytics-MessageTable-ic-1-Data.db
ContrailAnalytics-MessageTable-ic-2-Data.db

ContrailAnalytics-MessageTable-ic-1-Filter.db
ContrailAnalytics-MessageTable-ic-2-Filter.db

ContrailAnalytics-MessageTable-ic-1-Index.db
ContrailAnalytics-MessageTable-ic-2-Index.db

ContrailAnalytics-MessageTable-ic-1-Statistics.db
ContrailAnalytics-MessageTable-ic-2-Statistics.db

ContrailAnalytics-MessageTable-ic-1-Summary.db
ContrailAnalytics-MessageTable-ic-2-Summary.db

ContrailAnalytics-MessageTable-ic-1-TOC.txt
ContrailAnalytics-MessageTable-ic-2-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/StatsTableByStrStrTag//
=====

ContrailAnalytics-StatsTableByStrStrTag-ic-1-CompressionInfo.db
ContrailAnalytics-StatsTableByStrStrTag-ic-2-CompressionInfo.db

ContrailAnalytics-StatsTableByStrStrTag-ic-1-Data.db

```

```

ConrailAnalytics-StatsTableByStrStrTag-ic-2-Data.db

ConrailAnalytics-StatsTableByStrStrTag-ic-1-Filter.db
ConrailAnalytics-StatsTableByStrStrTag-ic-2-Filter.db

ConrailAnalytics-StatsTableByStrStrTag-ic-1-Index.db
ConrailAnalytics-StatsTableByStrStrTag-ic-2-Index.db

ConrailAnalytics-StatsTableByStrStrTag-ic-1-Statistics.db
ConrailAnalytics-StatsTableByStrStrTag-ic-2-Statistics.db

ConrailAnalytics-StatsTableByStrStrTag-ic-1-Summary.db
ConrailAnalytics-StatsTableByStrStrTag-ic-2-Summary.db

ConrailAnalytics-StatsTableByStrStrTag-ic-1-TOC.txt

=====check /home/cassandra/data/ConrailAnalytics/MessageTableTimestamp//
=====

ConrailAnalytics-MessageTableTimestamp-ic-1-CompressionInfo.db
ConrailAnalytics-MessageTableTimestamp-ic-2-CompressionInfo.db

ConrailAnalytics-MessageTableTimestamp-ic-1-Data.db
ConrailAnalytics-MessageTableTimestamp-ic-2-Data.db

ConrailAnalytics-MessageTableTimestamp-ic-1-Filter.db
ConrailAnalytics-MessageTableTimestamp-ic-2-Filter.db

ConrailAnalytics-MessageTableTimestamp-ic-1-Index.db
ConrailAnalytics-MessageTableTimestamp-ic-2-Index.db

ConrailAnalytics-MessageTableTimestamp-ic-1-Statistics.db
ConrailAnalytics-MessageTableTimestamp-ic-2-Statistics.db

ConrailAnalytics-MessageTableTimestamp-ic-1-Summary.db
ConrailAnalytics-MessageTableTimestamp-ic-2-Summary.db

ConrailAnalytics-MessageTableTimestamp-ic-1-TOC.txt

=====check /home/cassandra/data/ConrailAnalytics/MessageTableCategory//
=====

ConrailAnalytics-MessageTableCategory-ic-1-CompressionInfo.db
ConrailAnalytics-MessageTableCategory-ic-2-CompressionInfo.db

ConrailAnalytics-MessageTableCategory-ic-1-Data.db
ConrailAnalytics-MessageTableCategory-ic-2-Data.db

ConrailAnalytics-MessageTableCategory-ic-1-Filter.db
ConrailAnalytics-MessageTableCategory-ic-2-Filter.db

ConrailAnalytics-MessageTableCategory-ic-1-Index.db
ConrailAnalytics-MessageTableCategory-ic-2-Index.db

ConrailAnalytics-MessageTableCategory-ic-1-Statistics.db
ConrailAnalytics-MessageTableCategory-ic-2-Statistics.db

ConrailAnalytics-MessageTableCategory-ic-1-Summary.db
ConrailAnalytics-MessageTableCategory-ic-2-Summary.db

ConrailAnalytics-MessageTableCategory-ic-1-TOC.txt

```

```

=====check /home/cassandra/data/ContrailAnalytics/SystemObjectTable//
=====

ContrailAnalytics-SystemObjectTable-ic-1-CompressionInfo.db
ContrailAnalytics-SystemObjectTable-ic-1-Statistics.db

ContrailAnalytics-SystemObjectTable-ic-1-Data.db
ContrailAnalytics-SystemObjectTable-ic-1-Summary.db

ContrailAnalytics-SystemObjectTable-ic-1-Filter.db
ContrailAnalytics-SystemObjectTable-ic-1-TOC.txt

ContrailAnalytics-SystemObjectTable-ic-1-Index.db

=====check /home/cassandra/data/ContrailAnalytics/ObjectTable//
=====

ContrailAnalytics-ObjectTable-ic-1-CompressionInfo.db
ContrailAnalytics-ObjectTable-ic-2-CompressionInfo.db

ContrailAnalytics-ObjectTable-ic-1-Data.db
ContrailAnalytics-ObjectTable-ic-2-Data.db

ContrailAnalytics-ObjectTable-ic-1-Filter.db
ContrailAnalytics-ObjectTable-ic-2-Filter.db

ContrailAnalytics-ObjectTable-ic-1-Index.db
ContrailAnalytics-ObjectTable-ic-2-Index.db

ContrailAnalytics-ObjectTable-ic-1-Statistics.db
ContrailAnalytics-ObjectTable-ic-2-Statistics.db

ContrailAnalytics-ObjectTable-ic-1-Summary.db
ContrailAnalytics-ObjectTable-ic-2-Summary.db

ContrailAnalytics-ObjectTable-ic-1-TOC.txt

=====check /home/cassandra/data/config_db_uuid/obj_fq_name_table//
=====

config_db_uuid-obj_fq_name_table-ic-1-CompressionInfo.db
config_db_uuid-obj_fq_name_table-ic-2-CompressionInfo.db

config_db_uuid-obj_fq_name_table-ic-1-Data.db
config_db_uuid-obj_fq_name_table-ic-2-Data.db

config_db_uuid-obj_fq_name_table-ic-1-Filter.db
config_db_uuid-obj_fq_name_table-ic-2-Filter.db

config_db_uuid-obj_fq_name_table-ic-1-Index.db
config_db_uuid-obj_fq_name_table-ic-2-Index.db

config_db_uuid-obj_fq_name_table-ic-1-Statistics.db
config_db_uuid-obj_fq_name_table-ic-2-Statistics.db

config_db_uuid-obj_fq_name_table-ic-1-Summary.db
config_db_uuid-obj_fq_name_table-ic-2-Summary.db

config_db_uuid-obj_fq_name_table-ic-1-TOC.txt

```

```

=====check /home/cassandra/data/config_db_uuid/obj_uuid_table//
=====

config_db_uuid-obj_uuid_table-ic-1-CompressionInfo.db
config_db_uuid-obj_uuid_table-ic-2-CompressionInfo.db

config_db_uuid-obj_uuid_table-ic-1-Data.db
config_db_uuid-obj_uuid_table-ic-2-Data.db

config_db_uuid-obj_uuid_table-ic-1-Filter.db
config_db_uuid-obj_uuid_table-ic-2-Filter.db

config_db_uuid-obj_uuid_table-ic-1-Index.db
config_db_uuid-obj_uuid_table-ic-2-Index.db

config_db_uuid-obj_uuid_table-ic-1-Statistics.db
config_db_uuid-obj_uuid_table-ic-2-Statistics.db

config_db_uuid-obj_uuid_table-ic-1-Summary.db
config_db_uuid-obj_uuid_table-ic-2-Summary.db

config_db_uuid-obj_uuid_table-ic-1-TOC.txt

=====check /home/cassandra/data/system/schema_columns// =====

system-schema_columns-ic-5-CompressionInfo.db
system-schema_columns-ic-5-Summary.db
system-schema_columns-ic-6-Index.db

system-schema_columns-ic-5-Data.db
system-schema_columns-ic-5-TOC.txt
system-schema_columns-ic-6-Statistics.db

system-schema_columns-ic-5-Filter.db
system-schema_columns-ic-6-CompressionInfo.db
system-schema_columns-ic-6-Summary.db

system-schema_columns-ic-5-Index.db
system-schema_columns-ic-6-Data.db

system-schema_columns-ic-5-Statistics.db
system-schema_columns-ic-6-Filter.db

=====check /home/cassandra/data/system/local// =====

system-local-ic-6-CompressionInfo.db    system-local-ic-6-TOC.txt
system-local-ic-7-Summary.db            system-local-ic-8-Statistics.db

system-local-ic-6-Data.db                system-local-ic-7-CompressionInfo.db
system-local-ic-7-TOC.txt                system-local-ic-8-Summary.db

system-local-ic-6-Filter.db              system-local-ic-7-Data.db
system-local-ic-8-CompressionInfo.db     system-local-ic-8-TOC.txt

system-local-ic-6-Index.db                system-local-ic-7-Filter.db
system-local-ic-8-Data.db

system-local-ic-6-Statistics.db           system-local-ic-7-Index.db
system-local-ic-8-Filter.db

system-local-ic-6-Summary.db              system-local-ic-7-Statistics.db

```

```

system-local-ic-8-Index.db

=====check /home/cassandra/data/system/schema_columnfamilies// =====

system-schema_columnfamilies-ic-45-CompressionInfo.db
system-schema_columnfamilies-ic-47-CompressionInfo.db

system-schema_columnfamilies-ic-45-Data.db
system-schema_columnfamilies-ic-47-Data.db

system-schema_columnfamilies-ic-45-Filter.db
system-schema_columnfamilies-ic-47-Filter.db

system-schema_columnfamilies-ic-45-Index.db
system-schema_columnfamilies-ic-47-Index.db

system-schema_columnfamilies-ic-45-Statistics.db
system-schema_columnfamilies-ic-47-Statistics.db

system-schema_columnfamilies-ic-45-Summary.db
system-schema_columnfamilies-ic-47-Summary.db

system-schema_columnfamilies-ic-45-TOC.txt
system-schema_columnfamilies-ic-47-TOC.txt

system-schema_columnfamilies-ic-46-CompressionInfo.db
system-schema_columnfamilies-ic-48-CompressionInfo.db

system-schema_columnfamilies-ic-46-Data.db
system-schema_columnfamilies-ic-48-Data.db

system-schema_columnfamilies-ic-46-Filter.db
system-schema_columnfamilies-ic-48-Filter.db

system-schema_columnfamilies-ic-46-Index.db
system-schema_columnfamilies-ic-48-Index.db

system-schema_columnfamilies-ic-46-Statistics.db
system-schema_columnfamilies-ic-48-Statistics.db

system-schema_columnfamilies-ic-46-Summary.db
system-schema_columnfamilies-ic-48-Summary.db

system-schema_columnfamilies-ic-46-TOC.txt

=====check /home/cassandra/data/system/schema_keyspaces// =====

system-schema_keyspaces-ic-5-CompressionInfo.db
system-schema_keyspaces-ic-6-CompressionInfo.db
system-schema_keyspaces-ic-7-CompressionInfo.db

system-schema_keyspaces-ic-5-Data.db
system-schema_keyspaces-ic-6-Data.db
system-schema_keyspaces-ic-7-Data.db

system-schema_keyspaces-ic-5-Filter.db
system-schema_keyspaces-ic-6-Filter.db
system-schema_keyspaces-ic-7-Filter.db

system-schema_keyspaces-ic-5-Index.db
system-schema_keyspaces-ic-6-Index.db

```

```
system-schema_keyspaces-ic-7-Index.db

system-schema_keyspaces-ic-5-Statistics.db
system-schema_keyspaces-ic-6-Statistics.db
system-schema_keyspaces-ic-7-Statistics.db

system-schema_keyspaces-ic-5-Summary.db
system-schema_keyspaces-ic-6-Summary.db
system-schema_keyspaces-ic-7-Summary.db

system-schema_keyspaces-ic-5-TOC.txt
system-schema_keyspaces-ic-6-TOC.txt
```

Restore Steps Continued

9. Start Nova services.

```
user@host:~# service nova-api start
nova-api start/running, process 25075

user@host:~# service nova-compute start
nova-compute start/running, process 25527

user@host:~# service nova-scheduler start
nova-scheduler start/running, process 25509

user@host:~# service nova-conductor start
nova-conductor start/running, process 25545
```

10. Start Glance service.

```
user@host:~# service glance-api start
glance-api start/running, process 25779

user@host:~# service glance-registry start
glance-registry start/running, process 25793
```

11. Start Keystone service.

```
user@host:~# service keystone start

keystone start/running, process 25806
```

12. Start Config service.

```
fab start_cfgm
```

13. Start Collector service.


```
fab start_collector
```

14. Start Database service.

```
fab start_database
```

Finishing

Purpose Once all the services are started again, you should be able to restore all the VNs and policies in the new node setup.

CHAPTER 25

Contrail Application Programming Interfaces (APIs)

- [Contrail Analytics Application Programming Interfaces \(APIs\) and User-Visible Entities \(UVEs\) on page 635](#)
- [Contrail Node Status on page 645](#)
- [Log and Flow Information APIs on page 654](#)
- [Working with Neutron on page 660](#)
- [Support for Amazon VPC APIs on Contrail OpenStack on page 663](#)

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

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>`

- [User-Visible Entities on page 635](#)
- [Common UVEs in Contrail on page 637](#)
- [Virtual Network UVE on page 637](#)
- [Virtual Machine UVE on page 637](#)
- [vRouter UVE on page 637](#)
- [UVEs for Contrail Nodes on page 638](#)
- [Wild Card Query of UVEs on page 638](#)
- [Filtering UVE Information on page 639](#)

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://10.84.13.45:8081/analytics/uves/xmpp-peers",
  name: "xmpp-peers"
},
{
  href: "http://10.84.13.45:8081/analytics/uves/service-instances",
  name: "service-instances"
},
{
  href: "http://10.84.13.45:8081/analytics/uves/config-nodes",
  name: "config-nodes"
},
{
  href: "http://10.84.13.45:8081/analytics/uves/virtual-machines",
  name: "virtual-machines"
},
{
  href: "http://10.84.13.45:8081/analytics/uves/bgp-routers",
  name: "bgp-routers"
},
{
  href: "http://10.84.13.45:8081/analytics/uves/collectors",
  name: "collectors"
},
{
  href: "http://10.84.13.45:8081/analytics/uves/service-chains",
  name: "service-chains"
},
{
  href: "http://10.84.13.45:8081/analytics/uves/generators",
  name: "generators"
},
{
  href: "http://10.84.13.45:8081/analytics/uves/bgp-peers",
  name: "bgp-peers"
},
{
  href: "http://10.84.13.45:8081/analytics/uves/virtual-networks",
  name: "virtual-networks"
},
{
  href: "http://10.84.13.45:8081/analytics/uves/vrouters",
  name: "vrouters"
}
```

```

},
{
  href: "http://10.84.13.45:8081/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/routers`

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:

sfilt : filter by source (usually the hostname of the generator)

mfilt : filter by module (the module name of the generator)

cfilt : filter by content, useful when only part of a UVE needs to be retrieved

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
127.0.0.1:8081/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: Example output for a virtual machine UVE:
Virtual Machine UVE

```

[user@host ~]# curl
127.0.0.1:8081/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: Example output for a vRouter UVE:

vRouter UVE

```

[user@host ~]# curl 127.0.0.1:8081/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": "10.84.17.1",
          "@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"
      ]
    ]
  }
}

```

```

        "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": [
                        "10.84.17.2",
                        "10.84.17.3"
                    ]
                }
            },
            "a3s18:VRouterAgent"
        ]
    ]
}

```

Contrail Node Status

- [Overview on page 645](#)
- [UVE for NodeStatus on page 645](#)
- [Node Status Features on page 646](#)
- [Using Introspect to Get Process Status on page 651](#)
- [contrail-status script on page 652](#)

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"
                ],
                status: "Up",
                type: "IFMap",
                name: "IFMapServer",

```

```
        description: "Connection with IFMap Server (ironrond)"
      },
    {
      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



NOTE: The example output is the ProcessStatus of only one process of contrail-control. It does not show the full aggregated status of the control node through its UVE (as in the previous example).

```
root@a6s45:~# curl http://10.84.13.45:8083/Snh_SandeshU...q?x=NodeStatus
```

```
<?xml-stylesheet type="text/xsl"
href="/universal_parse.xsl"?><__NodeStatusUVE_list type="slist"><NodeStatusUVE
  type="sandesh"><data type="struct" identifier="1"><NodeStatus><name
type="string" identifier="1" key="ObjectBgpRouter">a6s45</name><process_status
  type="list" identifier="4" aggtype="union"><list type="struct"
size="1"><ProcessStatus><module_id type="string"
identifier="1">ControlNode</module_id><instance_id type="string"
identifier="2">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 - 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: The following is an example output from the `contrail-status` script.

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-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

Log and Flow Information APIs

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 on page 654](#)
- [HTTP POST API on page 654](#)
- [POST Data Format Example on page 655](#)
- [Query Types on page 656](#)
- [Examining Query Status on page 657](#)
- [Examining Query Chunks on page 657](#)
- [Example Queries for Log and Flow Data on page 657](#)

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:

```

SELECT field1, field2 ...
FROM table1
WHERE field1 = value1 AND field2 = value2 ...
FILTER BY ...
SORT BY ...
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:

start_time — the start of the time period

end_time — the end of the time period

table — the table from which to extract data

select_fields — the columns to display in the extracted data

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,
ObjectVNTTable, ObjectVMTable, FlowRecordTable)
    2: i64 start_time; // Microseconds in UTC since Epoch
    3: i64 end_time; // Microseconds in UTC since Epoch
    4: list<string> select_fields; // List of SELECT fields
    5: list<term> where; // WHERE (OR of terms)
    6: optional sort_op sort;
    7: optional list<string> sort_fields;
    8: optional i32 limit;
    9: optional flow_dir_t dir; // direction of flows being queried
    10: optional list<match> filter; // filter the processed result by value
}

struct flow_series_result_entry {
    1: optional i64 T; // Timestamp of the flow record
    2: optional string sourcevn;
    3: optional string sourceip;
    4: optional string destvn;
    5: optional string destip;
    6: optional i32 protocol;
    7: optional i32 sport;
    8: optional i32 dport;
    9: optional flow_dir_t direction_ing;
    10: optional i64 packets; // mutually exclusive to 12,13
    11: optional i64 bytes; // mutually exclusive to 12,13
    12: optional i64 sum_packets; // represented as "sum(packets)" in JSON
    13: optional i64 sum_bytes; // represented as "sum(bytes)" in JSON
};
typedef list<flow_series_result_entry> flow_series_result;
```

Query Types

The **analytics-api** supports two types of queries. Both types use the same POST parameters as described in POST API.

- **sync** — Default query mode. The results are sent inline with the query processing.
- **async** — To execute a query in async mode, attach the following header to the POST request: **Expect: 202-accepted**.

Examining Query Status

For an asynchronous query, the **analytics-api** responds with the code: **202 Accepted**. The response contents are a status entity href URL of the form: **/analytics/query/<QueryID>**. The QueryID is assigned by the **analytics-api**. To view the response contents, poll the status entity by performing a GET action on the URL. The status entity has a variable named **progress**, with a number between 0 and 100, representing the approximate percentage completion of the query. When progress is 100, the query processing is complete.

Examining Query Chunks

The status entity has an element named **chunks** that lists portions (chunks) of query results. Each element of this list has three fields: **start_time**, **end_time**, **href**. The **analytics-api** determines how many chunks to list to represent the query data. A chunk can include an empty string (""), to indicate that the data query is not yet available. If a partial result is available, the chunk href is of the form: **/analytics/query/<QueryID>/chunk-partial/<chunk number>**. When the final result of a chunk is available, the href is of the form: **/analytics/query/<QueryID>/chunk-final/<chunk number>**.

Example Queries for Log and Flow Data

The following example query lists the tables available for query.

```
[root@host ~]# curl 127.0.0.1:8081/analytics/tables | python -mjson.tool
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %    0     0     0         0      0      0     0
100    846    100    846     0     0    509k     0  --:--:--  --:--:--  --:--:--   826k
[
  {
    "href": "http://127.0.0.1:8081/analytics/table/MessageTable",
    "name": "MessageTable"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/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", "Messagetype", "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":
"Messagetype", "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,

```

```
    "MessageType": "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

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 on page 660](#)
- [Network Sharing in Neutron on page 661](#)
- [Commands for Neutron Network Sharing on page 661](#)
- [Support for Neutron APIs on page 662](#)
- [Contrail Neutron Plugin on page 662](#)
- [DHCP Options on page 663](#)
- [Incompatibilities on page 663](#)

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.

Commands for Neutron Network Sharing

The following table summarizes the most common Neutron commands used with Contrail.

Action	Command
List all shared networks.	neutron net-list --shared
Create a network that has the shared attribute.	neutron net-create <net-name> --shared
Set the shared attribute on an existing network.	neutron net-update <net-name> -shared
List all router:external networks.	neutron net-list --router:external

Action	Command
Create a network that has the router:external attribute.	neutron net-create <net-name> -router:external
Set the router:external attribute on an existing network.	neutron net-update <net-name> -router:external

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

- Overview of Amazon Virtual Private Cloud on page 664
- Mapping Amazon VPC Features to OpenStack Contrail Features on page 664
- VPC and Subnets Example on page 665
- Euca2ools CLI for VPC and Subnets on page 666
- Security in VPC: Network ACLs Example on page 666
- Euca2ools CLI for Network ACLs on page 667
- Security in VPC: Security Groups Example on page 667
- Euca2ools CLI for Security Groups on page 668
- Elastic IPs in VPC on page 669
- Euca2ools CLI for Elastic IPs on page 669
- Euca2ools CLI for Route Tables on page 669
- Supported Next Hops on page 670
- Internet Gateway Next Hop Euca2ools CLI on page 670

- [NAT Instance Next Hop Eucalyptus CLI on page 671](#)
- [Example: Creating a NAT Instance with Eucalyptus CLI on page 671](#)

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.

Eucalyptus are command-line tools for interacting with Amazon Web Services (AWS) and other AWS-compatible web services, such as OpenStack. **Eucalyptus** 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
- 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 70: 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 70: 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
AcId
----
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      AcId
      -----
      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
AcId
-----
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	AcId
-----	-----	-----


```
AcId
-----
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	AcId
-----	-----	-----
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**

Exzample: 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
```

