



Contrail™

Contrail Getting Started Guide

Release

4.0



Modified: 2017-06-06

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

Copyright © 2017, Juniper Networks, Inc. All rights reserved.

Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

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

Contrail™ Contrail Getting Started Guide

4.0

Copyright © 2017, 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 <http://www.juniper.net/support/eula.html>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

	About the Documentation	xxiii
	Documentation and Release Notes	xxiii
	Documentation Conventions	xxiii
	Documentation Feedback	xxv
	Requesting Technical Support	xxvi
	Self-Help Online Tools and Resources	xxvi
	Opening a Case with JTAC	xxvi
Part 1	Overview	
Chapter 1	Understanding Contrail	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
	Introduction to Containerized Contrail Modules	11
	Why Use Containers?	11
	Overview of Contrail Containers	12
	Contrail 4.0 Containers	13
	contrail-controller	13
	contrail-analytics	14
	contrail-analyticsdb	14
	contrail-lb	14
	Summary of Container Design, Configuration Management, and Orchestration	14
	Contrail Roles Overview	15
	Downloading Installation Software	15
	Installing the Operating System and Contrail Packages	16
	Installing Containerized Contrail Clusters Using Server Manager	17
	Installing Server Manager	17
	Creating Objects with Server Manager and JSONs	18
	Preparing the Target System for Provisioning	19

Provisioning the System	20
Installing Containerized Contrail Using Server Manager Lite (SM-Lite)	20
Preparing for SM-Lite Installation	21
Prerequisites	21
Installing SM-Lite	22
Provisioning Contrail Using SM-Lite	22
Sample JSONs and Testbed.py	23
Supporting Multiple Interfaces on Servers and Nodes	24
Support for Multiple Interfaces	24
Number of cfgm Nodes Supported	24
Uneven Number of Database Nodes Required	24
Support for VLAN Interfaces	24
Support for Bonding Options	25
Support for Static Route Options	25
Server Interface Examples	25
Interface Naming and Configuration Management	26
Configuring the Control Node	27
Adding a New Node to an Existing Containerized Contrail Cluster	31
Controller Configuration	32
Using contrailctl to add node configuration on existing containers	32
Manually configure contrailctl on all containers and sync the configs	33
Using contrailctl to Configure Services Within Containers	34
What is contrailctl?	34
Command Operations	34
Updating and Syncing Service Configurations Within the Container	35
Supporting Multiple Interfaces on Servers and Nodes	36
Support for Multiple Interfaces	36
Number of cfgm Nodes Supported	36
Uneven Number of Database Nodes Required	37
Support for VLAN Interfaces	37
Support for Bonding Options	37
Support for Static Route Options	37
Server Interface Examples	38
Interface Naming and Configuration Management	38
Contrail Global Controller	39
Resource Identifier Management	39
Multiple Location Resource Provisioning	39
Requirements, Assumptions, and Constraints	40
Platform Support	40
Installation	40

Chapter 4	Using Server Manager to Automate Provisioning	41
	Installing Server Manager	41
	Installation Requirements for Server Manager	41
	Platform Support	41
	Installation Prerequisites	42
	Installing Server Manager	42
	Finishing the Provisioning	43
	Starting the Server Manager Service	43
	Upgrading Server Manager Software	44
	Steps for Upgrading	44
	Server Manager Installation Completion Checks	44
	Server Manager Checks	44
	Server Manager Client Checks	45
	Server Manager WebUI Checks	45
	Sample Configurations for Server Manager Templates	45
	Sample Settings	45
	Sample dhcp.template File	45
	Sample named.conf.options File	46
	Sample named.template File	46
	The sendmail.cf File	46
	Using Server Manager to Automate Provisioning	46
	Overview of Server Manager	47
	Server Manager Requirements and Assumptions	47
	Server Manager Component Interactions	49
	Configuring Server Manager	50
	Configuring the Cobbler DHCP Template	52
	User-Defined Tags for Server Manager	52
	Server Manager Client Configuration File	53
	Restart Services	53
	Accessing Server Manager	54
	Communicating with the Server Manager Client	55
	Server Manager Commands for Configuring Servers	55
	Server Manager Commands Common Options	55
	Add New Servers or Update Existing Servers	56
	Delete Servers	57
	Display Server Configuration	57
	Server Manager Commands for Managing Clusters	58
	Server Manager Commands for Managing Tags	59
	Server Manager Commands for Managing Images	61
	Server Manager Operational Commands for Managing Servers	64
	Reimaging Server(s)	64
	Provisioning and Configuring Roles on Servers	65
	Restarting Server(s)	66
	Show Status of Server(s)	67
	Show Status of Provision	67
	Server Manager REST API Calls	68
	REST APIs for Server Manager Configuration Database Entries	68
	API: Add a Server	68
	API: Delete Servers	69

API: Retrieve Server Configuration	69
API: Add an Image	69
API: Upload an Image	70
API: Get Image Information	70
API: Delete an Image	70
API: Add or Modify a Cluster	71
API: Delete a Cluster	71
API: Get Cluster Configuration	71
API: Get All Server Manager Configurations	71
API: Reimage Servers	72
API: Provision Servers	72
API: Restart Servers	72
Example: Reimaging and Provisioning a Server	73
Using the Server Manager Web User Interface	74
Log In to Server Manager	75
Create a Cluster for Server Manager	75
Edit a Cluster through Edit JSON	84
Working with Servers in the Server Manager User Interface	84
Add a Server	85
Edit Tags for Servers	87
Using the Edit Config Option for Multiple Servers	87
Edit a Server through Server Manager, Edit JSON	88
Filter Servers by Tag	88
Viewing Server Details	89
Configuring Images and Packages	91
Add New Image or Package	91
Selecting Server Manager Actions for Clusters	92
Reimage a Cluster	92
Provision a Cluster	92
Installing and Using Server Manager Lite	92
Server Manager Lite Overview	92
Installing Server Manager Lite	93
Provisioning Using SM-Lite with Contrail 4.0	93
Displaying the Cluster Status	94
Displaying the SM-Lite Installation and Provisioning Log Files	94
Contrail Provisioning Log Files	94
Chapter 5	
Installing and Using Contrail Storage	97
Installing and Using Contrail Storage	97
Overview of the Contrail Storage Solution	97
Basic Storage Functionality with Contrail	98
Ceph Block and Object Storage Functionality	98
Using the Contrail Storage User Interface	98
Hardware Specifications	100
Software Files for Compute Storage Nodes	100
Contrail OpenStack Nova Modifications	100
Installing the Contrail Storage Solution	101
Using Fabric Commands to Install and Configure Storage	101
Fabric Installation Procedure	102

	Using Server Manager to Install and Configure Storage	103
	Server Manager Installation Procedure for Storage	104
	Example: Configurations for Storage for Reimaging and Provisioning a Server	105
	Storage Installation Limits	106
Chapter 6	Upgrading Contrail Software	107
	Upgrading Contrail 3.2 to 4.0	107
	Overview of Changes in Contrail Release 4.0	107
	Assumptions	108
	Using a Server Manager Script to Upgrade from Contrail 3.2 to 4.0	108
	Running the Upgrade Script	108
	Steps for Upgrading Contrail 3.2 to 4.0 (Without using Server-Manager for upgrade)	108
	Dynamic Kernel Module Support (DKMS) for vRouter	110
Part 3	Configuring Contrail	
Chapter 7	Configuring Virtual Networks	115
	Creating Projects in OpenStack for Configuring Tenants in Contrail	116
	Creating a Virtual Network with Juniper Networks Contrail	118
	Creating a Virtual Network with OpenStack Contrail	121
	Creating an Image for a Project in OpenStack Contrail	123
	Creating a Floating IP Address Pool	126
	Using Security Groups with Virtual Machines (Instances)	127
	Security Groups Overview	127
	Creating Security Groups and Adding Rules	127
	Support for IPv6 Networks in Contrail	130
	Overview: IPv6 Networks in Contrail	131
	Creating IPv6 Virtual Networks in Contrail	131
	Address Assignments	132
	Adding IPv6 Peers	133
	Configuring EVPN and VXLAN	133
	Configuring the VXLAN Identifier Mode	135
	Configuring Forwarding	137
	Configuring the VXLAN Identifier	138
	Configuring Encapsulation Methods	139
Chapter 8	Example of Deploying a Multi-Tier Web Application Using Contrail	143
	Example: Deploying a Multi-Tier Web Application	143
	Multi-Tier Web Application Overview	143
	Example: Setting Up Virtual Networks for a Simple Tiered Web Application	144
	Verifying the Multi-Tier Web Application	146
	Sample Addressing Scheme for Simple Tiered Web Application	147
	Sample Physical Topology for Simple Tiered Web Application	148
	Sample Physical Topology Addressing	148
	Sample Network Configuration for Devices for Simple Tiered Web Application	149

Chapter 9	Configuring Services	155
	Configuring DNS Servers	155
	DNS Overview	155
	Defining Multiple Virtual Domain Name Servers	156
	IPAM and Virtual DNS	156
	DNS Record Types	157
	Configuring DNS Using the Interface	158
	Configuring DNS Using Scripts	163
	Support for Multicast	164
	Subnet Broadcast	164
	All-Broadcast/Limited-Broadcast and Link-Local Multicast	165
	Host Broadcast	166
	Using Static Routes with Services	166
	Static Routes for Service Instances	166
	Configuring Static Routes on a Service Instance	167
	Configuring Static Routes on Service Instance Interfaces	168
	Configuring Static Routes as Host Routes	169
	Configuring Metadata Service	170
Chapter 10	Configuring Service Chaining	173
	Service Chaining	173
	Service Chaining Basics	173
	Service Chaining Configuration Elements	174
	Service Chaining MX Series Configuration	177
	ECMP Load Balancing in the Service Chain	178
	Customized Hash Field Selection for ECMP Load Balancing	179
	Overview: Custom Hash Feature	179
	Using ECMP Hash Fields Selection	181
	Configuring ECMP Hash Fields Over Service Chains	181
	Sample Flows	182
	Sample Traffic Flow Path Without Custom ECMP Hash Fields	182
	Sample Traffic Flow Path With Custom ECMP Hash Fields	182
	Using the Contrail Heat Template	183
	Introduction to Heat	183
	Heat Architecture	184
	Support for Heat Version 2 Resources	184
	Deprecation of Heat Version 1 Resources	185
	Heat Version 2 with Service Chaining and Port Tuple Sample Workflow	185
	Example: Creating a Service Template Using Heat	185
	Service Chain Route Reorigination	187
	Overview: Service Chaining in Contrail	187
	Route Aggregation	188
	Schema for Route Aggregation	189
	Configuring and Troubleshooting Route Aggregation	190
	Routing Policy	195
	Applying Routing Policy	196
	Routing Policy Configuration	198
	Configuring and Troubleshooting Routing Policy	199
	Using a VNC Script to Create Routing Policy	200

	Verify Routing Policy in API Server	201
	Verify Routing Policy in the Control Node	202
	Verify Routing Policy Configuration in the Control Node	202
	Verify Routing Policy Configuration on the Routing Instance	202
	Control for Route Reorigination	203
	Configuring and Troubleshooting Reorigination Control	204
	Service Instance Health Check	205
	Health Check Overview	205
	Health Check Object Configuration	206
	Health Check Modes	206
	Creating a Health Check with the Contrail User Interface	207
	Using the Health Check	208
	Health Check Process	208
Chapter 11	Examples: Configuring Service Chaining	209
	Example: Creating an In-Network or In-Network-NAT Service Chain	209
	Creating an In-Network or In-Network-NAT Service Chain	209
	Example: Creating a Transparent Service Chain	217
	Creating a Transparent Mode Service Chain	217
	Example: Creating a Service Chain With the CLI	221
	CLI for Creating a Service Chain	221
	CLI for Creating a Service Template	222
	CLI for Creating a Service Instance	222
	CLI for Creating a Service Policy	222
	Example: Creating a Service Chain with VSRX and In-Network or Routed Mode	223
Part 4	Monitoring and Troubleshooting the Network Using Contrail Analytics	
Chapter 12	Understanding Contrail Analytics	227
	Understanding Contrail Analytics	227
	Contrail Alerts	228
	Alert API Format	228
	Analytics APIs for Alerts	229
	Analytics APIs for SSE Streaming	230
	Built-in Node Alerts	230
	Underlay Overlay Mapping in Contrail	231
	Overview: Underlay Overlay Mapping using Contrail Analytics	232
	Underlay Overlay Analytics Available in Contrail	232
	Architecture and Data Collection	233
	New Processes/Services for Underlay Overlay Mapping	233
	External Interfaces Configuration for Underlay Overlay Mapping	234
	Physical Topology	234
	SNMP Configuration	235
	Link Layer Discovery Protocol (LLDP) Configuration	235
	IPFIX and sFlow Configuration	235
	Sending pRouter Information to the SNMP Collector in Contrail	237
	pRouter UVEs	237
	Contrail User Interface for Underlay Overlay Analytics	239

	Enabling Physical Topology on the Web UI	239
	Viewing Topology to the Virtual Machine Level	240
	Viewing the Traffic of any Link	240
	Trace Flows	241
	Search Flows and Map Flows	242
	Overlay to Underlay Flow Map Schemas	242
	Module Operations for Overlay Underlay Mapping	245
	SNMP Collector Operation	245
	Topology Module Operation	246
	IPFIX and sFlow Collector Operation	247
	Troubleshooting Underlay Overlay Mapping	247
	Script to add pRouter Objects	248
Chapter 13	Configuring Contrail Analytics	251
	Analytics Scalability	251
	High Availability for Analytics	252
	System Log Receiver in Contrail Analytics	253
	Overview	253
	Redirecting System Logs to Contrail Collector	253
	Exporting Logs from Contrail Analytics	253
	Sending Flow Messages to the Contrail System Log	254
	Ceilometer Support in a Contrail Cloud	255
	Overview	255
	Ceilometer Details	255
	Verification of Ceilometer Operation	256
	Contrail Ceilometer Plugin	258
	Ceilometer Installation and Provisioning	260
Chapter 14	Using Contrail Analytics to Monitor and Troubleshoot the Network	261
	Monitoring the System	262
	Debugging Processes Using the Contrail Introspect Feature	264
	Monitor > Infrastructure > Dashboard	268
	Monitor Dashboard	268
	Monitor Individual Details from the Dashboard	269
	Using Bubble Charts	269
	Color-Coding of Bubble Charts	270
	Monitor > Infrastructure > Control Nodes	270
	Monitor Control Nodes Summary	271
	Monitor Individual Control Node Details	271
	Monitor Individual Control Node Console	273
	Monitor Individual Control Node Peers	275
	Monitor Individual Control Node Routes	276
	Monitor > Infrastructure > Virtual Routers	277
	Monitor vRouters Summary	278
	Monitor Individual vRouters Tabs	279
	Monitor Individual vRouter Details Tab	279
	Monitor Individual vRouters Interfaces Tab	280
	Monitor Individual vRouters Networks Tab	281
	Monitor Individual vRouters ACL Tab	282
	Monitor Individual vRouters Flows Tab	283

Monitor Individual vRouters Routes Tab	284
Monitor Individual vRouter Console Tab	285
Monitor > Infrastructure > Analytics Nodes	287
Monitor Analytics Nodes	287
Monitor Analytics Individual Node Details Tab	288
Monitor Analytics Individual Node Generators Tab	289
Monitor Analytics Individual Node QE Queries Tab	290
Monitor Analytics Individual Node Console Tab	291
Monitor > Infrastructure > Config Nodes	292
Monitor Config Nodes	292
Monitor Individual Config Node Details	293
Monitor Individual Config Node Console	294
Monitor > Networking	295
Monitor > Networking Menu Options	295
Monitor -> Networking -> Dashboard	296
Monitor > Networking > Projects	297
Monitor Projects Detail	298
Monitor > Networking > Networks	300
Query > Flows	303
Query > Flows > Flow Series	303
Example: Query Flow Series	306
Query > Flow Records	307
Query > Flows > Query Queue	309
Query > Logs	310
Query > Logs Menu Options	310
Query > Logs > System Logs	311
Sample Query for System Logs	312
Query > Logs > Object Logs	313
Example: Debugging Connectivity Using Monitoring for Troubleshooting	315
Using Monitoring to Debug Connectivity	315

List of Figures

Part 2	Installing and Upgrading Contrail	
Chapter 3	Installing Contrail and Provisioning Roles	11
	Figure 1: Sample Configuration Containerized Contrail	13
	Figure 2: Configure > Infrastructure > BGP Routers	28
	Figure 3: BGP Routers Summary	28
	Figure 4: Create BGP Router	29
	Figure 5: Control Nodes	30
	Figure 6: Control Node Details	31
	Figure 7: Control Node Peers Tab	31
Chapter 4	Using Server Manager to Automate Provisioning	41
	Figure 8: Server Manager Component Interactions	49
	Figure 9: Server Manager > Clusters	75
	Figure 10: Add Cluster	75
	Figure 11: Add Servers to Cluster	76
	Figure 12: Add Servers to Cluster, Next	76
	Figure 13: Assign Roles	77
	Figure 14: Roles Assigned	77
	Figure 15: OpenStack Configuration	78
	Figure 16: Configure Contrail	78
	Figure 17: Configure High Availability	79
	Figure 18: Configure Analytics	79
	Figure 19: Configure Database	80
	Figure 20: Configure VMware	80
	Figure 21: Configure Virtual Gateway	81
	Figure 22: Configure Contrail Storage	81
	Figure 23: View Configured Clusters	81
	Figure 24: Select Cluster Action	82
	Figure 25: Display Cluster Details	82
	Figure 26: View Cluster JSON	83
	Figure 27: Link to View Cluster Details	83
	Figure 28: Display Servers for Cluster	84
	Figure 29: Edit Cluster JSON	84
	Figure 30: View Servers	85
	Figure 31: Add Server, System Management	85
	Figure 32: Add Server, Physical Interfaces	86
	Figure 33: Add Server, Contrail Storage	86
	Figure 34: Select Server Actions	87
	Figure 35: Edit Tags	87
	Figure 36: Edit Config, Multiple Servers	88

	Figure 37: Server Edit JSON	88
	Figure 38: Filter Servers by Tag	89
	Figure 39: View Server Details, System Management	89
	Figure 40: Server Monitoring	90
	Figure 41: Server Inventory	90
	Figure 42: Servers OS Images	91
	Figure 43: Add OS Image	91
	Figure 44: Reimage Cluster	92
	Figure 45: Provision Cluster	92
Part 3	Configuring Contrail	
Chapter 7	Configuring Virtual Networks	115
	Figure 46: OpenStack Projects	116
	Figure 47: Add Project	116
	Figure 48: Add IP Address Management	118
	Figure 49: Configure Networks	119
	Figure 50: Create Network	119
	Figure 51: Networks Window	121
	Figure 52: Create Network Window	121
	Figure 53: Create Network Window Subnet Tab	122
	Figure 54: OpenStack Images Window	123
	Figure 55: OpenStack Create An Image Window	124
	Figure 56: Configure > Networking > Networks	126
	Figure 57: Edit Network	126
	Figure 58: Security Groups	128
	Figure 59: Edit Security Group Rules	128
	Figure 60: Add Rule	129
	Figure 61: Create Security Group	130
	Figure 62: Associate Security Group at Launch Instance	130
	Figure 63: Global Config Window for VXLAN ID	136
	Figure 64: Edit Global Config Window for VXLAN Identifier Mode	136
	Figure 65: Edit Network Window	137
	Figure 66: Edit Network Window for VXLAN Identifier	139
	Figure 67: Edit Global Config Window for Encapsulation Priority Order	140
Chapter 8	Example of Deploying a Multi-Tier Web Application Using Contrail	143
	Figure 68: Simple Tiered Web Use Case	144
	Figure 69: Create Floating IP Pool	145
	Figure 70: Allocate Floating IP	145
	Figure 71: Sample Physical Topology for Simple Tiered Web Application	148
	Figure 72: Sample Physical Topology Addressing	149
Chapter 9	Configuring Services	155
	Figure 73: DNS Servers Examples	156
	Figure 74: IPAM and Virtual DNS	157
	Figure 75: Example Usage for NS Record Type	158
	Figure 76: Configure DNS Records	158
	Figure 77: Add DNS	159
	Figure 78: Add DNS Record	160

	Figure 79: Associate IPAMs to DNS	161
	Figure 80: Configure IP Address Management	162
	Figure 81: DNS Server	162
Chapter 10	Configuring Service Chaining	173
	Figure 82: Service Chaining	173
	Figure 83: Contrail Service Chain	174
	Figure 84: Load Balancing a Service Chain	178
	Figure 85: Route Reorigination	187
	Figure 86: Route Aggregate Relationships	189
	Figure 87: Create Route Aggregate	191
	Figure 88: Create Service Instance	192
	Figure 89: Create Health Check Screen	207
Chapter 11	Examples: Configuring Service Chaining	209
	Figure 90: Create Networks	209
	Figure 91: Add Service Template	210
	Figure 92: Add Service Template Shared IP	212
	Figure 93: Service Templates	212
	Figure 94: Create Service Instances	213
	Figure 95: Create Service Instances	214
	Figure 96: Service Instance Details	214
	Figure 97: Service Instance Console	215
	Figure 98: Create Policy	215
	Figure 99: Edit Network	216
	Figure 100: Launch Instances	216
	Figure 101: Create Networks	217
	Figure 102: Add Service Template	218
	Figure 103: Create Service Instances	219
	Figure 104: Service Instance Details	220
	Figure 105: Create Policy	220
	Figure 106: Launch Instances	221
Part 4	Monitoring and Troubleshooting the Network Using Contrail Analytics	
Chapter 12	Understanding Contrail Analytics	227
	Figure 107: Analytics Topology	235
	Figure 108: Add Physical Router Window	237
	Figure 109: Sample Output From a pRouter REST API	238
	Figure 110: Sample Output From a pRouter UVE	239
	Figure 111: Physical Topology Related to a vRouter	240
	Figure 112: Traffic Statistics Graph	241
	Figure 113: List of Active Flows	241
	Figure 114: Underlay Path	242
Chapter 13	Configuring Contrail Analytics	251
	Figure 115: Analytics Scalability	252
Chapter 14	Using Contrail Analytics to Monitor and Troubleshoot the Network	261
	Figure 116: Monitor Menu	262

Figure 117: Control Nodes Details Tab Window	265
Figure 118: Controller Introspect Window	266
Figure 119: BGP Peer Introspect Page	266
Figure 120: BGP Neighbor Summary Introspect Page	267
Figure 121: Agent Introspect Page	268
Figure 122: Monitor > Infrastructure > Dashboard	268
Figure 123: Dashboard Summary Boxes	269
Figure 124: Bubble Summary Information	270
Figure 125: Control Nodes Summary	271
Figure 126: Individual Control Node—Details Tab	272
Figure 127: Individual Control Node—Console Tab	273
Figure 128: Individual Control Node—Peers Tab	275
Figure 129: Individual Control Node—Routes Tab	276
Figure 130: vRouters Summary	278
Figure 131: Individual vRouters—Details Tab	279
Figure 132: Individual vRouters—Interfaces Tab	281
Figure 133: Individual vRouters—Networks Tab	282
Figure 134: Individual vRouters—ACL Tab	283
Figure 135: Individual vRouters—Flows Tab	284
Figure 136: Individual vRouters—Routes Tab	285
Figure 137: Individual vRouter—Console Tab	286
Figure 138: Analytics Nodes Summary	288
Figure 139: Monitor Analytics Individual Node Details Tab	289
Figure 140: Individual Analytics Node—Generators Tab	290
Figure 141: Individual Analytics Node—QE QueriesTab	290
Figure 142: Analytics Individual Node—Console Tab	291
Figure 143: Config Nodes Summary	293
Figure 144: Individual Config Nodes— Details Tab	293
Figure 145: Individual Config Node—Console Tab	294
Figure 146: Monitor Networking Menu Options	296
Figure 147: Traffic Statistics for Domain Window	296
Figure 148: Monitor > Networking > Projects	297
Figure 149: Monitor Projects Connectivity Details	298
Figure 150: Traffic Statistics Between Networks	298
Figure 151: Projects Instances Summary	299
Figure 152: Instance Traffic Statistics	300
Figure 153: Network Summary	300
Figure 154: Individual Network Connectivity Details—Summary Tab	301
Figure 155: Individual Network— Port Map Tab	301
Figure 156: Individual Network— Port Distribution Tab	302
Figure 157: Individual Network Instances Tab	302
Figure 158: Individual Network Details Tab	303
Figure 159: Query Flow Series Window	304
Figure 160: Flow Series Select	305
Figure 161: Flow Series Filter	306
Figure 162: Example: Query Flow Series	306
Figure 163: Query Flow Series Tabular Results	307
Figure 164: Query Flow Series Graphical Results	307
Figure 165: Flow Records	308

Figure 166: Flow Records Select Window	309
Figure 167: Where Clause Window	309
Figure 168: Flows Query Queue	310
Figure 169: Query > Logs	311
Figure 170: Query > Logs > System Logs	311
Figure 171: Edit Where Clause	313
Figure 172: Sample Query System Logs	313
Figure 173: Query > Logs > Object Logs	314
Figure 174: Navigate to Instance	315
Figure 175: Traffic Statistics for Instance	315
Figure 176: Navigate to Instance	316
Figure 177: Traffic Statistics for Instance	316
Figure 178: Navigate to a3s18 Interfaces	316
Figure 179: Navigate to a3s19 Interfaces	316
Figure 180: ACL Connectivity a3s18	317
Figure 181: ACL Connectivity a3s19	317
Figure 182: Routes default-domain:demo:vn0:vn0	317
Figure 183: Routes default-domain:demo:vn16:vn16	318
Figure 184: Verify Route and Next Hop a3s18	318
Figure 185: Verify Route and Next Hop a3s19	319
Figure 186: Flows for a3s18	319
Figure 187: Flows for a3s19	319

List of Tables

	About the Documentation	xxiii
	Table 1: Notice Icons	xxiv
	Table 2: Text and Syntax Conventions	xxiv
Part 2	Installing and Upgrading Contrail	
Chapter 2	Supported Platforms and Server Requirements	9
	Table 3: Supported Platforms	9
Chapter 3	Installing Contrail and Provisioning Roles	11
	Table 4: Create BGP Router Fields	29
Chapter 4	Using Server Manager to Automate Provisioning	41
	Table 5: Server Manager Parameters	50
	Table 6: Ansible Server Parameters	51
	Table 7: Common Command Options	56
	Table 8: Server Manager Add Server Command Options	56
	Table 9: Server Manager Delete Server Command Options	57
	Table 10: Server Manager Display Server Command Options	57
	Table 11: Server Manager Add Cluster Command Options	58
	Table 12: Server Manager Delete Cluster Command Options	59
	Table 13: Server Manager Display Cluster Command Options	59
	Table 14: Server Manager Add New Tag	60
	Table 15: Server Manager Image Types	61
	Table 16: Server Manager Add Image	62
	Table 17: Server Manager Upload Image	63
	Table 18: Server Manager Delete Image	63
	Table 19: Server Manager Display Image Configuration	64
	Table 20: Server Manager Reimage	65
	Table 21: Server Manager Provision	65
	Table 22: Server Manager Restart	66
	Table 23: Server Manager Status Server	67
	Table 24: Server Manager Status Provision	67
Part 3	Configuring Contrail	
Chapter 7	Configuring Virtual Networks	115
	Table 25: Add IP Address Management Fields	118
	Table 26: Create Network Fields	119
	Table 27: Create Network Fields	122
	Table 28: Create an Image Fields	124
	Table 29: Add Rule Fields	129

Chapter 8	Example of Deploying a Multi-Tier Web Application Using Contrail	143
	Table 30: Sample Addressing Scheme for Example	147
Chapter 9	Configuring Services	155
	Table 31: DNS Record Types Supported	157
	Table 32: Add DNS Fields	159
	Table 33: Add DNS Record Fields	160
	Table 34: Associate IPAMs to DNS Fields	161
	Table 35: DNS Modes	162
	Table 36: DNS Scripts	163
Chapter 10	Configuring Service Chaining	173
	Table 37: Health Check Configurable Parameters	206
	Table 38: Create Health Check Fields	207
Chapter 11	Examples: Configuring Service Chaining	209
	Table 39: Add Service Template Fields	210
	Table 40: Create Service Instances Fields	213
	Table 41: Add Service Template Fields	218
	Table 42: Create Service Instances Fields	219
Part 4	Monitoring and Troubleshooting the Network Using Contrail Analytics	
Chapter 14	Using Contrail Analytics to Monitor and Troubleshoot the Network	261
	Table 43: Monitor Menu Options	262
	Table 44: Dashboard Summary Boxes	269
	Table 45: Control Nodes Summary Fields	271
	Table 46: Individual Control Node—Details Tab Fields	272
	Table 47: Control Node: Console Tab Fields	273
	Table 48: Control Node: Peers Tab Fields	275
	Table 49: Control Node: Routes Tab Fields	276
	Table 50: vRouters Summary Fields	278
	Table 51: vRouters Details Tab Fields	279
	Table 52: vRouters: Interfaces Tab Fields	281
	Table 53: vRouters: Networks Tab Fields	282
	Table 54: vRouters: ACL Tab Fields	283
	Table 55: vRouters: Flows Tab Fields	284
	Table 56: vRouters: Routes Tab Fields	285
	Table 57: Control Node: Console Tab Fields	286
	Table 58: Fields on Analytics Nodes Summary	288
	Table 59: Monitor Analytics Individual Node Details Tab Fields	289
	Table 60: Monitor Analytics Individual Node Generators Tab Fields	290
	Table 61: Analytics Node QE Queries Tab Fields	290
	Table 62: Monitor Analytics Individual Node Console Tab Fields	291
	Table 63: Config Nodes Summary Fields	293
	Table 64: Individual Config Nodes— Details Tab Fields	294
	Table 65: Individual Config Node—Console Tab Fields	294
	Table 66: Projects Summary Fields	297
	Table 67: Projects Summary Fields	297

Table 68: Projects Instances Summary Fields	299
Table 69: Network Summary Fields	300
Table 70: Query Flow Series Fields	304
Table 71: Query Flow Records Fields	308
Table 72: Query Flow Records Fields	310
Table 73: Query System Logs Fields	311
Table 74: Object Logs Query Fields	314

About the Documentation

- Documentation and Release Notes on page xxiii
- Documentation Conventions on page xxiii
- Documentation Feedback on page xxv
- Requesting Technical Support on page xxvi

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 <http://www.juniper.net/techpubs/>.

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 <http://www.juniper.net/books>.

Documentation Conventions

Table 1 on page xxiv 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 xxiv 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.	<pre>[edit] routing-options { static { route default { nexthop <i>address</i>; retain; } } }</pre>
;(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 <http://www.juniper.net/techpubs/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 <http://www.juniper.net/techpubs/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 <http://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf>.
- Product warranties—For product warranty information, visit <http://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: <http://www.juniper.net/customers/support/>
- Search for known bugs: <http://www2.juniper.net/kb/>
- Find product documentation: <http://www.juniper.net/techpubs/>
- Find solutions and answer questions using our Knowledge Base: <http://kb.juniper.net/>
- Download the latest versions of software and review release notes: <http://www.juniper.net/customers/csc/software/>
- Search technical bulletins for relevant hardware and software notifications: <http://kb.juniper.net/InfoCenter/>
- Join and participate in the Juniper Networks Community Forum: <http://www.juniper.net/company/communities/>
- Open a case online in the CSC Case Management tool: <http://www.juniper.net/cm/>

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

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 <http://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 <http://www.juniper.net/support/requesting-support.html>.

PART 1

Overview

- [Understanding Contrail on page 3](#)

CHAPTER 1

Understanding Contrail

- [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](#)
 - [Contrail Roles Overview on page 15](#)

PART 2

Installing and Upgrading Contrail

- [Supported Platforms and Server Requirements on page 9](#)
- [Installing Contrail and Provisioning Roles on page 11](#)
- [Using Server Manager to Automate Provisioning on page 41](#)
- [Installing and Using Contrail Storage on page 97](#)
- [Upgrading Contrail Software on page 107](#)

CHAPTER 2

Supported Platforms and Server Requirements

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

Supported Platforms

[Table 3 on page 9](#) lists the operating system versions and the corresponding Linux or Ubuntu kernel versions supported by Contrail Release 4.0.

Table 3: Supported Platforms

Contrail Release	OpenStack Release	Operating System and Kernel Versions
Contrail Release 4.0	OpenStack Newton	<ul style="list-style-type: none">• Ubuntu 16.04.2—Linux kernel version 4.4-62• Red Hat 7.3—Linux kernel version 3.10.0-514.6.2• VMware vCenter 5.5, 6.0—Ubuntu 14.04.4 kernel version 3.13.0-106-generic
	OpenStack Mitaka	<ul style="list-style-type: none">• Ubuntu 14.04.5—Linux kernel versions 3.13.0-106-generic and 4.4.0-34-generic• VMware vCenter 5.5, 6.0—Ubuntu 16.04.2 kernel version 4.4.0-62-generic



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

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

Server Requirements

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

Each server must have a minimum of:

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



NOTE: If you are using Contrail Storage, additional hardware requirements can be found in [“Installing and Using Contrail Storage” on page 97](#), Hardware Specifications.

Related Documentation

- [Contrail Roles Overview on page 15](#)
- [Downloading Installation Software on page 15](#)

CHAPTER 3

Installing Contrail and Provisioning Roles

- [Introduction to Containerized Contrail Modules on page 11](#)
- [Contrail Roles Overview on page 15](#)
- [Downloading Installation Software on page 15](#)
- [Installing the Operating System and Contrail Packages on page 16](#)
- [Installing Containerized Contrail Clusters Using Server Manager on page 17](#)
- [Installing Containerized Contrail Using Server Manager Lite \(SM-Lite\) on page 20](#)
- [Supporting Multiple Interfaces on Servers and Nodes on page 24](#)
- [Configuring the Control Node on page 27](#)
- [Adding a New Node to an Existing Containerized Contrail Cluster on page 31](#)
- [Using contrailctl to Configure Services Within Containers on page 34](#)
- [Supporting Multiple Interfaces on Servers and Nodes on page 36](#)
- [Contrail Global Controller on page 39](#)

Introduction to Containerized Contrail Modules

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

- [Why Use Containers? on page 11](#)
- [Overview of Contrail Containers on page 12](#)
- [Contrail 4.0 Containers on page 13](#)
- [Summary of Container Design, Configuration Management, and Orchestration on page 14](#)

Why Use Containers?

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

These dependencies lead to complexities of deployment, including:

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

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

Overview of Contrail Containers

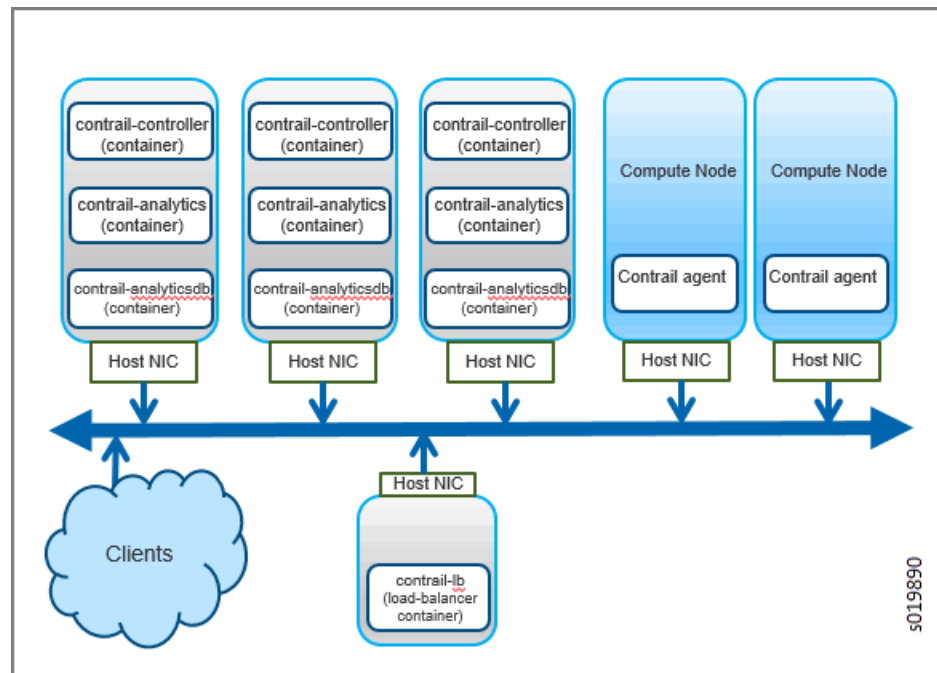
Starting with Contrail 4.0, some of the Contrail subsystems are delivered as Docker containers that group together related functional components. Each container file includes an INI-based configuration file for configuring the services within the container. The purpose of the INI is to provide enough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

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

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

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

Figure 1: Sample Configuration Containerized Contrail



Contrail 4.0 Containers

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

- [contrail-controller](#) on page 13
- [contrail-analytics](#) on page 14
- [contrail-analyticsdb](#) on page 14
- [contrail-lb](#) on page 14

contrail-controller

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

- All configuration services, such as **contrail api**, **config-nodemgr**, **device-manager**, **schema**, **svc-monitor**, and **CONFIGDB**.
- All control services, such as **contrail-control**, **control-nodemgr**, **contrail-dns**, and **contrail-named**.
- All Web UI services, such as **contrail-webui** and **contrail-webui-middleware**.
- Configuration database (Cassandra)
- Zookeeper
- RabbitMQ
- Redis for Web Ui

contrail-analytics

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

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

contrail-analyticsdb

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

contrail-lb

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

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

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

Summary of Container Design, Configuration Management, and Orchestration

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

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

- Related Documentation**
- [Using contrailctl to Configure Services Within Containers on page 34](#)
 - [Installing Containerized Contrail Clusters Using Server Manager on page 17](#)

Contrail Roles 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:

- *contrail-controller*—Runs the Contrail container service
- *openstack*—Runs OpenStack services such as Nova, Quantum, and the like
- *contrail-analytics*—Runs the Contrail analytics services
- *contrail-analyticsdb*—Runs the Contrail analytics database services
- *contrail-compute*—Runs the Contrail compute 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 16](#) 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**
- [Introduction to Containerized Contrail Modules on page 11](#)
 - [Installing the Operating System and Contrail Packages on page 16](#)
 - [Installing Containerized Contrail Clusters Using Server Manager on page 17](#)
 - [Installing Containerized Contrail Using Server Manager Lite \(SM-Lite\) on page 20](#)
 - [Upgrading Contrail 3.2 to 4.0 on page 107](#)

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*
- [Installing Containerized Contrail Clusters Using Server Manager on page 17](#)
- [Installing Containerized Contrail Using Server Manager Lite \(SM-Lite\) on page 20](#)
- [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. Create an image.json with the Ubuntu or CentOS image to be used to reimage the target server.

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

3. Use Server Manager to add the image.json, to be used for reimaging.

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

Related Documentation

- [Introduction to Containerized Contrail Modules on page 11](#)
- [Contrail Roles Overview on page 15](#)
- [Installing Containerized Contrail Clusters Using Server Manager on page 17](#)
- [Installing Containerized Contrail Using Server Manager Lite \(SM-Lite\) on page 20](#)
- [Upgrading Contrail 3.2 to 4.0 on page 107](#)
- [Download Software](#)

Installing Containerized Contrail Clusters Using Server Manager

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

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

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

The installation is completed using the following major activities:

- [Installing Server Manager on page 17](#)
- [Creating Objects with Server Manager and JSONs on page 18](#)
- [Preparing the Target System for Provisioning on page 19](#)
- [Provisioning the System on page 20](#)

Installing Server Manager

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

1. Install the Server Manager wrapper package:

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

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

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



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

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

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



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

4. Start the Server Manager process:
`service contrail-server-manager start`

For more details about the Server Manager installation process, refer to [“Installing Server Manager” on page 41](#).

Creating Objects with Server Manager and JSONs

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

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

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

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

The following procedure helps you create a target system that includes the components for OpenStack, Contrail controller, analytics, analytics database, and agent. The controller, analytics, and analytics database services are provisioned using Contrail containers, however, the agent service is configured on the bare-metal target host.

1. Configure the images needed for reimagining and provisioning.

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

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

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

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



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

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

2. Configure the cluster(s).

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

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

For a multi-node system:

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

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



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

3. Configure the server.

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

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

Preparing the Target System for Provisioning

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

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

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

- For a multi-node system:

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

Provisioning the System

Launch the system provisioning.

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

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

- For a multi-node system:

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

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

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

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

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

Related Documentation

- Sample JSONs for an All-In-One-Node Cluster:
[Sample JSONs for an all-in-one, single node with roles](#)
- Sample JSONs for a Multinode Cluster with Two Nodes:
[Sample JSONs for a Multinode Cluster](#)
- Sample JSONs for a Multinode Cluster with 7 Nodes and High Availability
[Sample JSONs for a Multinode Cluster with High Availability](#)
- [Introduction to Containerized Contrail Modules on page 11](#)
- [Contrail Roles Overview on page 15](#)
- [Installing the Operating System and Contrail Packages on page 16](#)
- [Installing Containerized Contrail Using Server Manager Lite \(SM-Lite\) on page 20](#)
- [Upgrading Contrail 3.2 to 4.0 on page 107](#)

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

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

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

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

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

- [Preparing for SM-Lite Installation on page 21](#)
- [Installing SM-Lite on page 22](#)
- [Provisioning Contrail Using SM-Lite on page 22](#)
- [Sample JSONs and Testbed.py on page 23](#)

Preparing for SM-Lite Installation

For Contrail 4.0, SM-Lite install is only supported on Ubuntu 14.04.5.

Before installing Contrail Release 4.0, you must install Server Manager SM-Lite on a server running Ubuntu 14.04.5.

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

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

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

Prerequisites

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

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

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

- Verify that the VM is set up correctly with hostname and domain details:
 - The hostname and domain name are present in **/etc/hosts** as follows:
<Host non mgmt IP> <server hostname>.<domain_name> <server hostname>
 - The domain name is present in **/etc/resolv.conf** as follows:

`search <domain_name>`

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

Installing SM-Lite

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

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

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

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

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

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

For more details about SM-Lite, refer to *Installing and Using Server Manager Lite*.

Provisioning Contrail Using SM-Lite

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

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

- Using JSONs

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

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

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

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

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

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

Sample JSONs and Testbed.py

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

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

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

- [Sample testbed.py for Provisioning Containers with SM-Lite](#)
- [Sample JSONs for an All-In-One-Node Cluster \(for demo\)](#)
- [Sample JSONs for a Multinode Cluster with Two Nodes](#)
- [Sample JSONs for a Multinode Cluster with 7 Nodes and High Availability](#)

Related Documentation

- [Sample JSONs for an All-In-One-Node Cluster \(for demo\)](#)
- [Sample JSONs for a Multinode Cluster with Two Nodes](#)
- [Sample JSONs for a Multinode Cluster with 7 Nodes and High Availability](#)
- [Sample testbed.py for Provisioning Containers with SM-Lite](#)
- [Introduction to Containerized Contrail Modules on page 11](#)
- [Contrail Roles Overview on page 15](#)
- [Installing the Operating System and Contrail Packages on page 16](#)
- [Installing Containerized Contrail Clusters Using Server Manager on page 17](#)
- [Upgrading Contrail 3.2 to 4.0 on page 107](#)

Supporting Multiple Interfaces on Servers and Nodes

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

- [Support for Multiple Interfaces on page 24](#)
- [Server Interface Examples on page 25](#)
- [Interface Naming and Configuration Management on page 26](#)

Support for Multiple Interfaces

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

Examples of control traffic include the following:

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

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

Number of cfm Nodes Supported

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

Uneven Number of Database Nodes Required

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

Support for VLAN Interfaces

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

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

Support for Bonding Options

Contrail provides support for bond interface options.

The default bond interface options are:

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

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

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

Support for Static Route Options

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

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

```
{
  "network": {
    "routes": [
      {
        "gateway": "3.3.2.254",
        "interface": "enp129s0f0",
        "netmask": "255.255.255.0",
        "network": "3.3.4.0"
      },
      {
        "gateway": "3.3.3.254",
        "interface": "enp129s0f1",
        "netmask": "255.255.255.0",
        "network": "3.3.5.0"
      }
    ]
  }
}
```

Server Interface Examples

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

- **mgmt=control=data** -- Single interface use case

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

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

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

Interface Naming and Configuration Management

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

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

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

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

Related Documentation

- *Juniper OpenStack High Availability*

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.

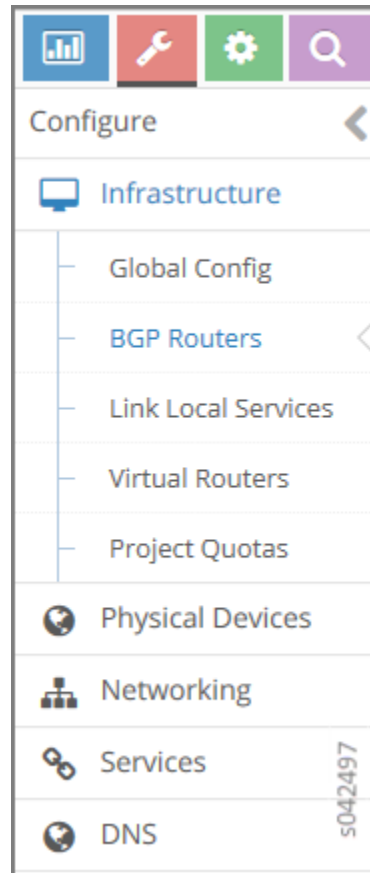
Before you begin, ensure that the following tasks are completed:

- 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 2 on page 28](#).

Figure 2: Configure > Infrastructure > BGP Routers



A summary screen of the control nodes and BGP routers is displayed; see [Figure 3 on page 28](#).

Figure 3: BGP Routers Summary

Configure > Infrastructure > BGP Routers				
BGP Routers				
<input type="checkbox"/> IP Address	Type	Vendor	HostName	
<input type="checkbox"/> 10.84.25.31	Control Node	contrail	b5s31	
<input type="checkbox"/> 10.84.11.252	BGP Router	mx	a3-mx80-1	
<input type="checkbox"/> 10.84.25.30	Control Node	contrail	b5s30	
<input type="checkbox"/> 10.84.25.29	Control Node	contrail	b5s29	
<input type="checkbox"/> 10.84.25.28	Control Node	contrail	b5s28	
<input type="checkbox"/> 10.84.25.27	Control Node	contrail	b5s27	
<input type="checkbox"/> 10.84.11.253	BGP Router	mx	mx1	
Total: 7 records 50 Records ▼				
Page 1 of 1				

2. (Optional) The global AS number is 64512 by default. To change the AS number, on the **BGP Router** summary screen click the gear wheel and select **Edit**. In the Edit BGP

Router window enter the new number.

3. To create control nodes and BGP routers, on the **BGP Routers** summary screen, click the **+** icon. The **Create BGP Router** window is displayed; see [Figure 4 on page 29](#).

Figure 4: Create BGP Router

4. In the **Create BGP Router** window, click **BGP Router** to add a new BGP router or click **Control Node** to add control nodes.

For each node you want to add, populate the fields with values for your system. See [Table 4 on page 29](#).

Table 4: 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 4: 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.

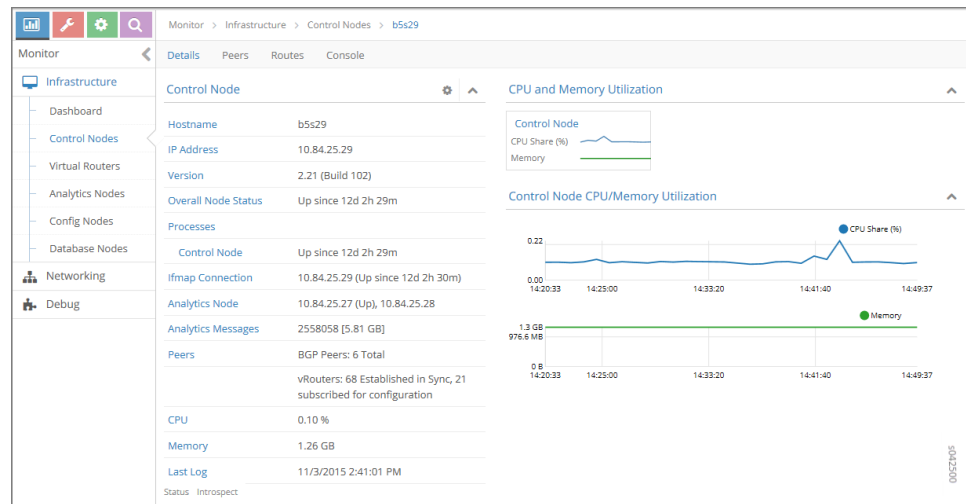
- Click **Save** to add each node that you create.
- To configure an existing node as a peer, select it from the list in the **Available Peers** box, then click >> to move it into the **Configured Peers** box.
Click << to remove a node from the **Configured Peers** box.
- You can check for peers by selecting **Monitor > Infrastructure > Control Nodes**; see [Figure 5 on page 30](#).

Figure 5: Control Nodes



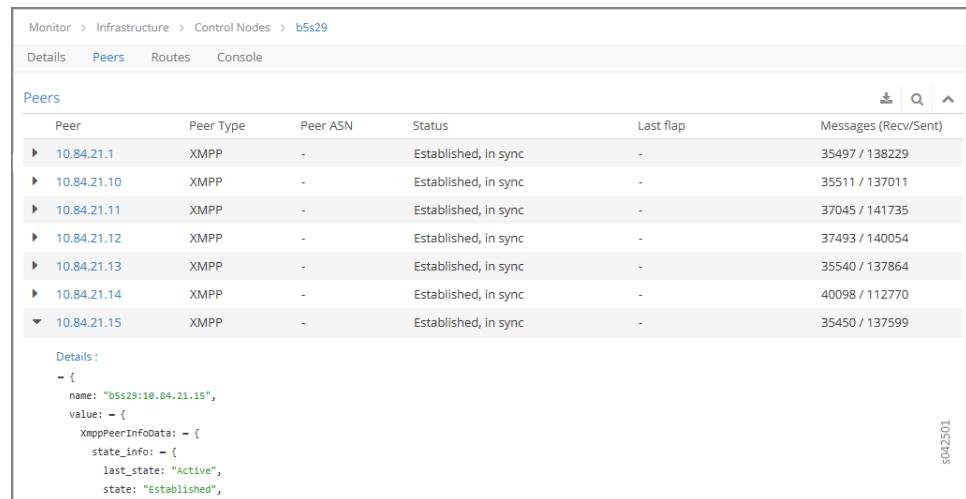
In the **Control Nodes** window, click any hostname in the memory map to view its details; see [Figure 6 on page 31](#).

Figure 6: Control Node Details



8. Click the **Peers** tab to view the peers of a control node; see [Figure 7 on page 31](#).

Figure 7: Control Node Peers Tab



Related Documentation

- [Creating a Virtual Network with Juniper Networks Contrail on page 118](#)
- [Creating a Virtual Network with OpenStack Contrail on page 121](#)

Adding a New Node to an Existing Containerized Contrail Cluster

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

- [Controller Configuration on page 32](#)

Controller Configuration

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

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

See examples on github at:

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

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

2. Configure the existing cluster nodes with new nodes.

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

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

- [Using contrailctl to add node configuration on existing containers on page 32](#)
- [Manually configure contrailctl on all containers and sync the configs on page 33](#)

Using contrailctl to add node configuration on existing containers

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



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

1. Determine which node is the leader node.

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

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

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

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

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

```

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

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

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

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

```

Manually configure contrailctl on all containers and sync the configs

1. Determine which node is the leader node.

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

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

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

2. Manually configure `/etc/contrailctl/controller.conf` with new nodes for various `*_list` configurations and `config_seed_list`. See examples at:

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

3. Run `contrailctl` within the containers.

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

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

Removing Nodes in an Existing Containerized Cluster

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

Using contrailctl to Configure Services Within Containers

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

- [What is contrailctl? on page 34](#)
- [Command Operations on page 34](#)

What is contrailctl?

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

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

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

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

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

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

Sample container configuration files can be seen at

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

Command Operations

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

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

Example:

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

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

Command Syntax and Options

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

optional arguments:
  -h, --help            show this help message and exit
  -f CONFIG_FILE, --config-file CONFIG_FILE
                        Master config file path
  -c {controller,analyticsdb,analytics,agent,lb,kubemanager,mesosmanager},
  --component {controller,analyticsdb,analytics,agent,lb,kubemanager,mesosmanager}
                        Component[s] to be configured
  -F, --force            Whether to apply config forcibly
  -t TAGS, --tags TAGS  comma separated list of tags to runspecific set of
                        ansible code
```

Updating and Syncing Service Configurations Within the Container

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

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

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

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

Syntax and Usage: config sync

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

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

Example: config sync

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

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

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

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

Related Documentation

- [Introduction to Containerized Contrail Modules on page 11](#)

Supporting Multiple Interfaces on Servers and Nodes

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

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

Support for Multiple Interfaces

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

Examples of control traffic include the following:

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

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

Number of cfm Nodes Supported

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

Uneven Number of Database Nodes Required

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

Support for VLAN Interfaces

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

```

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

```

Support for Bonding Options

Contrail provides support for bond interface options.

The default bond interface options are:

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

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

```

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

```

Support for Static Route Options

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

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

```

"network": {
  "routes": [
    {
      "gateway": "3.3.2.254",

```

```
        "interface": "enp129s0f0",
        "netmask": "255.255.255.0",
        "network": "3.3.4.0"
    },
    {
        "gateway": "3.3.3.254",
        "interface": "enp129s0f1",
        "netmask": "255.255.255.0",
        "network": "3.3.5.0"
    }
  ]
}
```

Server Interface Examples

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

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

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

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

Interface Naming and Configuration Management

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

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

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

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

Related Documentation

- *Juniper OpenStack High Availability*

Contrail Global Controller

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

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

- [Resource Identifier Management on page 39](#)
- [Multiple Location Resource Provisioning on page 39](#)

Resource Identifier Management

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

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

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

Multiple Location Resource Provisioning

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

Requirements, Assumptions, and Constraints

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

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

Platform Support

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

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

Installation

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

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

Related Documentation

- *Contrail Global Controller Web User Interface*

CHAPTER 4

Using Server Manager to Automate Provisioning

- [Installing Server Manager on page 41](#)
- [Using Server Manager to Automate Provisioning on page 46](#)
- [Using the Server Manager Web User Interface on page 74](#)
- [Installing and Using Server Manager Lite on page 92](#)

Installing Server Manager

- [Installation Requirements for Server Manager on page 41](#)
- [Installing Server Manager on page 42](#)
- [Upgrading Server Manager Software on page 44](#)
- [Server Manager Installation Completion Checks on page 44](#)
- [Sample Configurations for Server Manager Templates on page 45](#)

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 (deprecated)

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

- Kilo

- Liberty
- Mitaka

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 on 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.
- The system administrator might need to configure the Linux kernel security module AppArmor to allow **server-manager** access.

Installing Server Manager

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

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

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

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

1. Install the Server Manager packages:

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



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

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

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

- To set up all components:

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

- To set up only the Server Manager server:

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

- To set up only the Server Manager client:

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

- To set up only the Server Manager user interface:

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

- To set up only Server Manager monitoring:

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

Other options include:

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

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

Finishing the 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/settings
/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.

Steps for Upgrading

Use the following steps to upgrade your Server Manager installation.



NOTE: You do not need to manually delete your previous Server Manager installation before upgrading.

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

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

Server Manager Installation Completion Checks

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

Server Manager Checks

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

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

`service contrail-server-manager status`

`service cobblerd status`

`cobbler sync`

`service bind9 status`

`service isc-dhcp-server status`

`service apache2 status`

- Also verify processes using the following command:

`ps auwx | grep Passenger`

Server Manager Client Checks

- Verify the items listed:
which server-manager
- Check the client configuration at
/opt/contrail/server_manager/client/sm-client-config.ini
- Make sure that **listen_ip_addr** is configured with the correct Server Manager IP address.

Server Manager WebUI Checks

- Verify the status of the Server Manager WebUI:
service supervisor-webui-sm status
- Check the webui access from the browser:
 - Contrail release 2.2 and lower—**http:<server manager ip> :8080**
 - Contrail release 3.0 and greater—**http:<server manager ip> :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 parameters for DHCP, bind, and e-mail services.

Sample Settings

```
bind_master: 10.XX.11.6  
  
manage_forward_zones: ['contrail.juniper.net']  
  
manage_reverse_zones: ['10.XX.11']  
  
next_server: 10.XX.11.6  
  
server: 10.XX.11.6
```

Sample dhcp.template File

Add Server Manager hooks into the dhcp.template file, so that when DHCP actions occur, such as commit, release, or expire, the Server Manager is notified. The DHCP servers are detected on the Server Manager and the *Discovered* status is maintained.

Use the following sample to help define the subnet blocks that the DHCP server needs to support:

<https://github.com/Juniper/contrail-server-manager/blob/master/src/cobbler/dhcp.template>



NOTE: Your DHCP template must have a separate block for each subnet for which Server Manager will be the DHCP server.

Sample `named.conf.options` File

Use the following sample to help configure the `/etc/bind/named.conf.options`:

<https://github.com/Juniper/contrail-server-manager/blob/master/src/cobbler/named.conf.options>

You can also configure the following parameter:

```
forwarders {  
    0.0.0.0;  
};
```

Sample `named.template` File

Use the following sample to help configure the `/etc/cobbler/named.template`:

<https://github.com/Juniper/contrail-server-manager/blob/master/src/cobbler/named.template>

The `sendmail.cf` File

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

Related Documentation

- [Using Server Manager to Automate Provisioning on page 46](#)
- [Using the Server Manager Web User Interface on page 74](#)
- [Installing and Using Server Manager Lite on page 92](#)

Using Server Manager to Automate Provisioning

- [Overview of Server Manager on page 47](#)
- [Server Manager Requirements and Assumptions on page 47](#)
- [Server Manager Component Interactions on page 49](#)
- [Configuring Server Manager on page 50](#)
- [Configuring the Cobbler DHCP Template on page 52](#)
- [User-Defined Tags for Server Manager on page 52](#)
- [Server Manager Client Configuration File on page 53](#)
- [Restart Services on page 53](#)
- [Accessing Server Manager on page 54](#)
- [Communicating with the Server Manager Client on page 55](#)
- [Server Manager Commands for Configuring Servers on page 55](#)

- [Server Manager REST API Calls on page 68](#)
- [Example: Reimaging and Provisioning a Server on page 73](#)

Overview of Server Manager

The Contrail Server Manager is used to provision, configure, and reimage a Contrail virtual network system of servers, clusters, and nodes.

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

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

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

The Server Manager:

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

Server Manager Requirements and Assumptions

The following are requirements and assumptions for the Server Manager:

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

- Starting with Contrail Release 4.0, Server Manager uses Ansible software, an open source configuration management tool primarily used to automate the configuration and provisioning of Contrail components inside containers.
- The Server Manager also uses Docker to load and move these Contrail containers to the target servers. The Server Manager maintains a local registry on the Server Manager machine and users also have an option to use an external registry from which they can copy their Contrail Docker images directly onto the target servers.
- SQLite3 database management software is used to maintain and manage server configurations and it runs on the same machine where the Server Manager daemon is running.
- Because the server-manager process listens on port 9001, and the server-manager webui listens on ports 9080 and 9143, the firewall must be enabled for those ports.
- Server Manager needs a minimum of 4GB of RAM, 2 CPU cores, and 80GB of disks (to support multiple Contrail installations).
- Server Manager assumes that SSH is enabled on target nodes.

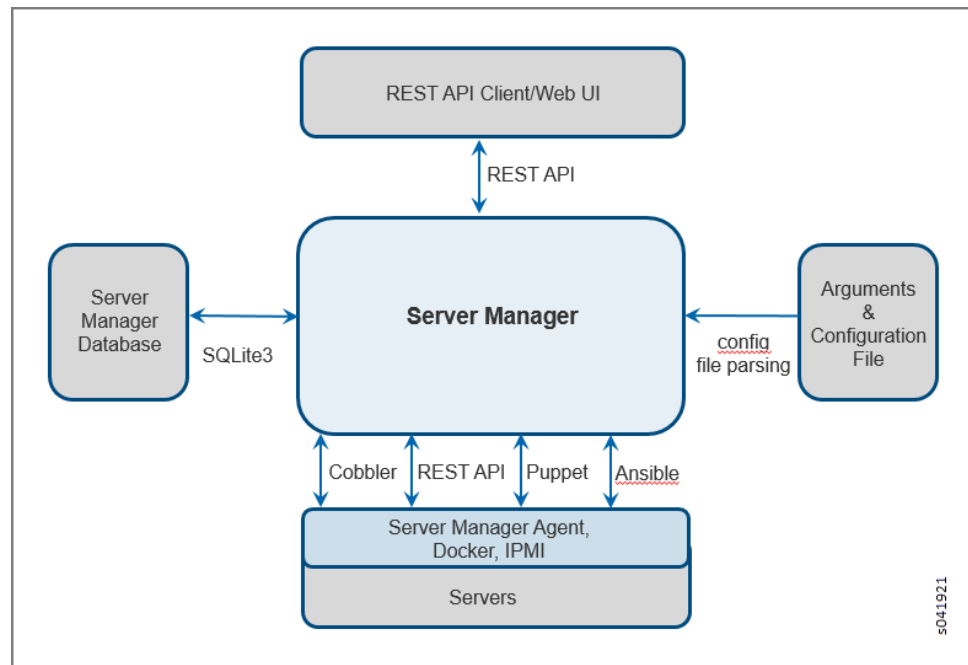
Server Manager Component Interactions

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

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

Figure 8 on page 49 illustrates several high-level components with which the Server Manager interacts.

Figure 8: 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, Docker to host Contrail containers, and Ansible and Puppet for provisioning, thereby ensuring necessary software packages are installed and configured, required services are running, and so on.

A Server Manager agent runs on each of the servers and communicates with the Server Manager, providing the information needed to monitor the operation of the servers. The Server Manager agent also uses REST APIs to communicate with the Server Manager, and it can use other software tools to fetch other information, such as Intelligent Platform Interface (IPMI). Monitoring functionality is enabled by default with Server Manager installation but can be skipped if the user wishes.

Configuring Server Manager

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

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

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

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

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

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

```
[SERVER-MANAGER]

listen_ip_addr = <SM-IP-address>

listen_port    = <port-number>

cobbler_ip_address = <cobbler-ip-address>
cobbler_port    = <cobbler-port-number>
cobbler_username = <cobbler-username>
cobbler_password = <cobbler-password>

ipmi_username = <IPMI username>
ipmi_password = <IPMI password>
ipmi_type     = <IPMI type>
```

[Table 5 on page 50](#) provides details for each of the parameters in the **SERVER-MANAGER** section.

Table 5: Server Manager Parameters

Parameter	Configuration
listen_ip_addr	Specify the IP address of the server on which the Server Manager is listening for REST API requests.
listen_port	The port number on which the Server Manager is listening for REST API requests. The default is 9001.

Table 5: Server Manager Parameters (*continued*)

Parameter	Configuration
<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>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, lanplus, or other Cobbler-supported types).

Starting with Contrail Release 4.0, there is an **ANSIBLE-SERVER** section for parameters for running the Server Manager Ansible daemon, which is used to set up a Docker registry. This registry is used by Ansible to provision Contrail Release 4.0 containers onto targets. These values can be modified to reflect any remote or non-Server Manager Docker registry that the user wants to use to host the Contrail Release 4.0 Docker containers. The following example shows the format and parameters of the **ANSIBLE-SERVER** section:

```
[ANSIBLE-SERVER]

docker_insecure_registries = <IP address:Port>

docker_registry = <IP address:Port>

ansible_srvr_ip = <IP address>

ansible_srvr_port = <Port>

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

[Table 6 on page 51](#) provides details for each of the parameters in the **ANSIBLE-SERVER** section.

Table 6: Ansible Server Parameters

Parameter	Configuration
<code>docker_insecure_registries</code>	Specify the IP address and port of the server on which the insecure Docker registry used by the Server Manager resides
<code>docker_registry</code>	Specify the IP address and port of the server on which the Docker registry used by the Server Manager resides

Table 6: Ansible Server Parameters (*continued*)

Parameter	Configuration
<code>ansible_svr_ip</code>	Specify the IP address of the Server Manager machine on which the Ansible daemon will run
<code>ansible_svr_port</code>	Specify the port on the Server Manager machine on which the Ansible daemon will run
<code>ansible_log_path</code>	Specify the log path where the Ansible daemon stores its log messages

Configuring the Cobbler DHCP Template

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

The following is a link to a sample `dhcp.template` file, which you can modify to match the subnets in your setup.



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

Sample dhcp.template

<https://github.com/Juniper/contrail-server-manager/blob/master/src/cobbler/dhcp.template.u.sample>

User-Defined Tags for Server Manager

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

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

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

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

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


Server Manager Client Configuration File

The Server Manager client application installation copies the `/etc/contrail/sm-client-config.ini` sample configuration file. The sample file contains parameter values such as the IP address to reach the Server Manager and the port used by Server Manager. You must modify the values in the `sm-client-config.ini` file to match your environment.

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

Starting with Contrail Release 3.0.2, if you don't explicitly specify a password, a password is automatically generated by the system. This makes the clusters provisioned by Server Manager more secure. There are no default passwords. The system administrator can specify the passwords to configure, or you can use the passwords that are automatically generated by Server Manager.

The following fields get an autogenerated password whenever an explicit password is not provided.

- Ceilometer Mongoddb password
- Ceilometer keystone auth password
- Cinder keystone auth password
- Glance keystone auth password
- Heat encryption key
- Heat keystone auth password
- Keystone admin password
- Keystone admin token
- MYSQL root password
- MYSQL service password
- Neutron keystone auth password
- Nova keystone auth password
- Swift keystone auth password

Restart Services

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

service contrail-server-manager restart

Accessing Server Manager

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

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

1. Add a boot image (ISO) to server-manager, along with the kickstart and preseed files compatible with your datacenter server. Each Server Manager release has a default kickstart file. If your system administrator doesn't provide the kickstart files, Server Manager default files will be used.
2. Add the Contrail image you are using to Server Manager.
3. Add the cluster(s) to Server Manager. You can add common provisioning parameters for servers to the cluster, and the parameters get passed to the server when provisioning starts.

4. Add the servers that will be managed by Server Manager. Remember to add the **cluster_id** to link with the cluster.

The following are the minimum parameters needed for reimaging or provisioning:

- ID
 - cluster
 - domain
 - interface details
 - roles assigned to each server
 - password
5. Specify the name and location of boot images, packages, and repositories used to bring up the servers with needed software of the supported versions.
 6. Provision or configure the servers by installing necessary packages, creating configuration files, and bringing up the correct services so that each server can perform the functions or role(s) configured for that server.

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

7. Set up API calls for monitoring servers.

Once the servers in the Contrail system are correctly reimaged and provisioned to run configured roles, the server monitoring REST API calls allow clients to monitor performance of the servers as they provide one or more role functions.

Communicating with the Server Manager Client

Server Manager provides a REST API interface for clients to talk to the Server Manager software. Any client that can send and receive REST API requests and responses can be used to communicate with Server Manager, for example, Curl or Postman. Additionally, the Server Manager software provides a client with a simplified CLI interface, in a separate package. The Server Manager client can be installed and run on the Server Manager machine itself or on another server with an IP connection to the Server Manager machine.

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

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

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

Server Manager Commands for Configuring Servers

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

- [Server Manager Commands Common Options on page 55](#)
- [Add New Servers or Update Existing Servers on page 56](#)
- [Delete Servers on page 57](#)
- [Display Server Configuration on page 57](#)
- [Server Manager Commands for Managing Clusters on page 58](#)
- [Server Manager Commands for Managing Tags on page 59](#)
- [Server Manager Commands for Managing Images on page 61](#)
- [Server Manager Operational Commands for Managing Servers on page 64](#)
- [Reimaging Server\(s\) on page 64](#)
- [Provisioning and Configuring Roles on Servers on page 65](#)
- [Restarting Server\(s\) on page 66](#)
- [Show Status of Server\(s\) on page 67](#)
- [Show Status of Provision on page 67](#)

Server Manager Commands Common Options

The common options in [Table 7 on page 56](#) are available with every Server Manager command.

Table 7: Common Command Options

Option	Description
-h, --help	Show the options available for the current command and exit.
--config_file CONFIG_FILE, -c CONFIG_FILE	The name of the Server Manager client configuration file. The default file is <code>/etc/contrail/sm-client-config.ini</code> .
--smgr_ip SMGR_IP	The IP address of the Server Manager process if different from that specified in the config file.
--smgr_port SMGR_PORT	The port that the Server Manager process is listening on if different from that in the config file.

Add New Servers or Update Existing Servers

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

```
server-manager [-h] [--smgr_ip SMGR_IP] [--smgr_port SMGR_PORT]
               [--config_file CONFIG_FILE] add server [-f FILE_NAME]
```

Table 8 on page 56 lists the optional arguments.

Table 8: Server Manager Add Server Command Options

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

The JSON file contains a number of server entries, in the format shown in the following example:

<https://github.com/Juniper/contrail-server-manager/blob/R3.1/src/client/new-server.json>

Most of the parameters in the JSON sample file are self-explanatory. **Cluster_id** defines the cluster to which the server belongs. The sample **roles** array in the example lists all valid role values. **Tag** defines the mapping of tag names and values for grouping and classifying the server.

The **server-manager add** command will add a new entry if the server with the given ID or `mac_address` does not exist in the Server Manager database. If an entry already exists, the add command modifies the fields in the existing entry with any new parameters specified.



NOTE: It is not possible to re-add an existing MAC address under a new server, even if the ID and IP address of that new server are unique.

Delete Servers

Use the **server-manager delete** command to delete one or more servers from the Server Manager database.

```
server-manager [-h] [--smgr_ip SMGR_IP] [--smgr_port SMGR_PORT][--config_file
CONFIG_FILE] delete server (--server_id SERVER_ID | --mac MAC | --ip IP |
--cluster_id CLUSTER_ID | --tag <tag_name=tag_value>.. )
```

Table 9 on page 57 lists the optional arguments.

Table 9: Server Manager Delete Server Command Options

Option	Description
--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.

Display Server Configuration

Use the **server-manager display** command to display the configuration of servers from the Server Manager database.

```
server-manager display [--smgr_ip SMGR_IP] [--smgr_port SMGR_PORT][--config_file
CONFIG_FILE]
server (--server_id SERVER_ID | --mac MAC | --ip IP |
--cluster_id CLUSTER_ID | --tag <tag_name=tag_value>.. ) [--detail]
```

Table 10 on page 57 lists the optional arguments.

Table 10: Server Manager Display Server Command Options

Option	Description
--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.

Table 10: Server Manager Display Server Command Options (*continued*)

Option	Description
<code>--cluster_id CLUSTER_ID</code>	The cluster ID for the server or servers to be displayed.
<code>--tag TagName=TagValue</code>	The TagName that is to be matched with the Tagvalue. Up to seven TagName and Tagvalue pairs separated by commas can be provided.
<code>--detail, -d</code>	Flag to indicate if details are requested.

The criteria for identifying servers to be displayed can be specified by providing the **server_id** or one of the following server parameters: **mac address**, **ip**, **cluster_id**, or **TagName=TagValue**.

Provide one or more of the server matching criteria to display a list of servers.

Server Manager Commands for Managing Clusters

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 58](#)
- [Delete a Cluster on page 59](#)
- [Display Cluster Configuration on page 59](#)

Create a New Cluster or Update an Existing Cluster

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

```
server-manager add cluster [--file_name FILE_NAME]
```

[Table 11 on page 58](#) lists the optional arguments.

Table 11: Server Manager Add Cluster Command Options

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

The JSON file contains a number of cluster entries, in the format shown in the following example:

<https://github.com/Juniper/contrail-server-manager/blob/master/src/client/new-cluster-contrail-4.x.json>

Server membership to a cluster is determined by specifying the ID corresponding to the cluster when defining the server. All of the cluster parameters are available to the server when provisioning roles on the server.

Delete a Cluster

Use the **server-manager delete** command to delete a cluster from the Server Manager database that are no longer needed. Use this command after all servers in the cluster have been deleted.



NOTE: A cluster can only be deleted if no servers are attached to it. If any servers are attached, deletion will fail.

```
server-manager delete cluster [--cluster_id CLUSTER_ID]
```

Table 12 on page 59 lists the optional arguments.

Table 12: Server Manager Delete Cluster Command Options

Option	Description
<code>--cluster_id CLUSTER_ID</code>	The cluster ID for the server or servers to be displayed.

Display Cluster Configuration

Use the **server-manager display** command to list the configuration of a cluster.

```
server-manager display cluster [--cluster_id CLUSTER_ID] [--detail]
```

Table 13 on page 59 lists the optional arguments.

Table 13: Server Manager Display Cluster Command Options

Option	Description
<code>--detail, -d</code>	Flag to indicate if details are requested.
<code>--cluster_id CLUSTER_ID</code>	The cluster ID for the cluster or clusters.

You can optionally specify a cluster ID to get server information about a particular cluster. If the optional parameter is not specified, server information about all clusters in the system is returned.

Server Manager Commands for Managing Tags

Tags are used for grouping servers together so that an operation such as show, reimage, provision, status, and so on can be easily performed on servers that have matching tags. The Server Manager provides a flexible way for you to define your own tags, and then use those tags to assign values to servers. Servers with matching tag values can be easily grouped together. The Server Manager can store a maximum of seven tag values. At initialization, the Server Manager reads the tag names from the configuration file. The tag names can be retrieved or modified using CLI commands. When modifying tag names,

the Server Manager ensures that the tag name being modified is not used by any of the server entries.

- [Create a New Tag or Update an Existing Tag on page 60](#)
- [Display Tag Configuration on page 60](#)

Create a New Tag or Update an Existing Tag

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

```
server-manager add tag [--file_name FILE_NAME] [--tags TAG_LIST]
```

[Table 14 on page 60](#) lists the optional arguments.

Table 14: Server Manager Add New Tag

Option	Description
--file_name FILE_NAME, -f FILE_NAME	The JSON file that contains the tag names.
--tags TAG_LIST	Comma separated list of tag number and tag. For example: tag0=abc,tag1=xyz

The sample JSON file contains a number of tag entries, in the format shown in the following example:

```
{
  "tag1" : "data-center",
  "tag2" : "floor",
  "tag3" : "",
  "tag4" : "pod",
  "tag5" : "rack",
}
```

In the example, you specify a JSON file to add or modify the tags, tag1 through tag5. For tag3, the "" value specifies that if the tag is defined prior to the CLI command, it is removed on execution of the command. The tag name for tag1 is set to data-center. This is allowed if, and only if, none of the server entries are using tag1.

Display Tag Configuration

Use the **server-manager display** command to list the configuration of a tag.

```
server-manager display tag
```

The following is sample output for the **display tag** command.

```
{
  "tag1": "datacenter",
  "tag2": "floor",
}
```



```

    "tag3": "hall",
    "tag4": "rack",
    "tag5": "user_tag"
  }

```

Server Manager Commands for Managing Images

In addition to servers and clusters, the Server Manager also manages information about images and packages that can be used to reimage and configure servers. Images and packages are both stored in the database as images. When new images are added to the database, or existing images are deleted, the Server Manager interfaces with Cobbler to make corresponding modifications in the Cobbler distribution profile for the specified image.

Table 15 on page 61 lists the image types supported.

Table 15: Server Manager Image Types

Image Type	Description
centos	Manages the CentOS ISO base.
contrail-centos-package	Maintains a repository of the package to be installed on the CentOS system image.
ubuntu	Manages the base Ubuntu ISO.
contrail-ubuntu-package	Maintains a repository of packages that contain Contrail and dependent packages to be installed on an Ubuntu base system.
ESXi5.1/ESXi5.5	Manages VMware ESXi 5.1 or 5.5 ISO.

- [Creating New Images or Updating Existing Images on page 61](#)
- [Add an Image on page 62](#)
- [Upload an Image on page 63](#)
- [Delete an Image on page 63](#)
- [Display Image Configuration on page 64](#)

Creating New Images or Updating Existing Images

The Server Manager maintains four types of images – CentOS ISO, Ubuntu ISO, Contrail CentOS package, and Contrail Ubuntu package.

Use the **server-manager add** command or the **server-manager upload** command to add new images to the Server Manager database.

- Use **add** when the new image is present locally on the Server Manager machine. The path provided is the image path on the Server Manager machine.

- Use **upload_image** when the new image is present on the machine where the client program is being invoked. The path provided is the image path on the client machine.

Add an Image

```
server-manager add image [--file_name FILE_NAME]
```

Table 16 on page 62 lists the optional arguments.

Table 16: Server Manager Add Image

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

The JSON file contains an array of possible entries, in the following sample format. The sample shows three images: one CentOS ISO containing Contrail packages, one Ubuntu base ISO, and one Contrail Ubuntu package. When the images are added, corresponding distribution, profile, and repository entries are created in Cobbler by the Server Manager.



NOTE: Release numbers are represented in the sample with `<x.xx>`. Be sure to use the correct release numbers for your image versions.

```
{
  "image": [
    {
      "id": "ubuntu-<x.xx.x>",
      "type": "ubuntu",
      "version": "ubuntu-<x.xx.x>",
      "path": "/iso/ubuntu-<x.xx.x>-server-amd64.iso"
    },
    {
      "id": "centos-<x.xx>",
      "type": "centos",
      "version": "centos-<x.xx>",
      "path": "/iso/CentOS-<x.xx>-x86_64-minimal.iso"
    },
    {
      "id": "contrail-ubuntu-<x.xx>",
```

```

        "type": "contrail-ubuntu-package",
        "version": "contrail-ubuntu-<x.xx>",
        "path": "/iso/contrail-cloud-docker_<x.xx-xx>_all.deb"
    }
]
}

```

Upload an Image

The `server-manager upload_image` command is similar to the `server-manager add` command, except that the path provided for the image being added is the local path on the client machine. This command is useful if the client is being run remotely, not on the Server Manager machine, and the image being added is not physically present on the Server Manager machine.

```
server-manager upload_image image_id image_version image_type file_name
```

[Table 17 on page 63](#) lists the optional arguments.

Table 17: Server Manager Upload Image

Option	Description
<code>image_id</code>	Name of the new image.
<code>image_version</code>	Version number of the new image.
<code>image_type</code>	Type of image: <code>fedora</code> , <code>centos</code> , <code>ubuntu</code> , <code>contrail-ubuntu-package</code> , <code>contrail-centos-package</code>
<code>file_name</code>	Complete path for the file.

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 image --image_id <image_id>
```

[Table 18 on page 63](#) lists the optional arguments.

Table 18: Server Manager Delete Image

Option	Description
<code>image_id</code>	The image ID for the image to be deleted.

Display Image Configuration

Use the **server-manager display** command to list the configuration of images from the Server Manager database. If the detail flag is specified, detailed information about the image is returned. If the optional **image_id** is not specified, information about all the images is returned.

```
server-manager display image [--image_id IMAGE_ID] [--detail]
```

Table 19 on page 64 lists the optional arguments.

Table 19: Server Manager Display Image Configuration

Option	Description
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, which also adds the images to Cobbler. The set of servers to be reimaged can be specified by providing match criteria for servers already added to the Server Manager database, using **server_id**.

You must identify the base image ID to be used to reimage.

The command prompts for a confirmation before making the REST API call to the Server Manager to start reimaging the servers. This confirmation message can be bypassed by specifying the optional **--no_confirm** or **-F** parameter on the command line.

```
server-manager reimage
    [--package_image_id PACKAGE_IMAGE_ID]

    [--no_reboot]

    (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag
    <tag_name=tag_value>)

    [--no_confirm]
    base_image_id
```

Options include the following:

Table 20 on page 65 lists the optional arguments.

Table 20: Server Manager Reimage

Option	Description
<code>base_image_id</code>	The image ID of the base image to be used.
<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 parameters from the database to achieve the task of configuring the servers with specific roles.

When the **server-manager provision** command is used, the Server Manager pushes the specified server configurations to the servers.

```
server-manager provision
  (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag
  <tag_name=tag_value> )
  [--no_confirm]
  package_image_id
```

Options include the following:

Table 21 on page 65 lists the optional arguments.

Table 21: Server Manager Provision

Option	Description
<code>package_image_id</code>	The Contrail package image ID to be used for provisioning.
<code>--server_id SERVER_ID</code>	The server ID for the server or servers to be provisioned.

Table 21: Server Manager Provision (*continued*)

Option	Description
<code>--cluster_id CLUSTER_ID</code>	The cluster ID for the server or servers to be provisioned.
<code>--tag TagName=TagValue</code>	TagName to be matched with Tagvalue.
<code>--provision_params_file PROVISION_PARAMS_FILE, -f PROVISION_PARAMS_FILE</code>	Optional JSON file containing the parameters for provisioning the server(s).
<code>--no_confirm, -F</code>	Flag to bypass confirmation message, default = do NOT bypass.



NOTE: Adding and deleting roles is not supported in Contrail Release 4.0.

Restarting Server(s)

Use the **server-manager restart** command to reboot identified server(s). Servers can be specified from the database by providing standard match conditions. The **restart** command provides a way to reboot or power-cycle the servers, using the Server Manager REST API interface. If reimaging is intended, use the **restart** command with the **net-boot** flag enabled. When netbooted, the Puppet agent is also installed and configured on the servers. If there are Puppet manifest files created for the server prior to rebooting, the agent pulls those from the Server Manager and executes the configured Puppet manifests. The **restart** command uses an IPMI mechanism to power cycle the servers, if available and configured. Otherwise, the **restart** command uses SSH to the server and the existing reboot command mechanism is used.

```
server-manager restart
    (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag
    <tag_name=tag_value>)

    [--net_boot]
    [--no_confirm]
```

Table 22 on page 66 lists the optional arguments.

Table 22: Server Manager Restart

Option	Description
<code>--server_id SERVER_ID</code>	The server ID for the server or servers to be restarted.
<code>--cluster_id CLUSTER_ID</code>	The cluster ID for the server or servers to be restarted.
<code>--tag TagName=TagValue</code>	TagName to be matched with Tagvalue.
<code>--net_boot, -n</code>	Optional parameter to indicate if the server should be netbooted.
<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 (--server_id SERVER_ID | --cluster_id CLUSTER_ID
| --tag <tag_name=tag_value>)
```

Table 23 on page 67 lists the optional arguments.

Table 23: Server Manager Status Server

Option	Description
--server_id SERVER_ID	The server ID for the server whose status is to be fetched.
--cluster_id CLUSTER_ID	The cluster ID for the server or servers to be restarted.
--tag TagName=TagValue	TagName to be matched with Tagvalue.

The status command provides a way to fetch the current status of a server.

Status outputs include the following:

```
restart_issued
reimage_started
provision_issued
provision_completed
openstack_started
openstack_completed
```

Show Status of Provision

Use the **server-manager status provision** to view the detailed provisioning status of servers or cluster. The **status** command provides a way to fetch the current status of a provision command.

```
server-manager status provision (--server_id SERVER_ID | --cluster_id
CLUSTER_ID | --tag <tag_name=tag_value>)
```

Table 24 on page 67 lists the optional arguments.

Table 24: Server Manager Status Provision

Option	Description
--server_id SERVER_ID	The server ID for the server whose status is to be fetched.
--cluster_id CLUSTER_ID	The cluster ID for the server or servers to be restarted.
--tag TagName=TagValue	TagName to be matched with Tagvalue.

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 68](#)
- [API: Add a Server on page 68](#)
- [API: Delete Servers on page 69](#)
- [API: Retrieve Server Configuration on page 69](#)
- [API: Add an Image on page 69](#)
- [API: Upload an Image on page 70](#)
- [API: Get Image Information on page 70](#)
- [API: Delete an Image on page 70](#)
- [API: Add or Modify a Cluster on page 71](#)
- [API: Delete a Cluster on page 71](#)
- [API: Get Cluster Configuration on page 71](#)
- [API: Get All Server Manager Configurations on page 71](#)
- [API: Reimage Servers on page 72](#)
- [API: Provision Servers on page 72](#)
- [API: Restart Servers on page 72](#)

REST APIs for Server Manager Configuration Database Entries

The REST API calls in this section help in configuring different elements in the Server Manager database.



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

API: Add a Server

To add a new server to the service manager configuration database:

URL: `http://<SM-IP-Address>:<SM-Port>/server`

Method: **PUT**

Payload: JSON payload containing an array of servers to be added. For each server in the array, all the parameters are specified as JSON fields. The mask, gateway, password, and domain fields are optional, and if not specified, the values of these fields are taken from the cluster to which the server belongs.

The following is a sample JSON file for adding a server in Contrail Release 4.0.

<https://github.com/Juniper/contrail-server-manager/blob/master/src/client/new-server-contrail-4.x.json>

API: Delete Servers

Use one of the following formats to delete a server.

URL: `http://<SM-IP-Address>:<SM-Port>/server?server_id=SERVER_ID`

`http://<SM-IP-Address>:<SM-Port>/server?cluster_id=CLUSTER_ID`

`http://<SM-IP-Address>:<SM-Port>/server?mac=MAC`

`http://<SM-IP-Address>:<SM-Port>/server?ip=IP`

`http://<SM-IP-Address>:<SM-Port>/server[?tag=<tag_name>=<tag_value>,,]`

Method : DELETE

Payload : None

API: Retrieve Server Configuration

Use one of the following methods to retrieve a server configuration. The detail argument is optional, and specified as part of the URL if details of the server entry are requested.

URL: `http://<SM-IP-Address>:<SM-Port>/server[?server_id=SERVER_ID&detail]`

`http://<SM-IP-Address>:<SM-Port>/server[?cluster_id=CLUSTER_ID&detail]`

`http://<SM-IP-Address>:<SM-Port>/server[?tag=<tag_name>=<tag_value>,,]`

`http://<SM-IP-Address>:<SM-Port>/server[?mac=MAC&detail]`

`http://<SM-IP-Address>:<SM-Port>/server[?ip=IP&detail]`

`http://<SM-IP-Address>:<SM-Port>/server[?tag=<tag_name>=<tag_value>,,]`

Method : GET

Payload : None

API: Add an Image

Use the following to add a new image to the Server Manager configuration database from the Server Manager machine.

An image is either an ISO for a CentOS or Ubuntu distribution or an Ubuntu Contrail package repository. When adding an image, the image file is assumed to be available on the Server Manager machine.

URL : `http://<SM-IP-Address>:<SM-Port>/image`

Method: PUT

Payload: Specifies all the parameters that define the image being added.

See sample payload in the following:

<https://github.com/Juniper/contrail-server-manager/blob/master/src/client/new-package.json>

API: Upload an Image

Use the following to upload a new image from a client to the Server Manager configuration database.

An image is an ISO for a CentOS or Ubuntu distribution or an Ubuntu Contrail package repository. Add image assumes the file is available on the Server Manager, whereas upload image transfers the image file from the client machine to the Server Manager machine.

URL : `http://<SM-IP-Address>:<SM-Port>/image/upload`

Method: **PUT**

Payload: Specifies all the parameters that define the image being added.

```
{
  "image": [
    {
      "id": "Image-id",
      "type": "image_type", <ubuntu or centos or esxi5.1 or esxi5.5 or
      contrail-ubuntu-package or contrail-centos-package>
      "version": "image_version",
      "path": "path-to-image-on-client-machine"
    }
  ]
}
```

API: Get Image Information

Use the following to get image information.

URL : `http://<SM-IP-Address>:<SM-Port>/image[?image_id=IMAGE_ID&detail]`

Method: **GET**

Payload: Specifies criteria for the image being sought. If no match criteria is specified, information about all the images is provided. The details field specifies if details of the image entry in the database are requested.

API: Delete an Image

Use the following to delete an image.

URL: `http://<SM-IP-Address>:<SM-Port>/image?image_id=IMAGE_ID`

Method: **DELETE**

Payload: Specifies criteria for the image being deleted.

API: Add or Modify a Cluster

Use the following to add a cluster to the Server Manager configuration database. A cluster maintains parameters for a set of servers that work together in different roles to provide complete functions for a Contrail cluster.

URL: **http://<SM-IP-Address>:<SM-Port>/cluster**

Method: **PUT**

Payload: Contains the definition of the cluster, including all the global parameters needed by all the servers in the cluster. The `subnet_mask`, `gateway`, `password`, and `domain` fields define parameters that apply to all servers in the VNS. These parameter values can be individually overridden for a server by specifying different values in the server entry.

A sample JSON for Contrail Release 4.0 is at the following:

<https://github.com/Juniper/contrail-server-manager/blob/master/src/client/new-cluster-contrail-4.x.json>

API: Delete a Cluster

Use this API to delete a cluster from the Server Manager database.

URL: **http://<SM-IP-Address>:<SM-Port>/cluster?cluster_id=CLUSTER_ID**

Method: **DELETE**

Payload: None

API: Get Cluster Configuration

Use this API to get a cluster configuration.

URL: **http://<SM-IP-Address>:<SM-Port>/cluster[?cluster_id=CLUSTER_ID&detail]**

Method: **GET**

Payload: None

The optional detail argument is specified as part of the URL if details of the VNS entry are requested.

API: Get All Server Manager Configurations

Use this API to get all configurations of Server Manager objects, including servers, clusters, images, and tags.

URL: **http://<SM-IP-Address>:<SM-Port>/all[?detail]**

Method: **GET**

Payload: None

The optional detail argument is specified as part of the URL if details of the Server Manager configuration are requested.

API: Reimage Servers

Use one of the following API formats to reimage one or more servers.

URL: `http://<SM-IP-Address>:<SM-Port>/server/reimage?server_id=SERVER_ID`
`http://<SM-IP-Address>:<SM-Port>/server/reimage?cluster_id=CLUSTER_ID`
`http://<SM-IP-Address>:<SM-Port>/server/reimage?mac=MAC`
`http://<SM-IP-Address>:<SM-Port>/server/reimage?ip=IP`
`http://<SM-IP-Address>:<SM-Port>/server/reimage [?tag=<tag_name>=<tag_value>,,]`

Method: **POST**

Payload: None

API: Provision Servers

Use this API to provision or configure one or more servers for roles configured on them.

URL: `http://<SM-IP-Address>:<SM-Port>/server/provision`

Method: **POST**

Payload: Specifies the criteria to be used to identify servers which are being provisioned. The servers can be identified by `server_id`, `mac`, `cluster_id` or `tags`. See the following example.

```
{
  server_id : <server_id> OR
  mac : <server_mac_address> OR
  cluster_id : <cluster_id> OR
  tag : {"data-center" : "dc1"}
}
```

API: Restart Servers

This REST API is used to power cycle the servers and reboot either with net-booting enabled or disabled.

If the servers are to be reimaged and reprovisioned, the **net-boot** flag should be set.

If servers are only being reprovisioned, the **net-boot** flag is not needed, however, the Puppet agent must be running on the target systems with the correct puppet configuration to communicate to the puppet master running on the Server Manager.

URL: `http://<SM-IP-Address>:<SM-Port>/server/restart?server_id=SERVER_ID`
`http://<SM-IP-Address>:<SM-Port>/server/restart?[netboot&]cluster_id=CLUSTER_ID`
`http://<SM-IP-Address>:<SM-Port>/server/restart? [netboot&]mac=MAC`
`http://<SM-IP-Address>:<SM-Port>/server/restart? [netboot&]ip=IP`
`http://<SM-IP-Address>:<SM-Port>/server/restart ?`
`[netboot&]tag=<tag_name>=<tag_value>`

Method: **POST**

Payload: Specifies the criteria to be used to identify servers which are being restarted. The servers can be identified by their **server_id**, **mac**, **cluster_id**, or **tag**. The **netboot** parameter specifies if the servers being power-cycled are to be booted from Cobbler or locally.

Example: Reimaging and Provisioning a Server

This example shows the steps used in Server Manager software to configure, reimage, and provision a server running all roles of the Contrail system in a single-node configuration.



NOTE: Component names and IP addresses in the following are used for example only. To use this example in your own environment, be sure to use addresses and names specific to your environment.

The Server Manager client configuration file used for the following CLI commands, is `/opt/contrail/server_manager/client/sm-client-config.ini`. It contains the values for the server IP address and port number as follows:

[SERVER-MANAGER]

`listen_ip_addr = 192.168.1.10 (Server Manager IP address)`

`listen_port = 9001`

Overview

The steps to be followed include:

1. Configure cluster.
2. Configure servers.
3. Configure images.
4. Reimage servers (either using servers configured above or using explicitly specified reimage parameters with the request).
5. Provision servers (either using servers configured above or using explicitly specified provision parameters with the request).

Procedure

1. Configure a cluster.

```
server-manager add cluster -f cluster.json
```
2. Configure the server.

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

3. Configure images.

In the example, the image files for **ubuntu-xx.xx.x** and **contrail-ubuntu-164** are located at the corresponding image path specified on the Server Manager.

```
server-manager add -c smgr_client_config.ini image -f image.json
```

4. Reimage servers.

This step can be performed after the configuration from the previous steps is in the Server Manager database.

```
server-manager reimage --server_id demo-server ubuntu-<x.xx.x>
```

5. Provision servers.

```
server-manager provision --server_id demo-server contrail-ubuntu-164
```



NOTE: The samples for all JSONs used in the procedure above are available as links in the documentation for the API calls for those respective commands.

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 75](#)
- [Create a Cluster for Server Manager on page 75](#)
- [Edit a Cluster through Edit JSON on page 84](#)
- [Working with Servers in the Server Manager User Interface on page 84](#)
- [Add a Server on page 85](#)
- [Edit Tags for Servers on page 87](#)
- [Using the Edit Config Option for Multiple Servers on page 87](#)
- [Edit a Server through Server Manager, Edit JSON on page 88](#)
- [Filter Servers by Tag on page 88](#)
- [Viewing Server Details on page 89](#)
- [Configuring Images and Packages on page 91](#)
- [Add New Image or Package on page 91](#)
- [Selecting Server Manager Actions for Clusters on page 92](#)
- [Reimage a Cluster on page 92](#)
- [Provision a Cluster on page 92](#)

Log In to Server Manager

The Server Manager user interface can be accessed using:

http://<server-manager-user-interface-ip>:9080

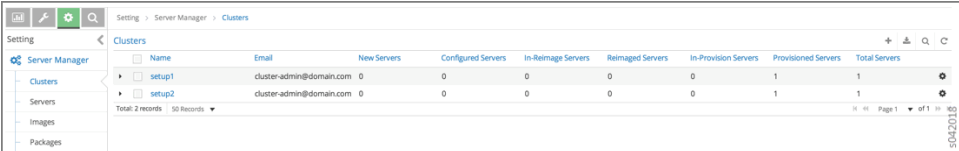
Where **<server-manager-user-interface-ip>** is the IP address of the server on which the Server Manager web user interface is installed.

From the Contrail user interface, select **Setting > Server Manager** to access the Server Manager home page. From this page you can manage Server Manager settings for clusters, servers, images, and packages.

Create a Cluster for Server Manager

Select **Add Cluster** to identify a cluster to be managed by the Server Manager. Select **Setting > Server Manager > Clusters**, to access the **Clusters** page, see [Figure 9 on page 75](#).

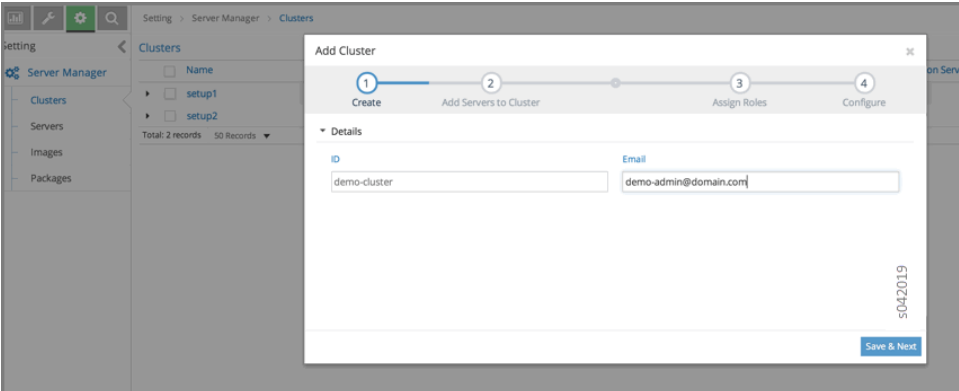
Figure 9: Server Manager > Clusters



Name	Email	New Servers	Configured Servers	In-Reimage Servers	Reimage Servers	In-Provision Servers	Provisioned Servers	Total Servers
setup1	cluster-admin@domain.com	0	0	0	0	0	1	1
setup2	cluster-admin@domain.com	0	0	0	0	0	1	1
Total: 2 records		50 Records						

To create a new cluster, click the plus icon in the upper right of the **Clusters** page. The **Add Cluster** window is displayed. In the **Add Cluster** window, you can add a new cluster ID and the domain e-mail address of the cluster. See [Figure 10 on page 75](#).

Figure 10: Add Cluster



Setting > Server Manager > Clusters

Clusters

setup1

setup2

Total: 2 records 50 Records

1 Create

2 Add Servers to Cluster

3 Assign Roles

4 Configure

Details

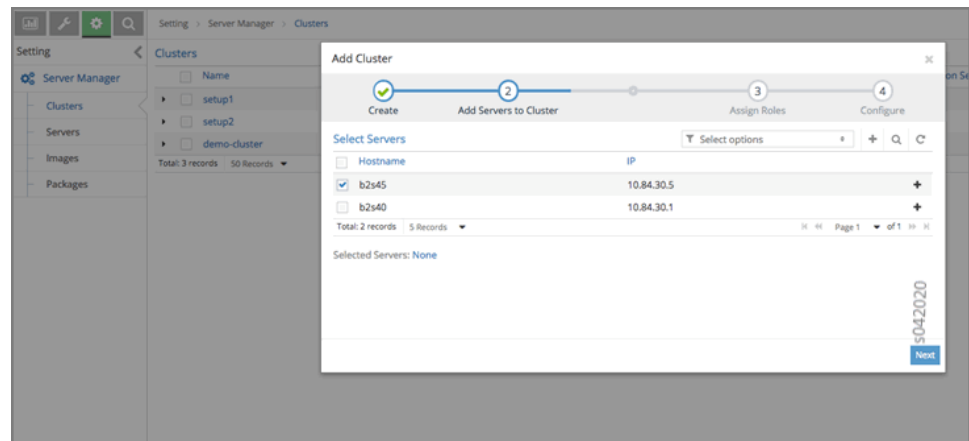
ID: demo-cluster

Email: demo-admin@domain.com

Save & Next

When you are finished adding information about the new cluster in the **Add Clusters** window, click **Save & Next**. Now you can add servers to the cluster, see [Figure 11 on page 76](#).

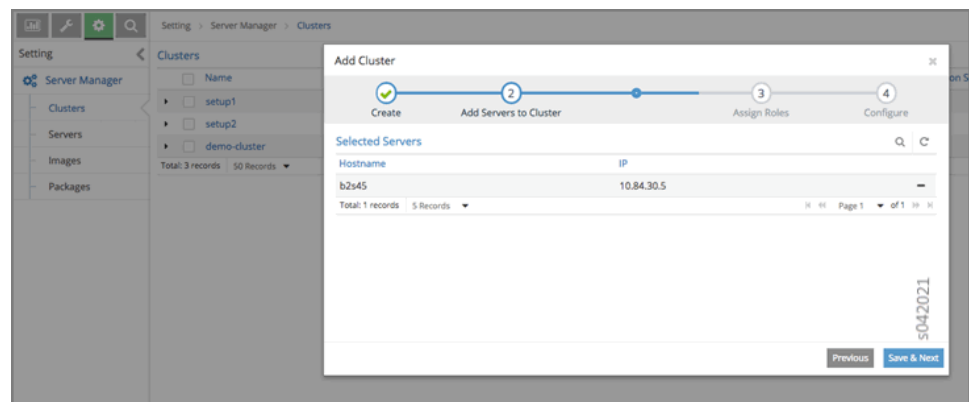
Figure 11: Add Servers to Cluster



Click the check box of each server to be added to the cluster.

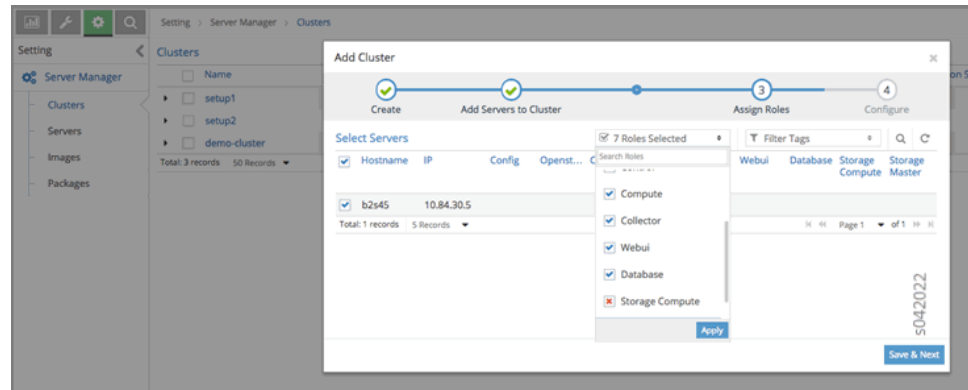
When you are finished, click **Next**. The selected servers are added to the cluster, see [Figure 12 on page 76](#).

Figure 12: Add Servers to Cluster, Next



When you are finished adding servers, click **Save & Next**. Now you can assign Contrail roles to servers that you select in the cluster. Roles available are Config, OpenStack, Control, Compute, and Collector. Select each role assignment for the selected server. You can also unselect any assigned role. The assigned roles correspond to the role functions in operation on the server, see [Figure 13 on page 77](#).

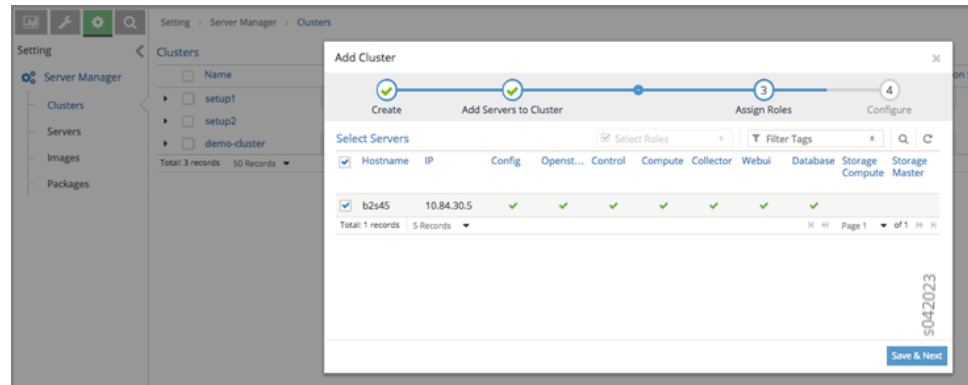
Figure 13: Assign Roles



When you are finished selecting roles for the selected server in the **Roles** window, click **Apply** to save your choices.

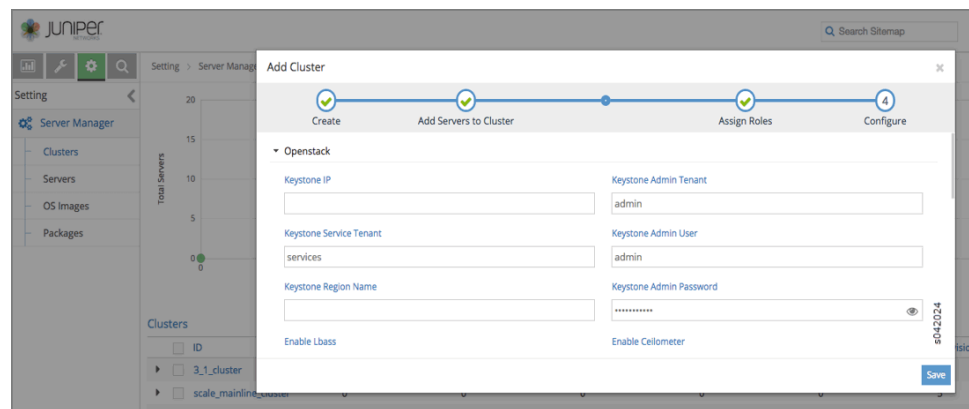
Click **Save & Next** to view your selections. Check marks are displayed in the columns of the **Add Cluster** window, see [Figure 14 on page 77](#).

Figure 14: Roles Assigned



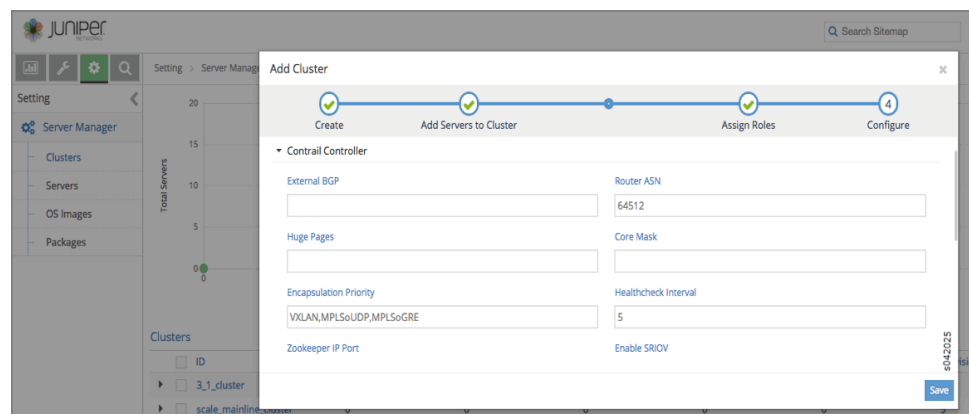
The next step after roles are assigned is to enter the cluster configuration information for OpenStack. After viewing the assigned roles, click **Save & Next**. The **Add Cluster** window is displayed. Click an icon that opens a set of fields where you can enter OpenStack or Contrail configuration information for the cluster. In the following image, the **Openstack** icon is selected. You can enter **Keystone** configuration information, such as IP, Admin tenant, user, and password, service tenant, and region name. You can also enable LBaaS and Ceilometer, see [Figure 15 on page 78](#).

Figure 15: OpenStack Configuration



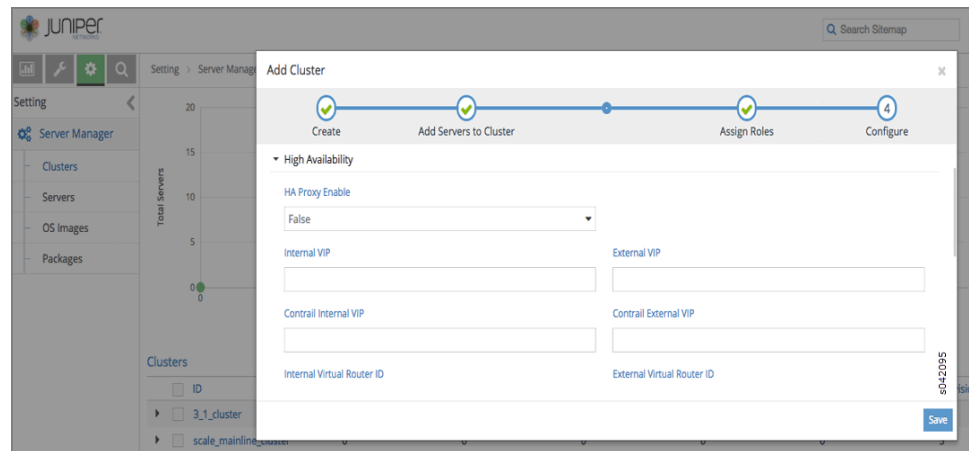
In the following image, the Contrail controller icon is selected. You can enter configuration information for Contrail, such as **External BGP, Router ASN, Huge Pages, Core Mask, Encapsulation Priority, Healthcheck Interval, Zookeeper IP Port, Enable SRIOV**, and so on, see [Figure 16 on page 78](#).

Figure 16: Configure Contrail



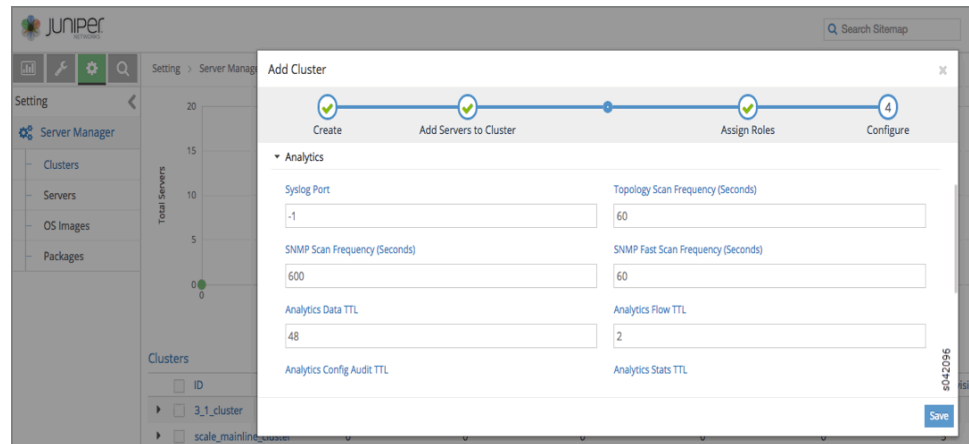
In the following image, the High Availability (HA) icon is selected. You can configure high availability parameters such as **HA Proxy Enable, Internal and External VIP**, and so on, see [Figure 17 on page 79](#).

Figure 17: Configure High Availability



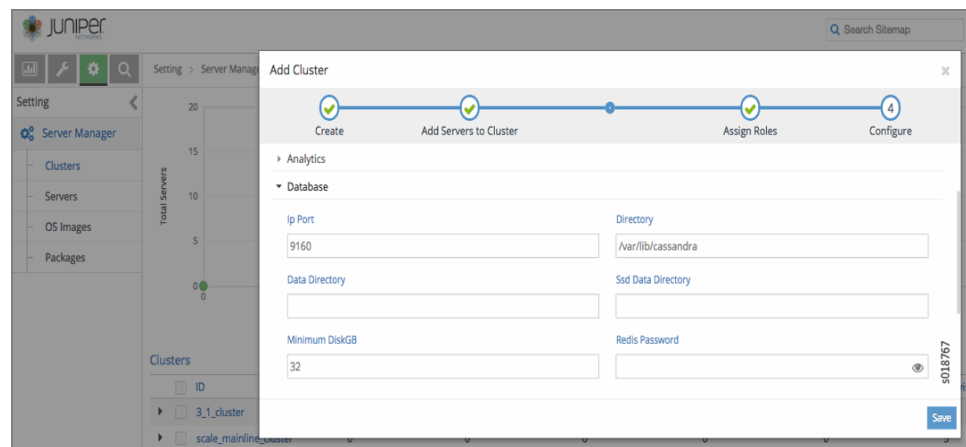
In the following image, the **Analytics** icon is selected. Here you can configure parameters for Contrail Analytics, including **Syslog Port**, various scan frequencies, and various TTL settings, see [Figure 18 on page 79](#).

Figure 18: Configure Analytics



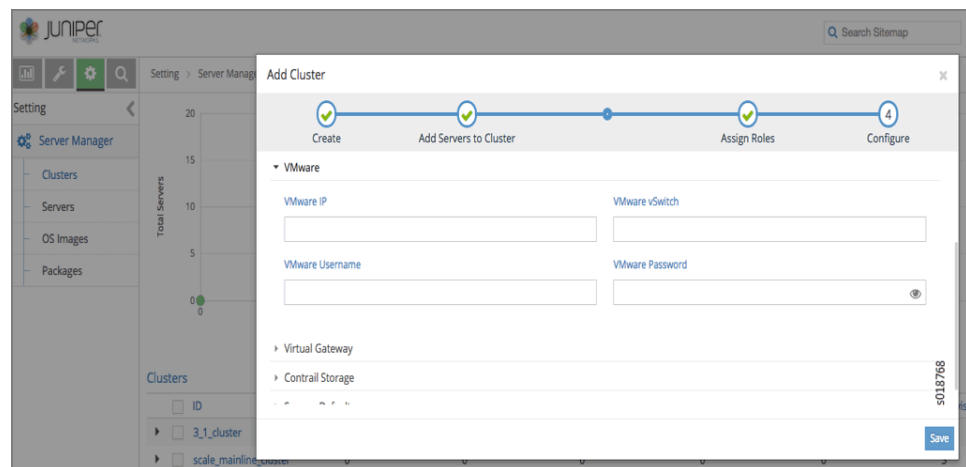
In following image, the **Database** icon is selected. You can configure parameters for the Contrail database, including **IP Port**, **Directory**, **Minimum Disk GB**, and so on, see [Figure 19 on page 80](#).

Figure 19: Configure Database



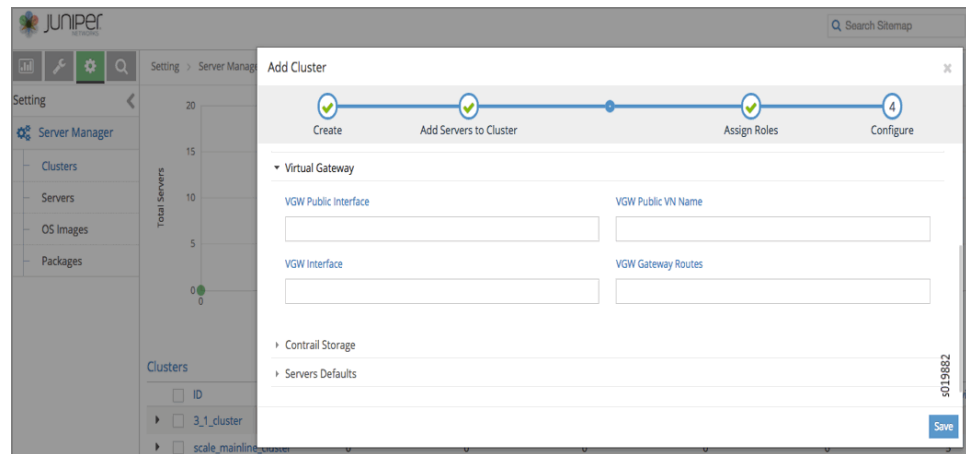
In following image, the **VMware** icon is selected. You can configure parameters for Contrail VMware , including **VMware IP**, **VMware vSwitch**, **Username**, **Password** , and so on, see [Figure 20 on page 80](#).

Figure 20: Configure VMware



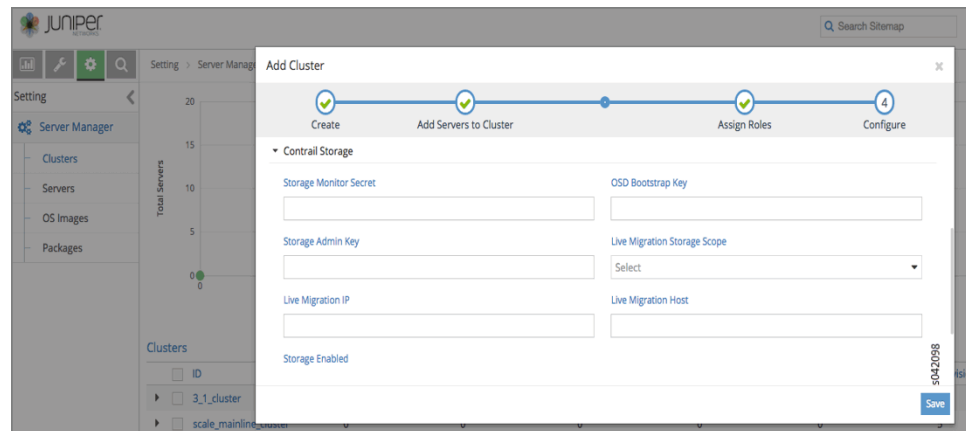
In following image, the **Virtual Gateway** icon is selected. You can configure parameters for the Contrail Virtual Gateway, including **VGW Public Interface**, **VGW Public VN Name**, **VGW Interface**, **Routes** , and so on, see [Figure 21 on page 81](#).

Figure 21: Configure Virtual Gateway



In following image, the **Contrail Storage** icon is selected. You can configure parameters for Contrail Storage, including **Storage Monitor Secret**, **OSD Bootstrap Key**, **Admin Key**, and so on, see [Figure 22 on page 81](#).

Figure 22: Configure Contrail Storage



When you are finished entering all of the cluster configuration information, click **Save** to submit the configurations. You can view all configured clusters on the **Clusters** window by selecting **Setting > Server Manager > Clusters**, see [Figure 23 on page 81](#).

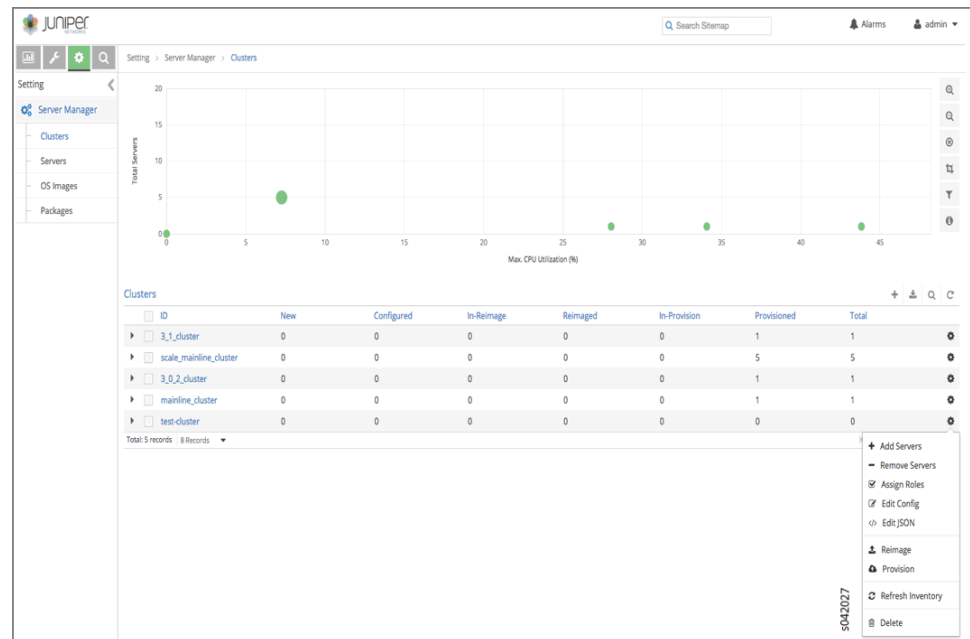
Figure 23: View Configured Clusters

Setting > Server Manager > Clusters								
Clusters								
<input type="checkbox"/>	Name	Email	New Servers	Configured Servers	In-Reimage Servers	Reimaged Servers	In-Provision Servers	Provisioned
<input type="checkbox"/>	setup1	cluster-admin@domain.com	0	0	0	0	0	0
<input type="checkbox"/>	setup2	cluster-admin@domain.com	0	0	0	0	0	1
<input type="checkbox"/>	demo-cluster	demo-admin@domain.com	0	0	0	0	0	1
Total: 3 records 50 Records								

To perform an action on one of the configured clusters, click the gear wheel icon at the right to select from a menu of actions available for that cluster, including **Add Servers**,

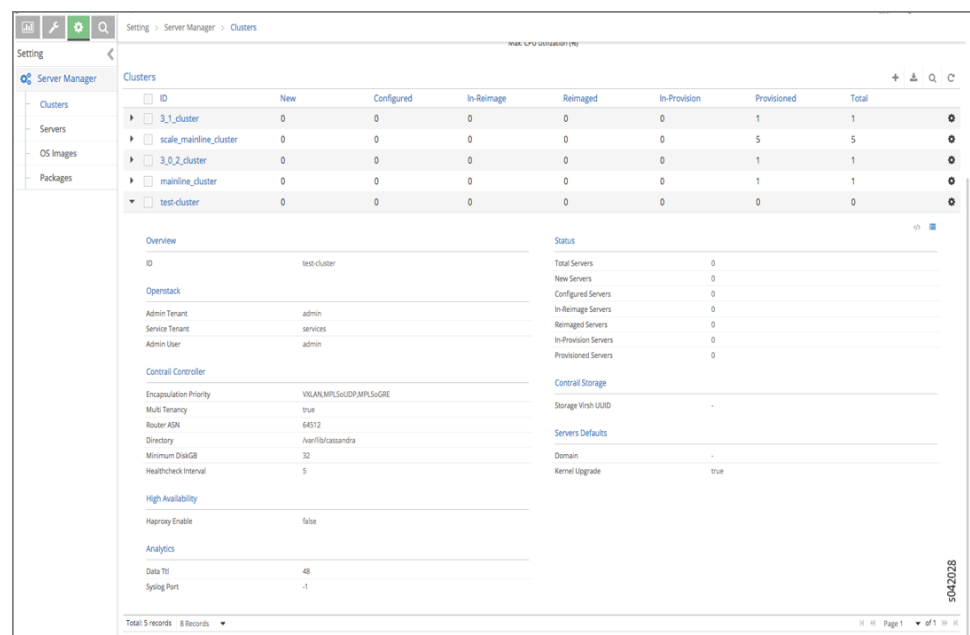
Remove Servers, Assign Roles, Edit Config, Reimage, Provision, and Delete, see [Figure 24 on page 82](#).

Figure 24: Select Cluster Action



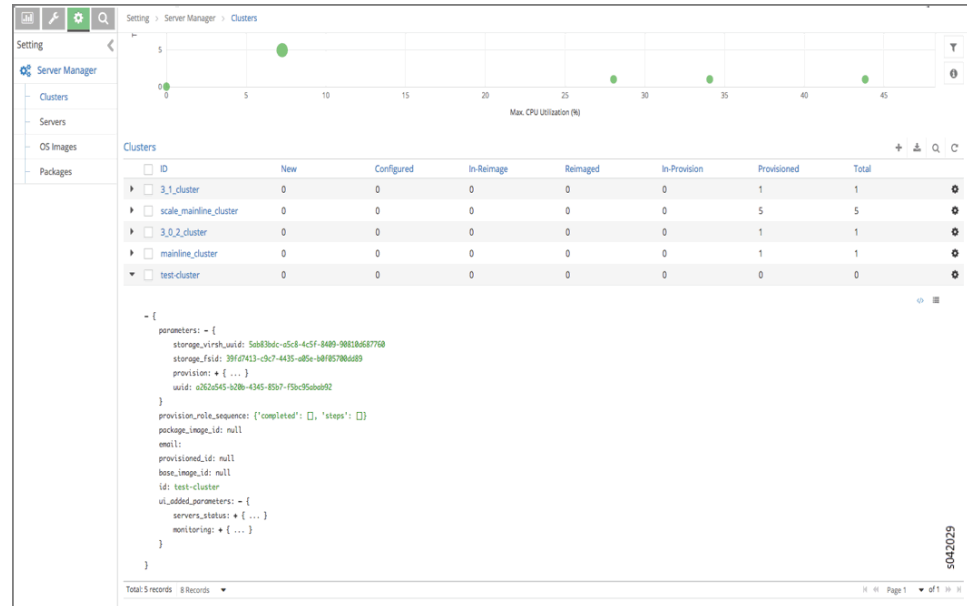
You can also click the expansion icon on the left side of the cluster name to display the details of that cluster in an area below the name line, see [Figure 25 on page 82](#).

Figure 25: Display Cluster Details



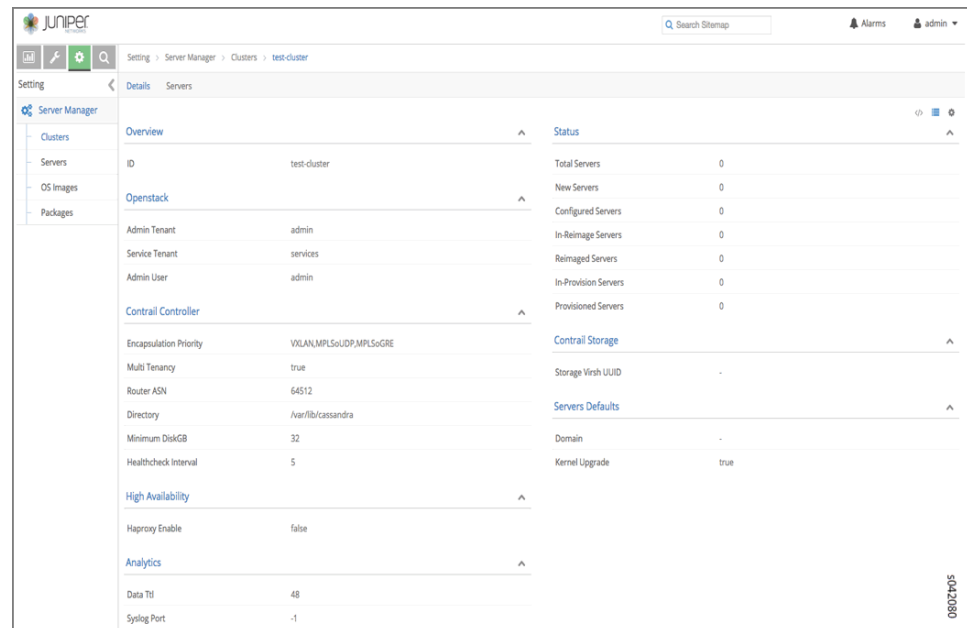
Click the upper right icon to switch to the JSON view to see the contents of the JSON file for the cluster, see [Figure 26 on page 83](#).

Figure 26: View Cluster JSON



The cluster name is a link, click the cluster name to display the cluster **Details** page, see [Figure 27 on page 83](#).

Figure 27: Link to View Cluster Details



Click the **Servers** tab to display the servers under that cluster, see [Figure 28 on page 84](#).

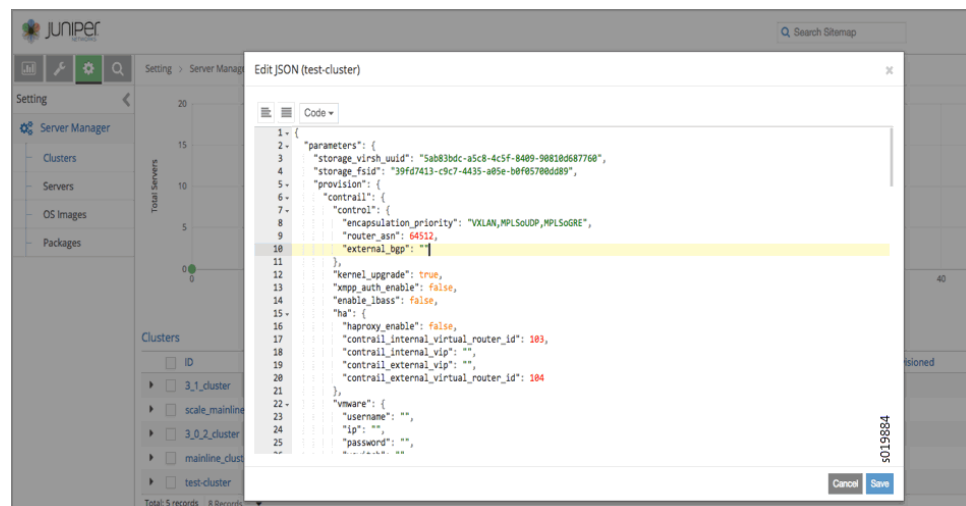
Figure 28: Display Servers for Cluster



Edit a Cluster through Edit JSON

Select **Edit JSON** to edit a cluster by editing the JSON file. Make changes to the JSON code and click **Save** to save the edited configuration for the cluster, see [Figure 29 on page 84](#).

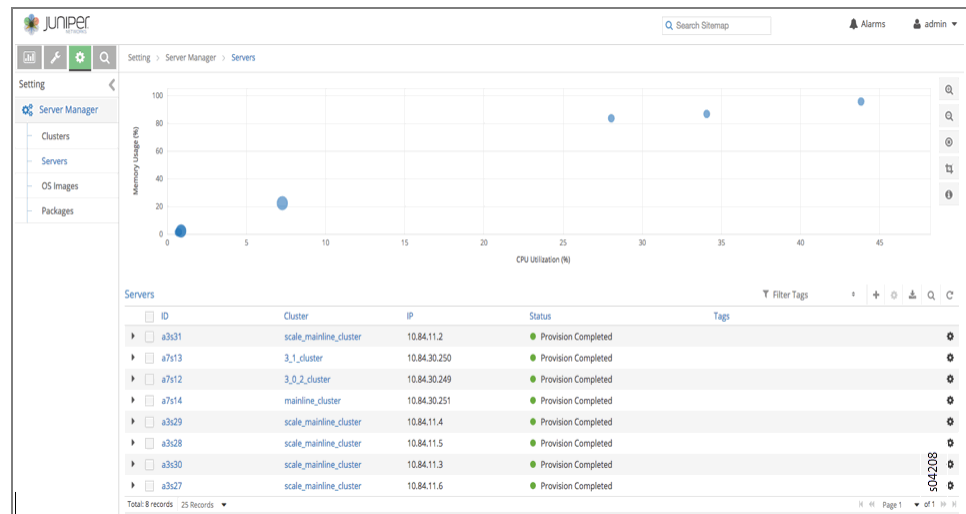
Figure 29: Edit Cluster JSON



Working with Servers in the Server Manager User Interface

Select **Setting > Server Manager** and click the **Servers** link in the left sidebar at to view a list of all servers, see [Figure 30 on page 85](#).

Figure 30: View Servers



Add a Server

To add a new server, select **Setting > Server Manager > Servers** and click the plus (+) icon at the upper right side in the header line. The **Add Server** window is displayed, see [Figure 31 on page 85](#), in which the **System Management** tab is expanded. Here you enter the details of **ID**, **Password**, **Domain**, **Partition**, and so on for the server.

Figure 31: Add Server, System Management

Add Server

System Management

ID: demo-server Password: [password field]

Host Name: demo-server Domain: englab.juniper.net

Static IP: [empty field] IPMI Address: 10.84.60.148

IPMI Username: ADMIN IPMI Password: [password field]

Partition: [empty field]

Interfaces

Cancel Save

In the following image, the **Physical Interfaces** icon is selected. You can add new interfaces or edit existing interfaces. To enable editing for any field, hover the cursor on any selected field to open it, see [Figure 32 on page 86](#).

Figure 32: Add Server, Physical Interfaces

Name	IP/Mask	MAC Address	Gateway	DHCP	TOR	TOR Port
eth01	1.2.3.4	aa:aa:aa:aa:aa:aa	2.2.3.4	<input checked="" type="checkbox"/>		-

+Add

Cancel Save

3_1_cluster 10.84.30.250 Provision Completed

In the following image, the **Contrail Storage** icon is selected. You can configure parameters for Contrail Storage, including selecting a package and adding storage disks locations, see [Figure 33 on page 86](#).

Figure 33: Add Server, Contrail Storage

Storage Repo ID

Select Repo ID

Chassis ID

Select Chassis ID

Add New Chassis ID

Storage Disks

+Add

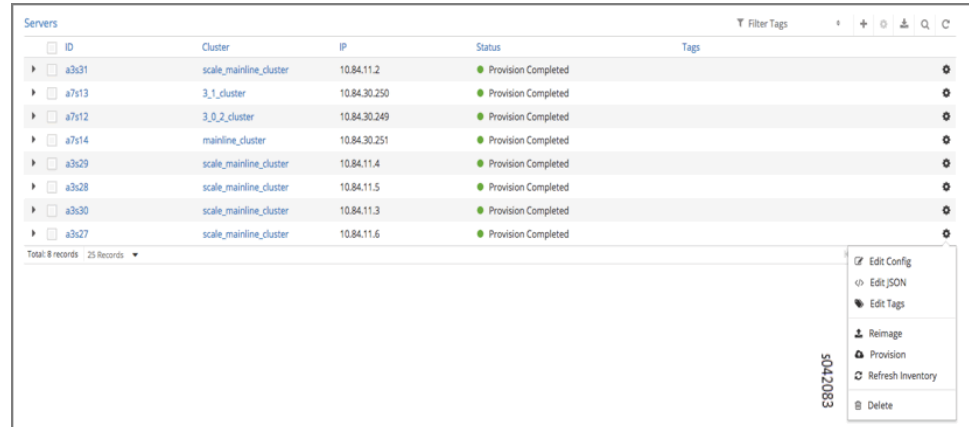
Cancel Save

scale_mainline_cluster 10.84.11.4 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 JSON**, **Edit Tags**, **Reimage**, **Provision**, **Refresh Inventory**, and **Delete**, see [Figure 34 on page 87](#).

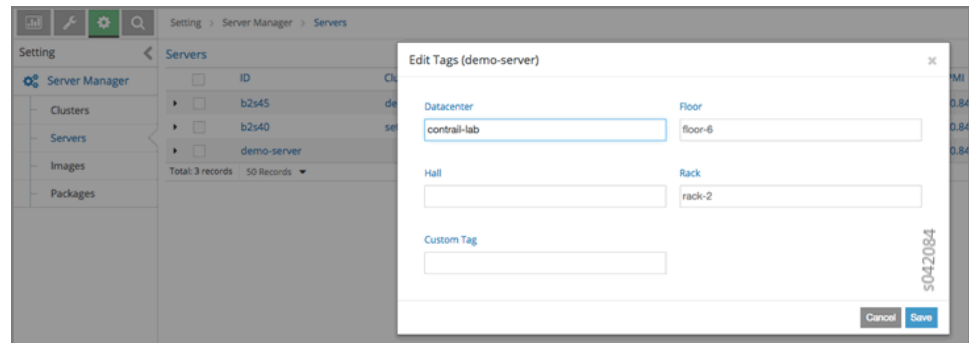
Figure 34: Select Server Actions



Edit Tags for Servers

Select **Edit Tags** from the gear wheel icon menu. The **Edit Tags** window is displayed. Enter any user-defined tags to be associated with the selected server, then click **Save** to add the tags to the server configuration, see [Figure 35 on page 87](#).

Figure 35: Edit Tags



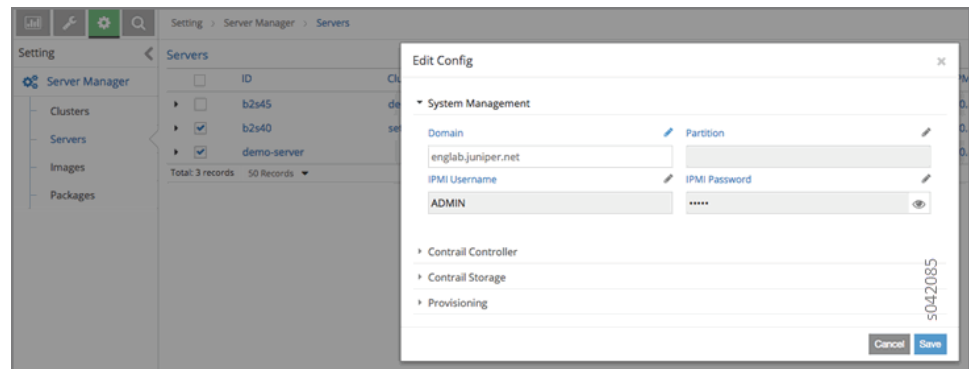
Using the Edit Config Option for Multiple Servers

You can also edit the configuration of multiple servers at one time. From the **Servers** window at **Setting > Server Manager > Servers**, select the servers you want to edit, then click a gear wheel icon at the right to open the action menu, and select **Edit Config**.

The **Edit Config** window is displayed, as shown.

Click a pencil icon to open configuration fields that can be edited. Fields include **System Management**, **Contrail Controller**, **Contrail Storage**, and so on, see [Figure 36 on page 88](#).

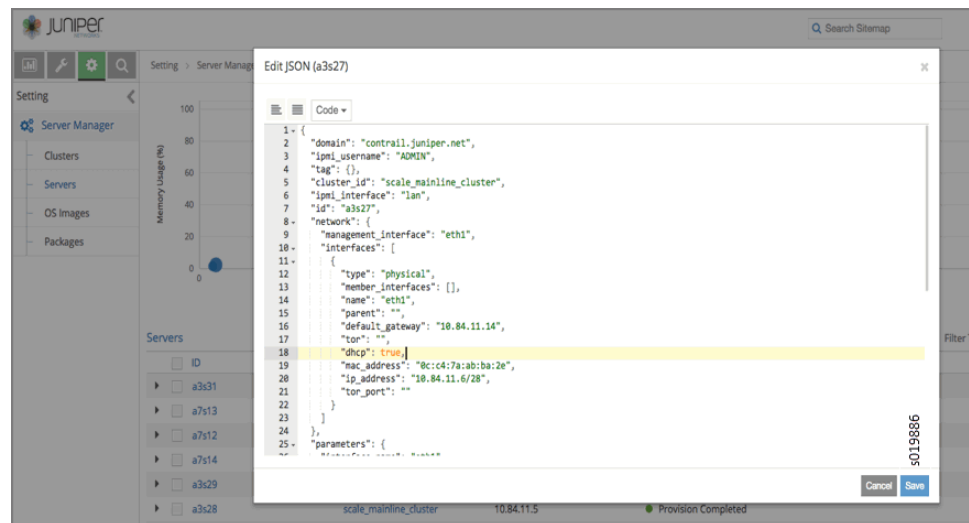
Figure 36: Edit Config, Multiple Servers



Edit a Server through Server Manager, Edit JSON

Select **Edit JSON** to edit the server through JSON file. Make changes to the server details in the JSON, then click **Save**, see [Figure 37 on page 88](#).

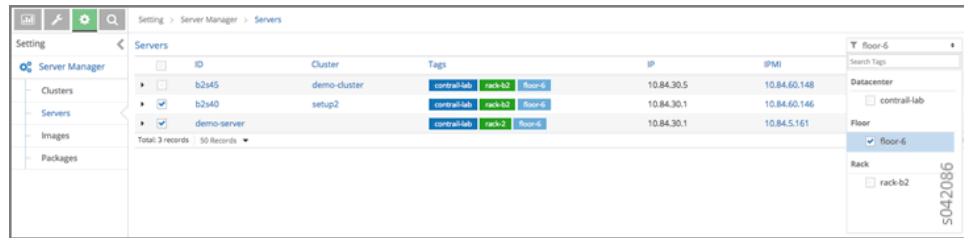
Figure 37: Server Edit JSON



Filter Servers by Tag

You can filter servers according to the tags defined for them. In the **Servers** window, click the **Filter Tags** field in the upper right heading. A list of configured tags is displayed. Select a tag by which to filter the list of servers, see [Figure 38 on page 89](#).

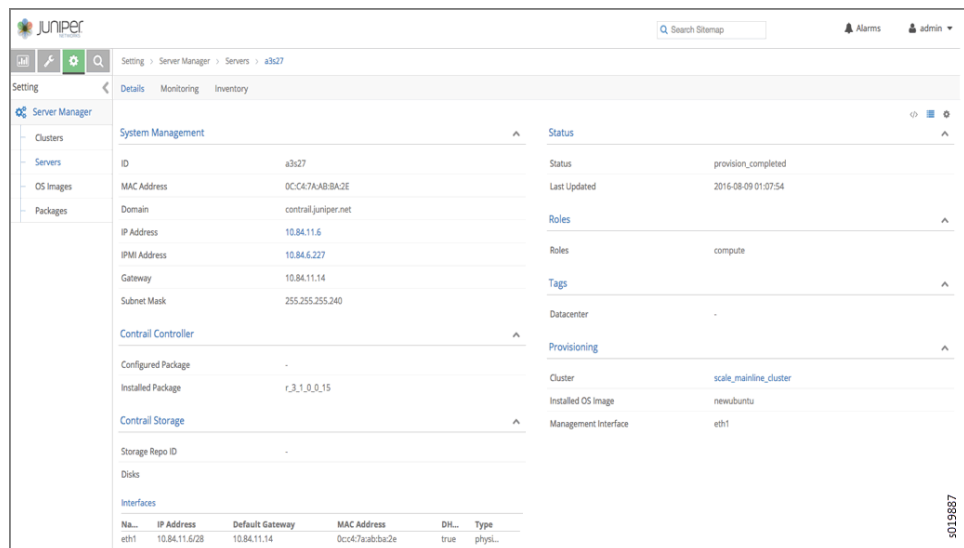
Figure 38: Filter Servers by Tag



Viewing Server Details

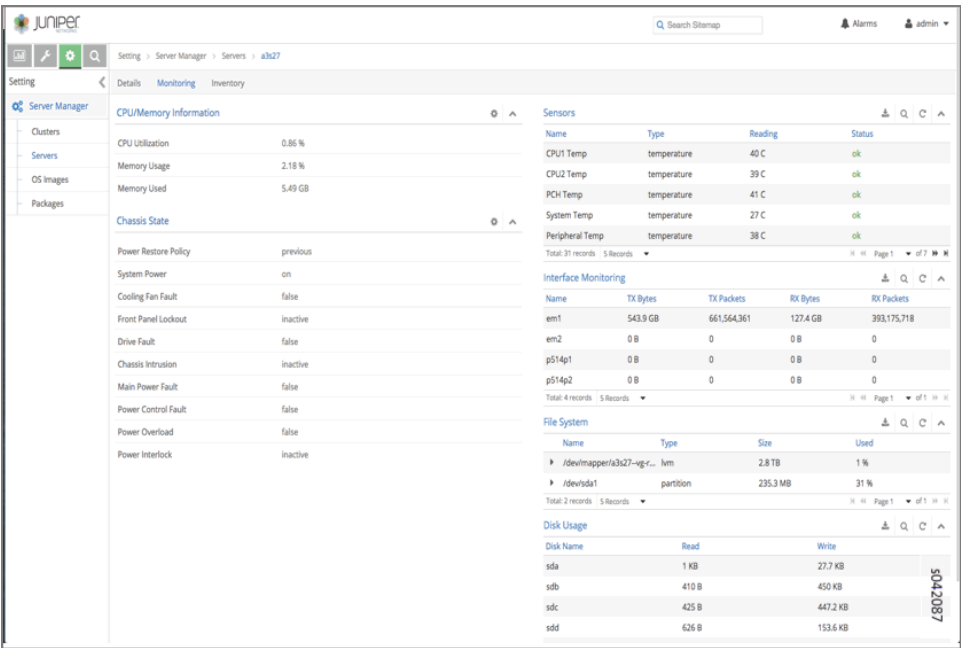
Each server name on the **Servers** page is a link to the details page for that server. Click any server name to open the details for that server, including **System Management** information, **Status**, **Contrail Controller**, **Contrail Storage**, **Roles**, **Tags**, and **Provisioning**, see Figure 39 on page 89.

Figure 39: View Server Details, System Management



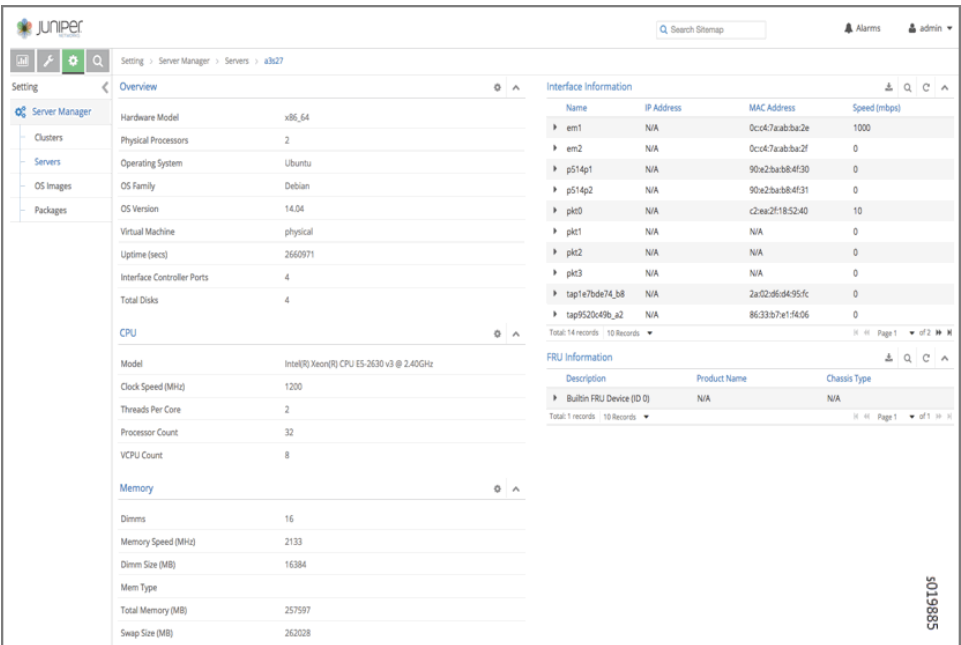
At the **Servers** page, click the **Monitoring** tab to see detailed information regarding **CPU/Memory Information**, **Chassis State**, **Sensors**, **Interface Monitoring**, **File System**, and **Disk Usage**, see Figure 40 on page 90.

Figure 40: Server Monitoring



At the **Servers** page, click the **Inventory** tab to see detailed information regarding **Overview** of the server, **Interface Information**, CPU information, Memory, and FRU Information, see [Figure 41 on page 90](#).

Figure 41: Server Inventory



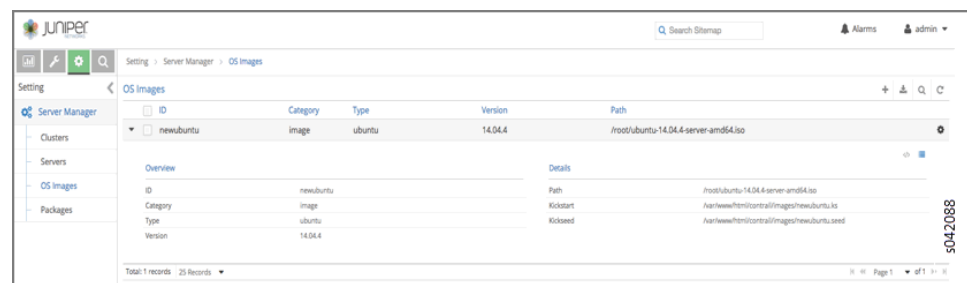
Configuring Images and Packages

Use the sidebar **Images and Packages** options to configure the software images and packages to be used by the Server Manager. Images are typically used to reimage clusters with an operating system version. Packages are used to provision clusters with a Contrail setup.

Both areas of the Server Manager user interface operate in a similar fashion. The figure shows the **Images** section. The **Packages** section has similar options.

Select **Images**. The Images page is displayed, see [Figure 42 on page 91](#).

Figure 42: Servers OS Images



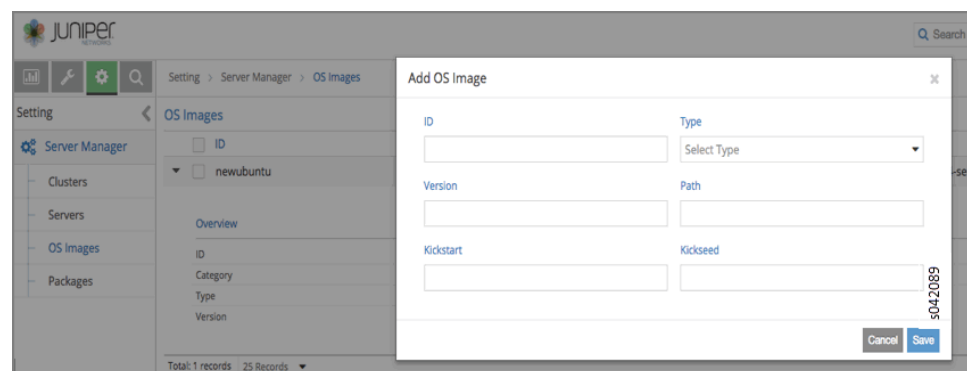
Add New Image or Package

To add a new image or package, on the respective **Images** or **Packages** page, click the plus (+) icon in the upper right header. The **Add Image** window is displayed. Enter the information for the new image (or package) and click **Save** to add the new item to the list of configured items, see [Figure 43 on page 91](#).



NOTE: The path field requires the path of the image where it is located on the server upon which the server-manager process is running.

Figure 43: Add OS Image



Selecting Server Manager Actions for Clusters

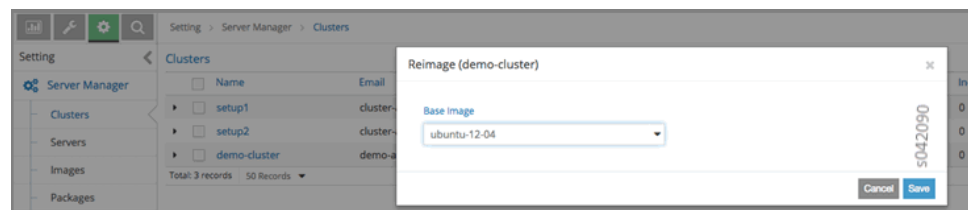
After all aspects of a cluster are configured, you can select actions for the Server Manager to perform on the cluster, such as **Reimage** or **Provision**.

Reimage a Cluster

Select **Setting > Servers > Clusters**. The **Clusters** window is displayed. Click the right side gear wheel icon of the cluster to be reimaged, then select **Reimage** from the action menu.

The **Reimage** dialog box is displayed, as shown. Verify that the correct image is selected in the **Default Image** field, then click **Save** to initiate the reimage action, see [Figure 44 on page 92](#).

Figure 44: Reimage Cluster

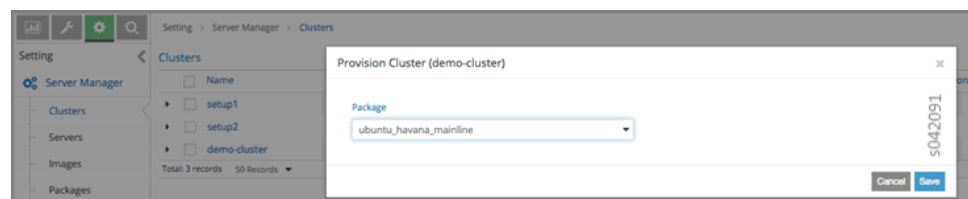


Provision a Cluster

The process to provision a cluster is similar to the process to reimage a cluster. Select **Setting > Servers > Clusters**. The **Clusters** window is displayed. Click the right side gear wheel icon of the cluster to be provisioned, then select **Provision** from the action menu.

The **Provision Cluster** dialog box is displayed, as shown. Verify that the correct package for provisioning is selected in the **Default Package** field, then click **Save** to initiate the provisioning action, see [Figure 45 on page 92](#).

Figure 45: Provision Cluster



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 reimage function.

SM-Lite supports the Server Manager functions of provisioning, monitoring, inventory, and WebUI. SM-Lite is intended to replace fab command provisioning. It allows easy deployment of Contrail provisioning and enables developers to work in isolated environments for Contrail provisioning.

SM-Lite eliminates installation and configuration of DHCP, DNS, and Cobbler services. Additionally, SM-Lite installation setup scripts are enhanced to reduce installation time.

SM-Lite provides a single command to install SM-Lite and provision a Contrail cluster.

SM-Lite introduces additional capabilities into Server Manager. The SM-Lite package is part of the Contrail Server Manager installer Debian package (**`contrail-server-manager-installer_<version string>.deb`**).

SM-Lite works with or without having a separate node for the SM-Lite installation, it can be installed on any Contrail node, but it is recommended to install it on the config node.

SM-Lite preserves the existing Server Manager WebUI functionality and it can be run on the same node as the Contrail WebUI. Because of that, the default port for the Server Manager WebUI has been changed to port **9080**.

It is important to note that the code base used for SM-Lite and Server Manager is common. Therefore, any changes or enhancements made to Server Manager provisioning functionality are automatically available in the SM-Lite software.

Installing Server Manager Lite

The SM-Lite package is included as part of the Server Manager installer package.

The installer package also has other packages such as Server Manager, Server Manager client, Server Manager WebUI, and Server Manager inventory. Before provisioning commands can be executed using SM-Lite, you need to install the Server Manager installer package.

Use the following command to install the Server Manager installer package.

```
dpkg -i <contrail-server-manager-installer-deb>
```

After the Server Manager installer package is installed, all necessary Server Manager packages, scripts, and so on are made available on the server where it is installed. You can then start using Server Manager Lite commands.

Provisioning Using SM-Lite with Contrail 4.0

For Contrail 4.0, to provision the target systems, use the `script`.

The **provision_containers.sh** script performs the following functions:

1. Installs SM-Lite.

Uses the **setup.sh** installation script with the **-smlite** option to install the SM-Lite package (**contrail-server-manager-lite_<version-sku>_all.deb**) and all other needed packages on the system.

2. Prepares the cluster for Contrail provisioning.

Translates the parameters in the **testbed.py** file into Server Manager objects and stores them in the Server Manager database. This specifies the servers in the cluster and the configuration parameters. The cluster-id value is used, if it is specified.

3. Performs a pre-check on the target systems to ensure that they are ready for running provisioning. SM-Lite uses from the Contrail package to provision the Contrail cluster.

4. This step issues provisioning commands for the cluster with the given Contrail package.

Server Manager Lite can be installed on any node. We recommend that you install it on the config node. Server Manager Lite can be installed on a separate node other than the Contrail cluster nodes.

The Server Manager WebUI default port is **9080**. You can change the port by editing the **/etc/contrail/config.global.sm.js** file, and then restarting the **supervisor-webui-sm** process.

Displaying the Cluster Status

The **server-manager cluster -detail** command displays the provisioning status of a cluster by role and by role progress.

Use the **server-manager status server** command to display the current status of the servers.

Displaying the SM-Lite Installation and Provisioning Log Files

Log files that provide information during installation and use of SM-Lite software are available at:

- **/var/log/contrail/install_logs/install_<timestamp>.log** (SM-Lite install)
- **/var/log/contrail/install_logs/provision_<timestamp>.log** (provisioning command logs)
- **testbed_parser.log** and **preconfig.log**

Contrail Provisioning Log Files

For each Puppet run, log files are automatically uploaded to the Server Manager at the following locations:

- **http:<sm-lite-ip-address>/logs**

- `/var/log/contrail_server_manager/<target>/<timestamp>.log`
- `/var/log/contrail/*`

You can also display the status of the processes and services using the **contrail-status** command.

**Related
Documentation**

- [Using Server Manager to Automate Provisioning on page 46](#)
- [Using the Server Manager Web User Interface on page 74](#)

CHAPTER 5

Installing and Using Contrail Storage

- [Installing and Using Contrail Storage on page 97](#)

Installing and Using Contrail Storage

- [Overview of the Contrail Storage Solution on page 97](#)
- [Basic Storage Functionality with Contrail on page 98](#)
- [Ceph Block and Object Storage Functionality on page 98](#)
- [Using the Contrail Storage User Interface on page 98](#)
- [Hardware Specifications on page 100](#)
- [Software Files for Compute Storage Nodes on page 100](#)
- [Contrail OpenStack Nova Modifications on page 100](#)
- [Installing the Contrail Storage Solution on page 101](#)
- [Using Fabric Commands to Install and Configure Storage on page 101](#)
- [Fabric Installation Procedure on page 102](#)
- [Using Server Manager to Install and Configure Storage on page 103](#)
- [Server Manager Installation Procedure for Storage on page 104](#)
- [Example: Configurations for Storage for Reimaging and Provisioning a Server on page 105](#)
- [Storage Installation Limits on page 106](#)

Overview of the Contrail Storage Solution

Contrail provides a storage support solution using OpenStack Cinder configured to work with Ceph. Ceph is a unified, distributed storage system whose infrastructure provides storage services to Contrail.

The Contrail storage solution has the following features:

- Provides storage class features to Contrail clusters, including replication, reliability, and robustness.
- Uses open source components.
- Uses Ceph block and object storage functionality.
- Integrates with OpenStack Cinder functionality.

- Does not require virtual machines (VMs) to configure mirrors for replication.
- Allows nodes to provide both compute and storage services.
- Provides easy installation of basic storage functionality based on Contrail roles.
- Provides 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.
- Provides native live-migration support if the VM is booted with Ceph storage as its root volume.
- Provides object storage support SwiftS3 APIs.

Basic Storage Functionality with Contrail

The following are basic interaction points between Contrail and the storage solution.

- Cinder volumes must be manually configured prior to installing the Contrail storage solution. The Cinder volumes can be attached to virtual machines (VMs) to provide additional storage.
- The storage solution stores virtual machine boot images and snapshots in Glance, using Ceph object storage functionality.
- All storage nodes can be monitored through a graphical user interface (GUI).
- It is possible to migrate virtual machines that have ephemeral storage in Ceph.

Ceph Block and Object Storage Functionality

Installing the Contrail storage solution creates the following Ceph configurations.

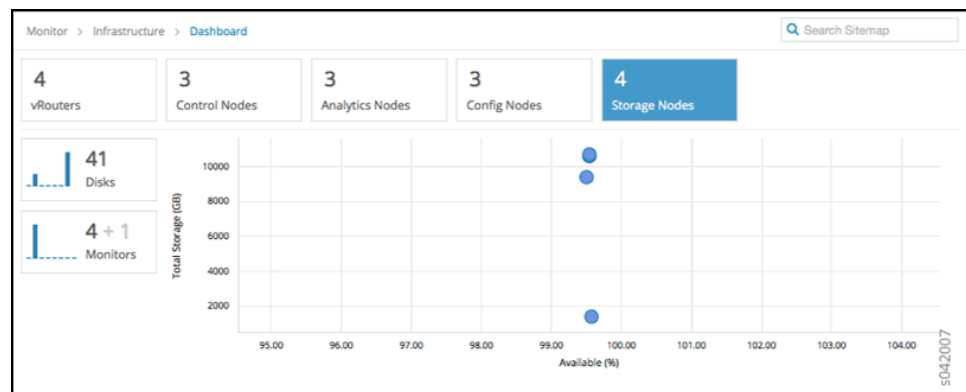
- Each disk is configured as a standalone storage device, enhancing optimal performance and creating proper failure boundaries. Ceph allocates and assigns a process called object storage daemon (OSD) to each disk.
- A replication factor of 2 is configured, consisting of one original instance plus one replica copy. Ceph ensures that each replica is on a different storage node.
-
- 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.

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.

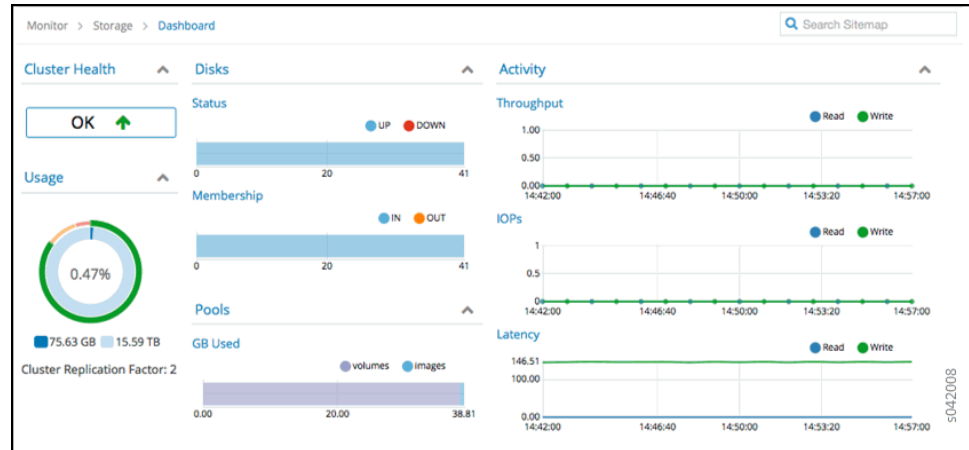
Select **Monitor > Infrastructure > Dashboard** to display an at-a-glance view of the system infrastructure components, including the numbers of virtual routers, control nodes, analytics nodes, config nodes, and storage nodes currently operational, and a bubble chart of storage nodes showing the Available (%) 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.



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: 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.
 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. live_migration = True

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 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

-
- 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:
{
  "a": "AQBDCCdpTsB5FChAA0zI2++uosfmtj7tjmhPu0g==",
  "osd_bootstrap_key":
  "AQBKCDpTmN+HGRAA16rmStq5iYoPnANzSXLcXA==",
  "b": "AQBLCdpTu0S6FhAAFDW0SsdzyDAUeuwOr/h61A=="
}
```

- e. `"live_migration": "enable",`

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

1. Define storage in the cluster. The following example configuration 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 2.10 and later:

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 2.10 and later:

Server.json :

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

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.

CHAPTER 6

Upgrading Contrail Software

- [Upgrading Contrail 3.2 to 4.0 on page 107](#)
- [Dynamic Kernel Module Support \(DKMS\) for vRouter on page 110](#)

Upgrading Contrail 3.2 to 4.0

Contrail Release 4.0 presents a number of differences from earlier versions of Contrail, the most notable is that many Contrail services are now run within containers.

This section provides the process for upgrading an existing Contrail Release 3.2 system to Contrail Release 4.0. You can perform an upgrade by using a convenient script, or by performing the upgrade manually. Both procedures are included in this topic.

- [Overview of Changes in Contrail Release 4.0 on page 107](#)
- [Using a Server Manager Script to Upgrade from Contrail 3.2 to 4.0 on page 108](#)
- [Steps for Upgrading Contrail 3.2 to 4.0 \(Without using Server-Manager for upgrade\) on page 108](#)

Overview of Changes in Contrail Release 4.0

For previous releases of Contrail, through Contrail Release 3.2.x, Contrail could be provisioned by using fab commands or by using the Contrail Server Manager. Starting with Contrail Release 4.0, many Contrail services have been containerized, and provisioning can only be accomplished by using the Contrail Server Manager.

Additionally, the use of fab commands is no longer supported in Contrail Release 4.0.

You can perform the upgrade by using the steps presented here, or by using a script available in Server Manager that automates the steps.

Significant differences between Contrail 3.2 and 4.0 include:

- A number of Contrail services are run in containers. Container default names in Contrail Release 4.0 include:
 - contrail-controller
 - contrail-analytics

- contrail-analyticsdb
- contrail-lb (load-balancer)

For more information about Contrail containers, see [“Introduction to Containerized Contrail Modules” on page 11](#).

Assumptions

The upgrade procedure assumes that RabbitMQ server and Neutron server are running on the OpenStack node by default.

Using a Server Manager Script to Upgrade from Contrail 3.2 to 4.0

A script is available in Server Manager that runs the steps provided in *Steps for Upgrading Contrail 3.2 to 4.0*.

The script location is: `/opt/contrail/server_manager/inplace_upgrade.py`.

Information regarding using the script:

- The script uses the **openstack-config** utility, you can install the contrail-setup package to get the utility.
- The timeout for cluster provisioning completion is 3600 seconds, you can increase the timeout for a cluster with a large number of nodes.

The default timeout parameter in the script:

PROV_TIMEOUT = 3600

- The script execution log is saved at `/var/log/contrail/inplace_upgrade.log`.

Running the Upgrade Script

1. Add the image, cluster, and servers to Server Manager running Contrail Release 4.0 code.
2. Run the script.

```
user@node:/opt/contrail/server_manager# python inplace_upgrade.py -h
usage: inplace_upgrade.py [-h] cluster_name image_name

positional arguments:
  cluster_name  cluster to be upgraded.The cluster and servers should be
                existing in SM
  image_name    version to upgrade to

optional arguments:
  -h, --help    show this help message and exit
```

Steps for Upgrading Contrail 3.2 to 4.0 (Without using Server-Manager for upgrade)

This procedure lists the steps needed for upgrading Contrail 3.2 to 4.0.

This procedure is an in-service upgrade from a running 3.2 system that will temporarily run in parallel with the new 4.0 system.

The services and roles referred to in this procedure could be running in separate nodes. Run the procedure commands in the nodes that are running the relevant roles or services.

1. Stop the listed services.

- supervisor-analytics
- supervisor-support-service
- supervisor-database
- contrail-database
- supervisor-webui
- supervisor-config
- supervisor-control
- haproxy (all nodes)
- redis-server (analytics/webui node)
- memcached (config nodes)
- neutron-server (config nodes)
- zookeeper (config nodes)

Example stop commands include:

service supervisor-analytics stop

service supervisor-support-service stop

2. Stop the epmd process. The epmd process is a RabbitMQ service that is usually running on a config node.

3. Use a **status** command to verify that none of the services that were stopped in Step 1 are still running.

Example status commands include:

service supervisor-analytics status

service supervisor-support-service status

4. Prepare the cluster.json and server.json, as required for Server Manager provisioning for Contrail Release 4.0, see [“Installing Containerized Contrail Clusters Using Server Manager” on page 17](#).

5. Provision the new 4.0 cluster and wait for provisioning to be completed.

6. Check the status of provisioning.

```
server-manager status server --cluster_id <cluster-id>
```

7. To prevent port conflict with Cassandra between the two versions of Contrail, on the 3.2 config node(s), edit the Cassandra configuration file at `/etc/cassandra/cassandra.yaml` to change the listening port from 9160 to 29160.

When finished, start the Cassandra service.

```
rpc_port: 29160
```

```
service cassandra start
```

8. Identify the new (4.0) and old (3.2) IP address lists for Cassandra.
 - a. Connect to the controller container.
 - b. Edit `/usr/lib/python2.7/dist-packages/contrail_issu/issu_contrail_config.py` and add the old Cassandra list and the new Cassandra list:

```
'old_cassandra_address_list': '192.xxx.xxx.102:29160',
```

```
'new_cassandra_address_list': '10.xx.5.xxx:9161',
```

It is sufficient to provide a single ip:port of the Contrail 3.2 Cassandra node.

9. Run a Cassandra sync between the old and new config nodes, to copy the cassandra config keyspaces from the release 3.2 nodes to the Contrail 4.0 Cassandra database.

```
contrail-issu-pre-sync
```

Wait for the command to complete.

10. Shut down the Cassandra service in the 3.2 controller.
11. The upgrade of the compute nodes includes a reboot of the compute nodes. The reboot turns off the guest virtual machine instances. To restart the virtual machine instances, log in to the Openstack node and restart the VMs:

```
nova start <instance-uuid>
```

Dynamic Kernel Module Support (DKMS) for vRouter

Dynamic Kernel Module Support (DKMS) is a framework provided by Linux to automatically build out-of-tree driver modules for Linux kernels whenever the Linux distribution upgrades the existing kernel to a newer version.

In Contrail, the vRouter kernel module is an out-of-tree, high performance packet forwarding module that provides advanced packet forwarding functionality in a reliable and stable manner. Contrail provides a DKMS-compatible source package for Ubuntu so that if you deploy an Ubuntu-based Contrail system you do not need to manually compile the kernel module each time the Linux deployment gets upgraded.

The **contrail-vrouter-dkms** package provides the DKMS compatibility for Contrail. Prior to installing the **contrail-vrouter-dkms** package, you must install both the DKMS package

and the **contrail-vrouter-utils** package, because the **contrail-vrouter-dkms** package is dependent on both. Installing the **contrail-vrouter-dkms** package adds the vRouter sources to the DKMS database, builds the vRouter module, and installs it in the existing kernel modules tree. When a kernel upgrade occurs, DKMS ensures that the module is compiled for the newer kernel and installed in the proper location so that upon reboot, the newer module can be used with the upgraded kernel.

For more information about DKMS, refer to:

- DKMS Ubuntu documentation at <https://help.ubuntu.com/community/DKMS>
- DKMS Ubuntu manual pages at <http://manpages.ubuntu.com/manpages/lucid/man8/dkms.8.html>
- Linux Journal article on DKMS at <http://www.linuxjournal.com/article/6896>

PART 3

Configuring Contrail

- [Configuring Virtual Networks on page 115](#)
- [Example of Deploying a Multi-Tier Web Application Using Contrail on page 143](#)
- [Configuring Services on page 155](#)
- [Configuring Service Chaining on page 173](#)
- [Examples: Configuring Service Chaining on page 209](#)

CHAPTER 7

Configuring Virtual Networks

- [Creating Projects in OpenStack for Configuring Tenants in Contrail on page 116](#)
- [Creating a Virtual Network with Juniper Networks Contrail on page 118](#)
- [Creating a Virtual Network with OpenStack Contrail on page 121](#)
- [Creating an Image for a Project in OpenStack Contrail on page 123](#)
- [Creating a Floating IP Address Pool on page 126](#)
- [Using Security Groups with Virtual Machines \(Instances\) on page 127](#)
- [Support for IPv6 Networks in Contrail on page 130](#)
- [Configuring EVPN and VXLAN on page 133](#)

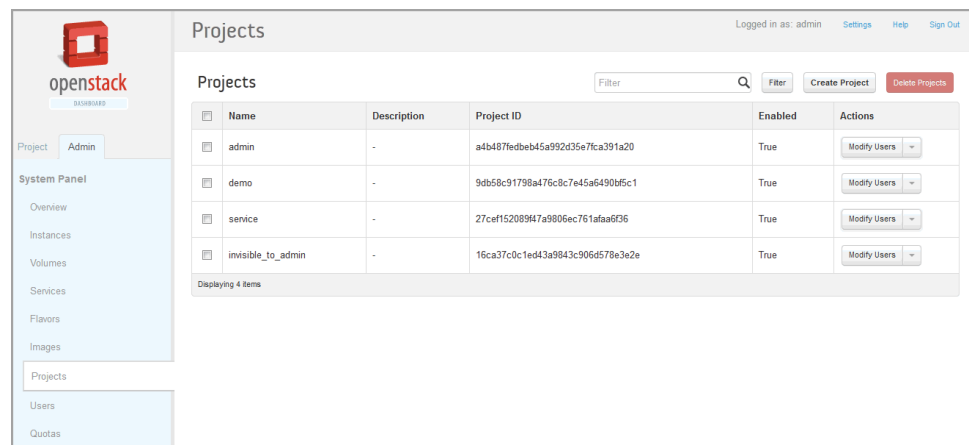
Creating Projects in OpenStack for Configuring Tenants in Contrail

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 46 on page 116](#).

Figure 46: OpenStack Projects



2. In the upper right, click the **Create Project** button to access the **Add Project** window; see [Figure 47 on page 116](#).

Figure 47: 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 a Virtual Network with Juniper Networks Contrail on page 118](#)
- [Creating a Virtual Network with OpenStack Contrail on page 121](#)
- [OpenStack documentation](#)

Creating a Virtual Network with Juniper Networks Contrail

Contrail makes creating a virtual network very easy for a self-service user. You create networks and network policies at the user dashboard, then associate policies with each network. The following procedure shows how to create a virtual network when using Juniper Networks Contrail.

1. You need to create an IP address management (IPAM) for your project for to create a virtual network. Select **Configure > Networking > IP Address Management**, then click the **Create** button.

The **Add IP Address Management** window appears, see [Figure 48 on page 118](#).

Figure 48: Add IP Address Management

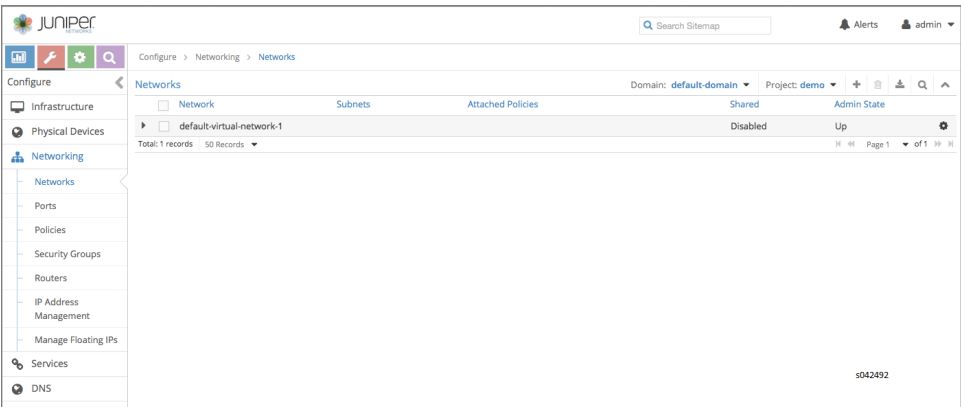
2. Complete the fields in **Add IP Address Management**: The fields are described in [Table 25 on page 118](#).

Table 25: Add IP Address Management Fields

Field	Description
Name	Enter a name for the IPAM you are creating.
DNS Method	Select from a list the domain name server method for this IPAM: 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** page; see [Figure 49 on page 119](#).

Figure 49: Configure Networks




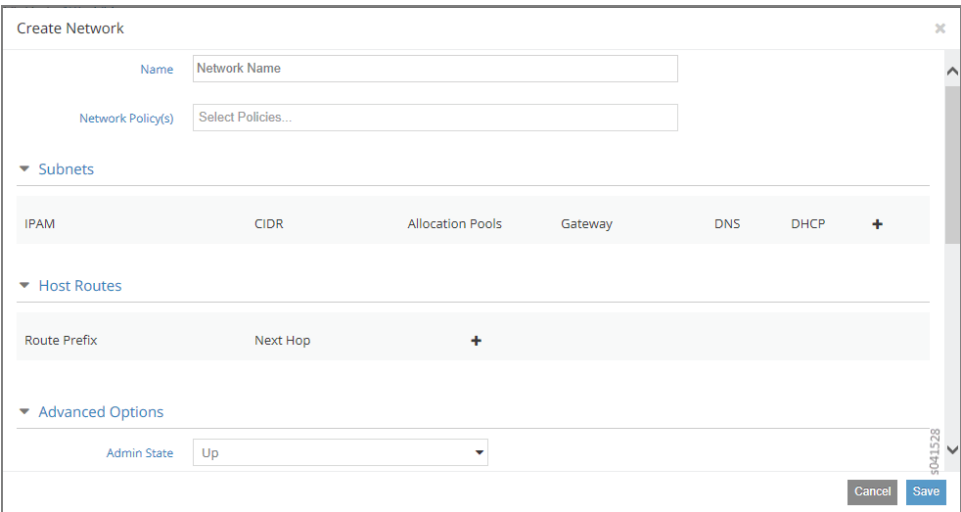
4. Verify that your project is displayed as active in the upper-right field, then click the  icon. The **Create Network** window is displayed. See [Figure 50 on page 119](#). Use the scroll bar to access all sections of this window.

Figure 50: Create Network



5. Complete the fields in the **Create Network** window with values that identify the network name, network policy, and IP options as needed. See field descriptions in [Table 26 on page 119](#).

Table 26: Create Network Fields

Field	Description
Name	Enter a name for the virtual network you are creating.
Network Policy	Select the policy to be applied to this network from the list of available policies. You can select more than one policy by clicking each one needed.

Table 26: 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. Alternatively, 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	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—Juniper Networks Contrail*.

- Related Documentation**
- [Creating an Image for a Project in OpenStack Contrail on page 123](#)
 - *Launching a Virtual Machine (Instance)*
 - *Creating a Network Policy—Juniper Networks Contrail*
 - *Deleting a Virtual Network—Juniper Networks Contrail*

Creating a Virtual Network with OpenStack Contrail

Contrail makes creating a virtual network very easy for you. You create networks and network policies at the user dashboard, then associate policies with each network. The following procedure shows how to create a virtual network when using OpenStack.

1. To create a virtual network when using OpenStack Contrail, select **Project > Other > Networking**. The **Networks** window is displayed. See [Figure 51 on page 121](#).

Figure 51: Networks Window

Network	Summary	Status	Actions
default-virtual-network	0 instances in 0 IP Blocks, 0 available IP addresses	Up	Edit IP Blocks

Displaying 1 item

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 52 on page 121](#) and [Figure 53 on page 122](#).

Figure 52: Create Network Window

Create Network

Network * Subnet * Subnet Detail Associate Network Policies >

Network Name *

☒ Admin State

From here you can create a new network. In addition a subnet associated with the network can be created in the next panel and Policies can be associated to this network

Cancel Create

Figure 53: 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 27 on page 122](#).

Table 27: 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 27: 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.

Related Documentation

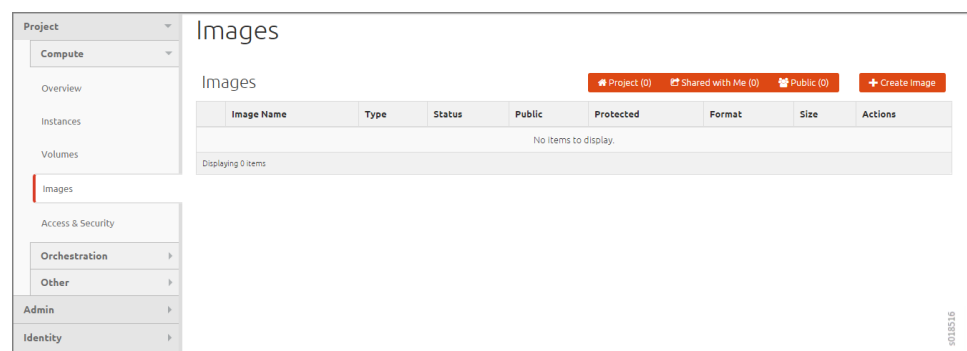
- *Deleting a Virtual Network—OpenStack Contrail*

Creating an Image for a Project in OpenStack Contrail

To specify an image to upload to the Image Service for a project in your system by using the OpenStack dashboard:

- In OpenStack, select **Project > Compute > Images**. The Images window is displayed. See [Figure 54 on page 123](#).

Figure 54: OpenStack Images Window



- Make sure you have selected the correct project to which you are associating an image.
- Click **Create Image**.

The **Create An Image** window is displayed. See [Figure 55 on page 124](#).

Figure 55: 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 28 on page 124](#) describes each of the fields on the window.



NOTE: Only images available through an HTTP URL are supported, and the image location must be accessible to the Image Service. Compressed image binaries are supported (*zip and *tar.gz).

Table 28: Create an Image Fields

Field	Description
Name	Enter a name for this image.

Table 28: 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– An unstructured image format VDI– Virtual Disk Image VHD– Virtual Hard Disk VMDK– Virtual Machine Disk</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	Select this check box if this is a public image. Leave unselected for a private image.
Protected	Select this check box for a protected image.

- When you are finished, click **Create Image**.

Related Documentation

- *Launching a Virtual Machine (Instance)*

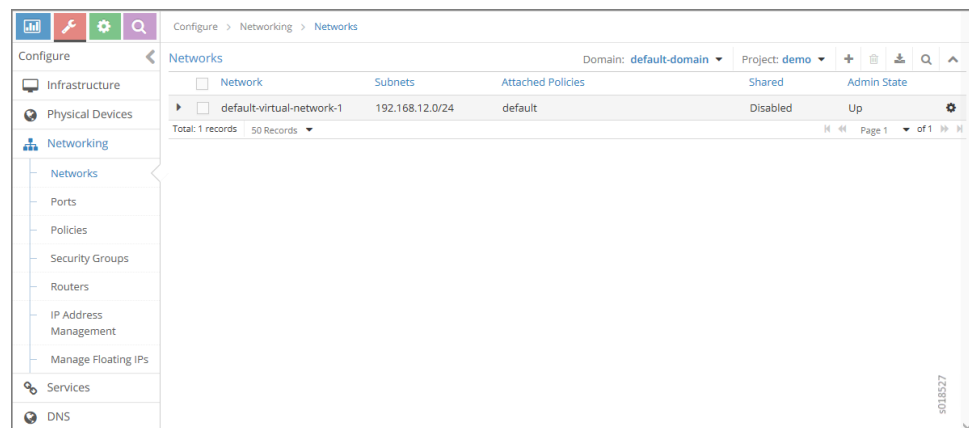
Creating a Floating IP Address Pool

A floating IP address is an IP address (typically public) that can be dynamically assigned to a running virtual instance.

To configure floating IP address pools in project networks in Contrail, then allocate floating IP addresses from the pool to virtual machine instances in other virtual networks:

1. Select **Configure > Networking > Networks**; see [Figure 56 on page 126](#). Make sure your project is the active project in the upper right.

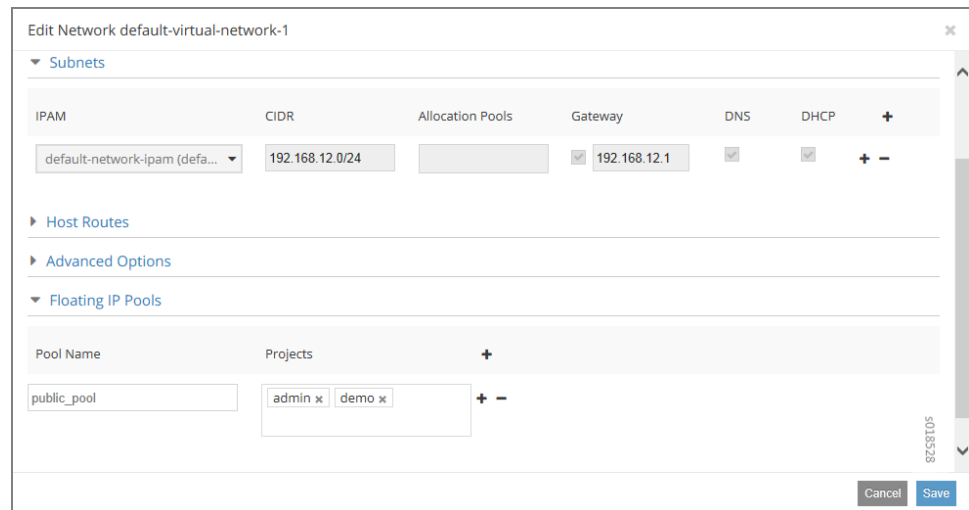
Figure 56: Configure > Networking > Networks



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 57 on page 126](#).

Figure 57: Edit Network



3. In the **Floating IP Pools** section, click the **Pool Name** field, enter a name for your floating IP pool, and click the + (plus sign) to add the IP pool to the table below the field.
 - Multiple floating IP pools can be created at the same time.
 - A floating IP pool can be associated with multiple projects.
4. Click **Save** to create the floating IP address pool, or click **Cancel** to remove your work and start over.

Using Security Groups with Virtual Machines (Instances)

- [Security Groups Overview on page 127](#)
- [Creating Security Groups and Adding Rules on page 127](#)

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

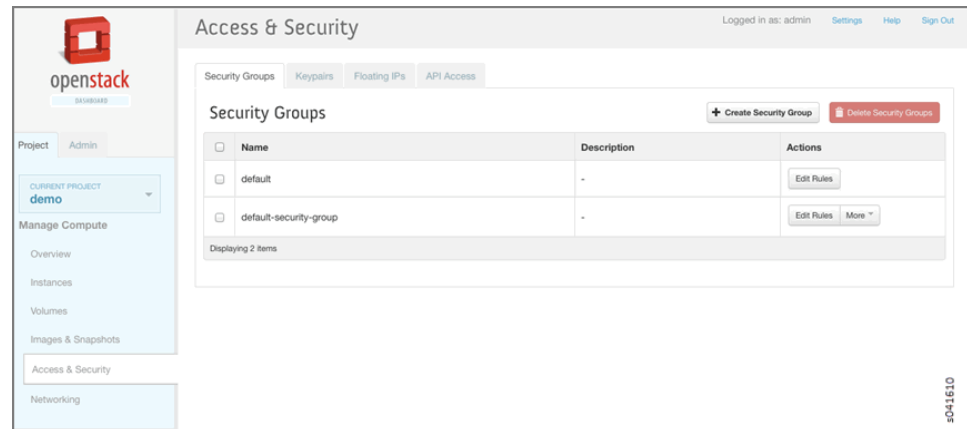
A default security group is created for each project. You can add security rules to the default security group and you can create additional security groups and add rules to them. The security groups are then associated with a VM, when the VM is launched or at a later date.

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 58 on page 128](#).

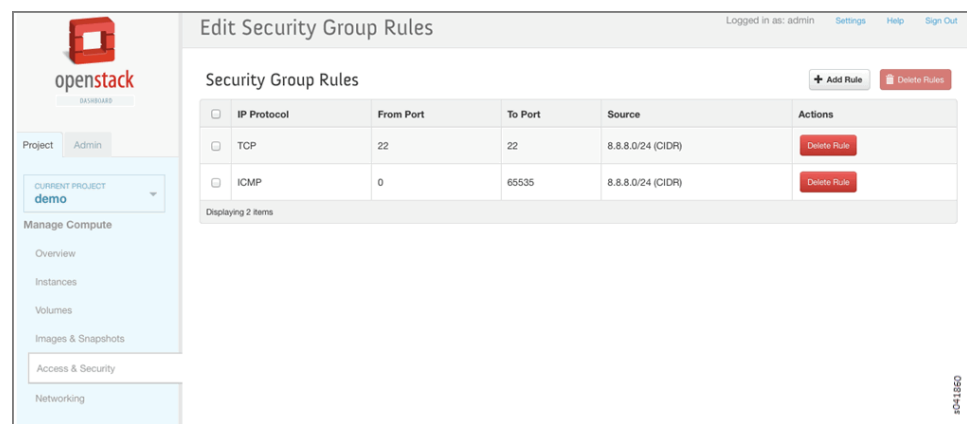
Figure 58: Security Groups



2. Select the **default-security-group** and click **Edit Rules** in the **Actions** column.

The **Edit Security Group Rules** window is displayed; see [Figure 59 on page 128](#). Any rules already associated with the security group are listed.

Figure 59: Edit Security Group Rules



3. Click **Add Rule** to add a new rule; see [Figure 60 on page 129](#).

Figure 60: Add Rule

Add Rule

IP Protocol

ICMP

Type

0

Code

0

Source

CIDR

CIDR

Security Group

0.0.0.0/0

Description:

Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

Protocol: You must specify the desired IP protocol to which this rule will apply; the options are TCP, UDP, or ICMP.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Source: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Cancel

Add

Table 29: 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 61 on page 130](#).

Each new security group has a unique 32-bit security group ID and an ACL is associated with the configured rules.

Figure 61: Create Security Group

- When an instance is launched, there is an opportunity to associate a security group; see [Figure 62 on page 130](#).

In the **Security Groups** list, select the security group name to associate with the instance.

Figure 62: Associate Security Group at Launch Instance

- 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

Starting with Contrail Release 2.0, support for IPv6 overlay networks is provided.

- [Overview: IPv6 Networks in Contrail on page 131](#)
- [Creating IPv6 Virtual Networks in Contrail on page 131](#)
- [Adding IPv6 Peers on page 133](#)

Overview: IPv6 Networks in Contrail

The following features are supported for IPv6 networks and overlay. The underlay network must be IPv4.

- Virtual machines with IPv6 and IPv4 interfaces
- Virtual machines with IPv6-only interfaces
- DHCPv6 and neighbor discovery
- Policy and Security groups
- IPv6 flow set up, tear down, and aging
- Flow set up and tear down based on TCP state machine
- Protocol-based flow aging
- Fat flow
- Allowed address pair configuration with IPv6 addresses
- IPv6 service chaining
- Equal Cost Multi-Path (ECMP)
- Connectivity with gateway (MX Series device)
- Virtual Domain Name Services (vDNS), name-to-IPv6 address resolution
- User-Visible Entities (UVEs)

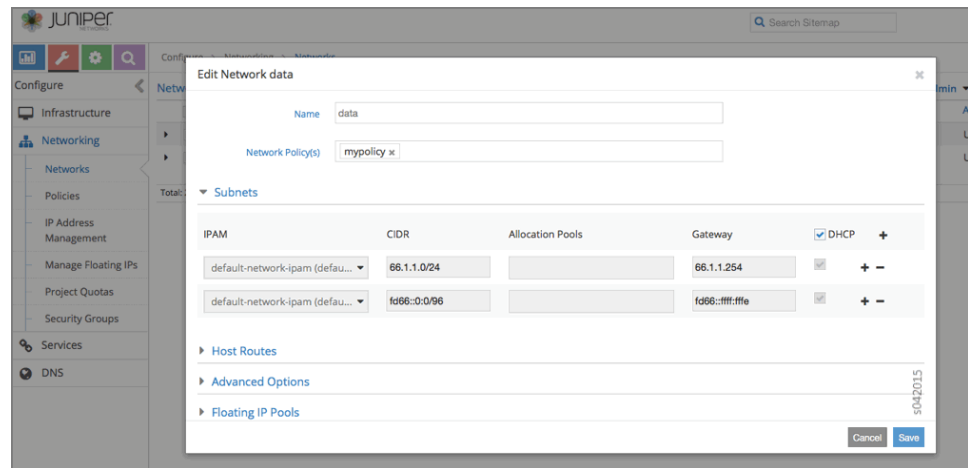
NOT present is support for the following:

- Source Network Address Translation (SNAT)
- Load Balancing as a Service (LBaaS)
- IPv6 fragmentation
- Floating IP
- Link-local and metadata services
- Diagnostics for IPv6
- Contrail Device Manager
- Virtual customer premises equipment (vCPE)

Creating IPv6 Virtual Networks in Contrail

You can create an IPv6 virtual network from the Contrail user interface in the same way you create an IPv4 virtual network. When you create a new virtual network by selecting

Configure > Networking > Networks, the Edit fields accept IPv6 addresses, as shown in the following image.



Address Assignments

When virtual machines are launched with an IPv6 virtual network created in the Contrail user interface, the virtual machine interfaces get assigned addresses from all the families configured in the virtual network.

The following is a sample of IPv6 instances with address assignments, as listed in the OpenStack Horizon user interface.

Instances									
Instances									
Instance Name	Image Name	IP Address	Size	Keypair	Status	Task	Power State	Uptime	Actions
Test-6b4281-ada9-415c-8609-bcd89578e03	ubuntu-jd4f	data 66.1.1.251 fd66::ffff:ffff 76.1.1.252	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	Create Snapshot More
Test-7a3b7f0b-e5a8-48a9-9346-29079e1ab0ba	ubuntu-jd4f	data 66.1.1.250 fd66::ffff:ffff 76.1.1.250	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	Create Snapshot More
Test-663309b7-1785-4cc4-9ecd-19025ec04ee5	ubuntu-jd4f	data 66.1.1.245 fd66::ffff:ffff 76.1.1.244	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	Create Snapshot More
Test-a20a46c7-5cd5-447e-8894-c794aa820ab	ubuntu-jd4f	data 66.1.1.252 fd66::ffff:ffff 76.1.1.251	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	Create Snapshot More
Test-43343608-455f-47a5-9346-5a81f5b2197	ubuntu-jd4f	data 66.1.1.247 fd66::ffff:ffff 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:

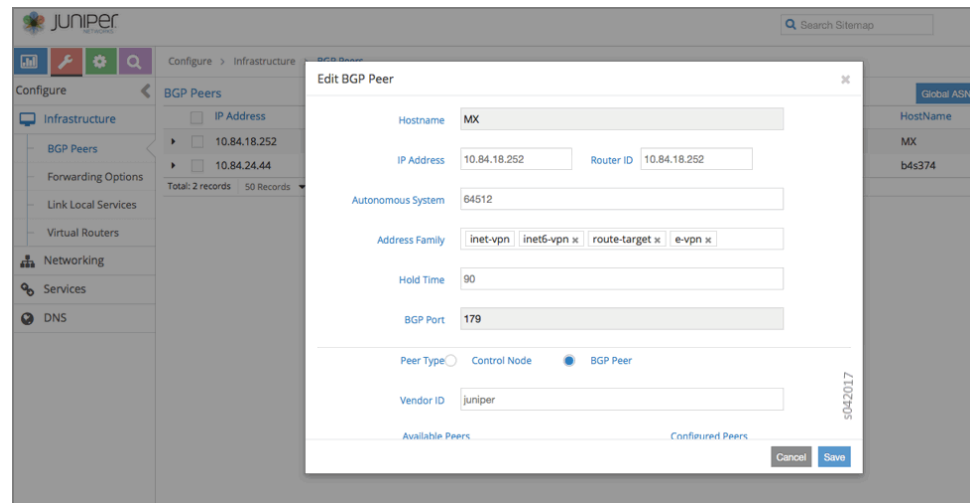
```
iface eht0 inet6 dhcp
```


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 135](#)
- [Configuring Forwarding on page 137](#)
- [Configuring the VXLAN Identifier on page 138](#)
- [Configuring Encapsulation Methods on page 139](#)

Configuring the VXLAN Identifier Mode

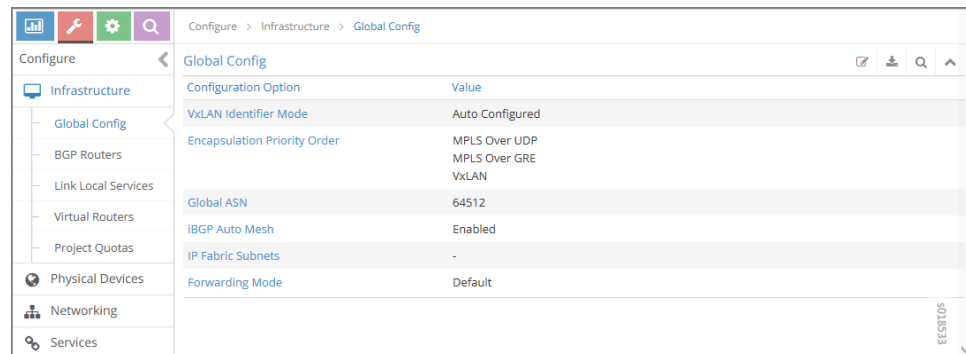
You can configure the global VXLAN identifier mode to select an auto-generated VNID or a user-generated VXLAN ID, either through the Contrail Web UI or by modifying a python file.

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.

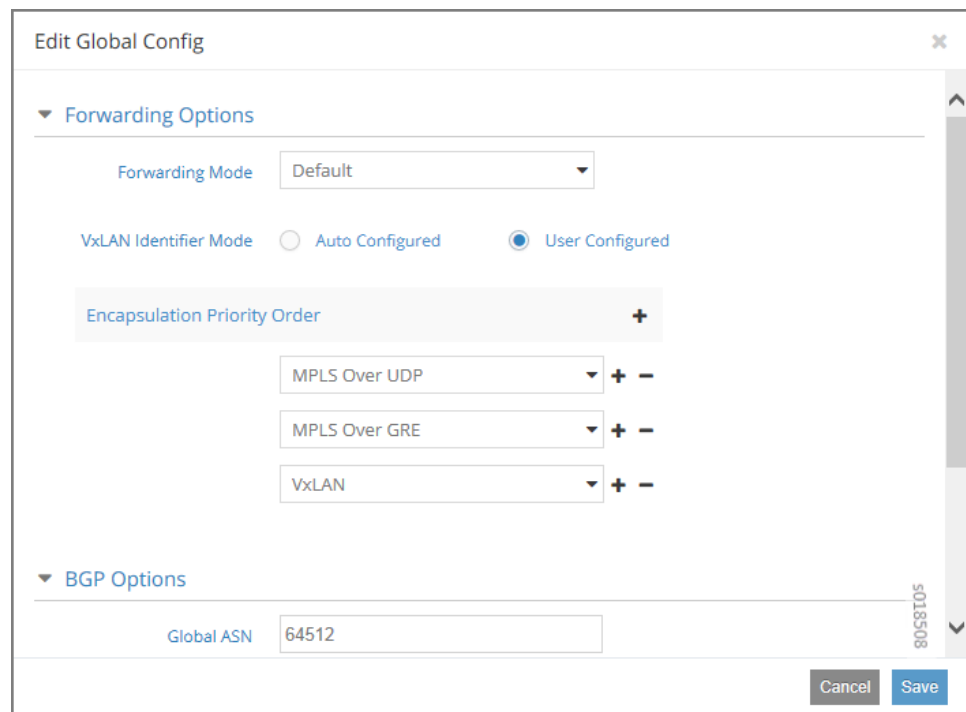
Figure 63: Global Config Window for VXLAN ID



2. Click the edit icon .

The Edit Global Config window is displayed as shown in [Figure 64 on page 136](#).

Figure 64: Edit Global Config Window for VXLAN Identifier Mode



3. Select one of the following:
 - **Auto Configured**— The VXLAN identifier is automatically assigned for the virtual network.
 - **User Configured**— You must provide the VXLAN identifier for the virtual network.



NOTE: When **User Configured** is selected, if you do not provide an identifier, then VXLAN encapsulation *is not used* and the mode falls back to MPLS.


Alternatively, you can set the VXLAN identifier mode by using Python to modify the `/opt/contrail/utils/encap.py` file as follows:

```
python encap.py <add | update | delete> <username> < password> < tenant_name> <
config_node_ip>
```

Configuring Forwarding

In Contrail, the default forwarding mode is enabled for fallback bridging (IP FIB and MAC FIB). The mode can be changed, either through the Contrail Web UI or by using python provisioning commands.

To change the forwarding mode:

1. From the Contrail Web UI, select **Configure > Networking > Networks**.
2. Select the virtual network that you want to change the forwarding mode for.
3. Click the gear icon  and select **Edit**.

The Edit Network window is displayed as shown in [Figure 65 on page 137](#).

Figure 65: Edit Network Window

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"/>

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.

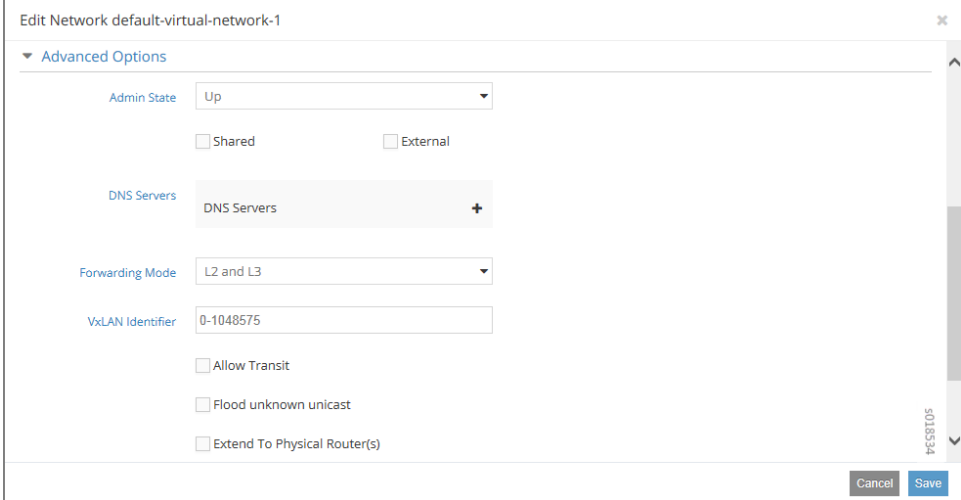
To configure the global VXLAN identifier:

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 66 on page 139](#).

Figure 66: Edit Network Window for VXLAN Identifier



4. Type the VXLAN identifier.
5. Click **Save**.

Alternatively, you can use the following Python provisioning command to configure the VXLAN identifier:

```
python provisioning_forwarding_mode --project_fq_name 'defaultdomain: admin' --vn_name vn1 --forwarding_mode < vxlan_id >
```


Configuring Encapsulation Methods

The default encapsulation mode for EVPN is MPLS over UDP. All packets on the fabric are encapsulated with the label allocated for the virtual machine interface. The label encoding and decoding is the same as for IP forwarding. Additional encapsulation methods supported for EVPN include MPLS over GRE and VXLAN. MPLS over UDP is different from MPLS over GRE only in the method of tunnel header encapsulation.

VXLAN has its own header and uses a VNID label to carry the traffic over the fabric. A VNID is assigned with every virtual network and is shared by all virtual machines in the virtual network. The VNID is mapped to the VRF of the virtual network to which it belongs.

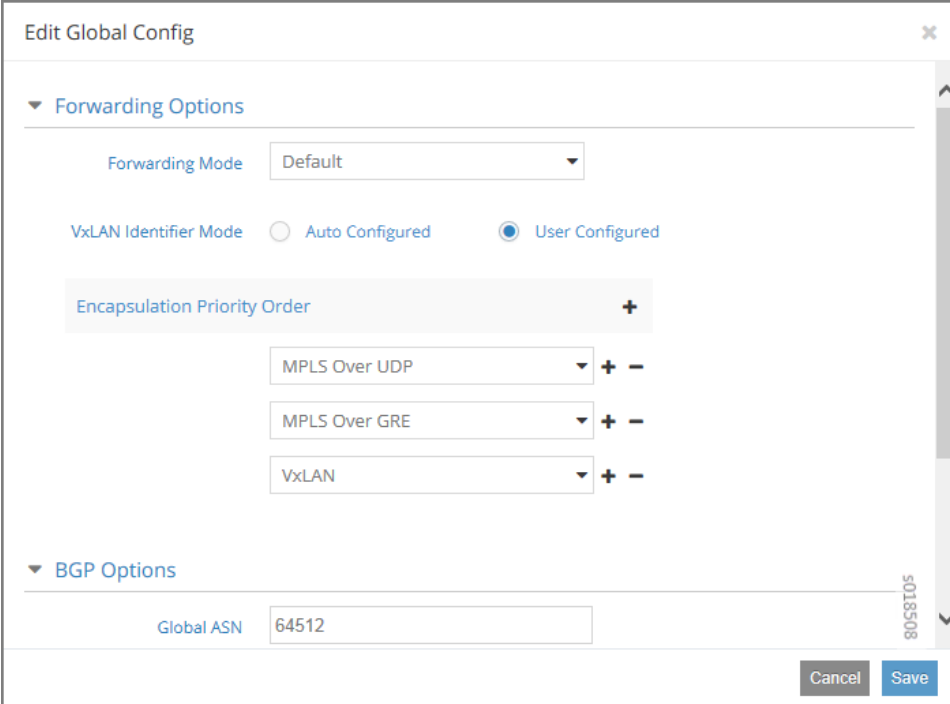
The priority order in which to apply encapsulation methods is determined by the sequence of methods set either from the Contrail Web UI or in the **encap.py** file.

To configure the global VXLAN identifier mode:

- From the Contrail Web UI, select **Configure > Infrastructure > Global Config**.
- The Global Config options are displayed.
- Click the edit icon  .

The Edit Global Config window is displayed as shown in [Figure 67 on page 140](#).

Figure 67: Edit Global Config Window for Encapsulation Priority Order



The screenshot shows the 'Edit Global Config' window. It has a title bar with a close button. The main content is divided into two sections: 'Forwarding Options' and 'BGP Options'. Under 'Forwarding Options', there is a 'Forwarding Mode' dropdown set to 'Default'. Below that, 'VxLAN Identifier Mode' has two radio buttons: 'Auto Configured' (unselected) and 'User Configured' (selected). Under 'Encapsulation Priority Order', there is a list of three items: 'MPLS Over UDP', 'MPLS Over GRE', and 'VxLAN'. Each item has a dropdown arrow and a '+ -' button to its right. Under 'BGP Options', there is a 'Global ASN' field with the value '64512'. At the bottom right, there are 'Cancel' and 'Save' buttons. A vertical scrollbar is on the right side of the window.

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.

CHAPTER 8

Example of Deploying a Multi-Tier Web Application Using Contrail

- [Example: Deploying a Multi-Tier Web Application on page 143](#)
- [Sample Network Configuration for Devices for Simple Tiered Web Application on page 149](#)

Example: Deploying a Multi-Tier Web Application

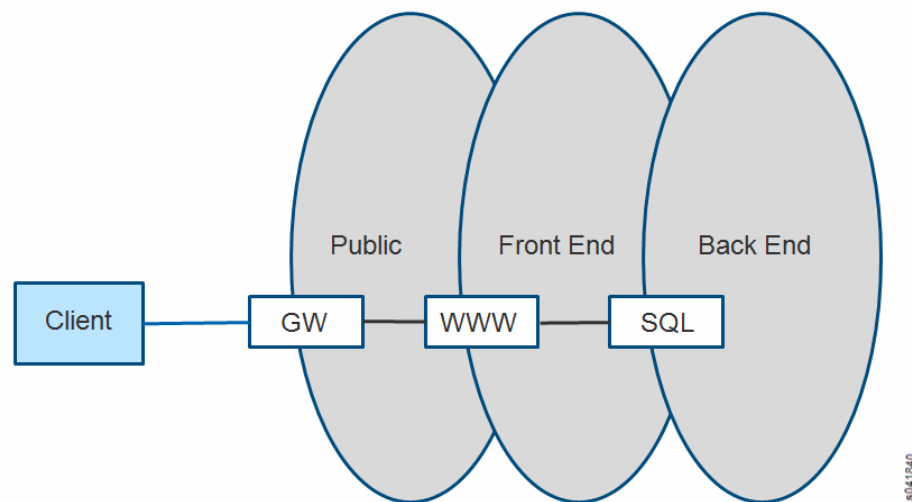
- [Multi-Tier Web Application Overview on page 143](#)
- [Example: Setting Up Virtual Networks for a Simple Tiered Web Application on page 144](#)
- [Verifying the Multi-Tier Web Application on page 146](#)
- [Sample Addressing Scheme for Simple Tiered Web Application on page 147](#)
- [Sample Physical Topology for Simple Tiered Web Application on page 148](#)
- [Sample Physical Topology Addressing on page 148](#)

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 68 on page 144](#).

Figure 68: 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 116](#).

2. In the **demo** project, create three virtual networks:

- a. A network named **public** with IP address **10.84.41.0/24**

This is a special use virtual network for floating IP addresses— it is assigned an address block from the public floating address pool that is assigned to each web server. The assigned block is the only address block advertised outside of the data center to clients that want to reach the web services provided.

- b. A network named **frontend** with IP address **192.168.1.0/24**

This network is the location where the web server virtual machine instances are launched and attached. The virtual machines are identified with private addresses that have been assigned to this virtual network.

- c. A network named **backend** with IP address **192.168.2.0/24**

This network is the location where the database server virtual machines instances are launched and attached. The virtual machines are identified with private addresses that have been assigned to this virtual network.

For more information; see “Creating a Virtual Network with OpenStack Contrail” on page 121 or “Creating a Virtual Network with Juniper Networks Contrail” on page 118.

3. Create a floating IP pool named **public_pool** for the **public** network within the **demo** project; see Figure 69 on page 145.

Figure 69: Create Floating IP Pool

Edit Network public

Network Name: public

Network Policy(s): Select Policies...

Address Management: default-network... xxx.xxx.xxx.xxx/xx + -

IPAM	IP Block
default-network-ipam	10.84.41.0/24

Floating IP Pools: public_pool demo x + -

Pool Name: admin

Cancel Save

4. Allocate the floating IP pool **public_pool** to the **demo** project; see Figure 70 on page 145.

Figure 70: Allocate Floating IP

Allocate Floating IP

Floating IP Pool: public:public_pool

Cancel Save

5. Verify that the floating IP pool has been allocated; see **Configure > Networking > Allocate Floating IPs**.

For more information; see [“Creating a Floating IP Address Pool”](#) on page 126 and [Allocating a Floating IP Address to a Virtual Machine](#).

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—Juniper Networks Contrail*, *Associating a Network to a Policy—Juniper Networks Contrail*, or *Creating a Network Policy—OpenStack Contrail*, and *Associating a Network to a Policy—OpenStack Contrail*.

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)*.

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 30 on page 147](#) as a guide for addressing devices in the simple tiered web example.

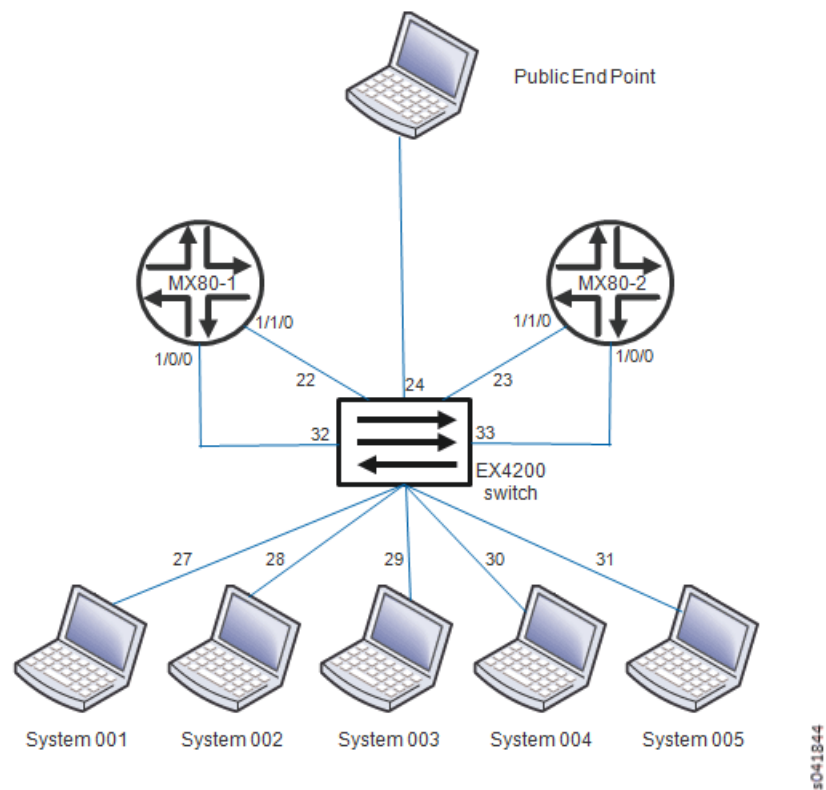
Table 30: Sample Addressing Scheme for Example

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 71 on page 148 provides a guideline diagram for the physical topology for the simple tiered web application example.

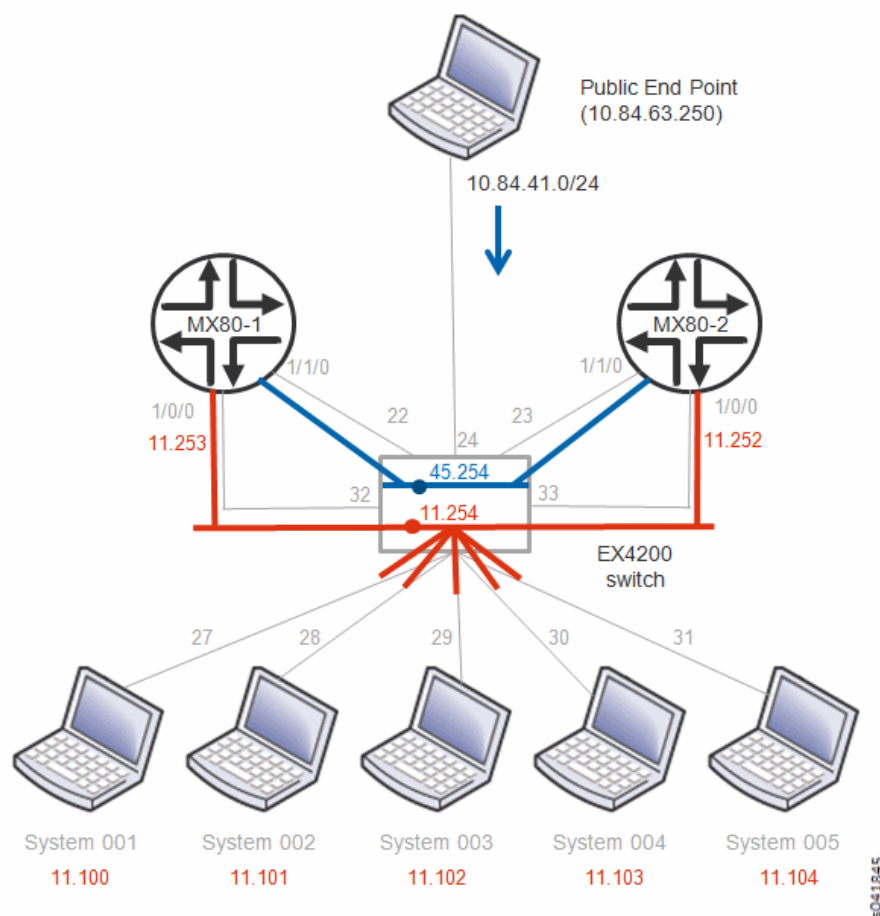
Figure 71: Sample Physical Topology for Simple Tiered Web Application



Sample Physical Topology Addressing

Figure 72 on page 149 provides a guideline diagram for addressing the physical topology for the simple tiered web application example.

Figure 72: Sample Physical Topology Addressing



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 143. 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 user1 {
      uid 2002;
      class read;
      authentication {
        encrypted-password "xxxxxxxxxxxx"; ## SECRET-DATA
      }
    }
  }
}
services {
  ssh {
    root-login allow;
  }
  telnet;
}

```

```
netconf {
  ssh;
}
web-management {
  http;
}
}
syslog {
  user * {
    any emergency;
  }
  file messages {
    any notice;
    authorization info;
  }
  file interactive-commands {
    interactive-commands any;
  }
}
}
chassis {
  aggregated-devices {
    ethernet {
      device-count 64;
    }
  }
}
}
```

CHAPTER 9

Configuring Services

- [Configuring DNS Servers on page 155](#)
- [Support for Multicast on page 164](#)
- [Using Static Routes with Services on page 166](#)
- [Configuring Metadata Service on page 170](#)

Configuring DNS Servers

- [DNS Overview on page 155](#)
- [Defining Multiple Virtual Domain Name Servers on page 156](#)
- [IPAM and Virtual DNS on page 156](#)
- [DNS Record Types on page 157](#)
- [Configuring DNS Using the Interface on page 158](#)
- [Configuring DNS Using Scripts on page 163](#)

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:

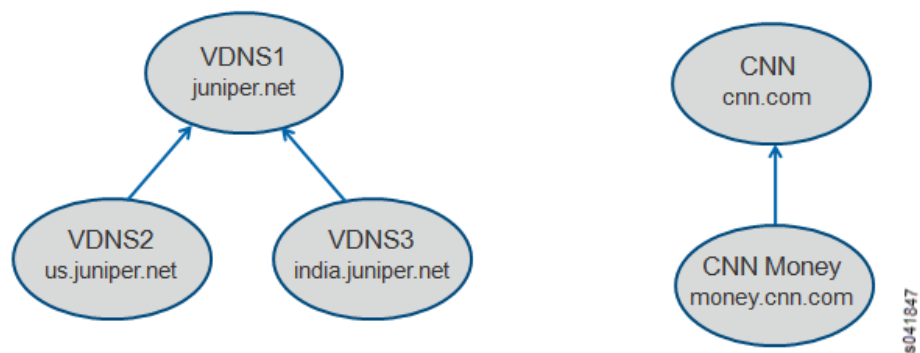
- It should be possible to configure multiple domain name servers to provide name resolution service for the virtual machines spawned in the system.
- It should be possible to configure the domain name servers to form DNS server hierarchies required by each tenant.

- 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 73 on page 156](#) shows examples of virtual DNS servers defined in **default-domain**, providing the name service for the DNS domains indicated.

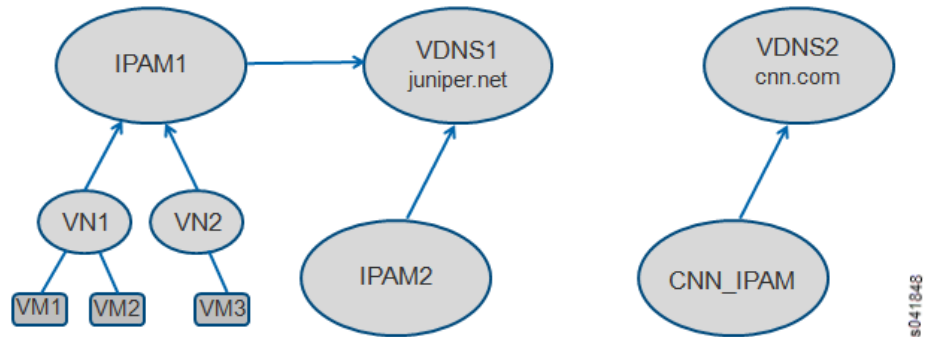
Figure 73: DNS Servers Examples



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 74 on page 157](#)

Figure 74: IPAM and Virtual DNS



DNS Record Types

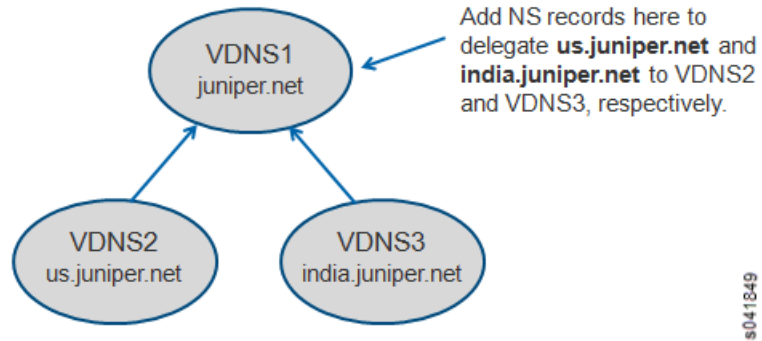
DNS records can be added statically. DNS record types **A**, **CNAME**, **PTR**, and **NS** are currently supported in the system. Each record includes the type, class (IN), name, data, and TTL values. See [Table 31 on page 157](#) for descriptions of the record types.

Table 31: 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 75 on page 158](#) shows an example usage for the DNS record type of **NS**.

Figure 75: 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 76 on page 158](#).

Figure 76: 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 77 on page 159](#)

Figure 77: Add DNS

Complete the fields for the new server; see [Table 32 on page 159](#).

Table 32: 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.

- 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 78 on page 160](#).

Figure 78: Add DNS Record

4. Complete the fields for the new record; see [Table 33 on page 160](#).

Table 33: 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 79 on page 161](#).

Figure 79: Associate IPAMs to DNS

Complete the IPAM associations, using the field descriptions in [Table 34 on page 161](#).

Table 34: 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 80 on page 162](#)) 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 80: Configure IP Address Management

7. To associate an IPAM to a virtual DNS server or to tenant's IP addresses, at the **IP Address Management** page, select the network associated with this IPAM, then click the **Action** button in the last column, and click **Edit**.

The **Edit IP Address Management** window appears; see [Figure 81 on page 162](#).

Figure 81: DNS Server

8. In the first field, select the **DNS Method** from a list (**None**, **Default DNS**, **Tenant DNS**, **Virtual DNS**; see [Table 35 on page 162](#)).

Table 35: 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 35: 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 36 on page 163](#).



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 36: 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 36: 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>

Support for Multicast

This section describes how the Contrail Controller supports broadcast and multicast.

- [Subnet Broadcast on page 164](#)
- [All-Broadcast/Limited-Broadcast and Link-Local Multicast on page 165](#)
- [Host Broadcast on page 166](#)

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.

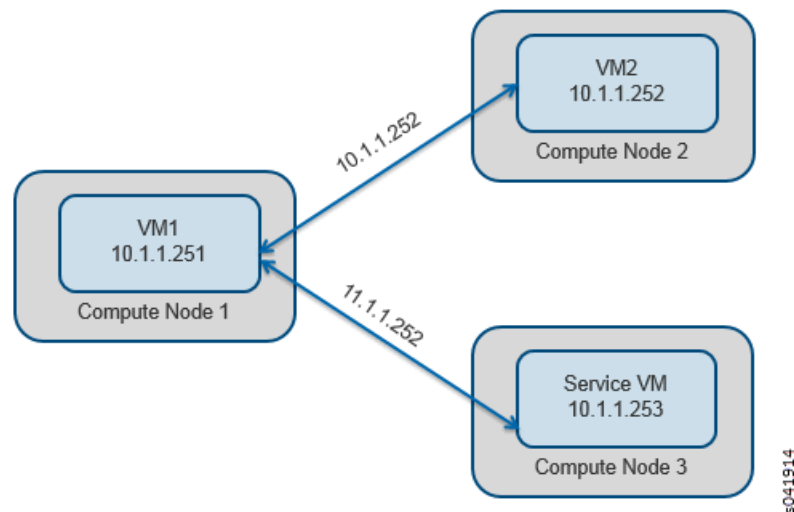
Using Static Routes with Services

- [Static Routes for Service Instances on page 166](#)
- [Configuring Static Routes on a Service Instance on page 167](#)
- [Configuring Static Routes on Service Instance Interfaces on page 168](#)
- [Configuring Static Routes as Host Routes on page 169](#)

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

In-Network

Image Name

nat-service

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

041915

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.

Create Service Instances

Instance Name: nat

Services Template: nat - [in-network (management, left, right)]

Interface 1: Management | Auto Configured

Interface 2: Left | vn1

▼ Static Routes

Prefix	Next hop	
11.1.1.0/24	Interface 2	+ -

Interface 3: Right | vn2

▼ Static Routes

Cancel Save

s041916

The following figure shows static route 10.1.1.0/24 configured on the right interface, so that all traffic from VN2 that is destined to VN1 reaches the right interface of the service virtual machine.

The screenshot shows the 'Create Service Instances' dialog box with two tabs: 'Interface 2' and 'Interface 3'. The 'Interface 2' tab is active, showing a dropdown for 'Left' and 'vn1'. Below it, the 'Static Routes' section has a table with columns 'Prefix', 'Next hop', and a '+' button. A row is added with '11.1.1.0/24' in the Prefix column and 'Interface 2' in the Next hop column. The 'Interface 3' tab is also visible, showing a dropdown for 'Right' and 'vn2'. Its 'Static Routes' section has a table with columns 'Prefix', 'Next hop', and a '+' button. A row is added with '10.1.1.0/24' in the Prefix column and 'Interface 3' in the Next hop column. At the bottom right, there are 'Cancel' and 'Save' buttons. A vertical scrollbar on the right side shows the value '041917'.

When the static routes are configured for both the left and the right interfaces, all inter-virtual network traffic is forwarded through the service instance.

Configuring Static Routes as Host Routes

You can also use static routes for host routes for a virtual machine, by using the classless static routes option in the DHCP server response that is sent to the virtual machine.

The routes to be sent in the DHCP response to the virtual machine can be configured for each virtual network as it is created.

To configure static routes as host routes:

1. Go to **Configure > Network > Networks** and click **Create**.
2. At **Create Network**, click the **Host Routes** option and add the host routes to be sent to the virtual machines.

An example is shown in the following figure.

Create Network

Address Management: ipam1 IP Block: 1.2.3.0/24 Gateway: 1.2.3.254

IPAM	IP Block	Gateway
ipam1	1.2.3.0/24	1.2.3.254

Route Targets

Floating IP Pools

Host Routes

IPAM	Route Prefix
ipam1	1.1.1.0/24
ipam1	2.2.2.0/24

Cancel Save

Configuring Metadata Service

OpenStack enables virtual machines to access metadata by sending an HTTP request to the link-local address 169.254.169.254. The metadata request from the virtual machine is proxied to Nova with additional HTTP header fields that Nova uses to identify the source instance, then responds with appropriate metadata.

In Contrail, the vRouter acts as the proxy, by trapping the metadata requests, adding the necessary header fields, and sending the requests to the Nova API server.

The metadata service is configured by setting the **linklocal-services** property on the **global-vrouter-config** object.

Use the following elements to configure the **linklocal-services** element for metadata service:

- **linklocal-service-name** = metadata
- **linklocal-service-ip** = 169.254.169.254
- **linklocal-service-port** = 80
- **ip-fabric-service-ip** = [server-ip-address]
- **ip-fabric-service-port** = [server-port]

The `linklocal-services` properties can be set from the Contrail UI (**Configure > Infrastructure > Link Local Services**) or by using the following command:

```
python /opt/contrail/utils/provision_linklocal.py --admin_user <user> --admin_password  
<passwd> --linklocal_service_name metadata --linklocal_service_ip 169.254.169.254  
--linklocal_service_port 80 --ipfabric_service_ip --ipfabric_service_port 8775
```


CHAPTER 10

Configuring Service Chaining

- [Service Chaining on page 173](#)
- [Service Chaining MX Series Configuration on page 177](#)
- [ECMP Load Balancing in the Service Chain on page 178](#)
- [Customized Hash Field Selection for ECMP Load Balancing on page 179](#)
- [Using the Contrail Heat Template on page 183](#)
- [Service Chain Route Reorigination on page 187](#)
- [Service Instance Health Check on page 205](#)

Service Chaining

Contrail Controller supports chaining of various Layer 2 through Layer 7 services such as firewall, NAT, IDP, and so on.

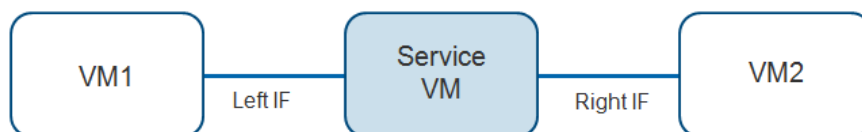
- [Service Chaining Basics on page 173](#)
- [Service Chaining Configuration Elements on page 174](#)

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.

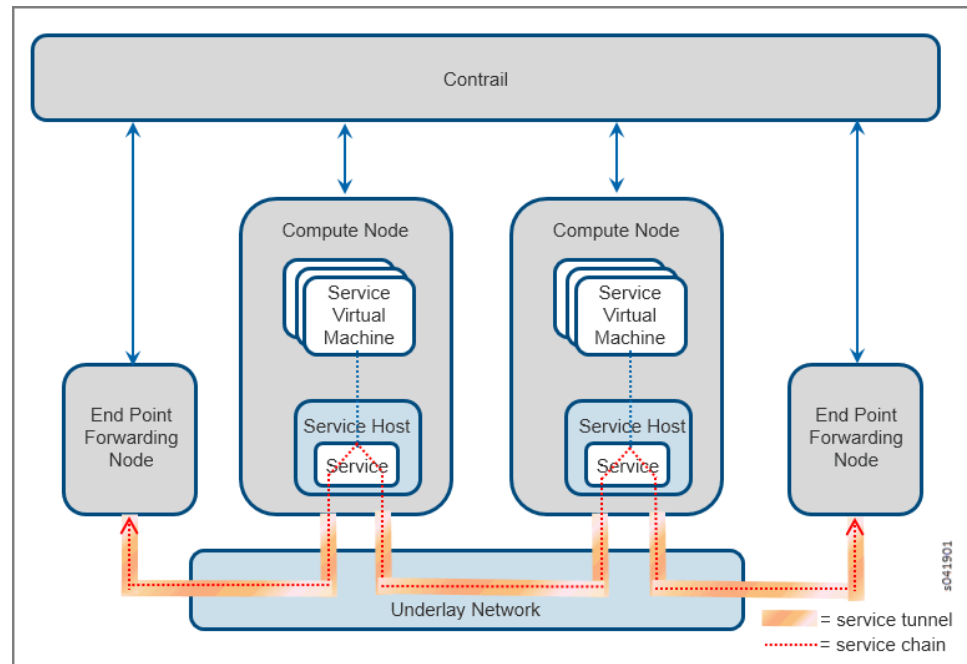
[Figure 82 on page 173](#) shows the basic service chain schema, with a single service. The service VM spawns the service, using the convention of left interface (left IF) and right interface (right IF). Multiple services can also be chained together.

Figure 82: 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 83 on page 174](#) 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 83: 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 209](#)
- [Example: Creating a Service Chain With the CLI on page 221](#)
- [ECMP Load Balancing in the Service Chain on page 178](#)

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;
}
}

```

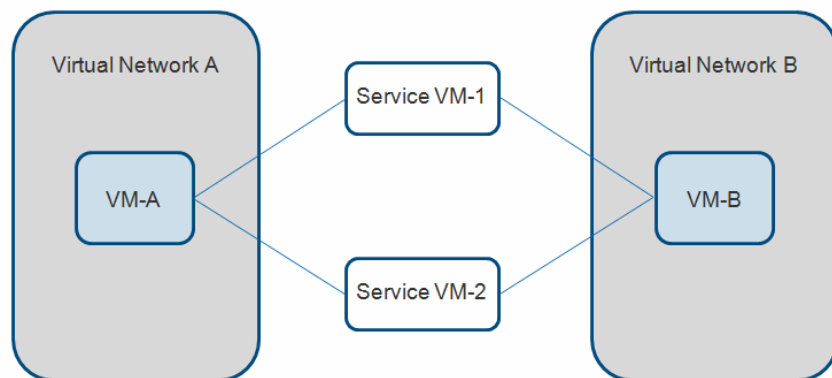
- The final step is to add **target:100:10000** to the public virtual network and **target:100:20000** to the enterprise virtual network, using the Contrail Juniper Networks interface.

A full MX Series router configuration for Contrail can be seen in [“Sample Network Configuration for Devices for Simple Tiered Web Application”](#) on page 149.

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 84 on page 178](#), 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 84: 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 173](#)
- [Example: Creating a Service Chain With the CLI on page 221](#)
- [Customized Hash Field Selection for ECMP Load Balancing on page 179](#)

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.

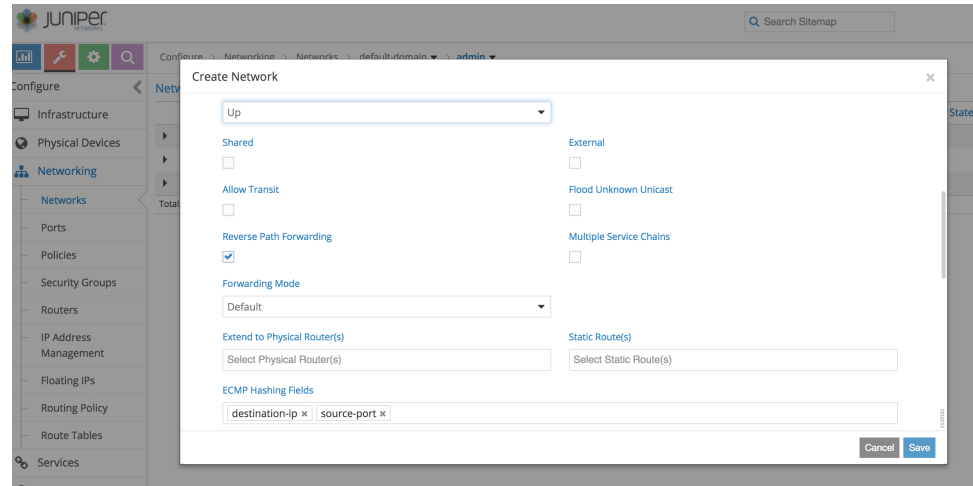
The custom hash configuration can be applied in the following ways:

- 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).

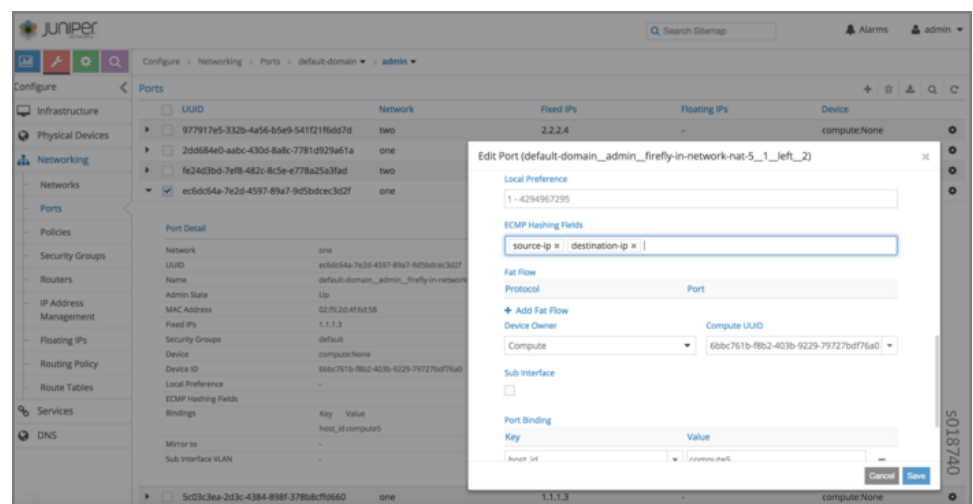
Custom hash is useful whenever packets originating from a particular source and addressed to a particular destination must go through the same set of service instances during transit. This might be required if source, destination, or transit nodes maintain a certain state based on the flow, and the state 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 183](#)
- [Heat Architecture on page 184](#)
- [Support for Heat Version 2 Resources on page 184](#)
- [Heat Version 2 with Service Chaining and Port Tuple Sample Workflow on page 185](#)
- [Example: Creating a Service Template Using Heat on page 185](#)

Introduction to Heat

A Heat template describes the infrastructure for a cloud application, such as networks, servers, floating IP addresses, and the like, and can be used to manage the entire life cycle of that application.

When the application infrastructure changes, the Heat templates can be modified to automatically reflect those changes. Heat can also delete all application resources if the system is finished with an application.

Heat templates can record the relationships between resources, for example, which networks are connected by means of policy enforcements, and consequently call OpenStack REST APIs that create the necessary infrastructure, in the correct order, needed to launch the application managed by the Heat template.

Heat Architecture

Heat is implemented by means of Python applications, including the following:

- **heat-client**—The CLI tool that communicates with the **heat-api** application to run Heat APIs.
- **heat-api**—Provides an OpenStack native REST API that processes API requests by sending them to the Heat engine over remote procedure calls (RPCs).
- **heat-engine**—Responsible for orchestrating the launch of templates and providing events back to the API consumer.

Support for Heat Version 2 Resources

Starting with Contrail Release 3.0.2, Contrail Heat resources and templates are autogenerated from the Contrail schema, using Heat Version 2 resources. Contrail Release 3.0.2 is the minimum required version for using Heat with Contrail in 3.x releases. The Contrail Heat Version 2 resources are of the following hierarchy:

OS::ContrailV2::<ResourceName>.

The generated resources and templates are part of the Contrail Python package, and are located in the following directory in the target installation:

/usr/lib/python2.7/dist-packages/vnc_api/gen/heat/

The **heat/** directory has the following subdirectories:

- **resources/**—Contains all the resources for the contrail-heat plugin, which runs in the context of the Heat engine service.
- **templates/**—Contains sample templates for each resource. Each sample template presents every possible parameter in the schema. Use the sample templates as a reference when you build up more complex templates for your network design.
- **env/**—Contains the environment for input to each template.

The following contains a list of all the generated plug-in resources that are supported by **contrail-heat** in Contrail Release 3.0.2 and greater:

<https://github.com/Juniper/contrail-heat/tree/master/generated/resources>

The following contains a list of new example templates:

https://github.com/Juniper/contrail-heat/tree/master/contrail_heat/new_templates

Deprecation of Heat Version 1 Resources

Heat Version 1 resources within the hierarchy `OS::Contrail::<ResourceName>` are being deprecated, and you should not create new service chains using the Heat Version 1 templates.

Heat Version 2 with Service Chaining and Port Tuple Sample Workflow

With Contrail service templates Version 2, the user can create ports and bind them to a virtual machine (VM)-based service instance, by means of a port-tuple object. All objects created with the Version 2 service template are directly visible to the Contrail Heat engine, and are directly managed by Heat.

The following shows the basic workflow steps for creating a port tuple and service instance that will be managed by Heat:

1. Create a service template. Select 2 in the Version field.
2. Create a service instance for the service template just created.
3. Create a port-tuple object.
4. Create ports, using Nova VM launch or without a VM launch.
5. Label each port as left, right, mgmt, and so on, and add the ports to the port-tuple object.

Use a unique label for each of the ports in a single port tuple. The labels named left and right are used for forwarding.

6. Link the port tuple to a service instance.
7. Launch the service instance.

Example: Creating a Service Template Using Heat

The following is an example of how to create a service template using Heat.

1. Define a template to create the service template.

```
service_template.yaml
heat_template_version: 2013-05-23
description: >
  HOT template to create a service template
parameters:
  name:
    type: string
    description: Name of service template
  mode:
    type: string
    description: service mode
```

```
    type:
      type: string
      description: service type
    image:
      type: string
      description: Name of the image
    flavor:
      type: string
      description: Flavor
    service_interface_type_list:
      type: string
      description: List of interface types
    shared_ip_list:
      type: string
      description: List of shared ip enabled- disabled
    static_routes_list:
      type: string
      description: List of static routes enabled- disabled

resources:
  service_template:
    type: OS::ContrailV2::ServiceTemplate
    properties:
      name: { get_param: name }
      service_mode: { get_param: mode }
      service_type: { get_param: type }
      image_name: { get_param: image }
      flavor: { get_param: flavor }
      service_interface_type_list: { "Fn::Split" : [ ",", Ref:
service_interface_type_list ] }
      shared_ip_list: { "Fn::Split" : [ ",", Ref: shared_ip_list ] }
      static_routes_list: { "Fn::Split" : [ ",", Ref: static_routes_list ]
}
    outputs:
      service_template_fq_name:
        description: FQ name of the service template
        value: { get_attr: [ service_template, fq_name] }

}
```

2. 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 Chaining in Contrail on page 187](#)
- [Route Aggregation on page 188](#)
- [Routing Policy on page 195](#)
- [Control for Route Reorigination on page 203](#)

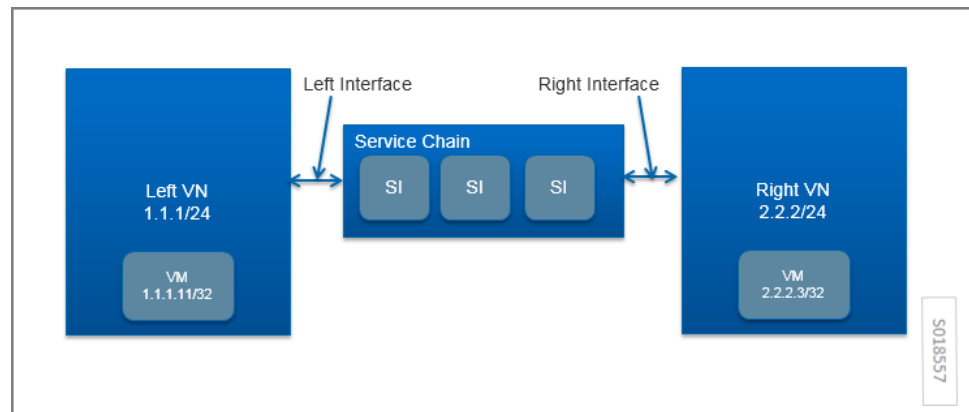
Overview: Service Chaining in Contrail

In Contrail, the service chaining feature allows the operator to insert dynamic services to control the traffic between two virtual networks. The service chaining works on a basic rule of next-hop stitching.

In [Figure 85 on page 187](#), the service chain is inserted between the Left VN and the Right VN. The service chain contains one or more service instances to achieve a required network policy.

In the example, the route for the VM in the Right VN is added to the routing table for the Left VN, with the next hop modified to ensure that the traffic is sent by means of the left interface of the service chain. This is an example of route reorigination.

Figure 85: Route Reorigination



Using reorigination of routes for service chaining (for example, putting the route for the right network in the left routing table) requires the following features:

- **Route aggregation**

For scaling purposes, it is useful to publish an aggregated route as the service chain route, rather than publishing every route of each VM (/32). This reduces the memory

footprint for the route table in the gateway router and also reduces route exchanges between control nodes and the gateway router. The route can be aggregated to the default route (0/0), to the VN subnet prefix, or to any arbitrary route prefix.

- **Path attribute modification for reoriginated routes**

There are cases where the **BgpPath** attribute for the service chain route needs to be modified. An example is the case of service chain failover, in which there are two service chains with identical services that are connected between the same two VNs. The operator needs to control which service chain is used for traffic between two networks, in addition to ensuring redundancy and high availability by providing failover support. Path attribute modification for reoriginated routes is implemented by means of routing policy, by providing an option to alter the MED (multi-exit discriminator) or **local-pref** of the reoriginated service chain route.

- **Control to enable and disable reorigination of the route**

In some scenarios, the operator needs a control to stop reorigination of the route as the service chain route, for example, when static routes are configured on service VM interfaces. Control to enable or disable reorigination of the route is implemented by tagging the routes with the **no-reoriginate** community. Routes with the **no-reoriginate** community tag are skipped for route reorigination.

Route Aggregation

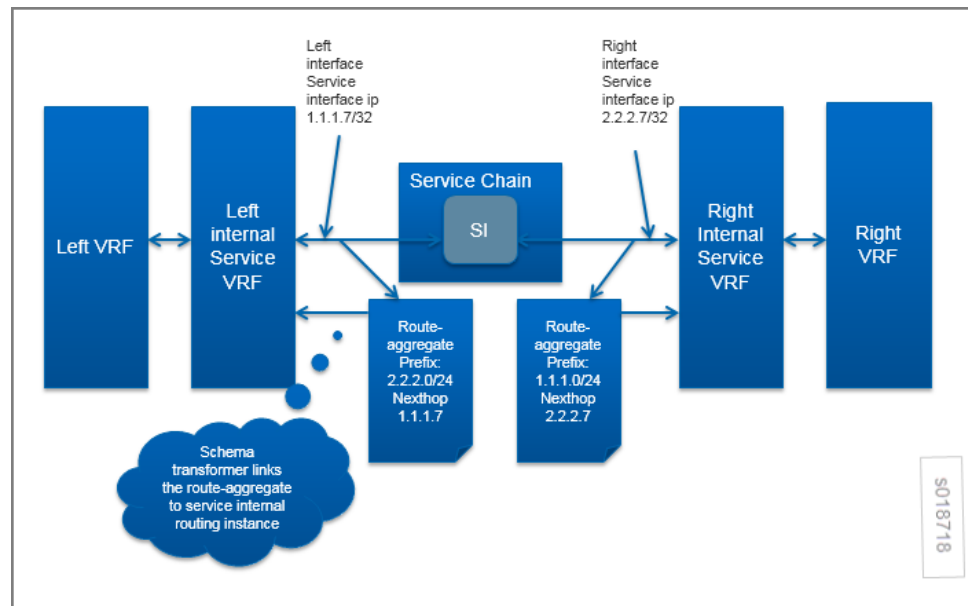
The route aggregation configuration object contains a list of prefixes to aggregate. The next-hop field in the route aggregate object contains the address of the route whose next hop is stitched as a next hop of the aggregate route.

Route aggregation is configured on the service instance. The operator can attach multiple route aggregation objects to a service instance. For example, if routes from the Right VN need to be aggregated and reoriginated in the route table of the Left VN, the route aggregate object is created with a prefix of the Right VN's subnet prefix and attached to the left interface of the service instance.

If the service chain has multiple service instances, the route aggregate object is attached to the left interface of the left-most service instance and to the right interface of the right-most service instance.

The relationships are shown in [Figure 86 on page 189](#).

Figure 86: Route Aggregate Relationships



The schema transformer sets the next-hop field of the route aggregate object to the service chain interface address. The schema transformer also links the route aggregate object to the internal routing instance created for the service instance.

Using the configuration as described, the Contrail control service reads the route aggregation object on the routing instance. When the first, more specific route or contributing route is launched (when the first VM is launched on the right VN), the aggregate route is published. Similarly, the aggregated route is deleted when the last, more specific route or contributing route is deleted (when the last VM is deleted in the right VN). The aggregated route is published when the next hop for the aggregated route gets resolved.

By default, in BGP or XMPP route exchanges, the control node will not publish contributing routes of an aggregate route.

Schema for Route Aggregation

- [Route Aggregate Object on page 189](#)
- [Service Instance Link to Route Aggregate Object on page 190](#)
- [Routing Instance Link to Route Aggregate Object on page 190](#)

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 191](#)
- [Configuring a Service Instance on page 191](#)
- [Create a Virtual Network and Network Policy on page 192](#)

- [Validate the Route Aggregate Object in the API Server on page 193](#)
- [Validate the Route Aggregate Object in the Control Node on page 194](#)

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 [Figure 87 on page 191](#).

Figure 87: Create Route Aggregate

Example VNC Script to Create a Route Aggregate Object

You can use a VNC script to create a route aggregate object, as in the following example:

```
from vnc_api.vnc_api import *
vnc_lib = VncApi("admin", "<password>.", "admin")
project=vnc_lib.project_read(fq_name=["default-domain", "admin"])
route_aggregate=RouteAggregate(name="left_to_right", parent_obj=project)
route_list=RouteListType(["<ip address>"])
route_aggregate.set_aggregate_route_entries(route_list)
vnc_lib.route_aggregate_create(route_aggregate)
```

Configuring a Service Instance

Create a service instance with the route aggregate object linked to the aggregate left network subnet prefix in the right virtual network. See the example in [Figure 88 on page 192](#).

Figure 88: Create Service Instance

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

5018720

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

5018721

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	
family	routing_instance	chain_address	prefixes	service_instance	static_routes	aggregate_routes	prefix	nexthop
inet	default-domain:admin:left:left	1.1.1.3	1.1.1.0/24	default-domain:admin:si-aggregate			1.1.1.0/24	1.1.1.3

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	
family	routing_instance	chain_address	prefixes	service_instance	static_routes	aggregate_routes	prefix	nexthop
inet	default-domain:admin:left:left	1.1.1.3	1.1.1.0/24	default-domain:admin:si-aggregate			1.1.1.0/24	1.1.1.3

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										
prefix	last_modified	paths								
1.1.1.0/24	2016-Feb-18 05:00:29.215876									
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.

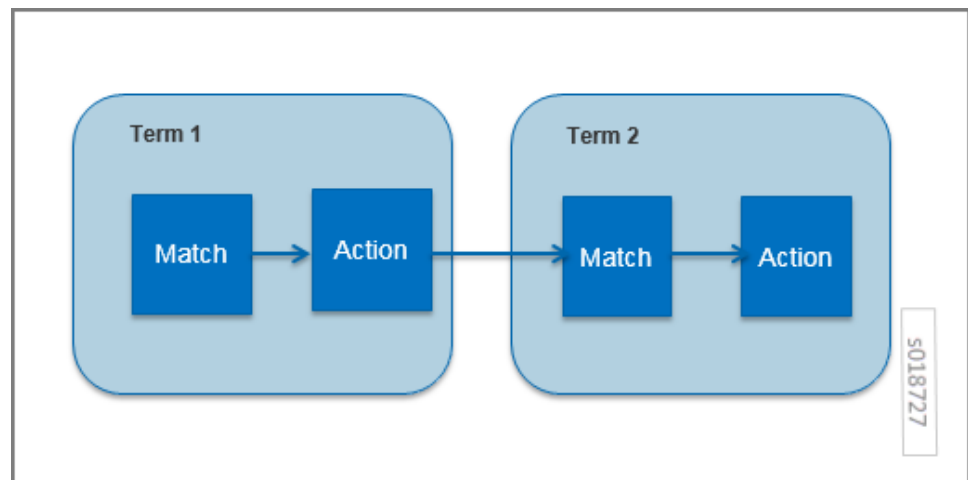
The routing policy contains list terms. A term can be a terminal rule, meaning that upon a match on the specified term, no further terms are evaluated and the route is dropped or accepted, based on the action in that term.

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.

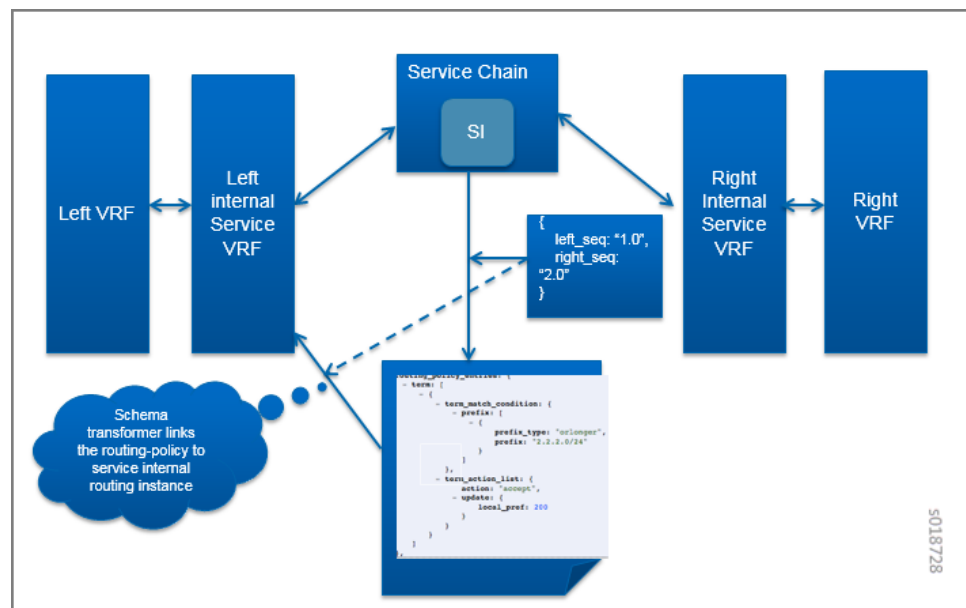
Update Action	User Input	Description
community	List of community	As part of the policy update, the following actions can be taken for community: <ul style="list-style-type: none"> • Add a list of community to the existing community. • Set a list of community. • Remove a list of community (if present) 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 199](#)
- [Configure Service Instance on page 200](#)
- [Configure the Network Policy for the Service Chain on page 200](#)

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.

Action	Protocol	Source	Ports	Direction	Destination	Ports	Log	Services	Mirror
PASS	ANY	left		right	right		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

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>true</div><div>matches</div><div>actions</div><div>prefix [2.2.2.0/24 orlonger]</div><div>accept</div><div>local-pref 200</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	prefixes	service_instance			policy_name	sequence
inet	default-domain:admin:right:right	1.1.1.6	prefixes 2.2.2.0/24	default-domain:admin: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

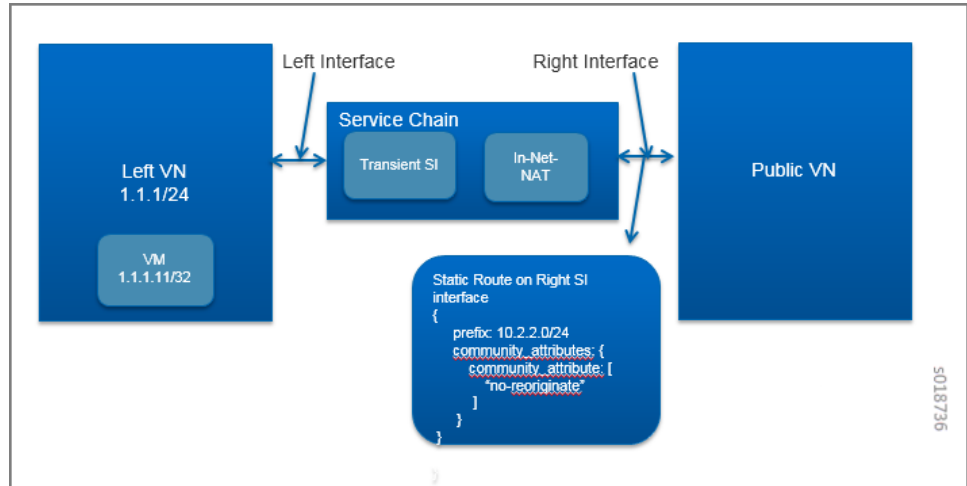
The ability to prevent reorigination of interface static routes is typically required when routes are configured on an interface that belongs to a service VM.

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
      }
    },
  },
}
```

s018738

Service Instance Health Check

In Contrail Release 3.0 and greater, a service instance health check can be used to determine the liveliness of a service provided by a virtual machine (VM).

- [Health Check Overview on page 205](#)
- [Health Check Object Configuration on page 206](#)
- [Creating a Health Check with the Contrail User Interface on page 207](#)
- [Using the Health Check on page 208](#)
- [Health Check Process on page 208](#)

Health Check Overview

The service instance health check is used to determine the liveliness of a service provided by a VM, checking whether the service is operationally up or down. The vRouter agent uses ping and an HTTP URL to the link-local address to check the liveliness of the interface.

If the health check determines that a service is no longer operational, it removes the routes for the VM, thereby disabling packet forwarding to the VM.

The service instance health check is used with service template Version 2.

Health Check Object Configuration

Table 37 on page 206 shows the configurable properties of the health check object.

Table 37: Health Check Configurable Parameters

Field	Description
- enabled	Indicates that health check is enabled. The default is False .
- health-check-type	Indicates the health check type: link-local 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 number of seconds to wait for a response.
- max-retries	The number of retries to attempt before declaring an instance health down.
- http-method	When the monitor protocol is HTTP, the type of HTTP method used, such as GET, PUT, POST, and so on.
- url-path	When the monitor protocol is HTTP, the URL to be used. For all other cases, such as ICMP, the destination IP address.
- expected-codes	When the monitor protocol is HTTP, the expected return code for HTTP operations.

Health Check Modes

The following modes are supported for the service instance health check:

- **link-local**—A local check for the service VM on the vRouter where the VM is running. In this case, the source IP of the packet is the service chain IP.
- **end-to-end**—A remote address or URL is provided for a service health check through a chain of services. The destination of the health check probe is allowed to be outside the service instance. However, the health check probe must be reachable through the interface of the service instance where the health check is attached. The end-to-end health check probe is transmitted all the way to the actual destination outside the service instance. The response to the health check probe is received and processed by the service health check to evaluate the status.

Restrictions include:

- This check is applicable for a chain where the services are not scaled out.
- When this mode is configured, a new health check IP is allocated and used as the source IP of the packet.

- The health check IP is allocated per **virtual-machine-interface** of the service VM where the health check is attached.
- The agent relies on the **service-health-check-ip** flag to use as the source IP.



NOTE: End-to-end health check is not supported on a transparent service chain. However, a link-local health check is possible on a transparent service instance if the corresponding service instance interface is configured with its IP address.

Creating a Health Check with the Contrail User Interface

To create a health check with the Contrail Web UI:

1. Navigate to **Configure > Services > Health Check Service**, and click to open the **Create** screen. See [Figure 89 on page 207](#).

Figure 89: Create Health Check Screen

The screenshot shows a 'Create' dialog box for a 'Health Check Service'. It has two tabs: 'Health Check Service' (selected) and 'Permissions'. The form contains the following fields:

- Name:** ext_hc_service
- Protocol:** PING (dropdown menu)
- Monitor Target:** 8.8.8.8 (dropdown menu)
- Delay (secs):** 3
- Timeout (secs):** 5
- Retries:** 2
- Health Check Type:** End-To-End (dropdown menu)

At the bottom right, there are 'Cancel' and 'Save' buttons. A small vertical text 's018766' is visible on the right edge of the dialog box.

2. Complete the fields to define the permissions for the health check, see [Table 38 on page 207](#).

Table 38: Create Health Check Fields

Field	Description
Name	Enter a name for the health check service you are creating.
Protocol	Select from the list the protocol to use for the health check, PING, HTTP, and so on.

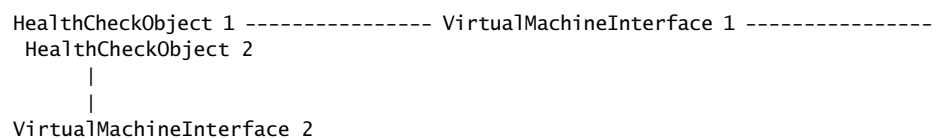
Table 38: Create Health Check Fields (*continued*)

Field	Description
Monitor Target	Select from the list the address of the target to be monitored by the health check.
Delay (secs)	The delay, in seconds, to repeat the health check.
Timeout (secs)	The number of seconds to wait for a response.
Retries	The number of retries to attempt before declaring an instance health down.
Health Check Type	Select from the list the type of health check—link-local or end-to-end.

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.

Examples: Configuring Service Chaining

- [Example: Creating an In-Network or In-Network-NAT Service Chain on page 209](#)
- [Example: Creating a Transparent Service Chain on page 217](#)
- [Example: Creating a Service Chain With the CLI on page 221](#)

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.

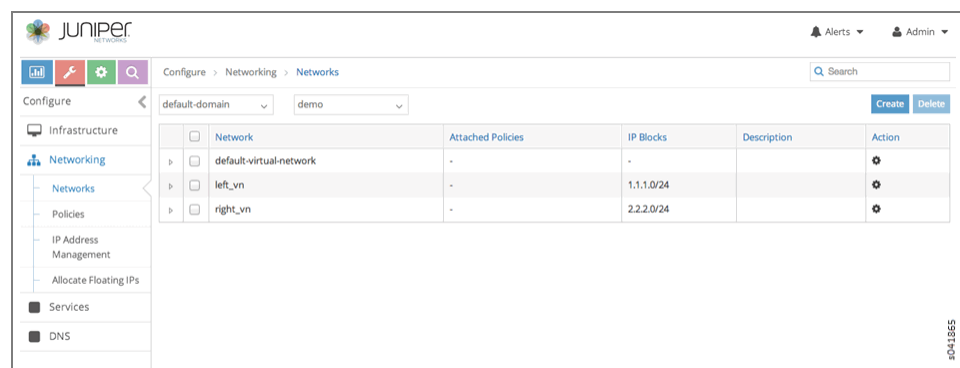
- [Creating an In-Network or In-Network-NAT Service Chain on page 209](#)

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 90 on page 209](#).

Figure 90: 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 91 on page 210](#).

Figure 91: 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 39: 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 39: 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.

3. 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

4. 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 92 on page 212](#) 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 92: Add Service Template Shared IP

Add Service Template

Name: nat-template

Service Mode: In-Network Service Type: Firewall

Service Scaling: ☒

Image Name: nat-service

Interface Types: Left Shared IP: ☒ + -

Service Interface	Shared IP
Management	Disabled
Right	Disabled
Left	Enabled

Cancel Save

s041903

5. Click **Save**.

The service template is created and appears on the **Service Templates** screen, see [Figure 93 on page 212](#).

Figure 93: Service Templates

Configure > Services > Service Templates

default-domain

Create Delete

Search Sitemap

Search Templates

Template	Service Mode	Service Scaling	Interfaces	Image Name	Flavor
nat-template	In-network	Enabled	Management, Left(Shared IP), Right	nat-service	m1.medium

s041900

6. 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 94 on page 213](#).

Figure 94: Create Service Instances

Table 40: 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 95 on page 214](#).

Figure 95: 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 96 on page 214](#) and [Figure 97 on page 215](#).

Figure 96: Service Instance Details

Instance Name	Template	Number of instances	Networks	Image	Flavor	Instance Details	Virtual Machine	Status	Power State	Networks
fw-instance	firewall-template (Transparent)	1	Management Network: Automatic, Left Network: Automatic, Right Network: Automatic	m1.medium	vnicbridge		fw-instance_1	ACTIVE	RUNNING	svc-vn-mgmt250.250.1.252 svc-vn-left250.250.2.253 svc-vn-right250.250.3.253

[View Console](#)

Figure 97: 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	
gr-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 98: 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 99: Edit Network

Network Name: left_vn

Network Policy(s): nat-policy

Address Management: default-domain:default-p...

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 100 on page 216).
 - 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 100: Launch Instances

Instances

Logged in as: admin Settings Help Sign Out

+ Launch Instance Terminate 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 173](#)
 - [Example: Creating a Transparent Service Chain on page 217](#)
 - [ECMP Load Balancing in the Service Chain on page 178](#)

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.

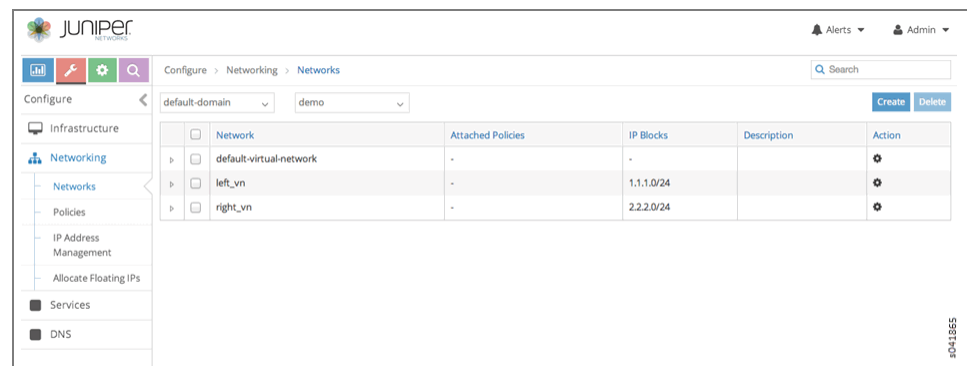
- [Creating a Transparent Mode Service Chain on page 217](#)

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 101 on page 217](#).

Figure 101: 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 102 on page 218](#).

Figure 102: Add Service Template

Add Service Template

Name: firewall-template

Service Mode: Transparent

Image Name: vsrxbridge

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: m1.medium(RAM:4096, CPU cores:2, Disk:...)

Cancel Save

Table 41: 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	<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 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 103 on page 219](#).

Figure 103: Create Service Instances

Table 42: Create Service Instances Fields

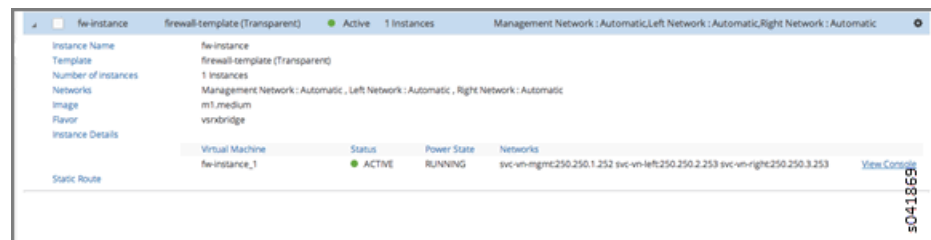
Field	Description
Instance Name	Enter a name for the service instance.
Services Template	Select from a list of available service templates the service template to use for this instance.
Left Network	Select from a list of available virtual networks the network to use for the left interface. For transparent mode, select Auto Configured .

Table 42: 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

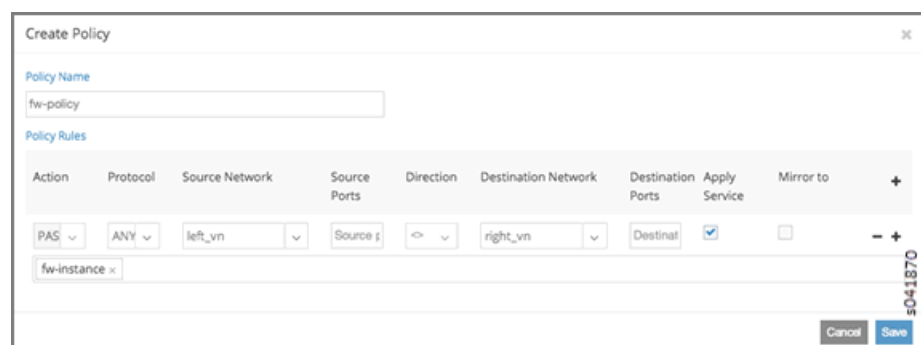
6. If scaling is enabled, enter a value in the **Number of Instances** field to define the number of instances of service virtual machines to launch; see [Figure 104 on page 220](#).

Figure 104: 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**).

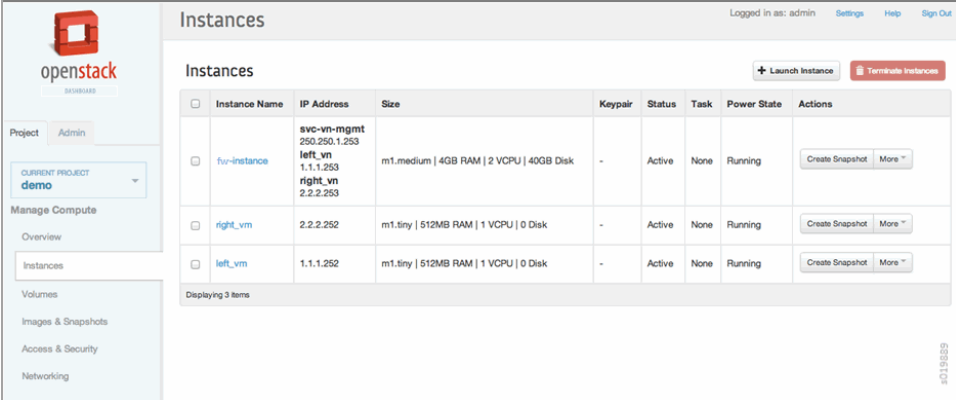
Figure 105: Create Policy



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 106 on page 221](#)).
 - e. A **TCPDUMP** on the **right_vm** should show that packets have the source IP set to 2.2.2.253.

Figure 106: Launch Instances



Instance Name	IP Address	Size	Keypair	Status	Task	Power State	Actions
svc-vn-mgmt 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 173](#)

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 221](#)
- [CLI for Creating a Service Template on page 222](#)
- [CLI for Creating a Service Instance on page 222](#)
- [CLI for Creating a Service Policy on page 222](#)
- [Example: Creating a Service Chain with VSRX and In-Network or Routed Mode on page 223](#)

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 173](#)

PART 4

Monitoring and Troubleshooting the Network Using Contrail Analytics

- [Understanding Contrail Analytics on page 227](#)
- [Configuring Contrail Analytics on page 251](#)
- [Using Contrail Analytics to Monitor and Troubleshoot the Network on page 261](#)

Understanding Contrail Analytics

- [Understanding Contrail Analytics on page 227](#)
- [Contrail Alerts on page 228](#)
- [Underlay Overlay Mapping in Contrail on page 231](#)

Understanding Contrail Analytics

Contrail is a distributed system of compute nodes, control nodes, configuration nodes, database nodes, web UI nodes, and analytics nodes.

The analytics nodes are responsible for the collection of system state information, usage statistics, and debug information from all of the software modules across all of the nodes of the system. The analytics nodes store the data gathered across the system in a database that is based on the Apache Cassandra open source distributed database management system. The database is queried by means of an SQL-like language and representational state transfer (REST) APIs.

System state information collected by the analytics nodes is aggregated across all of the nodes, and comprehensive graphical views allow the user to get up-to-date system usage information easily.

Debug information collected by the analytics nodes includes the following types:

- System log (syslog) messages—informational and debug messages generated by system software components.
- Object log messages—records of changes made to system objects such as virtual machines, virtual networks, service instances, virtual routers, BGP peers, routing instances, and the like.
- Trace messages—records of activities collected locally by software components and sent to analytics nodes only on demand.

Statistics information related to flows, CPU and memory usage, and the like is also collected by the analytics nodes and can be queried at the user interface to provide historical analytics and time-series information. The queries are performed using REST APIs.

Analytics data is written to a database in Contrail. The data expires after the default time-to-live (TTL) period of 48 hours. This default TTL time can be changed as needed by changing the value of the **database_ttl** value in the cluster configuration.

**Related
Documentation**

- [Contrail Alerts on page 228](#)
- [Analytics Scalability on page 251](#)
- [High Availability for Analytics on page 252](#)
- [Ceilometer Support in a Contrail Cloud on page 255](#)
- [Underlay Overlay Mapping in Contrail on page 231](#)
- [Monitoring the System on page 262](#)
- [Debugging Processes Using the Contrail Introspect Feature on page 264](#)
- [Monitor > Infrastructure > Dashboard on page 268](#)
- [Monitor > Infrastructure > Control Nodes on page 270](#)
- [Monitor > Infrastructure > Virtual Routers on page 277](#)
- [Monitor > Infrastructure > Analytics Nodes on page 287](#)
- [Monitor > Infrastructure > Config Nodes on page 292](#)
- [Monitor > Networking on page 295](#)
- [*Understanding Flow Sampling*](#)
- [Query > Flows on page 303](#)
- [Query > Logs on page 310](#)
- [System Log Receiver in Contrail Analytics on page 253](#)
- [Example: Debugging Connectivity Using Monitoring for Troubleshooting on page 315](#)

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: "eyJ0aW1lc3RhbnRhaXN0Y0tk1MzQ5MjUzMTc4LCAiaHR0cF9wb3J0Ijog
NTk5NSwgImhvc3RfaXAiOiAiMTAuODQuMTMuNDAiQ==",
        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 262](#)
- [Debugging Processes Using the Contrail Introspect Feature on page 264](#)
- [Monitor > Infrastructure > Dashboard on page 268](#)
- [Monitor > Infrastructure > Control Nodes on page 270](#)
- [Monitor > Infrastructure > Virtual Routers on page 277](#)
- [Monitor > Infrastructure > Analytics Nodes on page 287](#)
- [Monitor > Infrastructure > Config Nodes on page 292](#)
- [Monitor > Networking on page 295](#)
- [Understanding Flow Sampling](#)
- [Query > Flows on page 303](#)
- [Query > Logs on page 310](#)
- [Example: Debugging Connectivity Using Monitoring for Troubleshooting on page 315](#)

Underlay Overlay Mapping in Contrail

- [Overview: Underlay Overlay Mapping using Contrail Analytics on page 232](#)
- [Underlay Overlay Analytics Available in Contrail on page 232](#)
- [Architecture and Data Collection on page 233](#)
- [New Processes/Services for Underlay Overlay Mapping on page 233](#)
- [External Interfaces Configuration for Underlay Overlay Mapping on page 234](#)
- [Physical Topology on page 234](#)

- [SNMP Configuration on page 235](#)
- [Link Layer Discovery Protocol \(LLDP\) Configuration on page 235](#)
- [IPFIX and sFlow Configuration on page 235](#)
- [Sending pRouter Information to the SNMP Collector in Contrail on page 237](#)
- [pRouter UVEs on page 237](#)
- [Contrail User Interface for Underlay Overlay Analytics on page 239](#)
- [Enabling Physical Topology on the Web UI on page 239](#)
- [Viewing Topology to the Virtual Machine Level on page 240](#)
- [Viewing the Traffic of any Link on page 240](#)
- [Trace Flows on page 241](#)
- [Search Flows and Map Flows on page 242](#)
- [Overlay to Underlay Flow Map Schemas on page 242](#)
- [Module Operations for Overlay Underlay Mapping on page 245](#)
- [SNMP Collector Operation on page 245](#)
- [Topology Module Operation on page 246](#)
- [IPFIX and sFlow Collector Operation on page 247](#)
- [Troubleshooting Underlay Overlay Mapping on page 247](#)
- [Script to add pRouter Objects on page 248](#)

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.

contrail-status

```
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.

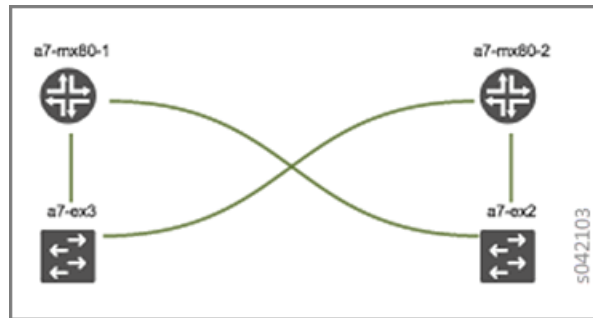
Physical Topology

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 107: Analytics Topology



SNMP Configuration

Configure SNMP on the physical devices so that the **contrail-snmp-collector** can read SNMP data.

The following shows an example SNMP configuration from a Juniper Networks device.

```
set snmp community public authorization read-only
```

Link Layer Discovery Protocol (LLDP) Configuration

Configure LLDP on the physical device so that the **contrail-snmp-collector** can read the neighbor information of the routers.

The following is an example of LLDP configuration on a Juniper Networks device.

```
set protocols lldp interface all
```

```
set protocols lldp-med interface all
```

IPFIX and sFlow Configuration

Flow samples are sent to the **contrail-collector** by the physical devices. Because the **contrail-collector** supports the sFlow and IPFIX protocols for receiving flow samples, the physical devices, such as MX Series devices or ToR switches, must be configured to send samples using one of those protocols.

Example: sFlow Configuration

The following shows a sample sFlow configuration. In the sample, the IP variable *<source ip>* refers to the loopback or IP that can be reachable of the device that acts as an sflow source, and the other IP variable *<collector_IP_data>* is the address of the collector device.

```

root@host> show configuration protocols sflow | display set

set protocols sflow polling-interval 0

set protocols sflow sample-rate ingress 10

set protocols sflow source-ip <source ip>4

set protocols sflow collector <collector_IP_data> udp-port 6343

set protocols sflow interfaces ge-0/0/0.0

```

```
set protocols sflow interfaces ge-0/0/1.0
set protocols sflow interfaces ge-0/0/2.0
set protocols sflow interfaces ge-0/0/3.0
set protocols sflow interfaces ge-0/0/4.0
```

Example: IPFIX Configuration

The following is a sample IPFIX configuration from a Juniper Networks device. The IP address variable `<ip_sflow collector>` represents the sflow collector (control-collector analytics node) and `<source ip>` represents the source (outgoing) interface on the router/switch device used for sending flow data to the collector. This could also be the lo0 address, if it is reachable from the Contrail cluster.

```
root@host> show configuration chassis | display set

set chassis tfeb slot 0 sampling-instance sample-ins1
set chassis network-services 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.

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 108: Add Physical Router Window

The screenshot shows the Juniper Contrail configuration interface. On the left is a navigation pane with categories like Infrastructure, Physical Devices, Networking, Services, and DNS. The 'Physical Devices' section is expanded, showing a list of physical routers. Overlaid on this is a modal window titled 'Add Physical Router'. The form in the modal includes:

- Name:** A text field containing 'new-router'.
- Vendor:** A text field.
- Model:** A text field.
- Management IP:** A text field containing '1.1.1.1'.
- Tunnel Source IP:** A text field.
- User Credentials:** An expandable section.
- Virtual Router:** An expandable section.
- BGP Router:** An expandable section.
- SNMP Credentials:** An expanded section containing:
 - Version:** Radio buttons for '2' (selected) and '3'.
 - Community:** A text field containing 'public'.

At the bottom right of the modal are 'Cancel' and 'Save' buttons. A small 'SQL3440' label is visible on the right edge of the modal.

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 109: Sample Output From a pRouter REST API

```
[
  - {
    href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-mx80-1?flat",
    name: "a7-mx80-1"
  },
  - {
    href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-mx80-2?flat",
    name: "a7-mx80-2"
  },
  - {
    href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-ex3?flat",
    name: "a7-ex3"
  },
  - {
    href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-ex2?flat",
    name: "a7-ex2"
  }
]
```

s042104

Details of a pRouter UVE can be obtained from your system, using a URL of the following form:

http://<host ip>:8081/analytics/uves/prouter/a7-ex3?flat

The following is sample output of a pRouter UVE.

Figure 110: Sample Output From a pRouter UVE

```

{
  - PRouterFlowEntry: {
    flow_export_source_ip: "10.84.63.114"
  },
  - PRouterLinkEntry: {
    - link_table: [
      - {
        remote_interface_name: "ge-1/0/1",
        local_interface_name: "ge-0/0/0.0",
        remote_interface_index: 517,
        local_interface_index: 503,
        type: 1,
        remote_system_name: "a7-mx80-1"
      },
      - {
        remote_interface_name: "ge-1/0/1",
        local_interface_name: "ge-0/0/1.0",
        remote_interface_index: 517,
        local_interface_index: 505,
        type: 1,
        remote_system_name: "a7-mx80-2"
      },
      - {
        remote_interface_name: "eth1",
        local_interface_name: "ge-0/0/2.0",
        remote_interface_index: 1,
        local_interface_index: 507,
        type: 2,
        remote_system_name: "a7s35"
      },
      - {
        remote_interface_name: "eth1",
        local_interface_name: "ge-0/0/3.0",
        remote_interface_index: 1,
        local_interface_index: 509,
        type: 2,
        remote_system_name: "a7s36"
      }
    ]
  },
  - PRouterEntry: {
    + ipMib: [...],
    + ifTable: [...],
    + ifXTable: [...],
    + arpTable: [...],
    + lldpTable: {...},
    + ifStats: [...]
  }
}

```

5042435

Contrail User Interface for Underlay Overlay Analytics

The topology view and related functionality is accessed from the Contrail Web user interface, **Monitor > Physical Topology**.

Enabling Physical Topology on the Web UI

To enable the **Physical Topology** section in the Contrail Web UI:

1. Add the following lines to the `/etc/contrail/config.global.js` file of all the **contrail-webui** nodes:

```

config.optFeatureList = {};
config.optFeatureList.mon_infra_underlay = true;

```

2. Restart webui supervisor.

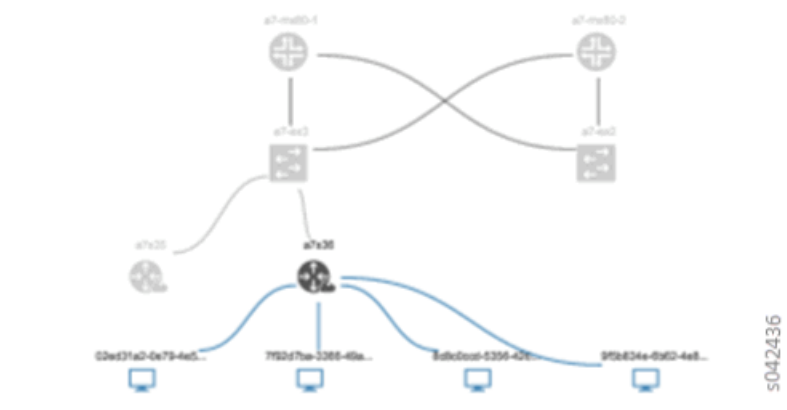
service supervisor-webui restart

The **Physical Topology** section is now available on the Contrail Web UI.

Viewing Topology to the Virtual Machine Level

In the Contrail user interface, it is possible to drill down through displayed topology to the virtual machine level. The following diagram shows the virtual machines instantiated on a7s36 vRouter and the full physical topology related to each.

Figure 111: 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.

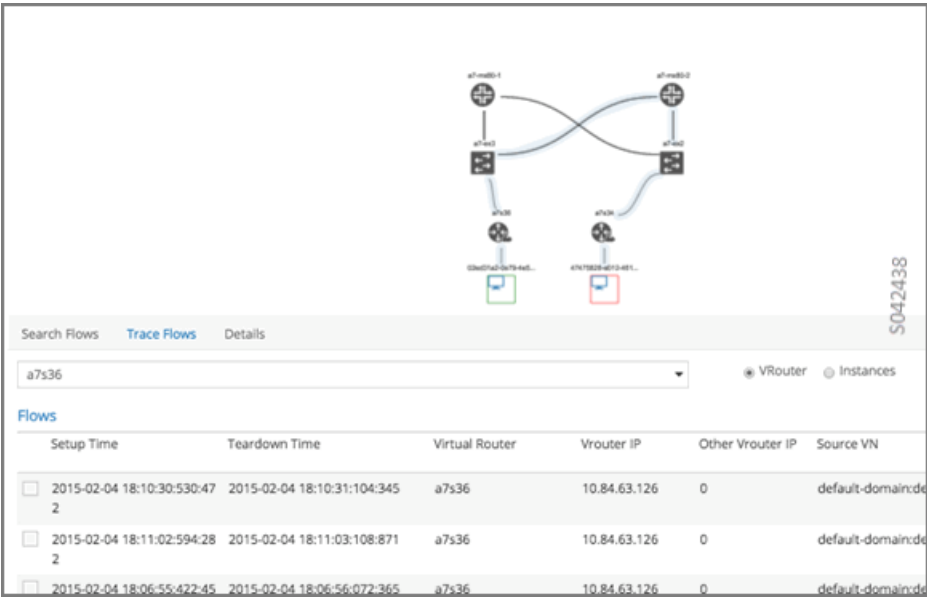
Figure 112: Traffic Statistics Graph



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 113: List of Active Flows



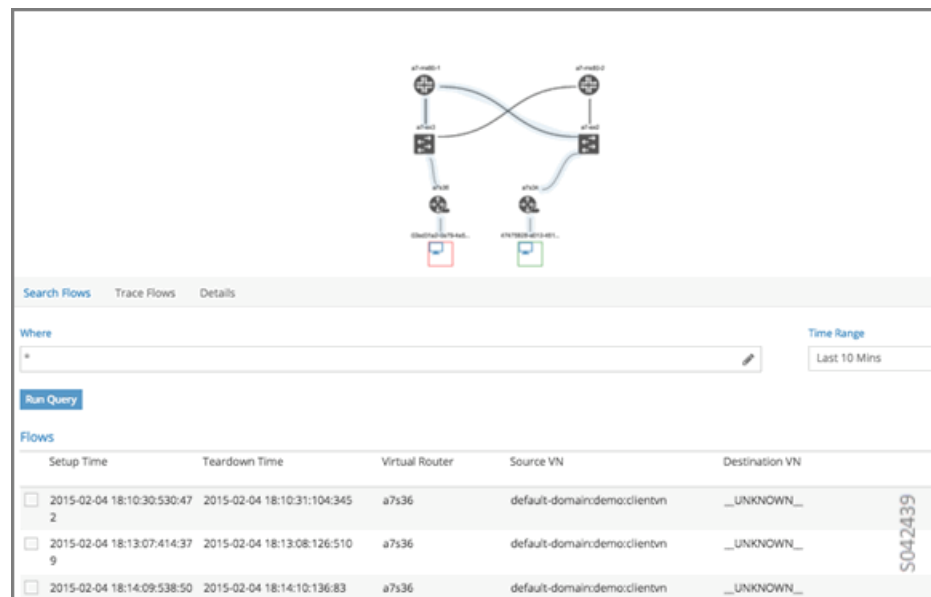
Limitations of Trace Flow Feature

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 114: Underlay Path



Overlay to Underlay Flow Map Schemas

The schema to query the underlay mapping information for an overlay flow is obtained from a REST API, which can be accessed on your system using a URL of the following form:

http://<host ip>:8081/analytics/table/OverlayToUnderlayFlowMap/schema

Example: Overlay to Underlay Flow Map Schema

```
{
  "type": "FLOW",
  "columns": [
    {
      "datatype": "string", "index": true, "name": "o_svn", "select": false,
      "suffixes": ["o_sip"]
    },
    {
      "datatype": "string", "index": false, "name": "o_sip", "select": false,
      "suffixes": null
    },
    {
      "datatype": "string", "index": true, "name": "o_dvn", "select": false,
      "suffixes": ["o_dip"]
    },
    {
      "datatype": "string", "index": false, "name": "o_dip", "select": false,

```

```

"suffixes": null},

{"datatype": "int", "index": false, "name": "o_sport", "select": false,
"suffixes": null},

{"datatype": "int", "index": false, "name": "o_dport", "select": false,
"suffixes": null},

{"datatype": "int", "index": true, "name": "o_protocol", "select": false,
"suffixes": ["o_sport", "o_dport"]},

{"datatype": "string", "index": true, "name": "o_vrouter", "select": false,
"suffixes": null},

{"datatype": "string", "index": false, "name": "u_prouter", "select": null,
"suffixes": null},

{"datatype": "int", "index": false, "name": "u_pifindex", "select": null,
"suffixes": null},

{"datatype": "int", "index": false, "name": "u_vlan", "select": null,
"suffixes": null},

{"datatype": "string", "index": false, "name": "u_sip", "select": null,
"suffixes": null},

{"datatype": "string", "index": false, "name": "u_dip", "select": null,
"suffixes": null},

{"datatype": "int", "index": false, "name": "u_sport", "select": null,
"suffixes": null},

{"datatype": "int", "index": false, "name": "u_dport", "select": null,
"suffixes": null},

{"datatype": "int", "index": false, "name": "u_protocol", "select": null,
"suffixes": null},

{"datatype": "string", "index": false, "name": "u_flowtype", "select": null,
"suffixes": null},

{"datatype": "string", "index": false, "name": "u_otherinfo", "select": null,
"suffixes": null}}}

```

The schema for underlay data across pRouters is defined in the Contrail installation at:

<http://<host ip>:8081/analytics/table/StatTable.UFlowData.flow/schema>

Example: Flow Data Schema for Underlay

```

{"type": "STAT",
"columns": [

{"datatype": "string", "index": true, "name": "Source", "suffixes": null},

{"datatype": "int", "index": false, "name": "T", "suffixes": null},

{"datatype": "int", "index": false, "name": "CLASS(T)", "suffixes": null},

{"datatype": "int", "index": false, "name": "T=", "suffixes": null},

```

```
{ "datatype": "int", "index": false, "name": "CLASS(T=)", "suffixes": null },
{ "datatype": "uuid", "index": false, "name": "UUID", "suffixes": null },
{ "datatype": "int", "index": false, "name": "COUNT(flow)", "suffixes": null },
{ "datatype": "string", "index": true, "name": "name", "suffixes":
  ["flow.pifindex"] },
{ "datatype": "int", "index": false, "name": "flow.pifindex", "suffixes": null },
{ "datatype": "int", "index": false, "name": "SUM(flow.pifindex)", "suffixes":
  null },
{ "datatype": "int", "index": false, "name": "CLASS(flow.pifindex)", "suffixes":
  null },
{ "datatype": "int", "index": false, "name": "flow.sport", "suffixes": null },
{ "datatype": "int", "index": false, "name": "SUM(flow.sport)", "suffixes":
  null },
{ "datatype": "int", "index": false, "name": "CLASS(flow.sport)", "suffixes":
  null },
{ "datatype": "int", "index": false, "name": "flow.dport", "suffixes": null },
{ "datatype": "int", "index": false, "name": "SUM(flow.dport)", "suffixes":
  null },
{ "datatype": "int", "index": false, "name": "CLASS(flow.dport)", "suffixes":
  null },
{ "datatype": "int", "index": true, "name": "flow.protocol", "suffixes":
  ["flow.sport", "flow.dport"] },
{ "datatype": "int", "index": false, "name": "SUM(flow.protocol)", "suffixes":
  null },
{ "datatype": "int", "index": false, "name": "CLASS(flow.protocol)", "suffixes":
  null },
{ "datatype": "string", "index": true, "name": "flow.sip", "suffixes": null },
{ "datatype": "string", "index": true, "name": "flow.dip", "suffixes": null },
{ "datatype": "string", "index": true, "name": "flow.vlan", "suffixes": null },
{ "datatype": "string", "index": false, "name": "flow.flowtype", "suffixes":
  null },
{ "datatype": "string", "index": false, "name": "flow.otherinfo", "suffixes":
  null } }
```


Example: Typical Query for Flow Map

The following is a typical query. Internally, the **analytics-api** performs a query into the **FlowRecordTable**, then into the **StatTable.UFlowData.flow**, to return list of (**prouter**, **pifindex**) pairs that give the underlay path taken for the given overlay flow.

```
FROM
  OverlayToUnderlayFlowMap

SELECT
  prouter, pifindex

WHERE
  o_svn, o_sip, o_dvn, o_dip, o_sport, o_dport, o_protocol = <overlay flow>
```

Module Operations for Overlay Underlay Mapping

SNMP Collector Operation

The Contrail SNMP collector uses a Net-SNMP library to talk to a physical router or any SNMP agent. Upon receiving SNMP packets, the data is translated to the Python dictionary, and corresponding UVE objects are created. The UVE objects are then posted to the SNMP collector.

The SNMP module sleeps for some configurable period, then forks a collector process and waits for the process to complete. The collector process goes through a list of devices to be queried. For each device, it forks a greenlet task (Python coroutine), accumulates SNMP data, writes the summary to a JSON file, and exits. The parent process then reads the JSON file, creates UVEs, sends the UVEs to the collector, then goes to sleep again.

The pRouter UVE sent by the SNMP collector carries only the raw MIB information.

Example: pRouter Entry Carried in pRouter UVE

The definition below shows the **pRouterEntry** carried in the **pRouterUVE**. Additionally, an example **LldpTable** definition is shown.

The following create a virtual table as defined by:

```
http://<host ip>:8081/analytics/table/StatTable.UFlowData.flow/schema

struct LldpTable {
  1: LldpLocalSystemData lldpLocalSystemData
  2: optional list<LldpRemoteSystemsData> lldpRemoteSystemsData
}

struct PRouterEntry {
  1: string name (key="ObjectPRouter")
  2: optional bool deleted
  3: optional LldpTable lldpTable
  4: optional list<ArpTable> arpTable
```

```
5: optional list<IfTable> ifTable
6: optional list<IfXTable> ifXTable
7: optional list<IfStats> ifStats (tags="name:.ifIndex")
8: optional list<IpMib> ipMib
}
uve sandesh PRouterUVE {
  1: PRouterEntry data
}
```

Topology Module Operation

The topology module reads UVEs posted by the SNMP collector and computes the neighbor table, populating the table with remote system name, local and remote interface names, the remote type (pRouter or vRouter) and local and remote ifindices. The topology module sleeps for a while, reads UVEs, then computes the neighbor table and posts the UVE to the collector.

The pRouter UVE sent by the topology module carries the neighbor list, so the clients can put together all of the pRouter neighbor lists to compute the full topology.

The corresponding pRouter UVE definition is the following.

```
struct LinkEntry {
  1: string remote_system_name
  2: string local_interface_name
  3: string remote_interface_name
  4: RemoteType type
  5: i32 local_interface_index
  6: i32 remote_interface_index
}
struct PRouterLinkEntry {
  1: string name (key="ObjectPRouter")
  2: optional bool deleted
  3: optional list<LinkEntry> link_table
}
uve sandesh PRouterLinkUVE {
  1: PRouterLinkEntry data
}
```

```
}
```

IPFIX and sFlow Collector Operation

An IPFIX and sFlow collector has been implemented in the Contrail collector. The collector receives the IPFIX and sFlow samples and stores them as statistics samples in the analytics database.

Example: IPFIX sFlow Collector Data

The following definition shows the data stored for the statistics samples and the indices that can be used to perform queries.

```
struct UFlowSample {
    1: u64 pifindex
    2: string sip
    3: string dip
    4: u16 sport
    5: u16 dport
    6: u16 protocol
    7: u16 vlan
    8: string flowtype
    9: string otherinfo
}

struct UFlowData {
    1: string name (key="ObjectPRouterIP")
    2: optional bool deleted
    3: optional list<UFlowSample> flow (tags="name:.pifindex, .sip, .dip,
        .protocol:.sport, .protocol:.dport, .vlan")
}
```

Troubleshooting Underlay Overlay Mapping

This section provides a variety of links where you can research errors that may occur with underlay overlay mapping.

System Logs Logs for **contrail-snmp-collector** and **contrail-topology** are in the following locations on an installed Contrail system:

`/var/log/contrail/contrail-snmp-collector-stdout.log`

`/var/log/contrail/contrail-topology.log`

Introspect Utility Use URLs of the following forms on your Contrail system to access the introspect utilities for SNMP data and for topology data.

- SNMP data introspect

`http://<host ip>:5920/Snh_SandeshUVECacheReq?x=PRouterEntry`

- Topology data introspect

`http://<host ip>:5921/Snh_SandeshUVECacheReq?x=PRouterLinkEntry`

Script to add pRouter Objects

The usual mechanism for adding pRouter objects to **contrail-config** is through Contrail UI. But you also have the ability to add these objects using the Contrail **vnc-api**. To add one pRouter, save the file with the name **cfg-snmp.py**, and then execute the command as shown:

python cfg-snmp.py

**Example: Content for
cfg-snmp.py**

```
#!/python

from vnc_api import vnc_api

from vnc_api.gen.resource_xsd import SNMPCredentials

vnc = vnc_api.VncApi('admin', 'abcde123', 'admin')
apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-mx80-1')
apr.set_physical_router_management_ip('ip_address')
apr.set_physical_router_dataplane_ip('ip_address')
apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2,
v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-mx80-2')
apr.set_physical_router_management_ip('ip_address')
apr.set_physical_router_dataplane_ip('ip_address')
apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2,
v2_community='public'))
```

```
vnc.physical_router_create(apr)

#$ABC123'

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-ex3')

apr.set_physical_router_management_ip('source_ip')

apr.set_physical_router_dataplane_ip('source_ip')

apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2,
v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123'

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-ex2')

apr.set_physical_router_management_ip('ip_address')

apr.set_physical_router_dataplane_ip('ip_address')

apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2,
v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123'
```

- Related Documentation**
- [Understanding Contrail Analytics on page 227](#)
 - [Contrail Alerts on page 228](#)

Configuring Contrail Analytics

- [Analytics Scalability on page 251](#)
- [High Availability for Analytics on page 252](#)
- [System Log Receiver in Contrail Analytics on page 253](#)
- [Sending Flow Messages to the Contrail System Log on page 254](#)
- [Ceilometer Support in a Contrail Cloud on page 255](#)

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).

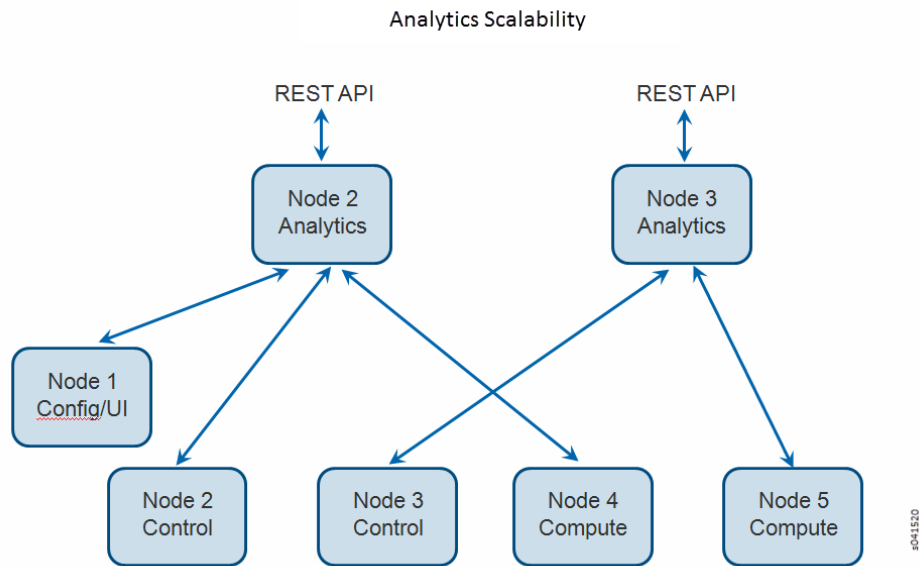
This section provides a brief description of the recommended configuration of analytics in Contrail to achieve horizontal scalability.

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 115 on page 252](#) 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 115: Analytics Scalability



The analytics nodes collect and store data and provide this data through various REST API queries. Scalability is provided for the control nodes, the compute nodes, and the REST API users, with the API output displayed in the Contrail user interface. As the number of control and compute nodes increase in the system, the analytics nodes can also be increased.

High Availability for Analytics

Contrail supports multiple instances of analytics for high availability and load balancing.

Contrail analytics provides two broad areas of functionality:

- **contrail-collector** —Receives status, logs, and flow information from all Contrail processing elements (for example, generators) and records them.

Every generator is connected to one of the **contrail-collector** instances at any given time. If an instance fails (or is shut down), all the generators that are connected to it are automatically moved to another functioning instance, typically in a few seconds or less. Some messages may be lost during this movement. UVEs are resilient to message loss, so the state shown in a UVE is kept consistent to the state in the generator.

- **contrail-opserver** —Provides an external API to report UVEs and to query logs and flows.

Each analytics component exposes a northbound REST API represented by the **contrail-opserver** service (port 8081) so that the failure of one analytics component or one **contrail-opserver** service should not impact the operation of other instances.

These are the ways to manage connectivity to the **contrail-opserver** endpoints:

- Periodically poll the **contrail-opserver** service on a set of analytics nodes to determine the list of functioning endpoints, then make API requests from one or more of the functioning endpoints.
- The Contrail user interface makes use of the same northbound REST API to present dashboards, and reacts to any **contrail-opserver** high availability event automatically.

System Log Receiver in Contrail Analytics

- [Overview on page 253](#)
- [Redirecting System Logs to Contrail Collector on page 253](#)
- [Exporting Logs from Contrail Analytics on page 253](#)

Overview

The **contrail-collector** process on the Contrail Analytics node can act as a system log receiver.

Redirecting System Logs to Contrail Collector

You can enable the **contrail-collector** to receive system logs by giving a valid **syslog_port** as a command line option:

```
--DEFAULT.syslog_port <arg>
```

or by adding **syslog_port** in the **DEFAULT** section of the configuration file at **/etc/contrail/contrail-collector.conf**.

For nodes to send system logs to the **contrail-collector**, the system log configuration for the node should be set up to direct the system logs to **contrail-collector**.

Example Add the following line in **/etc/rsyslog.d/50-default.conf** on an Ubuntu system to redirect the system logs to **contrail-collector**.

```
*.* @<collector_ip>:<collector_syslog_port> :: @ for udp, @@ for tcp
```

The logs can be retrieved by using Contrail tool, either by using the **contrail-logs** utility on the analytics node or by using the Contrail user interface on the system log query page.

Exporting Logs from Contrail Analytics

You can also export logs stored in Contrail analytics to another system log receiver by using the **contrail-logs** utility.

The **contrail-logs** utility can take these options: **--send-syslog**, **--syslog-server**, **--syslog-port**, to query Contrail analytics, then send the results as system logs to a system log server. This is an on-demand command, one can write a cron job or a job that continuously invokes **contrail-logs** to achieve continuous sending of logs to another system log server.

Sending Flow Messages to the Contrail System Log

The **contrail-vrouter-agent** can be configured to send flow messages and other messages to the system log (syslog). To send flow messages to syslog, configure the following parameters in **/etc/contrail/contrail-vrouter-agent.conf**.

The following parameters are under the section **DEFAULT**:

- **log_flow=1**—Enables logging of all flow messages.
- **use_syslog=1**—Enables sending of all messages, including flow messages, to syslog.
- **syslog_facility=LOG_LOCAL0**—Enables sending messages from the **contrail-vrouter-agent** to the syslog, using the facility **LOCAL0**. You can configure **LOCAL0** to your required facility.
- **log_level=SYS_INFO**—Changes the logging level of **contrail-vrouter-agent** to **INFO**.

If syslog is enabled, flow messages are *not* sent to Contrail Analytics because the two destinations are mutually exclusive.

Flow log sampling settings apply regardless of the flow log destination specified. If sampling is enabled, the syslog messages will be sampled using the same rules that would apply to Contrail Analytics. If non-sampled flow data is required, sampling must be disabled by means of configuration settings.

Flow events for termination will include both the appropriate tear-down fields and the appropriate setup fields.

The flow messages will be sent to the syslog with a severity of **INFO**.

The user can configure the remote system log (**rsyslog**) on the compute node to send syslog messages with facility **LOCAL0**, severity of **INFO** (and lower), to the remote syslog server. Messages with a higher severity than **INFO** can be logged to a local file to allow for debugging.

Flow messages appear in the syslog in a format similar to the following log example:

```
May 24 14:40:13 a7s10 contrail-vrouter-agent[29930]: 2016-05-24 Tue 14:40:13:921.098
PDT a7s10 [Thread 139724471654144, Pid 29930]: [SYS_INFO]: FlowLogDataObject:
flowdata= [ [ [ flowuuid = 7ea8bf8f-b827-496e-b93e-7622a0c8eeea direction_ing = 1
sourcevn = default-domain:mock-gen-test:vn8 sourceip = 1.0.0.9 destvn =
default-domain:mock-gen-test:vn58 destip = 1.0.0.59 protocol = 1 sport = -29520 dport =
20315 setup_time = 1464125225556930 bytes = 1035611592 packets = 2024830 diff_bytes
= 27240 diff_packets = 40 ], ] ]
```



NOTE: Several individual flow messages might be packed into a single syslog message for improved efficiency.

Ceilometer Support in a Contrail Cloud

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 255](#)
- [Ceilometer Details on page 255](#)
- [Verification of Ceilometer Operation on page 256](#)
- [Contrail Ceilometer Plugin on page 258](#)
- [Ceilometer Installation and Provisioning on page 260](#)

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 Server Manager Lite 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 | 15c0240142084d16b3127d6f844adb9 |
ded208991de34fe4bb7dd725097f1c7e |
| subnet | gauge | subnet |
09105ed1-1654-4b5f-8c12-f0f2666fa304 | 15c0240142084d16b3127d6f844adb9 |
ded208991de34fe4bb7dd725097f1c7e |
| subnet | gauge | subnet |
4bf00aac-407c-4266-a048-6ff52721ad82 | 15c0240142084d16b3127d6f844adb9 |
ded208991de34fe4bb7dd725097f1c7e |
| subnet.create | delta | subnet |
09105ed1-1654-4b5f-8c12-f0f2666fa304 | 15c0240142084d16b3127d6f844adb9 |
ded208991de34fe4bb7dd725097f1c7e |
| subnet.create | delta | subnet |
4bf00aac-407c-4266-a048-6ff52721ad82 | 15c0240142084d16b3127d6f844adb9 |
ded208991de34fe4bb7dd725097f1c7e |

```



NOTE: The `ceilometer meter-list` command lists the meters only if images have been created, or instances have been launched, or if subnet, port, floating IP addresses have been created, otherwise the meter list is empty. You also need to source the `/etc/contrail/openstackrc` file when executing the command.

Contrail Ceilometer Plugin

The Contrail Ceilometer plugin adds the capability to meter the traffic statistics of floating IP addresses in Ceilometer. The following meters for each floating IP resource are added by the plugin in Ceilometer.

```

ip.floating.receive.bytes
ip.floating.receive.packets
ip.floating.transmit.bytes
ip.floating.transmit.packets

```

The Contrail Ceilometer plugin configuration is done in the `/etc/ceilometer/pipeline.yaml` file when Contrail is installed by the Fabric provisioning scripts.

The following example shows the configuration that is added to the file:

```

sources:
- name: contrail_source
  interval: 600
  meters:
  - "ip.floating.receive.packets"
  - "ip.floating.transmit.packets"
  - "ip.floating.receive.bytes"
  - "ip.floating.transmit.bytes"
  resources:
  - contrail://<IP-address-of-Contrail-Analytics-Node>:8081
  sinks:
  - contrail_sink
sinks:
- name: contrail_sink
  publishers:
  - rpc://
  transformers:

```

The following example shows the Ceilometer meter list output for the floating IP meters:

Name	Type	Unit	Resource ID	User ID
Project ID				
ip.floating.receive.bytes	cumulative	B		
451c93eb-e728-4ba1-8665-6e7c7a8b49e2	None			None
ip.floating.receive.bytes	cumulative	B		
9cf76844-8f09-4518-a09e-e2b8832bf894	None			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      |
| Duration End          |
+-----+
| 0      | 2015-02-13T19:50:40.795000 | 2015-02-13T19:50:40.795000 | 2892 |
| 0.0 | 325.0 | 1066.0 | 0.368603042877 | 439069.674 | 2015-02-13T19:50:40.795000
| 2015-02-18T21:48:30.469000 |
+-----+
```

```
+-----+
| Resource ID          | Name                  | Type
| Volume | Unit  | Timestamp          |
+-----+
| 9cf76844-8f09-4518-a09e-e2b8832bf894 | ip.floating.receive.packets |
cumulative | 208.0 | packet | 2015-02-18T21:48:30.469000 |
| 451c93eb-e728-4ba1-8665-6e7c7a8b49e2 | ip.floating.receive.packets |
cumulative | 325.0 | packet | 2015-02-18T21:48:28.354000 |
| 9cf76844-8f09-4518-a09e-e2b8832bf894 | ip.floating.receive.packets |
cumulative | 0.0   | packet | 2015-02-18T21:38:30.350000 |
```

Ceilometer Installation and Provisioning

There are two scenarios possible for Contrail Ceilometer plugin installation.

1. If you install your own OpenStack distribution, you can install the Contrail Ceilometer plugin on the OpenStack controller node.
2. When using Contrail Cloud services, the Ceilometer controller services are installed and provisioned as part of the OpenStack controller node and the compute agent service is installed as part of the compute node when **enable_ceilometer** is set as **True** in the cluster **config** or **testbed** files.

CHAPTER 14

Using Contrail Analytics to Monitor and Troubleshoot the Network

- [Monitoring the System on page 262](#)
- [Debugging Processes Using the Contrail Introspect Feature on page 264](#)
- [Monitor > Infrastructure > Dashboard on page 268](#)
- [Monitor > Infrastructure > Control Nodes on page 270](#)
- [Monitor > Infrastructure > Virtual Routers on page 277](#)
- [Monitor > Infrastructure > Analytics Nodes on page 287](#)
- [Monitor > Infrastructure > Config Nodes on page 292](#)
- [Monitor > Networking on page 295](#)
- [Query > Flows on page 303](#)
- [Query > Logs on page 310](#)
- [Example: Debugging Connectivity Using Monitoring for Troubleshooting on page 315](#)

Monitoring the System

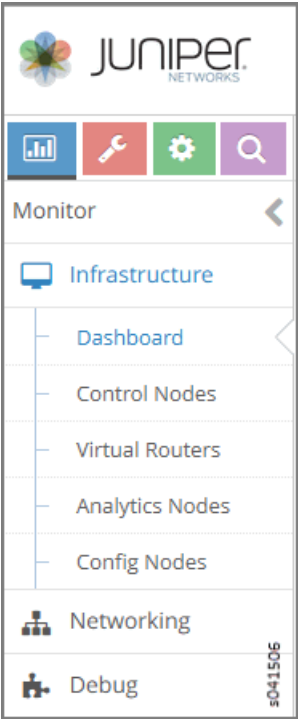
The **Monitor** icon on the Contrail Controller provides numerous options so you can view and analyze usage and other activity associated with all nodes of the system, through the use of reports, charts, and detailed lists of configurations and system activities.

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 116 on page 262](#)

Figure 116: Monitor Menu



See [Table 43 on page 262](#) for descriptions of the items available under each of the menu options from the **Monitor** icon.

Table 43: 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 268.

Table 43: 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 270.</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 277.</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 287.</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 292.</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 295.</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 295.</p>
Networking > Projects	<p>View essential information about projects in the system including name, associated networks, and traffic in and out.</p>

Table 43: 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 268](#)
- [Monitor > Infrastructure > Control Nodes on page 270](#)
- [Monitor > Infrastructure > Virtual Routers on page 277](#)
- [Monitor > Networking on page 295](#)
- [Query > Logs on page 310](#)
- [Query > Flows on page 303](#)

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.

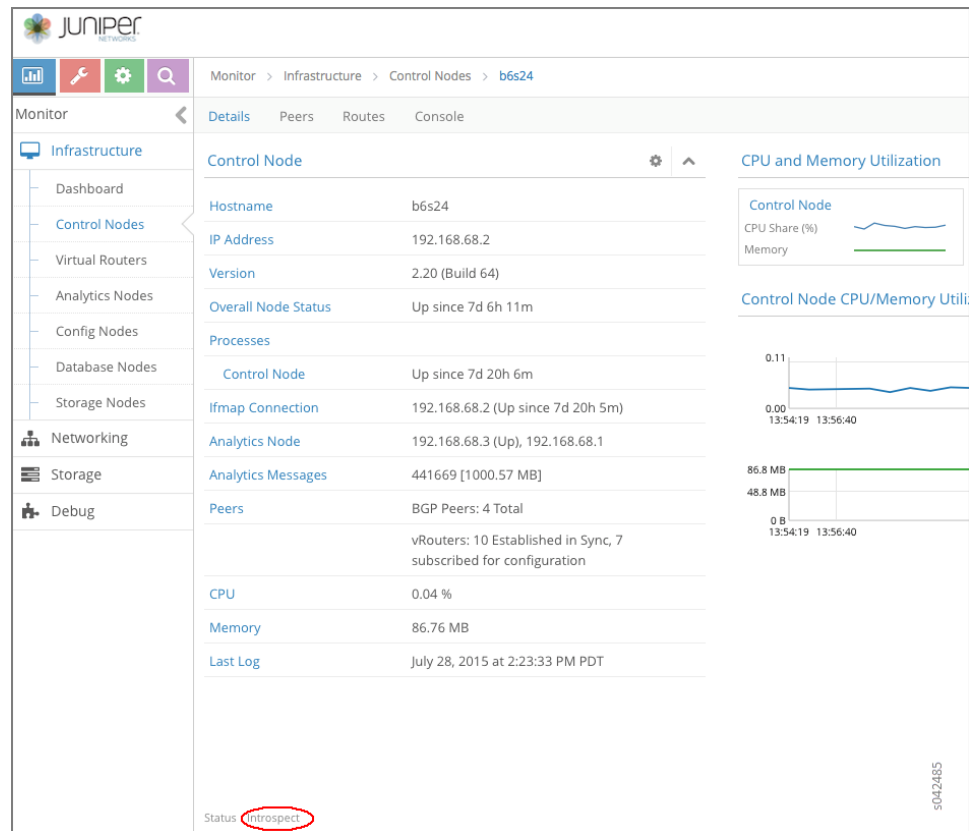
You can display the HTTP introspect of a Contrail daemon directly by accessing the following Introspect ports:

- `<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 117 on page 265 shows the *contrail-control* infrastructure page. Notice the Introspect link at the bottom of the Control Nodes Details tab window.

Figure 117: 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 118 on page 266 shows the Controller Introspect window.

Figure 118: Controller Introspect Window

Figure 119 on page 266 shows an example of the BGP Peer (bgp_peer.xml) Introspect page.

Figure 119: BGP Peer Introspect Page

Figure 120 on page 267 shows an example of the BGP Neighbor Summary Introspect page.

Figure 120: 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 121 on page 268 shows an example of the Agent (agent.xml) Introspect page.

Figure 121: 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 q k u d
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 q k u d

Monitor > Infrastructure > Dashboard

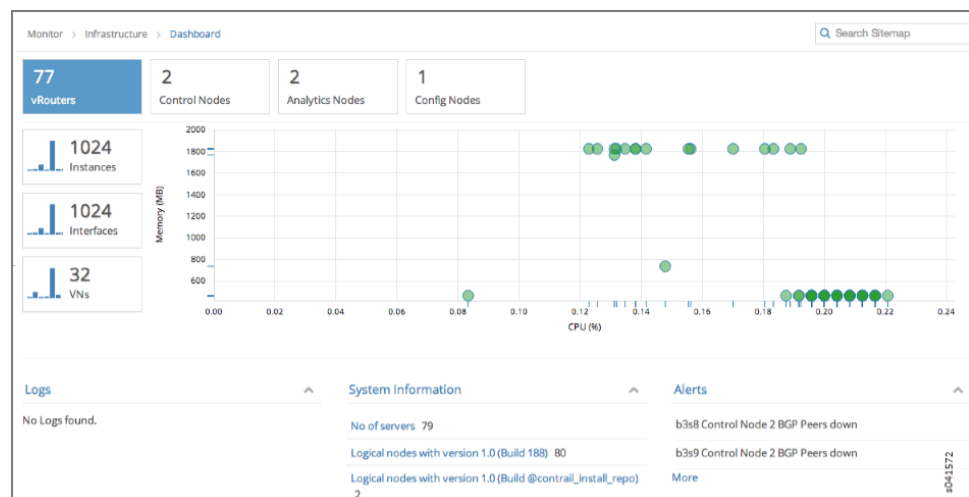
Use **Monitor > Infrastructure > Dashboard** to get an “at-a-glance” view of the system infrastructure components, including the numbers of virtual routers, control nodes, analytics nodes, and config nodes currently operational, a bubble chart of virtual routers showing the CPU and memory utilization, log messages, system information, and alerts.

- [Monitor Dashboard on page 268](#)
- [Monitor Individual Details from the Dashboard on page 269](#)
- [Using Bubble Charts on page 269](#)
- [Color-Coding of Bubble Charts on page 270](#)

Monitor Dashboard

Click **Monitor > Infrastructure > Dashboard** on the left to view the **Dashboard**. See [Figure 122 on page 268](#).

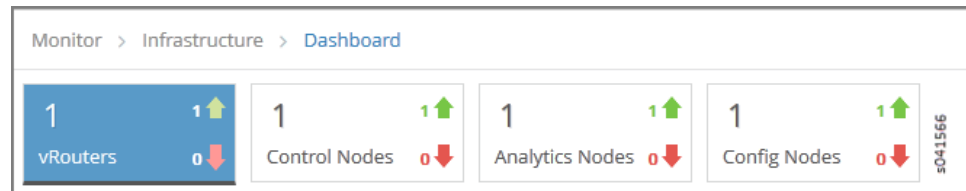
Figure 122: Monitor > Infrastructure > Dashboard



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 123 on page 269](#). 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 123: Dashboard Summary Boxes



Detailed information about monitoring each of the areas represented by the boxes is provided in the links in [Table 44 on page 269](#).

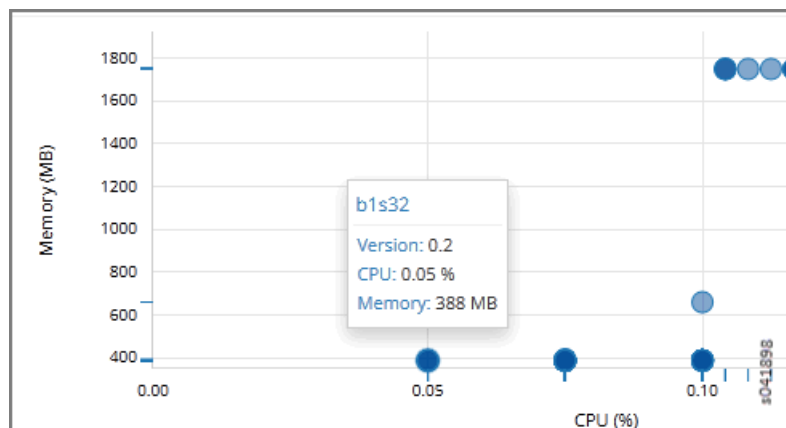
Table 44: Dashboard Summary Boxes

Box	For More Information
vRouters	"Monitor > Infrastructure > Virtual Routers" on page 277
Control Nodes	"Monitor > Infrastructure > Control Nodes" on page 270
Analytics Nodes	"Monitor > Infrastructure > Analytics Nodes" on page 287
Config Nodes	"Monitor > Infrastructure > Config Nodes" on page 292

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 124 on page 270](#). You can click through the summary information to get more details about the component.

Figure 124: 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 277](#)
- [Monitor > Infrastructure > Control Nodes on page 270](#)
- [Monitor > Infrastructure > Analytics Nodes on page 287](#)
- [Monitor > Infrastructure > Config Nodes on page 292](#)

Monitor > Infrastructure > Control Nodes

Use **Monitor > Infrastructure > Control Nodes** to gain insight into usage statistics for control nodes.

- [Monitor Control Nodes Summary on page 271](#)
- [Monitor Individual Control Node Details on page 271](#)
- [Monitor Individual Control Node Console on page 273](#)
- [Monitor Individual Control Node Peers on page 275](#)
- [Monitor Individual Control Node Routes on page 276](#)

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 125 on page 271](#). See [Table 45 on page 271](#) for descriptions of the fields on this screen.

Figure 125: Control Nodes Summary



Table 45: Control Nodes Summary Fields

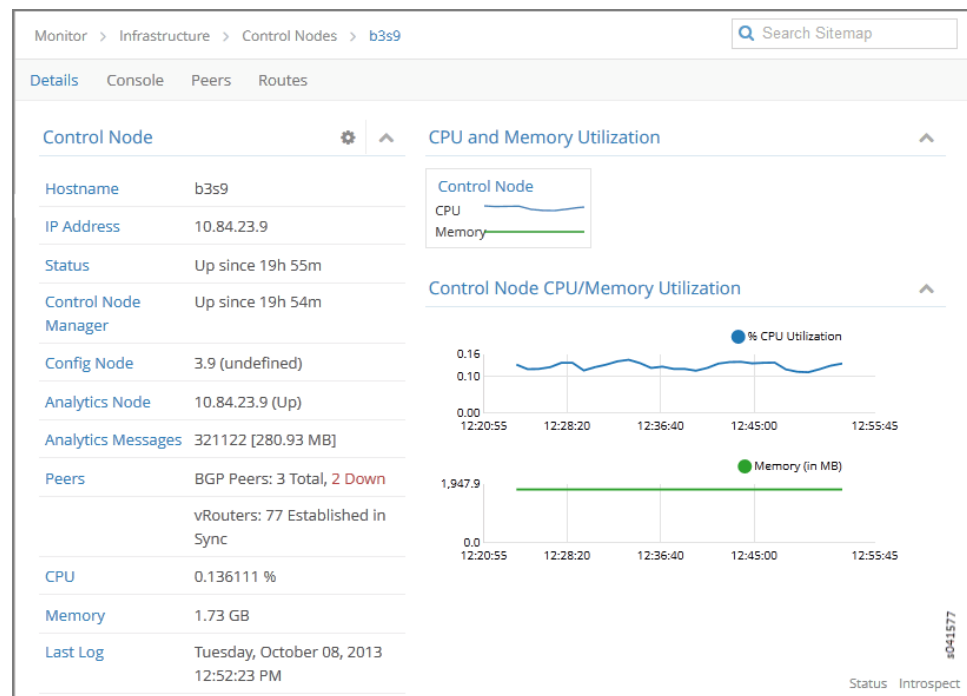
Field	Description
Host name	The name of the control node.
IP Address	The IP address of the control node.
Version	The software version number that is installed on the control node.
Status	The current operational status of the control node — Up or Down.
CPU (%)	The CPU percentage currently in use by the 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 126 on page 272](#).

Figure 126: 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 46 on page 272](#) for descriptions of the fields on this tab.

Table 46: 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 46: 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 127 on page 273](#).

Figure 127: 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 47 on page 273](#) for descriptions of the fields on the **Console** tab screen.

Table 47: 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 .

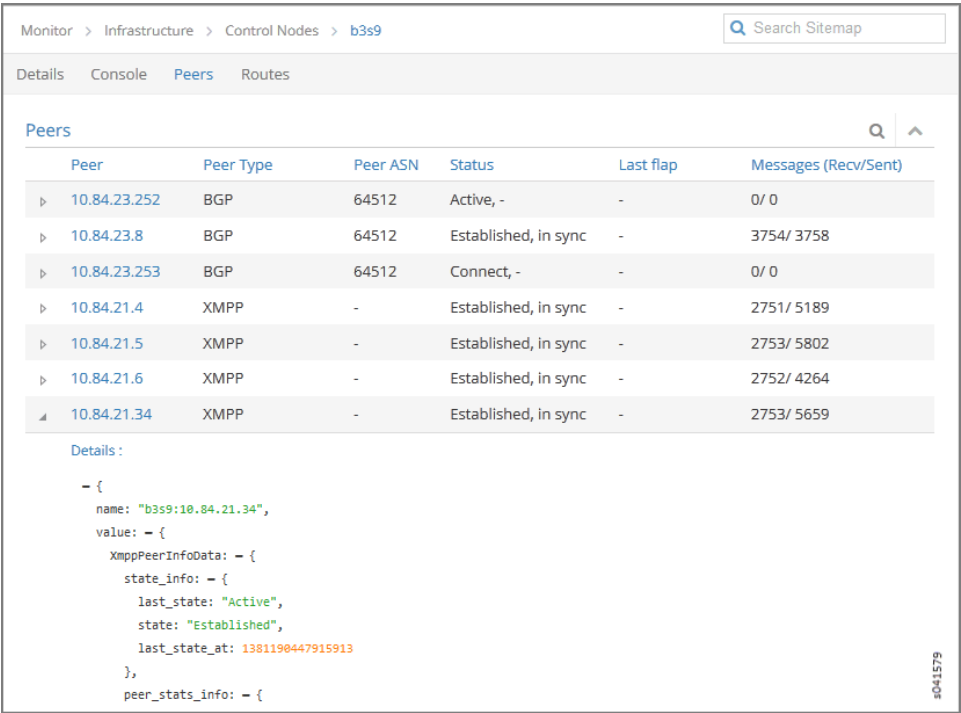
Table 47: Control Node: Console Tab Fields (*continued*)

Field	Description
Log Category	Select a log category to display: All _default_ XMPP 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 128 on page 275](#).

Figure 128: Individual Control Node—Peers Tab



See [Table 48 on page 275](#) for descriptions of the fields on the **Peers** tab screen.

Table 48: 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 129 on page 276](#).

Figure 129: Individual Control Node—Routes Tab

Routing Table	Prefix	Protocol	Source	Next hop	Label	Secur...	Origin VN
bgp.l3vpn.0	10.84.21.1:13:192.168.30.240/32	XMPP	b1s1	10.84.21.1	28	3	default-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 49 on page 276](#) for descriptions of the fields on the **Routes** tab screen.

Table 49: 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 49: 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

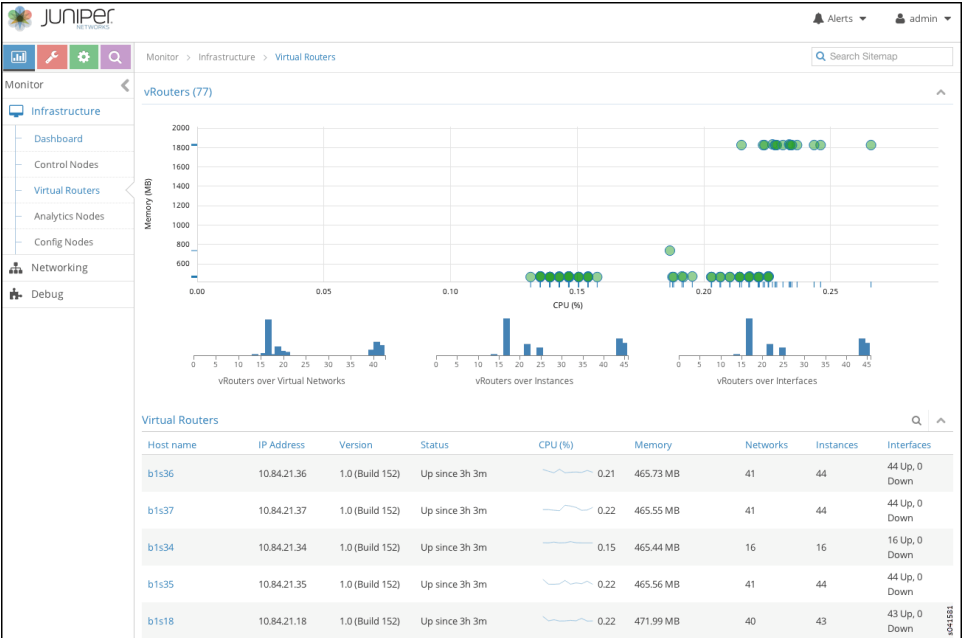
- [Monitor vRouters Summary on page 278](#)
- [Monitor Individual vRouters Tabs on page 279](#)
- [Monitor Individual vRouter Details Tab on page 279](#)
- [Monitor Individual vRouters Interfaces Tab on page 280](#)
- [Monitor Individual vRouters Networks Tab on page 281](#)
- [Monitor Individual vRouters ACL Tab on page 282](#)
- [Monitor Individual vRouters Flows Tab on page 283](#)

- [Monitor Individual vRouters Routes Tab on page 284](#)
- [Monitor Individual vRouter Console Tab on page 285](#)

Monitor vRouters Summary

Click **Monitor > Infrastructure > Virtual Routers** to view the **vRouters** summary screen. See [Figure 130 on page 278](#).

Figure 130: vRouters Summary



See [Table 50 on page 278](#) for descriptions of the fields on the **vRouters Summary** screen.

Table 50: 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.
Instances	The total number of instances for this vRouter.

Table 50: vRouters Summary Fields (*continued*)

Field	Description
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 131 on page 279](#).
- **Console**—similar display of information as on individual control nodes **Console** tab. See [Figure 137 on page 286](#).
- **Interfaces**—details about associated interfaces. See [Figure 132 on page 281](#).
- **Networks**—details about associated networks. See [Figure 133 on page 282](#).
- **ACL**—details about access control lists. See [Figure 134 on page 283](#).
- **Flows**—details about associated traffic flows. See [Figure 135 on page 284](#).
- **Routes**—details about associated routes. See [Figure 136 on page 285](#).

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 131 on page 279](#). See [Table 51 on page 279](#) for descriptions of the fields on this tab.

Figure 131: Individual vRouters—Details Tab

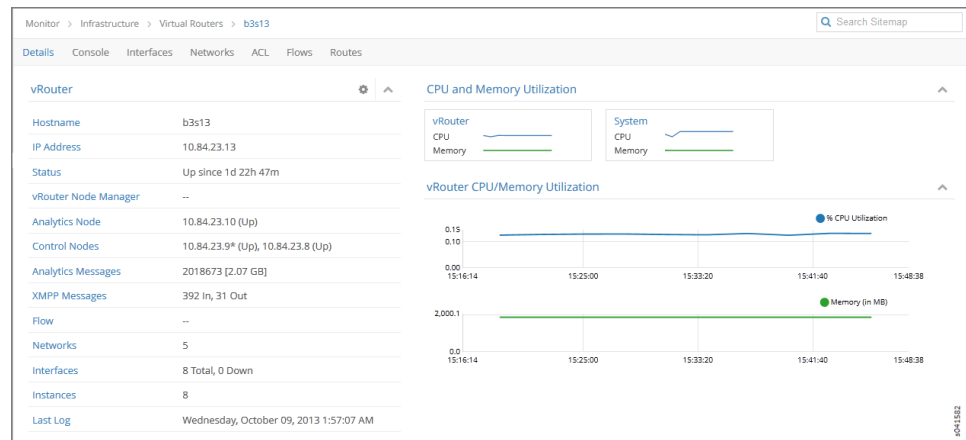


Table 51: vRouters Details Tab Fields

Field	Description
Hostname	The hostname of the vRouter.

Table 51: vRouters Details Tab Fields (*continued*)

Field	Description
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 132 on page 281](#). See [Table 52 on page 281](#) for descriptions of the fields on the **Interfaces** tab screen.

Figure 132: Individual vRouters—Interfaces Tab

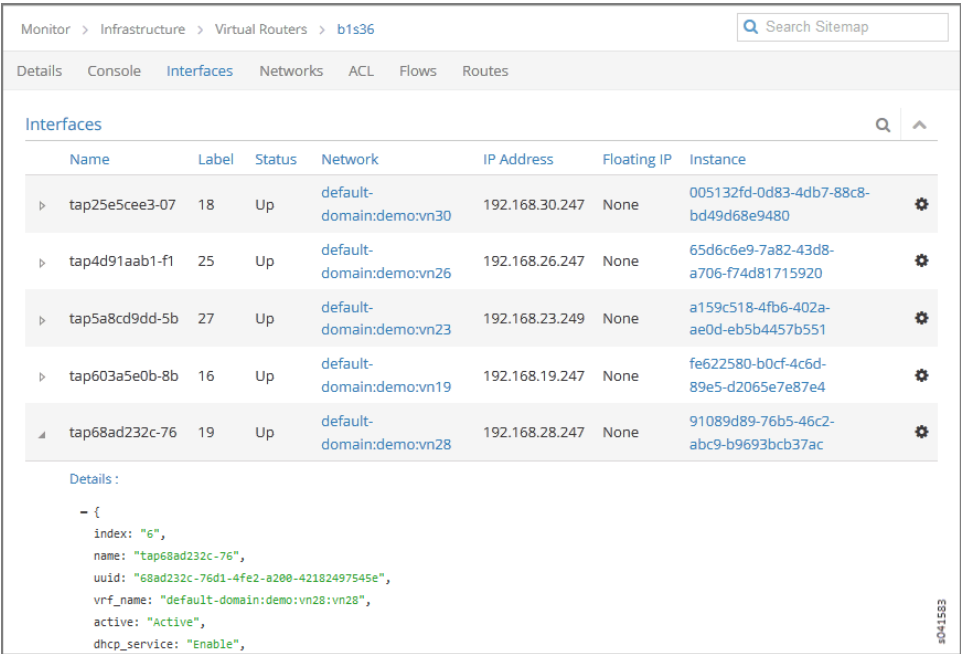


Table 52: vRouters: Interfaces Tab Fields

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.

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 133 on page 282](#). See [Table 53 on page 282](#) for descriptions of the fields on the **Networks** tab screen.

Figure 133: 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 53: 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 134 on page 283](#). See [Table 54 on page 283](#) for descriptions of the fields on the **ACL** tab screen.

Figure 134: 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 54: 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 135 on page 284. See Table 55 on page 284 for descriptions of the fields on the **Flows** tab screen.

Figure 135: Individual vRouters—Flows Tab

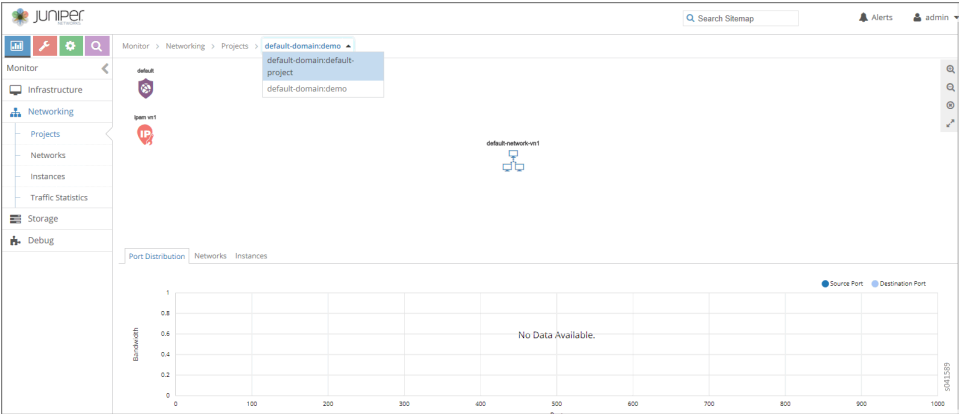


Table 55: vRouters: Flows Tab Fields

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 136 on page 285. See Table 56 on page 285 for descriptions of the fields on the **Routes** tab screen.

Figure 136: Individual vRouters—Routes Tab

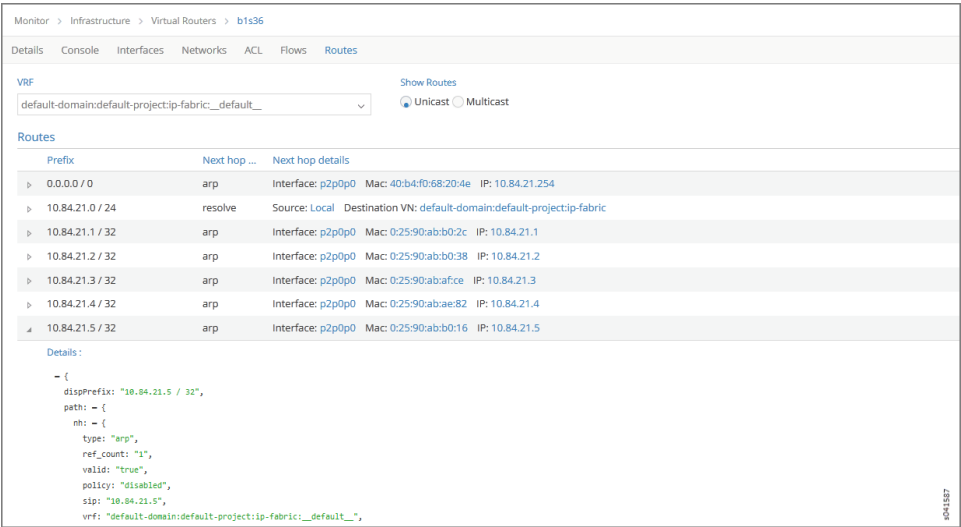


Table 56: vRouters: Routes Tab Fields

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 137 on page 286](#). See [Table 57 on page 286](#) for descriptions of the fields on the **Console** tab screen.

Figure 137: 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-domaindemo: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-domaindemo: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-domaindemo: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-domaindemo: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-domaindemo: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-domaindemo:vn31:vn31 10.84.23.9

Table 57: 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 • 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 57: 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 287](#)
- [Monitor Analytics Individual Node Details Tab on page 288](#)
- [Monitor Analytics Individual Node Generators Tab on page 289](#)
- [Monitor Analytics Individual Node QE Queries Tab on page 290](#)
- [Monitor Analytics Individual Node Console Tab on page 291](#)

Monitor Analytics Nodes

Select **Monitor > Infrastructure > Analytics Nodes** to view a summary of activities for the analytics nodes; see [Figure 138 on page 288](#). See [Table 58 on page 288](#) for descriptions of the fields on the analytics summary.

Figure 138: Analytics Nodes Summary



Table 58: 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 139 on page 289](#).

See [Table 59 on page 289](#) for descriptions of the fields on this screen.

Figure 139: Monitor Analytics Individual Node Details Tab

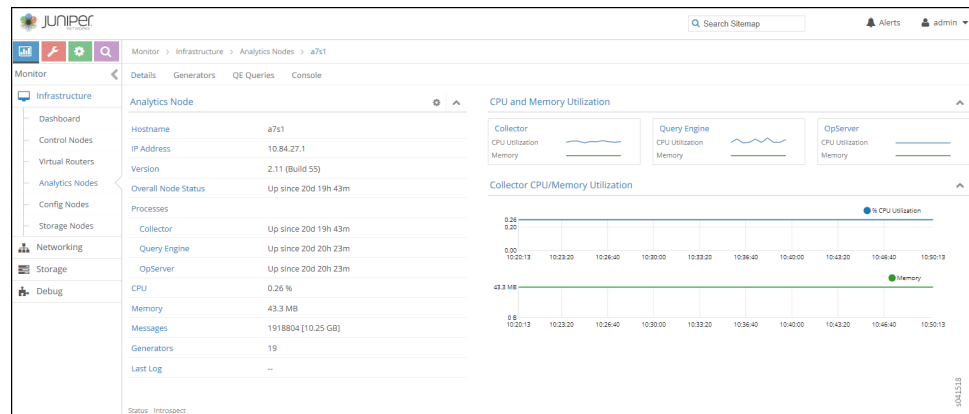


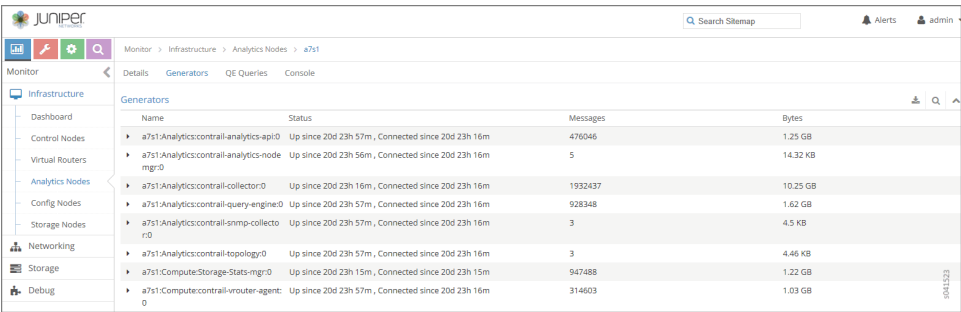
Table 59: Monitor Analytics Individual Node Details Tab Fields

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 140 on page 290](#). Click the expansion arrow next to any generator name to reveal more details. See [Table 60 on page 290](#) for descriptions of the fields on the **Peers** tab screen.

Figure 140: 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 60: 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 141 on page 290](#).

See [Table 61 on page 290](#) for descriptions of the fields on the **QE Queries** tab screen.

Figure 141: Individual Analytics Node—QE QueriesTab



Enqueue Time	Query	Progress
No QE Queries to display		

Table 61: 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 142 on page 291](#). See [Table 62 on page 291](#) for descriptions of the fields on the **Console** tab screen.

Figure 142: Analytics Individual Node—Console Tab

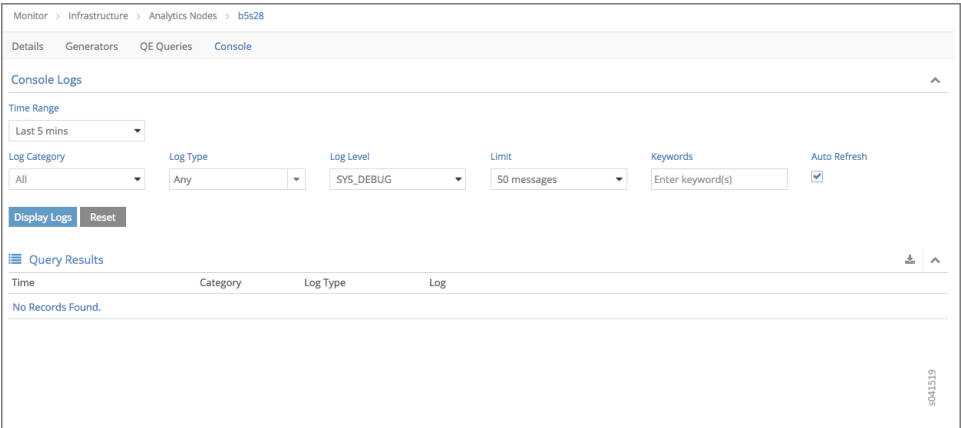


Table 62: Monitor Analytics Individual Node Console Tab Fields

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: All _default_ XMPP 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
Keywords	Enter any text string to search for and display logs containing that string.

Table 62: 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 292](#)
- [Monitor Individual Config Node Details on page 293](#)
- [Monitor Individual Config Node Console on page 294](#)

Monitor Config Nodes

Select **Monitor > Infrastructure > Config Nodes** to view a summary of activities for the analytics nodes. See [Figure 143 on page 293](#).

Figure 143: Config Nodes Summary

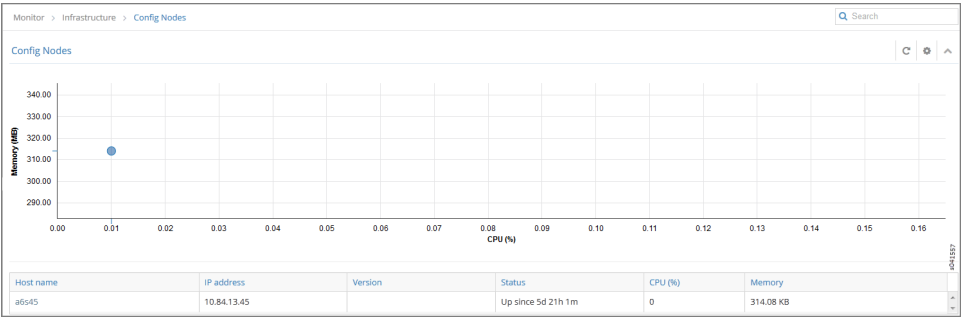


Table 63 on page 293 describes the fields in the Config Nodes summary.

Table 63: 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 144 on page 293](#).

Figure 144: Individual Config Nodes— Details Tab

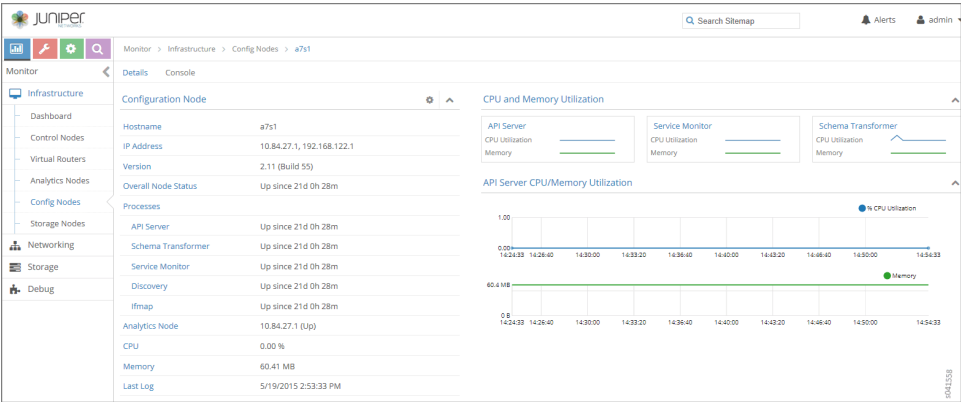


Table 64 on page 294 describes the fields on the Details screen.

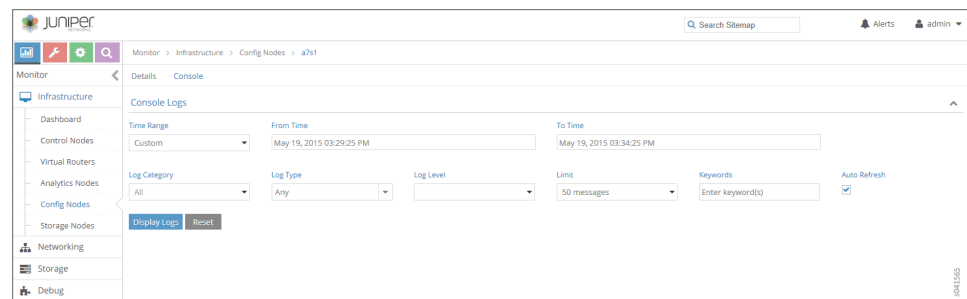
Table 64: Individual Config Nodes— Details Tab Fields

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, and the like.
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 145 on page 294](#).

Figure 145: Individual Config Node—Console Tab



See [Table 65 on page 294](#) for descriptions of the fields on the **Console** tab screen.

Table 65: 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 65: 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

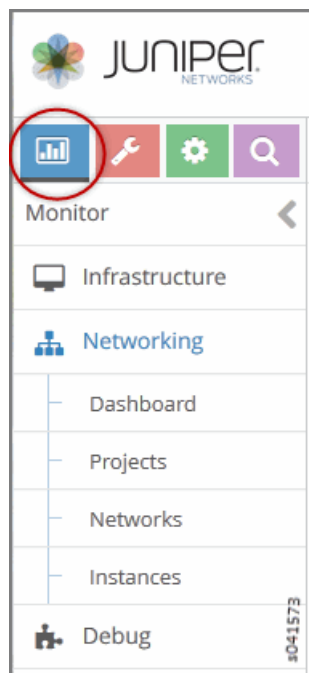
The **Monitor -> Networking** pages give an overview of the networking traffic statistics and health of domains, projects within domains, virtual networks within projects, and virtual machines within virtual networks.

- [Monitor > Networking Menu Options on page 295](#)
- [Monitor -> Networking -> Dashboard on page 296](#)
- [Monitor > Networking > Projects on page 297](#)
- [Monitor Projects Detail on page 298](#)
- [Monitor > Networking > Networks on page 300](#)

Monitor > Networking Menu Options

Figure 146 on page 296 shows the menu options available under **Monitor > Networking**.

Figure 146: 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 147 on page 296](#).

Figure 147: Traffic Statistics for Domain Window

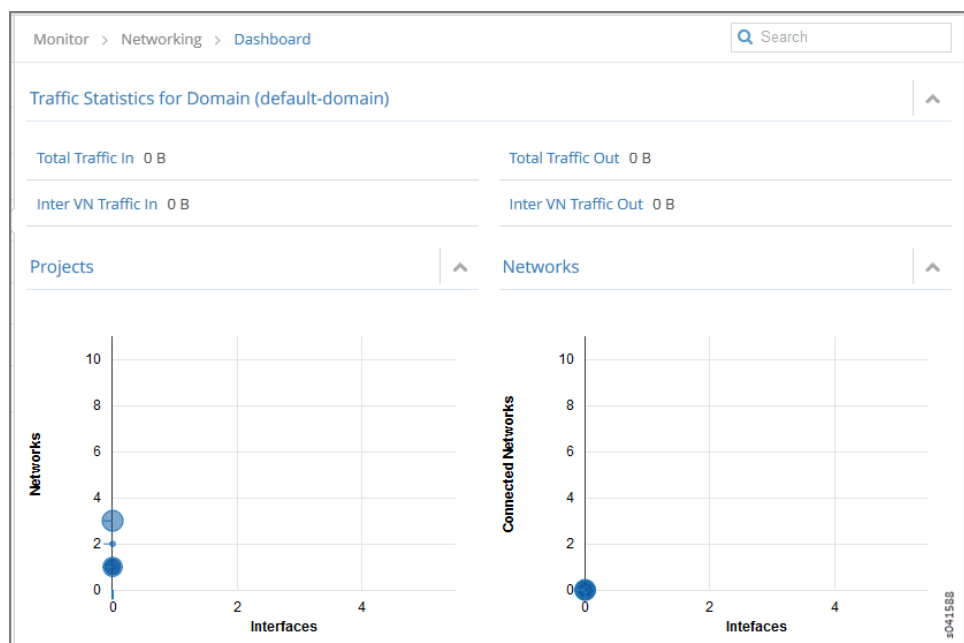


Table 66 on page 297 describes the fields in the Traffic Statistics for Domain window.

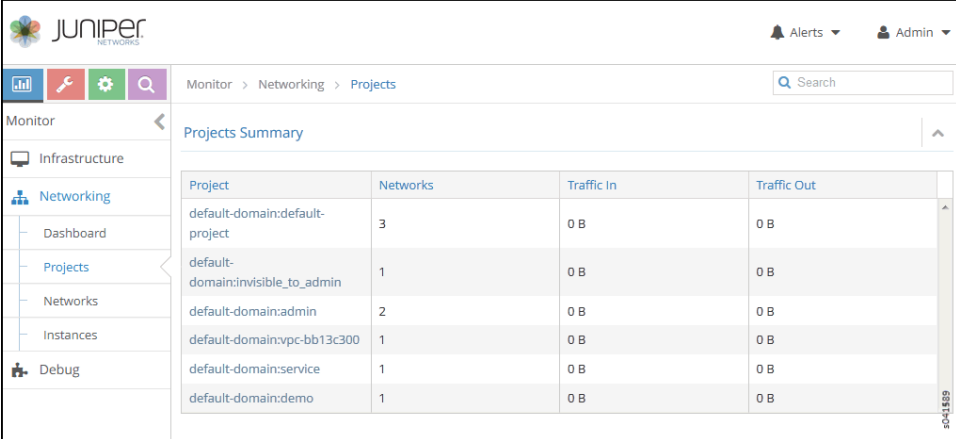
Table 66: Projects Summary Fields

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 148 on page 297.

Figure 148: 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 67 on page 297 for descriptions of the fields on this screen.

Table 67: 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.

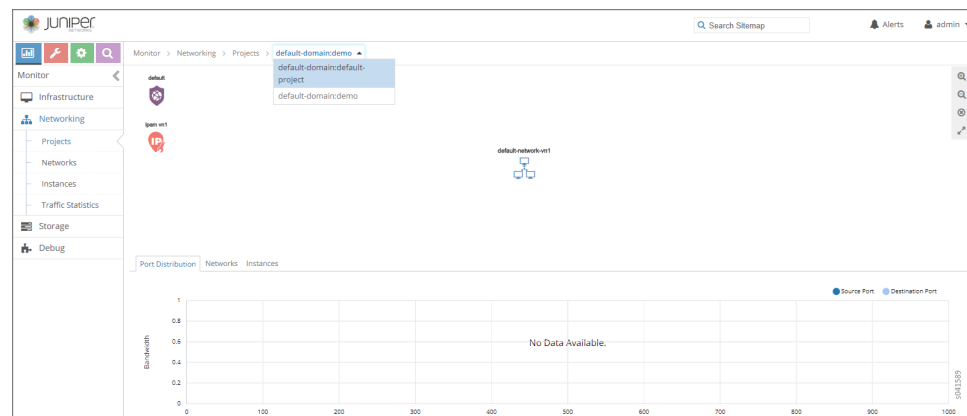
Table 67: Projects Summary Fields (*continued*)

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 149 on page 298](#). Hover over any of the connections to get more details.

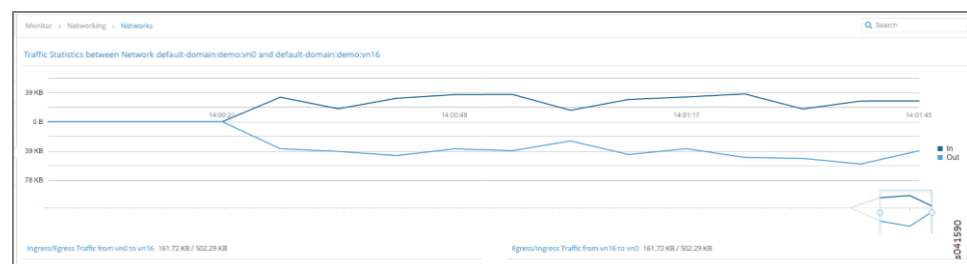
Figure 149: Monitor Projects Connectivity Details



In the Connectivity Details window you can click the links between the virtual networks to view the traffic statistics between the virtual networks.

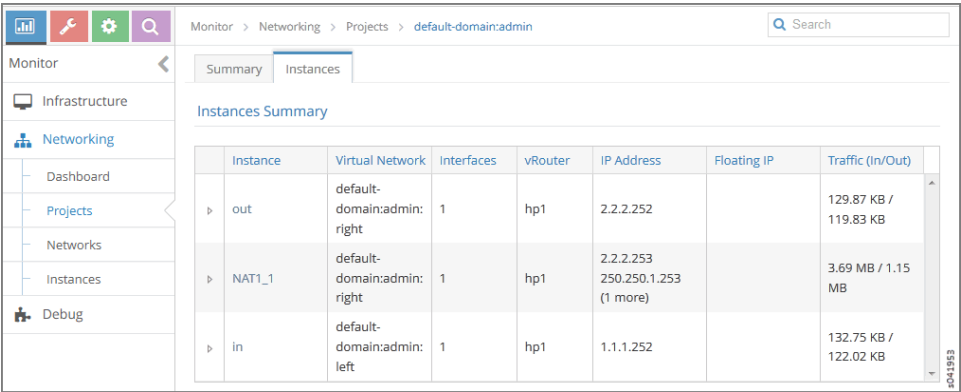
The Traffic Statistics information is also available when you select **Monitor > Networking > Networks** as shown in [Figure 150 on page 298](#).

Figure 150: 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 151: Projects Instances Summary



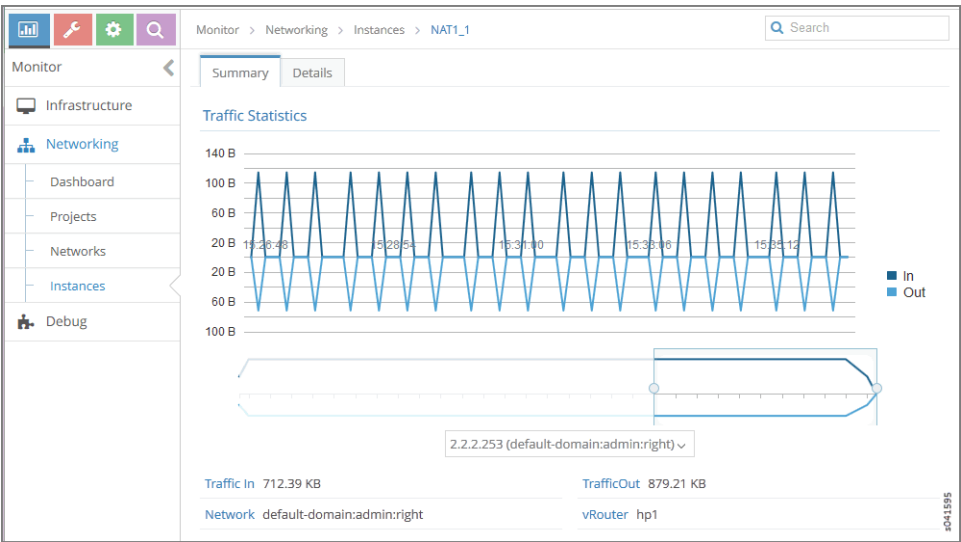
See Table 3 for a description of the fields on this screen.

Table 68: 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 152 on page 300](#).

Figure 152: Instance Traffic Statistics



Monitor > Networking > Networks

Select **Monitor > Networking > Networks** to view a summary of the virtual networks in your system. See [Figure 153 on page 300](#).

Figure 153: Network Summary

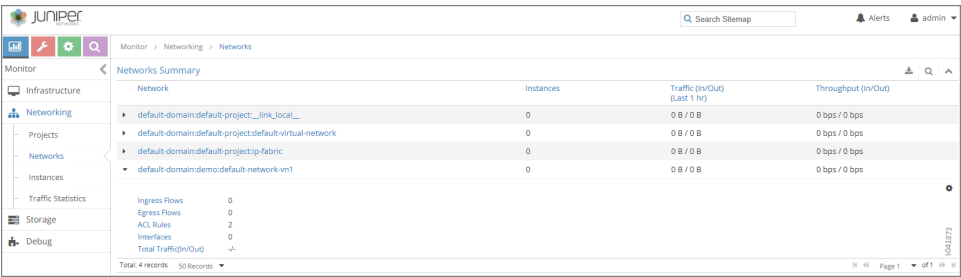


Table 69: Network Summary Fields

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 154 on page 301 shows the **Summary** tab for an individual network, which displays connectivity details and traffic statistics for the selected network.

Figure 154: Individual Network Connectivity Details—Summary Tab

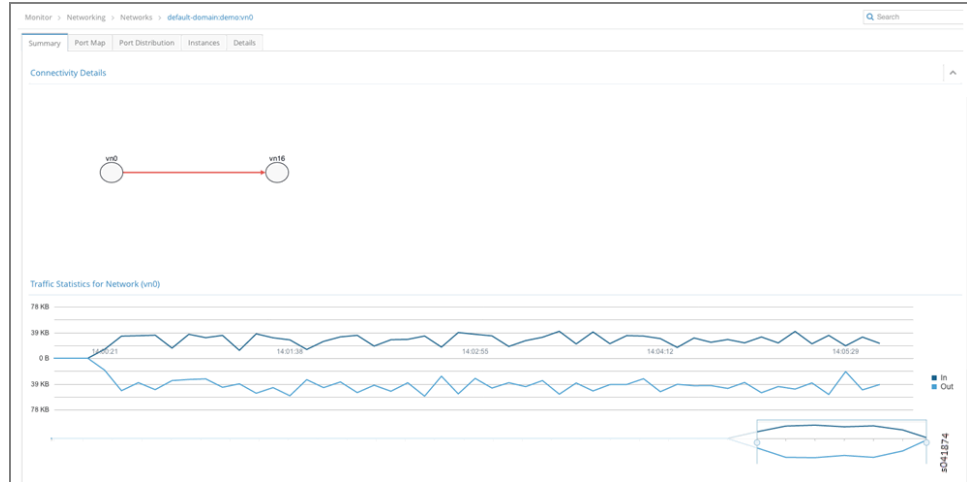


Figure 155 on page 301 shows the **Port Map** tab for an individual network, which displays the relative distribution of traffic for this network by protocol, by port.

Figure 155: Individual Network— Port Map Tab

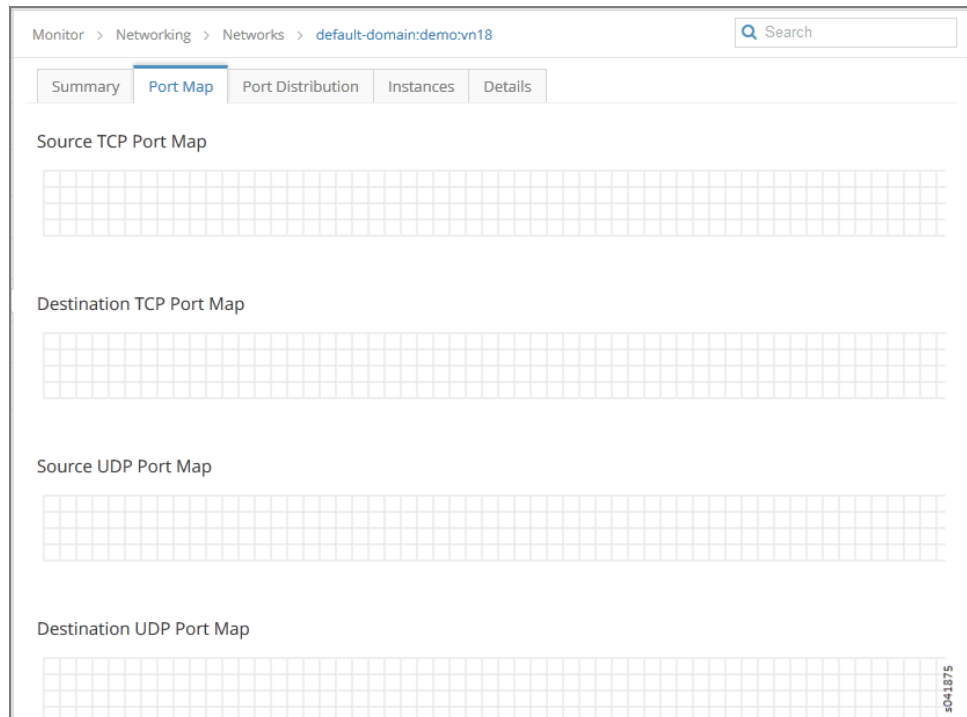


Figure 156 on page 302 shows the **Port Distribution** tab for an individual network, which displays the relative distribution of traffic in and out by source port and destination port.

Figure 156: Individual Network-- Port Distribution Tab

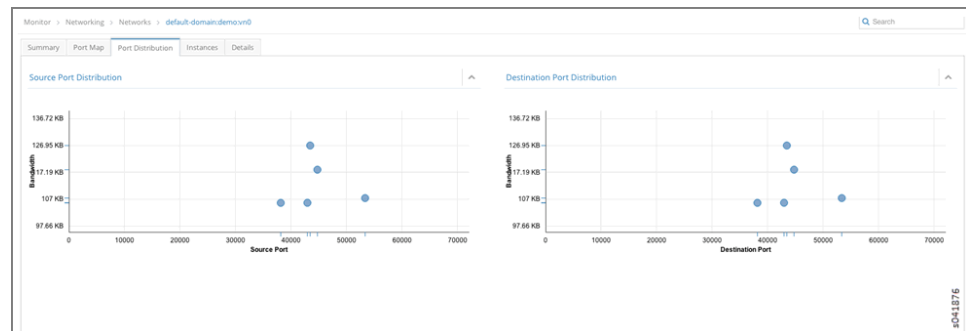


Figure 157 on page 302 shows the **Instances** tab for an individual network, which displays details for each instance associated with this network, including the number of interfaces, the associated vRouter, the instance IP address, and the volume of traffic in and out.

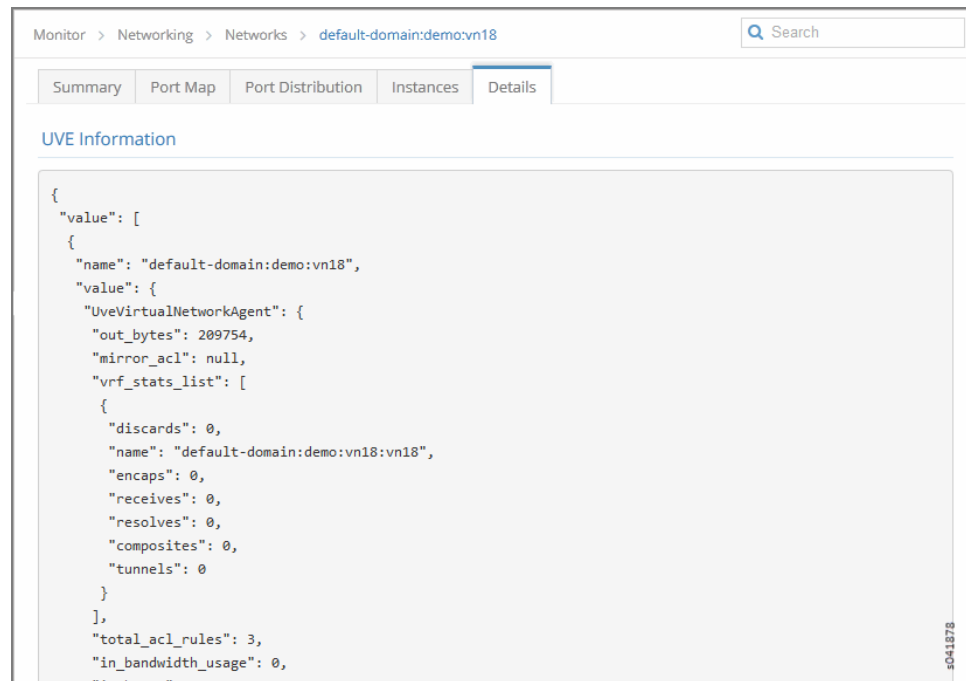
Additionally, you can click the arrow near the instance name to reveal even more details about the instance—the interfaces and their addresses, UUID, CPU (usage), and memory used of the total amount available.

Figure 157: Individual Network Instances Tab

Monitor > Networking > Networks > default-domain:demo:vn18						
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 158 on page 303 shows the **Details** tab for an individual network, which displays the code used to define this network --the User Virtual Environment (UVE) code.

Figure 158: 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 303](#)
- [Example: Query Flow Series on page 306](#)
- [Query > Flow Records on page 307](#)
- [Query > Flows > Query Queue on page 309](#)

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 159 on page 304](#)

Figure 159: Query Flow Series Window

The query fields available on the screen for the **Flow Series** tab are described in [Table 70 on page 304](#). Enter query data into the fields to create a SQL-like query to display and analyze flows.

Table 70: 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 160 on page 305), 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 161 on page 306), 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 70: Query Flow Series Fields (*continued*)

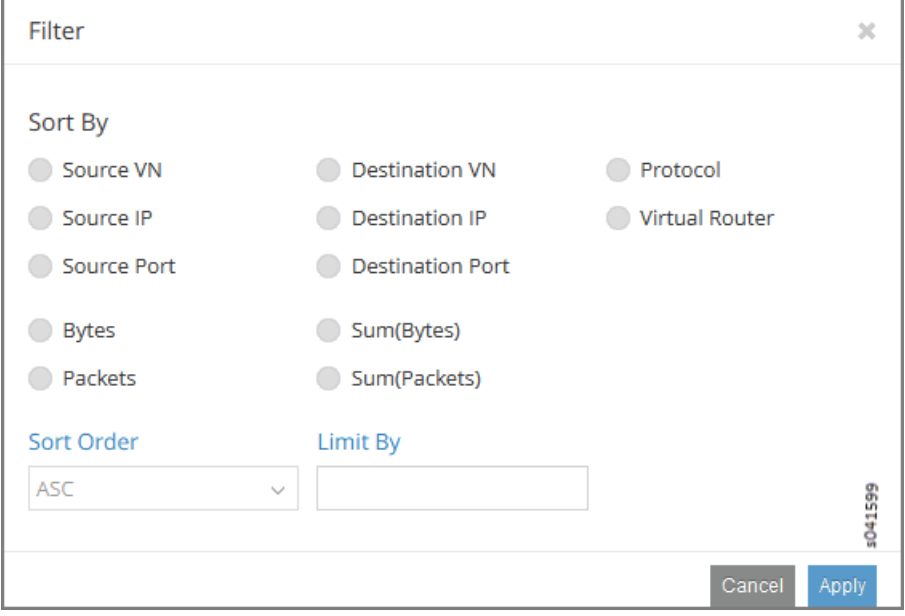
Field	Description
Export	The Export button is displayed after you click Run Query . This allows you to export the list of flows to a text .csv file.

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 160 on page 305](#). 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 160: 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 161 on page 306](#).

Figure 161: Flow Series Filter



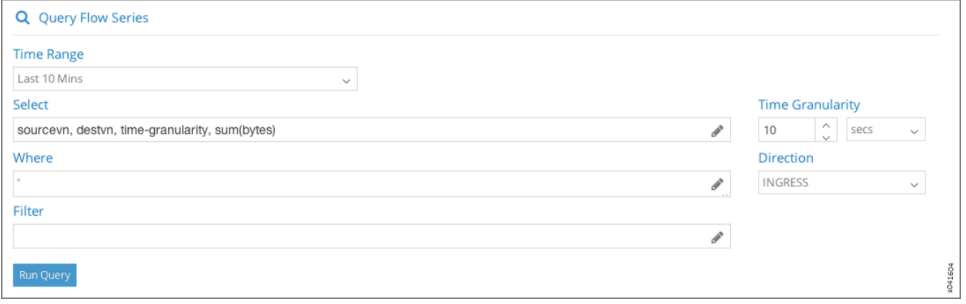
The 'Filter' dialog box contains the following elements:

- 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.
- Count:** A vertical label '1041599' on the right side of the dialog.

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 162 on page 306](#).

Figure 162: 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:** A text input field containing an asterisk: `*`.
- 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 'Cancel' button at the bottom right.

The query returns tabular time series data, see [Figure 163 on page 307](#), for the following combinations of Source VN and Dest VN:

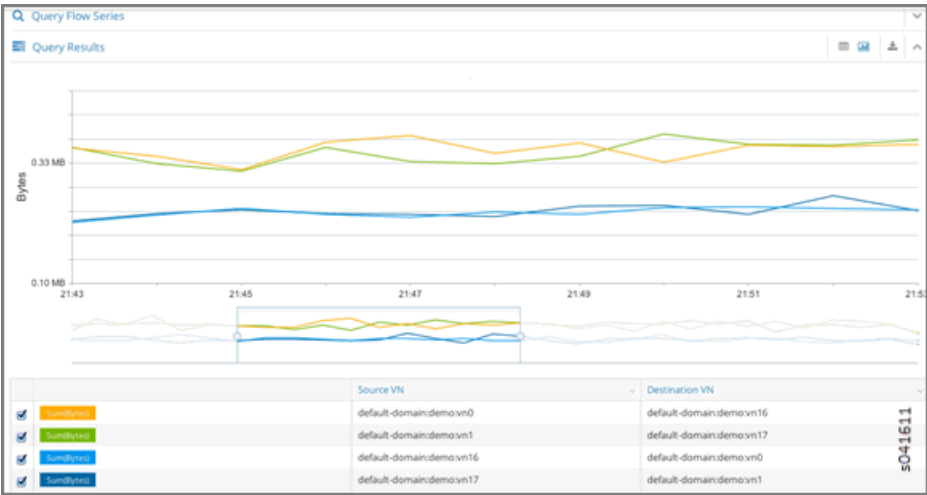
1. Flow Class 1: Source VN = default-domain:demo:front-end, Dest VN= __UNKNOWN__
2. Flow Class 2: Source VN = default-domain:demo:front-end, Dest VN=default-domain:demo:back-end

Figure 163: 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 164 on page 307](#).

Figure 164: 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 165: Flow Records

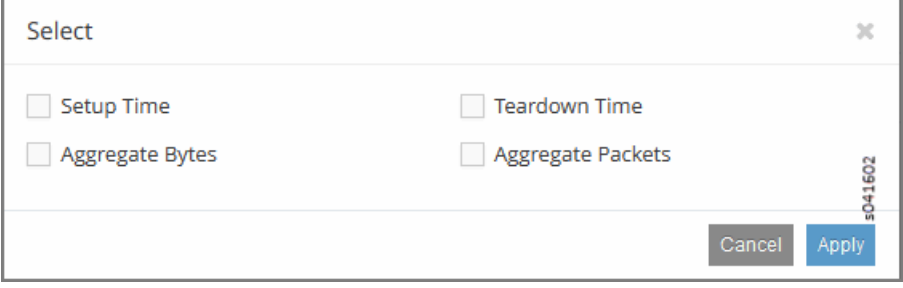
The query fields available on the screen for the **Flow Records** tab are described in [Table 71 on page 308](#). Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 71: 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 166 on page 309), 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 166 on page 309](#).

Figure 166: Flow Records Select Window



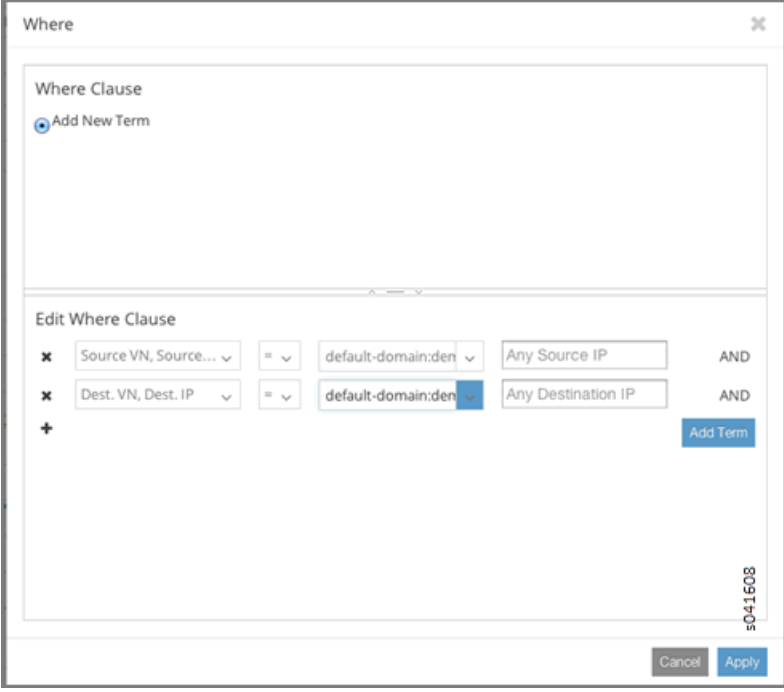
The 'Select' window contains four checkboxes: 'Setup Time', 'Teardown Time', 'Aggregate Bytes', and 'Aggregate Packets'. At the bottom right, there are 'Cancel' and 'Apply' buttons. A vertical label 's041602' is positioned on the right side of the window.

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 167: Where Clause Window



The 'Where' window shows a 'Where Clause' section with a radio button for 'Add New Term'. Below it is an 'Edit Where Clause' section. This section contains two rows of conditions:

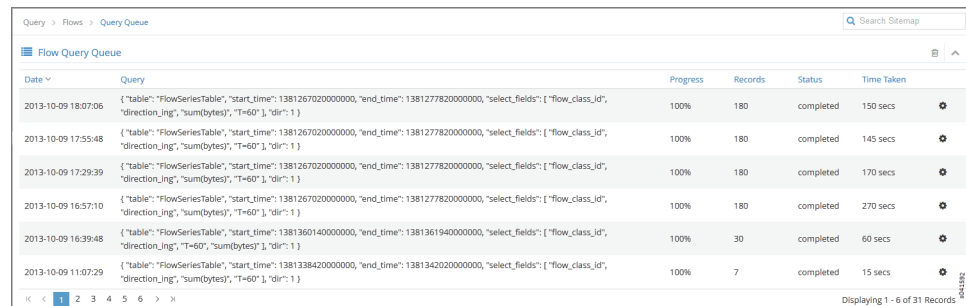
- Row 1: 'Source VN, Source...' selected, followed by '=', 'default-domain:den', 'Any Source IP', and 'AND'.
- Row 2: 'Dest. VN, Dest. IP' selected, followed by '=', 'default-domain:den', 'Any Destination IP', and 'AND'.

 There is a plus sign (+) to add more terms and an 'Add Term' button. At the bottom right, there 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 168 on page 310](#).

Figure 168: 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 72 on page 310](#). Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 72: 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.

Query > Logs

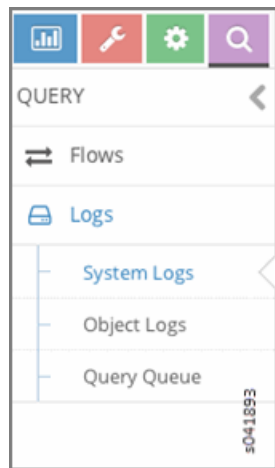
The **Query > Logs** option allows you to access the system log and object log activity of any Contrail Controller component from one central location.

- [Query > Logs Menu Options on page 310](#)
- [Query > Logs > System Logs on page 311](#)
- [Sample Query for System Logs on page 312](#)
- [Query > Logs > Object Logs on page 313](#)

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 169 on page 311](#).

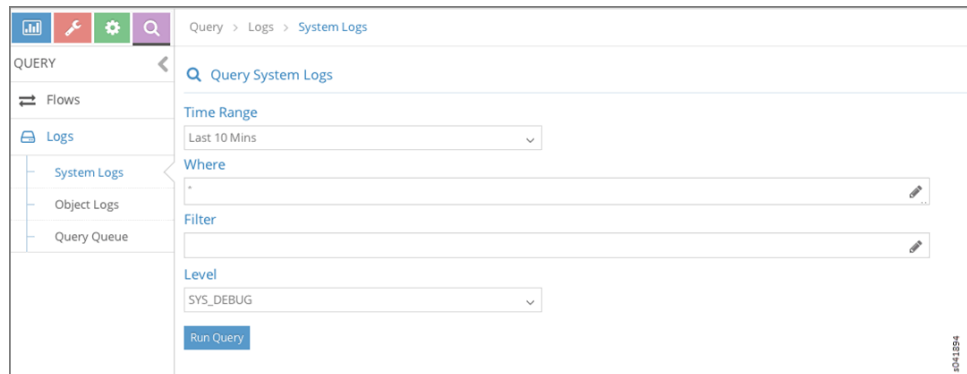
Figure 169: Query > Logs



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 170 on page 311](#).

Figure 170: Query > Logs > System Logs



The query fields available on the **Query System Logs** screen are described in [Table 73 on page 311](#).

Table 73: 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>

Table 73: Query System Logs Fields (*continued*)

Field	Description
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 171 on page 313](#).

Figure 171: Edit Where Clause

Where

Where Clause

☒ Add New Term

Edit Where Clause

✕ ModuleId = VRouterAgent AND

✕ Source = b1s16 AND

+

Add Term

OK Cancel

s041895

2. The information you defined at the Where screen displays on the **Query System Logs**. Enter any more defining information needed; see [Figure 172 on page 313](#). When finished, click **Run Query** to display the results.

Figure 172: Sample Query System Logs

Q Query System Logs

Time Range

Last 10 Mins

Where

(ModuleId = VRouterAgent AND Source = b1s16)

Filter

Level

SYS_DEBUG

Run Query

s041896

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 173 on page 314](#).

Figure 173: Query > Logs > Object Logs

The query fields available on the **Object Logs** screen are described in [Table 74 on page 314](#).

Table 74: 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 74: 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.

Example: Debugging Connectivity Using Monitoring for Troubleshooting

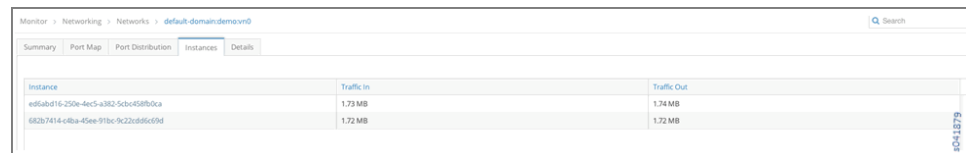
- [Using Monitoring to Debug Connectivity on page 315](#)

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.

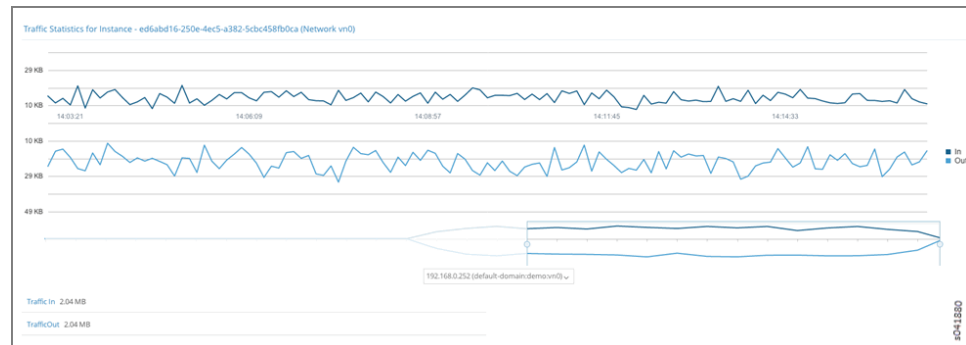
1. Navigate to **Monitor -> Networking -> Networks -> default-domain:demo:vn0**, Instance **ed6abd16-250e-4ec5-a382-5cbc458fb0ca** with IP address **192.168.0.252** in the virtual network **vn0**; see [Figure 174 on page 315](#)

Figure 174: Navigate to Instance



2. Click the instance to view **Traffic Statistics for Instance**. see [Figure 175 on page 315](#).

Figure 175: Traffic Statistics for Instance



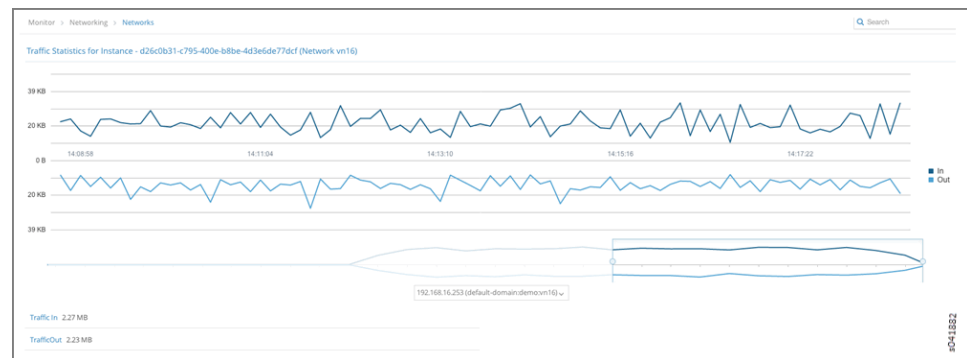
- Instance **d26c0b31-c795-400e-b8be-4d3e6de77dcf** with IP address **192.168.0.253** in the virtual network **vn16**. see [Figure 176 on page 316](#) and [Figure 177 on page 316](#).

Figure 176: Navigate to Instance



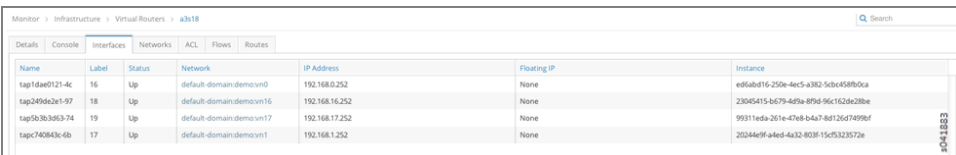
Instance	Traffic In	Traffic Out
d26c0b31-c795-400e-b8be-4d3e6de77dcf	2.18 MB	2.13 MB
23045415-b679-4d8a-8f9d-96c1629c28be	2.11 MB	2.16 MB

Figure 177: Traffic Statistics for Instance



- From **Monitor->Infrastructure->Virtual Routers->a3s18->Interfaces**, we can see that Instance **ed6abd16-250e-4ec5-a382-5cbc458fb0ca** is hosted on Virtual Router **a3s18**; see [Figure 178 on page 316](#).

Figure 178: 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-96c1629c28be
tap1b7b3d83-74	19	Up	default-domain:demo:vn17	192.168.17.252	None	99111eda-261e-47e8-b4a7-8d13d67499df
tapc740843c-6b	17	Up	default-domain:demo:vn1	192.168.1.252	None	20244ef9-afed-4a32-8039-15c75323572e

- From **Monitor->Infrastructure->Virtual Routers->a3s19->Interfaces**, we can see that Instance **d26c0b31-c795-400e-b8be-4d3e6de77dcf** is hosted on Virtual Router **a3s19**; see [Figure 179 on page 316](#).

Figure 179: Navigate to a3s19 Interfaces



Name	Label	Status	Network	IP Address	Floating IP	Instance
tap29595b2f-42	19	Up	default-domain:demo:vn16	192.168.16.253	None	d26c0b31-c795-400e-b8be-4d3e6de77dcf
tapb257621d-43	18	Up	default-domain:demo:vn1	192.168.1.253	None	eebce321-7536-4ee7-a454-eef113ac095
tapc3e3b87-66	17	Up	default-domain:demo:vn17	192.168.17.253	None	b242595-677e-4060-9478-81a4a8275d41
tapc5eaf7e3-55	16	Up	default-domain:demo:vn0	192.168.0.253	None	682b7414-c4ba-45ee-91bc-9c22cd86c9d

- 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 180 on page 317](#) and [Figure 181 on page 317](#).

Figure 180: 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 181: 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 182 on page 317](#) and [Figure 183 on page 318](#).

Figure 182: 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 183: 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>Peer Source</div> <div>All</div> </div> <div> <div>Prefix</div> <div>Prefix</div> </div> <div> <div>Display Routes</div> <div>Reset</div> </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
192.168.0.253/32	inet	XMPP	a3s19	10.84.17.5	16	100	-
192.168.16.252/32	inet	BGP	10.84.17.3	10.84.17.5	16	100	AS_PATH: 0
192.168.16.252/32	inet	XMPP	a3s18	10.84.17.4	17	100	-
192.168.16.253/32	inet	BGP	10.84.17.3	10.84.17.4	17	100	AS_PATH: 0
192.168.16.253/32	inet	XMPP	a3s19	10.84.17.5	17	100	-
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 184 on page 318](#).

Figure 184: 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 185 on page 319](#).

Figure 185: 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 186 on page 319 and Figure 187 on page 319.

Figure 186: 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	184508/5417	21:00:22.131180 2013-Aug-06
TCP	vn16	192.168.16.253	9100	vn0	192.168.0.252	43434	190968/5891	21:00:22.131192 2013-Aug-06
TCP	vn16	192.168.16.253	9101	vn0	192.168.0.252	53369	190950/5805	21:00:22.206222 2013-Aug-06
TCP	vn0	192.168.0.252	53369	vn16	192.168.16.253	9101	189088/5302	21:00:22.206207 2013-Aug-06
UDP	vn0	192.168.0.252	39522	vn16	192.168.16.252	9200	0/0	21:00:22.362861 2013-Aug-06
UDP	vn0	192.168.0.252	44794	vn16	192.168.16.253	9201	1707392/3144	21:00:24.104277 2013-Aug-06
UDP	vn16	192.168.16.253	9201	vn0	192.168.0.252	44794	1735788/3107	21:00:24.104293 2013-Aug-06
UDP	vn0	192.168.0.252	40561	vn16	192.168.16.253	9200	1693476/3067	21:00:22.037877 2013-Aug-06
UDP	vn16	192.168.16.253	9200	vn0	192.168.0.252	40561	1643324/3061	21:00:22.037887 2013-Aug-06
UDP	vn0	192.168.0.252	39522	vn16	192.168.16.252	9200	1676616/3074	21:00:22.306703 2013-Aug-06
TCP	vn0	192.168.0.252	34236	vn16	192.168.16.252	9100	1891368/5686	21:00:22.395695 2013-Aug-06
TCP	vn0	192.168.0.252	34236	vn16	192.168.16.252	9100	0/0	21:00:22.400371 2013-Aug-06

Figure 187: 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	1069380/1975	21:00:24.111374 2013-Aug-06
UDP	vn16	192.168.16.253	9201	vn0	192.168.0.252	44794	1100604/1963	21:00:24.111380 2013-Aug-06
UDP	vn0	192.168.0.252	40561	vn16	192.168.16.253	9200	1046756/1877	21:00:22.047467 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	21:00:22.047566 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	1196336/3400	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	21:00:23.440990 2013-Aug-06
UDP	vn16	192.168.16.253	9201	vn0	192.168.0.253	53390	1088892/1953	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	1304073/3904	21:00:23.514651 2013-Aug-06

