



---

Contrail™

# Juniper Networks Contrail Feature Guide

Release  
2.0



---

Modified: 2016-06-09

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

Copyright © 2016, 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™ Juniper Networks Contrail Feature Guide*

2.0

Copyright © 2016, 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 . . . . .	xxv
	Documentation and Release Notes . . . . .	xxv
	Documentation Conventions . . . . .	xxv
	Documentation Feedback . . . . .	xxvii
	Requesting Technical Support . . . . .	xxviii
	Self-Help Online Tools and Resources . . . . .	xxviii
	Opening a Case with JTAC . . . . .	xxviii
<b>Part 1</b>	<b>Overview</b>	
<b>Chapter 1</b>	<b>Understanding Contrail Controller . . . . .</b>	<b>3</b>
	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
<b>Part 2</b>	<b>Installation</b>	
<b>Chapter 2</b>	<b>Installing and Provisioning Roles . . . . .</b>	<b>9</b>
	Installation Overview . . . . .	9
	Server Requirements . . . . .	10
	Supported Platforms . . . . .	11
	Downloading Installation Software . . . . .	11
	Installing the Operating System and Contrail Packages . . . . .	12
	Configuring System Settings . . . . .	13
	Configuring Contrail on VMware ESXi . . . . .	13
	Introduction . . . . .	13
	Using Server Manager to Configure Contrail Compute Nodes on VMware ESXi . . . . .	14
	Using Fab Commands to Configure Contrail Compute Nodes on VMware ESXi . . . . .	22
	Fab Installation Guidelines for ESXi . . . . .	22
	Installing the Contrail Packages, Part One (CentOS or Ubuntu) . . . . .	24
	Populating the Testbed Definitions File . . . . .	26
	Supporting Multiple Interfaces on Servers and Nodes . . . . .	29
	Support for Multiple Interfaces . . . . .	29
	Number of cfgm Nodes Supported . . . . .	29
	Uneven Number of Database Nodes Required . . . . .	29
	Support for VLAN Interfaces . . . . .	29

Support for Bonding Options . . . . .	30
Support for Static Route Options . . . . .	30
Server Interface Examples . . . . .	30
Interface Naming and Configuration Management . . . . .	31
Setting Up Interfaces and Installing . . . . .	31
Sample testbed.py File With Exclusive Interfaces . . . . .	32
High Availability Support . . . . .	33
Contrail High Availability Features . . . . .	34
Configuration Options for Enabling Contrail High Availability . . . . .	34
Supported Cluster Topologies for High Availability . . . . .	35
Deploying OpenStack and Contrail on the Same High Available Nodes . . . . .	35
Deploying OpenStack and Contrail on Different High Available Nodes . . . . .	35
Deploying Contrail Only on High Available Nodes . . . . .	36
Testbed Definitions File Settings for Deploying Contrail with an Existing	
OpenStack Node . . . . .	36
Setting Up and Using a Simple Virtual Gateway with Contrail . . . . .	38
Introduction to the Simple Gateway . . . . .	38
How the Simple Gateway Works . . . . .	38
Setup Without Simple Gateway . . . . .	38
Setup With Simple Gateway . . . . .	39
Simple Gateway Configuration Features . . . . .	40
Packet Flows with the Simple Gateway . . . . .	41
Packet Flow Process From the Virtual Network to the Public Network . . . . .	41
Packet Flow Process From the Public Network to the Virtual Network . . . . .	42
Four Methods for Configuring the Simple Gateway . . . . .	42
Using Fab Provisioning to Configure the Simple Gateway . . . . .	42
Using the vRouter Configuration File to Configure the Simple Gateway . . . . .	44
Using Thrift Messages to Dynamically Configure the Simple Gateway . . . . .	44
How to Dynamically Create a Virtual Gateway . . . . .	45
How to Dynamically Delete a Virtual Gateway . . . . .	45
Using Devstack to Configure the Simple Gateway . . . . .	46
Common Issues with Simple Gateway Configuration . . . . .	47
Installing the Contrail Packages, Part Two (CentOS or Ubuntu) — Installing on	
the Remaining Machines . . . . .	47
Configuring the Control Node . . . . .	50
Using Server Manager to Automate Provisioning . . . . .	55
Overview of Server Manager . . . . .	55
Server Manager Requirements and Assumptions . . . . .	55
Server Manager Component Interactions . . . . .	57
Configuring Server Manager . . . . .	58
Configuring the Cobbler DHCP Template . . . . .	59
User-Defined Tags for Server Manager . . . . .	60
Server Manager Client Configuration File . . . . .	60
Restart Services . . . . .	61
Accessing Server Manager . . . . .	61
Communicating with the Server Manager Client . . . . .	62
Server Manager Commands for Configuring Servers . . . . .	62
Create New Servers or Update Existing Servers . . . . .	63
Delete Servers . . . . .	64

Show Server Configuration . . . . .	65
Server Manager Commands for Managing Clusters . . . . .	66
Server Manager Commands for Managing Tags . . . . .	69
Server Manager Commands for Managing Images . . . . .	71
Server Manager Operational Commands for Managing Servers . . . . .	74
Reimaging Server(s) . . . . .	75
Provisioning and Configuring Roles on Servers . . . . .	76
Restarting Server(s) . . . . .	78
Show Status of Server(s) . . . . .	79
Server Manager REST API Calls . . . . .	80
REST APIs for Server Manager Configuration Database Entries . . . . .	81
API: Add a Server . . . . .	81
API: Delete Servers . . . . .	82
API: Retrieve Server Configuration . . . . .	82
API: Add an Image . . . . .	82
API: Upload an Image . . . . .	83
API: Get Image Information . . . . .	83
API: Delete an Image . . . . .	84
API: Add or Modify a Cluster . . . . .	84
API: Delete a Cluster . . . . .	85
API: Get Cluster Configuration . . . . .	85
API: Get All Server Manager Configurations . . . . .	85
API: Reimage Servers . . . . .	85
API: Provision Servers . . . . .	85
API: Restart Servers . . . . .	86
Example: Reimaging and Provisioning a Server . . . . .	86
Installing Server Manager . . . . .	89
Overview: Installation Requirements for Server Manager . . . . .	89
Platform Support . . . . .	89
Installation Prerequisites . . . . .	90
Installing Server Manager . . . . .	90
Finishing the Provisioning . . . . .	91
Starting the Server Manager Service . . . . .	91
Upgrading Server Manager Software . . . . .	91
Prerequisite to Upgrading . . . . .	91
Use Steps for New Installation . . . . .	92
Server Manager Installation Completion Checks . . . . .	92
Server Manager Checks . . . . .	92
Server Manager Client Checks . . . . .	92
Server Manager Webui Checks . . . . .	92
Sample Configurations for Server Manager Templates . . . . .	93
Sample Settings . . . . .	93
dhcp.template . . . . .	93
named.conf.options . . . . .	93
named.template . . . . .	93
sendmail.cf . . . . .	93

	Using the Server Manager Web User Interface . . . . .	94
	Log In to Server Manager . . . . .	94
	Create a Cluster for Server Manager . . . . .	94
	Working with Servers in the Server Manager User Interface . . . . .	101
	Add a Server . . . . .	101
	Edit Tags for Servers . . . . .	103
	Using the Edit Config Option for Multiple Servers . . . . .	103
	Filter Servers by Tag . . . . .	104
	Viewing Server Details . . . . .	104
	Configuring Images and Packages . . . . .	104
	Add New Image or Package . . . . .	105
	Selecting Server Manager Actions for Clusters . . . . .	105
	Reimage a Cluster . . . . .	105
	Provision a Cluster . . . . .	106
	Installing Contrail with Red Hat OpenStack . . . . .	106
	Overview: Contrail with Red Hat OpenStack . . . . .	106
	Procedure for Installing RHOSP5 . . . . .	107
	Install and Configure Contrail . . . . .	108
	Download and Install Contrail-Install-Packages to the First Config Node . . . . .	108
	Update the Testbed.py . . . . .	109
	Complete the Installation on Remaining Nodes . . . . .	111
	Appendix: Installing with RDO . . . . .	112
	Install All-in-One OpenStack . . . . .	112
	Installing and Using Contrail Storage . . . . .	113
	Overview of the Contrail Storage Solution . . . . .	113
	Basic Storage Functionality with Contrail . . . . .	114
	Ceph Block and Object Storage Functionality . . . . .	114
	Using the Contrail Storage User Interface . . . . .	114
	Hardware Specifications . . . . .	115
	Software Files for Compute Storage Nodes . . . . .	116
	Contrail OpenStack Nova Modifications . . . . .	116
	Installing the Contrail Storage Solution . . . . .	116
	Using Fabric Commands to Install and Configure Storage . . . . .	117
	Fabric Installation Procedure . . . . .	117
	Using Server Manager to Install and Configure Storage . . . . .	119
	Server Manager Installation Procedure for Storage . . . . .	120
	Example Configurations for Storage for Reimaging and Provisioning a Server . . . . .	121
	Storage Installation Limits . . . . .	126
<b>Chapter 3</b>	<b>Upgrading Contrail Software . . . . .</b>	<b>129</b>
	Upgrading Contrail Software from Release 1.20 or Greater to Release 2.10 . . . . .	130
	Adding or Removing a Compute Node in an Existing Contrail Cluster . . . . .	133
	DKMS for vRouter Kernel Module . . . . .	134
<b>Part 3</b>	<b>Configuration</b>	
<b>Chapter 4</b>	<b>Configuring Virtual Networks . . . . .</b>	<b>137</b>
	Creating Projects in OpenStack for Configuring Tenants in Contrail . . . . .	138
	Creating a Virtual Network—Juniper Networks Contrail . . . . .	139

	Deleting a Virtual Network—Juniper Networks Contrail . . . . .	142
	Creating a Virtual Network—OpenStack Contrail . . . . .	143
	Deleting a Virtual Network—OpenStack Contrail . . . . .	144
	Creating an Image . . . . .	146
	Launching a Virtual Machine (Instance) . . . . .	148
	Creating a Network Policy—Juniper Networks Contrail . . . . .	151
	Associating a Network to a Policy—Juniper Networks Contrail . . . . .	153
	Associating Network Policies Overview . . . . .	153
	Associating a Network Policy to a Network from the Edit Network . . . . .	153
	Associating Networks with Network Policies from the Edit Policy . . . . .	156
	Creating a Network Policy—OpenStack Contrail . . . . .	157
	Associating a Network to a Policy—OpenStack Contrail . . . . .	160
	Associating Network Policies Overview . . . . .	161
	Associating a Network Policy to a Network . . . . .	161
	Creating a Floating IP Address Pool . . . . .	162
	Allocating a Floating IP Address to a Virtual Machine . . . . .	164
	Using Security Groups with Virtual Machines (Instances) . . . . .	165
	Security Groups Overview . . . . .	165
	Creating Security Groups and Adding Rules . . . . .	165
	Support for IPv6 Networks in Contrail . . . . .	168
	Overview: IPv6 Networks in Contrail . . . . .	169
	Creating IPv6 Virtual Networks in Contrail . . . . .	169
	Address Assignments . . . . .	169
	Adding IPv6 Peers . . . . .	170
	Configuring EVPN and VXLAN . . . . .	171
	Configuring Forwarding . . . . .	173
	Configuring the VXLAN Identifier Mode . . . . .	174
	Configuring the VXLAN Identifier . . . . .	175
	Configuring Encapsulation Methods . . . . .	175
	Configuring Network QoS Parameters . . . . .	177
	Overview . . . . .	177
	QoS Configuration Examples . . . . .	177
	Limitations . . . . .	178
<b>Chapter 5</b>	<b>Examples of Tiered Web Configurations . . . . .</b>	<b>181</b>
	Example: Deploying a Multi-Tier Web Application . . . . .	181
	Multi-Tier Web Application Overview . . . . .	181
	Example: Setting Up Virtual Networks for a Simple Tiered Web Application . . . . .	182
	Verifying the Multi-Tier Web Application . . . . .	184
	Sample Addressing Scheme for Simple Tiered Web Application . . . . .	184
	Sample Physical Topology for Simple Tiered Web Application . . . . .	185
	Sample Physical Topology Addressing . . . . .	186
	Sample Network Configuration for Devices for Simple Tiered Web Application . . . . .	187
<b>Chapter 6</b>	<b>High Availability . . . . .</b>	<b>193</b>
	Juniper OpenStack High Availability . . . . .	193
	Introduction . . . . .	193
	Contrail High Availability . . . . .	194

	OpenStack High Availability . . . . .	194
	Supported Platforms . . . . .	194
	Juniper OpenStack High Availability Architecture . . . . .	194
	Juniper OpenStack Objectives . . . . .	195
	Limitations . . . . .	195
	Solution Components . . . . .	196
	Virtual IP with Load Balancing . . . . .	196
	Failure Handling . . . . .	196
	Deployment . . . . .	197
	Minimum Hardware Requirement . . . . .	197
	Compute . . . . .	197
	Network . . . . .	198
	Installation . . . . .	198
	Testbed File for Fab . . . . .	199
	Using Headless vRouter to Improve Redundancy . . . . .	202
	Configuring the Headless vRouter . . . . .	202
<b>Part 4</b>	<b>Administration</b>	
<b>Chapter 7</b>	<b>Service Chaining . . . . .</b>	<b>205</b>
	Service Chaining . . . . .	205
	Service Chaining Basics . . . . .	205
	Service Chaining Configuration Elements . . . . .	206
	Service Chaining MX Series Configuration . . . . .	209
	Example: Creating an In-Network or In-Network-NAT Service Chain . . . . .	210
	Creating an In-Network or In-Network-NAT Service Chain . . . . .	210
	Example: Creating a Transparent Service Chain . . . . .	217
	Creating a Transparent Mode Service Chain . . . . .	217
	ECMP Load Balancing in the Service Chain . . . . .	221
	Example: Creating a Service Chain With the CLI . . . . .	222
	CLI for Creating a Service Chain . . . . .	222
	CLI for Creating a Service Template . . . . .	222
	CLI for Creating a Service Instance . . . . .	223
	CLI for Creating a Service Policy . . . . .	223
	Example: Creating a Service Chain with VSRX and In-Network or Routed Mode . . . . .	223
	Using the Juniper Networks Heat Template with Contrail . . . . .	225
	Introduction to Heat . . . . .	225
	Heat Architecture . . . . .	225
	Juniper Heat Plugin . . . . .	225
	Example: Creating a Service Template Using Heat . . . . .	226
<b>Chapter 8</b>	<b>Configuring Services . . . . .</b>	<b>229</b>
	Configuring DNS Servers . . . . .	229
	DNS Overview . . . . .	229
	Defining Multiple Virtual Domain Name Servers . . . . .	230
	IPAM and Virtual DNS . . . . .	230
	DNS Record Types . . . . .	231
	Configuring DNS Using the Interface . . . . .	232



	Configuring DNS Using Scripts . . . . .	237
	Support for Multicast . . . . .	238
	Subnet Broadcast . . . . .	238
	All-Broadcast/Limited-Broadcast and Link-Local Multicast . . . . .	239
	Host Broadcast . . . . .	240
	Using Static Routes with Services . . . . .	240
	Static Routes for Service Instances . . . . .	240
	Configuring Static Routes on a Service Instance . . . . .	241
	Configuring Static Routes on Service Instance Interfaces . . . . .	242
	Configuring Static Routes as Host Routes . . . . .	243
	Configuring Metadata Service . . . . .	244
<b>Chapter 9</b>	<b>Load Balancing-as-a-Service . . . . .</b>	<b>247</b>
	Configuring Load-Balancing-as-a-Service in Contrail . . . . .	247
	Overview: Load-Balancing-as-a-Service . . . . .	247
	Contrail LBaaS Implementation . . . . .	248
<b>Chapter 10</b>	<b>Multitenancy Support . . . . .</b>	<b>251</b>
	Configuring Multitenancy Support . . . . .	251
	Multitenancy Permissions . . . . .	251
	API Server . . . . .	252
	API Library Keystone Integration . . . . .	252
	Supporting Utilities . . . . .	252
<b>Part 5</b>	<b>Optimizing and Troubleshooting</b>	
<b>Chapter 11</b>	<b>Traffic Mirroring . . . . .</b>	<b>257</b>
	Configuring Traffic Analyzers and Packet Capture for Mirroring . . . . .	257
	Traffic Analyzer Images . . . . .	257
	Configuring Traffic Analyzers . . . . .	258
	Setting Up Traffic Mirroring Using Monitor > Debug > Packet Capture . . . . .	258
	Setting Up Traffic Mirroring Using Configure > Networking > Services . . . . .	261
	Configuring Interface Monitoring and Mirroring . . . . .	266
	Analyzer Service Virtual Machine (analyzer-vm-console.qcow2) . . . . .	267
	Packet Format for Analyzer . . . . .	268
	Metadata Format . . . . .	268
	Wireshark Changes . . . . .	269
	Troubleshooting Packet Display . . . . .	269

<b>Chapter 12</b>	<b>Monitoring and Troubleshooting</b>	<b>271</b>
	Contrail Analytics Overview	271
	Monitoring the System	273
	Monitor > Infrastructure > Dashboard	275
	Monitor Dashboard	275
	Monitor Individual Details from the Dashboard	276
	Using Bubble Charts	277
	Color-Coding of Bubble Charts	277
	Monitor > Infrastructure > Control Nodes	278
	Monitor Control Nodes Summary	278
	Monitor Individual Control Node Details	279
	Monitor Individual Control Node Console	280
	Monitor Individual Control Node Peers	282
	Monitor Individual Control Node Routes	283
	Monitor > Infrastructure > Virtual Routers	285
	Monitor vRouters Summary	286
	Monitor Individual vRouters Tabs	287
	Monitor Individual vRouter Details Tab	287
	Monitor Individual vRouters Interfaces Tab	288
	Configuring Interface Monitoring and Mirroring	289
	Monitor Individual vRouters Networks Tab	290
	Monitor Individual vRouters ACL Tab	291
	Monitor Individual vRouters Flows Tab	292
	Monitor Individual vRouters Routes Tab	293
	Monitor Individual vRouter Console Tab	294
	Monitor > Infrastructure > Analytics Nodes	296
	Monitor Analytics Nodes	296
	Monitor Analytics Individual Node Details Tab	297
	Monitor Analytics Individual Node Generators Tab	298
	Monitor Analytics Individual Node QE Queries Tab	299
	Monitor Analytics Individual Node Console Tab	300
	Monitor > Infrastructure > Config Nodes	301
	Monitor Config Nodes	301
	Monitor Individual Config Node Details	302
	Monitor Individual Config Node Console	303
	Monitor > Networking	304
	Monitor > Networking Menu Options	304
	Monitor -> Networking -> Dashboard	305
	Monitor > Networking > Projects	306
	Monitor Projects Detail	307
	Monitor > Networking > Networks	309
	Query > Flows	312
	Query > Flows > Flow Series	312
	Example: Query Flow Series	315
	Query > Flow Records	316
	Query > Flows > Query Queue	318

	Query > Logs . . . . .	319
	Query > Logs Menu Options . . . . .	319
	Query > Logs > System Logs . . . . .	320
	Sample Query for System Logs . . . . .	321
	Query > Logs > Object Logs . . . . .	322
	System Log Receiver in Contrail Analytics . . . . .	324
	Overview . . . . .	324
	Redirecting System Logs to Contrail Collector . . . . .	324
	Exporting Logs from Contrail Analytics . . . . .	324
	Example: Debugging Connectivity Using Monitoring for Troubleshooting . . . . .	325
	Using Monitoring to Debug Connectivity . . . . .	325
	Analytics Scalability . . . . .	329
	High Availability for Analytics . . . . .	330
<b>Chapter 13</b>	<b>Application Programming Interfaces (APIs) . . . . .</b>	<b>333</b>
	Contrail Analytics Application Programming Interfaces (APIs) and User-Visible	
	Entities (UVEs) . . . . .	333
	User-Visible Entities . . . . .	333
	Common UVEs in Contrail . . . . .	335
	Virtual Network UVE . . . . .	335
	Virtual Machine UVE . . . . .	335
	vRouter UVE . . . . .	335
	UVEs for Contrail Nodes . . . . .	336
	Wild Card Query of UVEs . . . . .	336
	Filtering UVE Information . . . . .	337
	Contrail Node Status . . . . .	344
	Overview . . . . .	344
	UVE for NodeStatus . . . . .	344
	Node Status Features . . . . .	345
	Using Introspect to Get Process Status . . . . .	350
	contrail-status script . . . . .	351
	Log and Flow Information APIs . . . . .	353
	HTTP GET APIs . . . . .	353
	HTTP POST API . . . . .	354
	POST Data Format Example . . . . .	354
	Query Types . . . . .	356
	Examining Query Status . . . . .	356
	Examining Query Chunks . . . . .	356
	Example Queries for Log and Flow Data . . . . .	356
	Working with Neutron . . . . .	359
	Data Structure . . . . .	359
	Network Sharing in Neutron . . . . .	360
	Commands for Neutron Network Sharing . . . . .	360
	Support for Neutron APIs . . . . .	361
	Contrail Neutron Plugin . . . . .	361
	DHCP Options . . . . .	362
	Incompatibilities . . . . .	362

	Support for Amazon VPC APIs on Contrail OpenStack . . . . .	362
	Overview of Amazon Virtual Private Cloud . . . . .	363
	Mapping Amazon VPC Features to OpenStack Contrail Features . . . . .	363
	VPC and Subnets Example . . . . .	364
	Euca2ools CLI for VPC and Subnets . . . . .	365
	Security in VPC: Network ACLs Example . . . . .	365
	Euca2ools CLI for Network ACLs . . . . .	366
	Security in VPC: Security Groups Example . . . . .	366
	Euca2ools CLI for Security Groups . . . . .	367
	Elastic IPs in VPC . . . . .	368
	Euca2ools CLI for Elastic IPs . . . . .	368
	Euca2ools CLI for Route Tables . . . . .	368
	Supported Next Hops . . . . .	369
	Internet Gateway Next Hop Euca2ools CLI . . . . .	369
	NAT Instance Next Hop Euca2ools CLI . . . . .	369
	Example: Creating a NAT Instance with Euca2ools CLI . . . . .	370
<b>Chapter 14</b>	<b>Optimizing . . . . .</b>	<b>371</b>
	vRouter Command Line Utilities . . . . .	371
	Overview . . . . .	371
	vif Command . . . . .	372
	flow Command . . . . .	374
	vrfstats Command . . . . .	376
	rt Command . . . . .	376
	dropstats Command . . . . .	377
	mpls Command . . . . .	381
	mirror Command . . . . .	383
	vxlan Command . . . . .	384
	nh Command . . . . .	385
	Route Target Filtering . . . . .	387
	Introduction . . . . .	387
	Debugging and Troubleshooting Route Target Filtering . . . . .	388
	RTF Limitations in Contrail 1.10 . . . . .	389
	Source Network Address Translation (SNAT) . . . . .	389
	Overview . . . . .	389
	Neutron APIs for Routers . . . . .	390
	Network Namespace . . . . .	391
	Using Web UI to Configure Routers with SNAT . . . . .	391
<b>Chapter 15</b>	<b>Common Support Answers . . . . .</b>	<b>393</b>
	Debugging Ping Failures for Policy-Connected Networks . . . . .	393
	Debugging BGP Peering and Route Exchange in Contrail . . . . .	399
	Example Cluster . . . . .	400
	Verifying the BGP Routers . . . . .	400
	Verifying the Route Exchange . . . . .	403
	Debugging Route Exchange with Policies . . . . .	405
	Debugging Peering with an MX Series Router . . . . .	407
	Debugging a BGP Peer Down Error with Incorrect Family . . . . .	408
	Configuring MX Peering (iBGP) . . . . .	410
	Checking Route Exchange with an MX Series Peer . . . . .	411

Checking the Route in the MX Series Router . . . . .	413
Troubleshooting the Floating IP Address Pool in Contrail . . . . .	414
Example Cluster . . . . .	415
Example . . . . .	415
Example: MX80 Configuration for the Gateway . . . . .	416
Ping the Floating IP from the Public Network . . . . .	419
Troubleshooting Details . . . . .	419
Get the UUID of the Virtual Network . . . . .	419
View the Floating IP Object in the API Server . . . . .	420
View floating-ips in floating-ip-pools in the API Server . . . . .	423
Check Floating IP Objects in the Virtual Machine Interface . . . . .	426
View Floating IP Objects in the IFMAP Server View . . . . .	429
View the BGP Peer Status on the Control Node . . . . .	433
Querying Routes in the Public Virtual Network . . . . .	434
Verification from the MX80 Gateway . . . . .	435
Viewing the Compute Node Vnsw Agent . . . . .	437
Advanced Troubleshooting . . . . .	440
Removing Stale Virtual Machines and Virtual Machine Interfaces . . . . .	442
Problem Example . . . . .	442
Show Virtual Machines . . . . .	443
Show Virtual Machines Using Python API . . . . .	445
Delete Methods . . . . .	446
Troubleshooting Link-Local Services in Contrail . . . . .	446
Overview of Link-Local Services . . . . .	446
Troubleshooting Procedure for Link-Local Services . . . . .	447
Metadata Service . . . . .	449
Troubleshooting Procedure for Link-Local Metadata Service . . . . .	449
<b>Chapter 16</b>	
<b>Contrail Commands . . . . .</b>	<b>451</b>
contrail-logs (Accessing Log File Messages) . . . . .	451
Command-Line Options for Contrail-Logs . . . . .	451
Option Descriptions . . . . .	452
Example Uses . . . . .	453
contrail-status (Viewing Node Status) . . . . .	454
contrail-version (Viewing Version Information) . . . . .	455
service (Managing Services) . . . . .	457
Backing Up and Restoring Configurations . . . . .	458
Back up Procedure . . . . .	458
Restore Procedure . . . . .	459
Restore Steps Continued . . . . .	469
Finishing . . . . .	469
<b>Part 6</b>	
<b>Index</b>	
Index . . . . .	473



# List of Figures

<b>Part 2</b>	<b>Installation</b>	
<b>Chapter 2</b>	<b>Installing and Provisioning Roles</b>	<b>9</b>
	Figure 1: Configure> Infrastructure > BGP Routers	51
	Figure 2: BGP Routers Summary	51
	Figure 3: Create BGP Router	52
	Figure 4: Control Nodes	53
	Figure 5: Control Node Details	54
	Figure 6: Control Node Peers Tab	54
	Figure 7: Server Manager Component Interactions	57
<b>Part 3</b>	<b>Configuration</b>	
<b>Chapter 4</b>	<b>Configuring Virtual Networks</b>	<b>137</b>
	Figure 8: OpenStack Projects	138
	Figure 9: Add Project	138
	Figure 10: Add IP Address Management	139
	Figure 11: Configure Networks	140
	Figure 12: Create Network	140
	Figure 13: Configure Networks	142
	Figure 14: Networks	143
	Figure 15: Create Network	143
	Figure 16: OpenStack Networks	144
	Figure 17: OpenStack Network Detail , Associated Instances Tab	145
	Figure 18: Instances	145
	Figure 19: Images & Snapshots	146
	Figure 20: Create An Image	147
	Figure 21: OpenStack Instances	148
	Figure 22: Launch Instance , Details Tab	149
	Figure 23: Launch Instance, Networking Tab	150
	Figure 24: Network Policy	151
	Figure 25: Create Policy	151
	Figure 26: Configure > Networking > Networks	154
	Figure 27: Edit Network	155
	Figure 28: Configure > Networking > Policies	156
	Figure 29: Edit Policy	156
	Figure 30: Network Policy	157
	Figure 31: Create Network Policy	158
	Figure 32: Network Policy	158
	Figure 33: Edit Policy Rules	159
	Figure 34: Networks Screen	161

	Figure 35: Edit Network Policy . . . . .	161
	Figure 36: Configure > Networking > Networks . . . . .	162
	Figure 37: Edit Network . . . . .	163
	Figure 38: Allocate Floating IPs . . . . .	164
	Figure 39: Allocate Floating IP . . . . .	164
	Figure 40: Associate Floating IP to Instance . . . . .	165
	Figure 41: Security Groups . . . . .	166
	Figure 42: Edit Security Group Rules . . . . .	166
	Figure 43: Add Rule . . . . .	167
	Figure 44: Create Security Group . . . . .	168
	Figure 45: Associate Security Group at Launch Instance . . . . .	168
<b>Chapter 5</b>	<b>Examples of Tiered Web Configurations . . . . .</b>	<b>181</b>
	Figure 46: Simple Tiered Web Use Case . . . . .	182
	Figure 47: Create Floating IP Pool . . . . .	183
	Figure 48: Allocate Floating IP . . . . .	183
	Figure 49: Sample Physical Topology for Simple Tiered Web Application . . . . .	186
	Figure 50: Sample Physical Topology Addressing . . . . .	187
<b>Part 4</b>	<b>Administration</b>	
<b>Chapter 7</b>	<b>Service Chaining . . . . .</b>	<b>205</b>
	Figure 51: Service Chaining . . . . .	205
	Figure 52: Contrail Service Chain . . . . .	206
	Figure 53: Create Networks . . . . .	210
	Figure 54: Add Service Template . . . . .	211
	Figure 55: Add Service Template Shared IP . . . . .	212
	Figure 56: Service Templates . . . . .	213
	Figure 57: Create Service Instances . . . . .	213
	Figure 58: Create Service Instances . . . . .	214
	Figure 59: Service Instance Details . . . . .	214
	Figure 60: Service Instance Console . . . . .	215
	Figure 61: Create Policy . . . . .	215
	Figure 62: Edit Network . . . . .	216
	Figure 63: Launch Instances . . . . .	216
	Figure 64: Create Networks . . . . .	217
	Figure 65: Add Service Template . . . . .	218
	Figure 66: Create Service Instances . . . . .	219
	Figure 67: Service Instance Details . . . . .	220
	Figure 68: Create Policy . . . . .	220
	Figure 69: Launch Instances . . . . .	221
	Figure 70: Load Balancing a Service Chain . . . . .	221
<b>Chapter 8</b>	<b>Configuring Services . . . . .</b>	<b>229</b>
	Figure 71: DNS Servers Examples . . . . .	230
	Figure 72: IPAM and Virtual DNS . . . . .	231
	Figure 73: Example Usage for NS Record Type . . . . .	232
	Figure 74: Configure DNS Records . . . . .	232
	Figure 75: Add DNS . . . . .	233
	Figure 76: Add DNS Record . . . . .	234



	Figure 77: Associate IPAMs to DNS . . . . .	235
	Figure 78: Configure IP Address Management . . . . .	236
	Figure 79: DNS Server . . . . .	236
<b>Part 5</b>	<b>Optimizing and Troubleshooting</b>	
<b>Chapter 11</b>	<b>Traffic Mirroring . . . . .</b>	<b>257</b>
	Figure 80: Packet Capture . . . . .	258
	Figure 81: Create Analyzer . . . . .	258
	Figure 82: Analyzer Rules . . . . .	259
	Figure 83: Create Analyzer Associate Networks . . . . .	260
	Figure 84: Launch Analyzer VM . . . . .	261
	Figure 85: Packet Capture Display . . . . .	261
	Figure 86: Service Templates . . . . .	262
	Figure 87: Add Service Template . . . . .	262
	Figure 88: Create Service Instances . . . . .	263
	Figure 89: Create Policy . . . . .	264
	Figure 90: Policy Rules . . . . .	264
	Figure 91: Service Instances View Console . . . . .	266
	Figure 92: Individual vRouter . . . . .	267
	Figure 93: Interfaces . . . . .	267
	Figure 94: Wireshark Packet Display . . . . .	269
<b>Chapter 12</b>	<b>Monitoring and Troubleshooting . . . . .</b>	<b>271</b>
	Figure 95: Monitor Menu . . . . .	273
	Figure 96: Monitor > Infrastructure > Dashboard . . . . .	276
	Figure 97: Dashboard Summary Boxes . . . . .	276
	Figure 98: Bubble Summary Information . . . . .	277
	Figure 99: Control Nodes Summary . . . . .	278
	Figure 100: Individual Control Node—Details Tab . . . . .	279
	Figure 101: Individual Control Node—Console Tab . . . . .	281
	Figure 102: Individual Control Node—Peers Tab . . . . .	283
	Figure 103: Individual Control Node—Routes Tab . . . . .	284
	Figure 104: vRouters Summary . . . . .	286
	Figure 105: Individual vRouters—Details Tab . . . . .	287
	Figure 106: Individual vRouters—Interfaces Tab . . . . .	289
	Figure 107: Individual vRouter . . . . .	290
	Figure 108: Interfaces . . . . .	290
	Figure 109: Individual vRouters—Networks Tab . . . . .	291
	Figure 110: Individual vRouters—ACL Tab . . . . .	292
	Figure 111: Individual vRouters—Flows Tab . . . . .	293
	Figure 112: Individual vRouters—Routes Tab . . . . .	294
	Figure 113: Individual vRouter—Console Tab . . . . .	295
	Figure 114: Analytics Nodes Summary . . . . .	297
	Figure 115: Monitor Analytics Individual Node Details Tab . . . . .	298
	Figure 116: Individual Analytics Node—Generators Tab . . . . .	299
	Figure 117: Individual Analytics Node—QE Queries Tab . . . . .	299
	Figure 118: Analytics Individual Node—Console Tab . . . . .	300
	Figure 119: Config Nodes Summary . . . . .	302
	Figure 120: Individual Config Nodes—Details Tab . . . . .	302

	Figure 121: Individual Config Node—Console Tab . . . . .	303
	Figure 122: Monitor Networking Menu Options . . . . .	305
	Figure 123: Traffic Statistics for Domain Window . . . . .	305
	Figure 124: Monitor > Networking > Projects . . . . .	306
	Figure 125: Monitor Projects Connectivity Details . . . . .	307
	Figure 126: Traffic Statistics Between Networks . . . . .	307
	Figure 127: Projects Instances Summary . . . . .	308
	Figure 128: Instance Traffic Statistics . . . . .	309
	Figure 129: Network Summary . . . . .	309
	Figure 130: Individual Network Connectivity Details—Summary Tab . . . . .	310
	Figure 131: Individual Network— Port Map Tab . . . . .	310
	Figure 132: Individual Network— Port Distribution Tab . . . . .	311
	Figure 133: Individual Network Instances Tab . . . . .	311
	Figure 134: Individual Network Details Tab . . . . .	312
	Figure 135: Query Flow Series . . . . .	313
	Figure 136: Flow Series Select . . . . .	314
	Figure 137: Flow Series Filter . . . . .	315
	Figure 138: Example: QueryFlow Series . . . . .	315
	Figure 139: Query Flow Series Tabular Results . . . . .	316
	Figure 140: Query Flow Series Graphical Results . . . . .	316
	Figure 141: Flow Records . . . . .	317
	Figure 142: Flow Records Select Window . . . . .	318
	Figure 143: Where Clause Window . . . . .	318
	Figure 144: Flows Query Queue . . . . .	319
	Figure 145: Query > Logs . . . . .	320
	Figure 146: Query > Logs > System Logs . . . . .	320
	Figure 147: Edit Where Clause . . . . .	322
	Figure 148: Sample Query System Logs . . . . .	322
	Figure 149: Query > Logs > Object Logs . . . . .	323
	Figure 150: Navigate to Instance . . . . .	325
	Figure 151: Traffic Statistics for Instance . . . . .	325
	Figure 152: Navigate to Instance . . . . .	325
	Figure 153: Traffic Statistics for Instance . . . . .	326
	Figure 154: Navigate to a3s18 Interfaces . . . . .	326
	Figure 155: Navigate to a3s19 Interfaces . . . . .	326
	Figure 156: ACL Connectivity a3s18 . . . . .	326
	Figure 157: ACL Connectivity a3s19 . . . . .	327
	Figure 158: Routes default-domain:demo:vn0:vn0 . . . . .	327
	Figure 159: Routes default-domain:demo:vn16:vn16 . . . . .	327
	Figure 160: Verify Route and Next Hop a3s18 . . . . .	328
	Figure 161: Verify Route and Next Hop a3s19 . . . . .	328
	Figure 162: Flows for a3s18 . . . . .	329
	Figure 163: Flows for a3s19 . . . . .	329
	Figure 164: Analytics Scalability . . . . .	330
<b>Chapter 14</b>	<b>Optimizing . . . . .</b>	<b>371</b>
	Figure 165: Virtual Network With a Private Subnet . . . . .	390
	Figure 166: Edit Router Window to Enable SNAT . . . . .	391
	Figure 167: Router Status for SNAT . . . . .	392

## Chapter 15

Figure 168: Instance Details Window . . . . .	392
<b>Common Support Answers . . . . .</b>	<b>393</b>
Figure 169: Virtual Machine Status Window . . . . .	394
Figure 170: Tap Interface Status Window . . . . .	394
Figure 171: Policies, Attachments, and Traffic Rule Status Window . . . . .	395
Figure 172: Virtual Network Policy Configuration Window . . . . .	395
Figure 173: Virtual Network Route Information Window . . . . .	395
Figure 174: Flow and Dropstats Command List . . . . .	396
Figure 175: Flow Command Output Window . . . . .	396
Figure 176: Fetch Flow Record Window . . . . .	397
Figure 177: Unresolved IP Address Window . . . . .	397
Figure 178: Unresolved Flow Details Window . . . . .	398
Figure 179: Protocol-Specific Flow Sample . . . . .	398
Figure 180: Protocol-Specific Flow Sample With Deny Action . . . . .	399
Figure 181: Sample Output, BGP Routers: . . . . .	401
Figure 182: Sample Output, BGP Router References: . . . . .	402
Figure 183: Sample Output, BGP Neighbor Config: . . . . .	402
Figure 184: Sample Output, BGP Peering Config: . . . . .	403
Figure 185: Sample Output, BGP Neighbor States: . . . . .	403
Figure 186: Sample Output, Show Routing Instance: . . . . .	404
Figure 187: Sample Output, Validate Route: . . . . .	404
Figure 188: Sample Output, Validate L3vpn Table: . . . . .	405
Figure 189: Sample Output, Validate L3vpn Table, Scrolled: . . . . .	405
Figure 190: Create Policy Window . . . . .	406
Figure 191: Sample Output, Validate Import Target: . . . . .	406
Figure 192: Sample Output, Route Import: . . . . .	406
Figure 193: Edit Global ASN Window . . . . .	407
Figure 194: Create BGP Peer Window . . . . .	407
Figure 195: Sample BGP Peer UVE . . . . .	408
Figure 196: Sample Established BGP Peer UVE . . . . .	409
Figure 197: Edit Global ASN Window . . . . .	410
Figure 198: Create BGP Peer Window . . . . .	410
Figure 199: Sample Established IBGP Peer UVE . . . . .	411
Figure 200: Sample Established IBGP Peer Introspect Window . . . . .	411
Figure 201: Routing Instance Route Table . . . . .	411
Figure 202: Routing Instance Route Table . . . . .	412
Figure 203: Routing Instance Public IPv4 Route Table . . . . .	412
Figure 204: Virtual Machine Routing Instance Public IPv4 Route Table . . . . .	412
Figure 205: BGP Routing Instance Route Table . . . . .	413



# List of Tables

	<b>About the Documentation</b> . . . . .	<b>xxv</b>
	Table 1: Notice Icons . . . . .	xxvi
	Table 2: Text and Syntax Conventions . . . . .	xxvi
<b>Part 2</b>	<b>Installation</b>	
<b>Chapter 2</b>	<b>Installing and Provisioning Roles</b> . . . . .	<b>9</b>
	Table 3: Create BGP Router Fields . . . . .	52
	Table 4: Server Manager Parameters . . . . .	59
	Table 5: Server Manager Add Server Command Options . . . . .	63
	Table 6: Server Manager Delete Server Command Options . . . . .	64
	Table 7: Server Manager Show Server Command Options . . . . .	65
	Table 8: Server Manager Add Cluster Command Options . . . . .	66
	Table 9: Server Manager Delete Cluster Command Options . . . . .	68
	Table 10: Server Manager Show Cluster Command Options . . . . .	68
<b>Part 3</b>	<b>Configuration</b>	
<b>Chapter 4</b>	<b>Configuring Virtual Networks</b> . . . . .	<b>137</b>
	Table 11: Add IP Address Management Fields . . . . .	139
	Table 12: Create Network Fields . . . . .	141
	Table 13: Create Network Fields . . . . .	143
	Table 14: Create An Image Fields . . . . .	147
	Table 15: Launch Instance Details Tab Fields . . . . .	149
	Table 16: Create Policy Fields . . . . .	152
	Table 17: Edit Policy Rules Fields . . . . .	159
	Table 18: Add Rule Fields . . . . .	167
<b>Chapter 5</b>	<b>Examples of Tiered Web Configurations</b> . . . . .	<b>181</b>
	Table 19: Sample Addressing Scheme for Example . . . . .	185
<b>Part 4</b>	<b>Administration</b>	
<b>Chapter 7</b>	<b>Service Chaining</b> . . . . .	<b>205</b>
	Table 20: Add Service Template Fields . . . . .	211
	Table 21: Create Service Instances Fields . . . . .	213
	Table 22: Add Service Template Fields . . . . .	218
	Table 23: Create Service Instances Fields . . . . .	219
<b>Chapter 8</b>	<b>Configuring Services</b> . . . . .	<b>229</b>
	Table 24: DNS Record Types Supported . . . . .	231
	Table 25: Add DNS Fields . . . . .	233

	Table 26: Add DNS Record Fields . . . . .	234
	Table 27: Associate IPAMs to DNS Fields . . . . .	235
	Table 28: DNS Modes . . . . .	236
	Table 29: DNS Scripts . . . . .	237
<b>Part 5</b>	<b>Optimizing and Troubleshooting</b>	
<b>Chapter 11</b>	<b>Traffic Mirroring . . . . .</b>	<b>257</b>
	Table 30: Analyzer Rule Fields . . . . .	259
	Table 31: Add Service Template Fields . . . . .	262
	Table 32: Create Service Instances Fields . . . . .	263
	Table 33: Add Rule Fields . . . . .	265
<b>Chapter 12</b>	<b>Monitoring and Troubleshooting . . . . .</b>	<b>271</b>
	Table 34: Monitor Menu Options . . . . .	273
	Table 35: Dashboard Summary Boxes . . . . .	276
	Table 36: Control Nodes Summary Fields . . . . .	278
	Table 37: Individual Control Node—Details Tab Fields . . . . .	280
	Table 38: Control Node: Console Tab Fields . . . . .	281
	Table 39: Control Node: Peers Tab Fields . . . . .	283
	Table 40: Control Node: Routes Tab Fields . . . . .	284
	Table 41: vRouters Summary Fields . . . . .	286
	Table 42: vRouters Details Tab Fields . . . . .	287
	Table 43: vRouters: Interfaces Tab Fields . . . . .	289
	Table 44: vRouters: Networks Tab Fields . . . . .	291
	Table 45: vRouters: ACL Tab Fields . . . . .	292
	Table 46: vRouters: Flows Tab Fields . . . . .	293
	Table 47: vRouters: Routes Tab Fields . . . . .	294
	Table 48: Control Node: Console Tab Fields . . . . .	295
	Table 49: Fields on Analytics Nodes Summary . . . . .	297
	Table 50: Monitor Analytics Individual Node Details Tab Fields . . . . .	298
	Table 51: Monitor Analytics Individual Node Generators Tab Fields . . . . .	299
	Table 52: Analytics Node QE Queries Tab Fields . . . . .	299
	Table 53: Monitor Analytics Individual Node Console Tab Fields . . . . .	300
	Table 54: Config Nodes Summary Fields . . . . .	302
	Table 55: Individual Config Nodes— Details Tab Fields . . . . .	303
	Table 56: Individual Config Node—Console Tab Fields . . . . .	303
	Table 57: Projects Summary Fields . . . . .	306
	Table 58: Projects Summary Fields . . . . .	306
	Table 59: Projects Instances Summary Fields . . . . .	308
	Table 60: Network Summary Fields . . . . .	309
	Table 61: Query Flow Series Fields . . . . .	313
	Table 62: Query Flow Records Fields . . . . .	317
	Table 63: Query Flow Records Fields . . . . .	319
	Table 64: Query System Logs Fields . . . . .	320
	Table 65: Object Logs Query Fields . . . . .	323
<b>Chapter 13</b>	<b>Application Programming Interfaces (APIs) . . . . .</b>	<b>333</b>
	Table 66: Amazon VPC and OpenStack Contrail Feature Comparison . . . . .	363
<b>Chapter 14</b>	<b>Optimizing . . . . .</b>	<b>371</b>

Table 67: vif Fields .....	373
----------------------------	-----





# About the Documentation

- Documentation and Release Notes on page xxv
- Documentation Conventions on page xxv
- Documentation Feedback on page xxvii
- Requesting Technical Support on page xxviii

## 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 xxvi 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 xxvi defines the text and syntax conventions used in this guide.

Table 2: Text and Syntax Conventions

Convention	Description	Examples
<b>Bold text like this</b>	Represents text that you type.	To enter configuration mode, type the <b>configure</b> command:  user@host> <b>configure</b>
Fixed-width text like this	Represents output that appears on the terminal screen.	user@host> <b>show chassis alarms</b>  No alarms currently active
<i>Italic text like this</i>	<ul style="list-style-type: none"> <li>Introduces or emphasizes important new terms.</li> <li>Identifies guide names.</li> <li>Identifies RFC and Internet draft titles.</li> </ul>	<ul style="list-style-type: none"> <li>A policy <i>term</i> is a named structure that defines match conditions and actions.</li> <li><i>Junos OS CLI User Guide</i></li> <li>RFC 1997, <i>BGP Communities Attribute</i></li> </ul>
<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@# <b>set system domain-name</b> <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"><li>To configure a stub area, include the <b>stub</b> statement at the <b>[edit protocols ospf area area-id]</b> hierarchy level.</li><li>The console port is labeled <b>CONSOLE</b>.</li></ul>
< > (angle brackets)	Encloses optional keywords or variables.	<b>stub &lt;default-metric <i>metric</i>&gt;;</b>
(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.	<b>broadcast   multicast</b>  <b>(<i>string1</i>   <i>string2</i>   <i>string3</i>)</b>
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	<b>rsvp { # Required for dynamic MPLS only</b>
[ ] (square brackets)	Encloses a variable for which you can substitute one or more values.	<b>community name members [ <i>community-ids</i> ]</b>
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"><li>In the Logical Interfaces box, select <b>All Interfaces</b>.</li><li>To cancel the configuration, click <b>Cancel</b>.</li></ul>
> (bold right angle bracket)	Separates levels in a hierarchy of menu selections.	In the configuration editor hierarchy, select <b>Protocols&gt;Ospf</b> .

## 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](mailto: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 Controller on page 3](#)





## CHAPTER 1

# Understanding Contrail Controller

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

### Contrail Overview

---

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

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

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

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

Contrail can be used with open cloud orchestration systems such as OpenStack or CloudStack. It can also interact with other systems and applications based on Operations

Support System (OSS) and Business Support Systems (BSS), using northbound APIs. Contrail allows customers to build elastic architectures that leverage the benefits of cloud computing — agility, self-service, efficiency, and flexibility — while providing an interoperable, scale-out control plane for network services within and across network domains.

**Related Documentation**

- [Contrail Description on page 4](#)

---

## Contrail Description

- [Contrail Major Components on page 4](#)
- [Contrail Solution on page 4](#)

### Contrail Major Components

The following are the major components of Contrail.

---

#### Contrail Control Nodes

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

---

#### Contrail Compute Nodes – XMPP Agent and vRouter

- Responsible for managing the data plane.
- Functionality can reside on a host OS.

### Contrail Solution

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

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

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

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

- Related Documentation**
- [Contrail Overview on page 3](#)
  - [Installation Overview on page 9](#)



## PART 2

# Installation

- [Installing and Provisioning Roles on page 9](#)
- [Upgrading Contrail Software on page 129](#)



## CHAPTER 2

# Installing and Provisioning Roles

- [Installation Overview on page 9](#)
- [Server Requirements on page 10](#)
- [Supported Platforms on page 11](#)
- [Downloading Installation Software on page 11](#)
- [Installing the Operating System and Contrail Packages on page 12](#)
- [Configuring System Settings on page 13](#)
- [Configuring Contrail on VMware ESXi on page 13](#)
- [Installing the Contrail Packages, Part One \(CentOS or Ubuntu\) on page 24](#)
- [Populating the Testbed Definitions File on page 26](#)
- [Supporting Multiple Interfaces on Servers and Nodes on page 29](#)
- [High Availability Support on page 33](#)
- [Testbed Definitions File Settings for Deploying Contrail with an Existing OpenStack Node on page 36](#)
- [Setting Up and Using a Simple Virtual Gateway with Contrail on page 38](#)
- [Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines on page 47](#)
- [Configuring the Control Node on page 50](#)
- [Using Server Manager to Automate Provisioning on page 55](#)
- [Installing Server Manager on page 89](#)
- [Using the Server Manager Web User Interface on page 94](#)
- [Installing Contrail with Red Hat OpenStack on page 106](#)
- [Installing and Using Contrail Storage on page 113](#)

## Installation Overview

---

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

The roles used for the installed system include:

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

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

[“Installing the Operating System and Contrail Packages” on page 12](#) describes installing the Contrail Controller software onto multiple servers.

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

**Related  
Documentation**

- [Server Requirements on page 10](#)
- [Downloading Installation Software on page 11](#)
- [Installing the Operating System and Contrail Packages on page 12](#)

---

## Server Requirements

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

Each server must have a minimum of:

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

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

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





**NOTE:** If you are using Contrail Storage, additional hardware requirements can be found in “[Installing and Using Contrail Storage](#)” on page 113, Hardware Specifications.

**Related  
Documentation**

- [Installation Overview on page 9](#)
- [Downloading Installation Software on page 11](#)

## Supported Platforms

Contrail Release 2.21, is supported on the OpenStack Juno and Icehouse releases. Juno is supported on Ubuntu 14.04.2 and Centos 7.1.

Contrail networking is supported on Red Hat RHOSP 5.0, which is supported only on OpenStack Icehouse

In Contrail Release 2.21, support for VMware vCenter 5.5. vCenter is limited to Ubuntu 14.04.2 (Linux kernel version: 3.13.0-40-generic).

Other supported platforms include:

- CentOS 6.5 (Linux kernel version: 2.6.32-358.el6.x86\_64)
- CentOS 7.1 (Linux kernel version: 3.10.0-229.el7)
- Redhat 7/RHOSP 5.0 (Linux kernel version: 3.10.0-123.el7.x86\_64)
- Ubuntu 12.04.04 (Ubuntu kernel version: 3.13.0-34-generic)
- Ubuntu 14.04. (Linux kernel version: 3.13.0-40-generic)

## Downloading Installation Software

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

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

Versions are available for each Contrail release, for the supported Linux operating systems and versions, and for the supported versions of OpenStack.

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

The Contrail image includes the following software:

- All dependent software packages needed to support installation and operation of OpenStack and Contrail

- Contrail Controller software – all components
- OpenStack release currently in use for Contrail

**Related Documentation**

- [Installing the Operating System and Contrail Packages on page 12](#)
- [Configuring System Settings on page 13](#)
- [Populating the Testbed Definitions File on page 26](#)
- [Installing the Contrail Packages, Part One \(CentOS\)](#)
- [Download Software](#)

---

## Installing the Operating System and Contrail Packages

Install the stock CentOS or Ubuntu operating system image appropriate for your version of Contrail (CentOS 6.4 or 6.5 or Ubuntu 12.04.4 for Contrail 1.2 and greater) onto the server, then install Contrail packages separately.

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

1. Install a CentOS or Ubuntu minimal distribution as desired on all servers. Follow the published operating system installation procedure for the selected operating system; refer to the website for the operating system.
2. After rebooting all of the servers after installation, verify that you can log in to each of them using the root password defined during installation.
3. After the initial installations on all servers, configure some items specific to your systems, (see [“Configuring System Settings” on page 13](#)), then begin the first part of the installation (see [“Installing the Contrail Packages, Part One \(CentOS or Ubuntu\)” on page 24](#)).

**Related Documentation**

- [Configuring System Settings on page 13](#)
- [Installing the Contrail Packages, Part One \(CentOS or Ubuntu\) on page 24](#)
- [Populating the Testbed Definitions File on page 26](#)
- [Download Software](#)

## Configuring System Settings

---

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

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

To configure system settings:

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

### Related Documentation

- [Installing the Contrail Packages, Part One \(CentOS\)](#)
- [Populating the Testbed Definitions File on page 26](#)

## Configuring Contrail on VMware ESXi

---

- [Introduction on page 13](#)
- [Using Server Manager to Configure Contrail Compute Nodes on VMware ESXi on page 14](#)
- [Using Fab Commands to Configure Contrail Compute Nodes on VMware ESXi on page 22](#)
- [Fab Installation Guidelines for ESXi on page 22](#)

### Introduction

A Contrail cluster of nodes consists of one or more servers, each configured to provide certain role functionalities for the cluster, including control, config, database, analytics, web-ui, and compute. The cluster provides virtualized network functionality in a cloud

computing environment, such as OpenStack. Typically, the servers running Contrail components are using the Linux operating system and a KVM hypervisor.

As of Contrail Release 2.0 and greater, limited capability is provided for extending the Contrail compute node functionality to servers running the VMware ESXi virtualization platform. To run Contrail on ESXi, a virtual machine is spawned on a physical ESXi server, and a compute node is configured on the virtual machine. For Contrail on ESXi, only compute node functionality is provided at this time.

There are two methods for configuring and provisioning nodes in a Contrail cluster: using the Contrail Server Manager to automate provisioning or using fab (Fabric) commands. Both methods can also be used to configure Contrail compute nodes on VMWare ESXi.

### Using Server Manager to Configure Contrail Compute Nodes on VMware ESXi

The following procedure provides guidelines and steps used to configure compute nodes on ESXi when using the Contrail Server Manager.

Using Server Manager to provision nodes for ESXi is similar to the procedure for provisioning nodes on the KVM hypervisor. However, because an ESXi server is represented by a virtual machine for compute nodes in the Contrail environment, there are additional items to be identified in the setup. The following procedure describes how to configure the ESXi servers using Server Manager.

For more details regarding the use and functionality of the Server Manager, refer to [Using Server Manager to Automate Provisioning](#).

**Installation Guidelines  
Using Server Manager**

The following procedure provides guidelines for using the normal Server Manager installation process and applying it to an environment that includes compute nodes on ESXi server(s).

1. Define the cluster and the cluster parameters. There are no additional cluster parameters needed for ESXi hosts.
2. Define servers and configure them to be part of the cluster already defined. For a KVM-only environment, server entries are needed only for physical servers. However, when there is one or more ESXi servers in the cluster, in addition to a server entry for each physical ESXi server, there must also be an entry for a virtual machine on each ESXi server. The virtual machine is used to configure a Contrail compute node in OpenStack. Refer to the sample file following this procedure for examples of the additional entry and fields needed for ESXi servers and their virtual machines.
3. Add images for Ubuntu and ESXi to the server manager database. These images are the base images for configuring the KVM and ESXi servers.
4. Use the Server Manager **add image** command to add the Contrail package to be used to provision the Contrail nodes.
5. In addition to the base OS (ESXi) for the ESXi server, also provide a modified Ubuntu VMDK image that will be used to spawn the virtual machine on ESXi. The Contrail compute node functionality runs on the spawned virtual machine. The location of the VMDK image is provided as part of the parameters of the server entry that corresponds to the ESXi virtual machine.
6. Add all additional configuration objects that are needed as described in the Server Manager installation instructions. See Using Server Manager to Automate Provisioning.
7. When all configuration objects are created in the Server Manager database, issue the **reimage** command, as described in Using Server Manager to Automate Provisioning, to boot the Ubuntu and ESXi hosts.
8. Issue the Server Manager **provision** command to provision the nodes on Ubuntu and ESXi hosts. During provisioning, the virtual machine is spawned on the ESXi server, using the VMDK image, then that node is configured as a compute node.
9. Upon completion of provisioning, on an Ubuntu machine, all the nodes are up and operational. On the ESXi server, only compute node functionality is supported, consequently, only the virtual machine is seen as one of the compute nodes within the OpenStack cluster.

**Example: json Files for  
Configuring with Server  
Manager**

The following example demonstrates sample configuration parameters needed to configure ESXi hosts along with Ubuntu servers. Only compute node functionality is supported for ESXi at this time.

1. Create a file, cluster.json, for the cluster configuration. The following shows sample parameters to include.

```
"cluster" : [
  {
    "id" : "clusteresx",
    "email" : "test@testco.net",
    "parameters" : {
      "router_asn": "<asn number>",
      "database_dir": "/home/cassandra",
      "db_initial_token": "",
      "openstack_mgmt_ip": "",
      "use_certs": "False",
      "multi_tenancy": "False",
      "encapsulation_priority": "MPLSoUDP,MPLSoGRE,VXLAN",
      "service_token": "<password>",
      "keystone_user": "admin",
      "keystone_passwd": "<password>",
      "keystone_tenant": "admin",
      "openstack_password": "<password>",
      "analytics_data_ttl": "168",
      "compute_non_mgmt_ip": "",
      "compute_non_mgmt_gway": "",
      "haproxy": "disable",
      "subnet_mask": "<ip address>",
      "gateway": "<ip address>",
      "password": "<password>",
      "external_bgp": "",
      "domain": "<domain name>"
    }
  }
]
```

```
}
```

```
}
```

2. Add the cluster, using the json file just created:

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

3. Create a file, server.json, for the servers configuration.

The following shows sample parameters to include. ESXi-specific parameters are highlighted. Note that the Ubuntu server is configured to have all Contrail role definitions, however, the ESXi virtual machine is configured for a compute role only:

```
"server": [  
  {  
    "id": "nodea10",  
    "mac_address": "<mac address>",  
    "ip_address": "<ip address>",  
    "parameters" : {  
      "interface_name": "eth1"  
    },  
    "roles" :  
    ["config","openstack","control","compute","collector","webui","database"],  
    "cluster_id": "clusteresx",  
    "subnet_mask": "<ip address>",  
    "gateway": "<ip address>",  
    "password": "<password>",  
    "domain": "<domain name>",  
    "ipmi_address": "<ip address>"  
  },  
  {  
    "id": "nodeh6",  
    "mac_address": "<mac address>",  
    "ip_address": "<ip address>",  
    "parameters": {  
      "interface_name": "eth0",  
      "server_license": "",  
      "esx_nicname": "vmnic0"  
    },  
    "roles": [  
  
    ],  
  },  
]
```



```

    "cluster_id": "clusteresx",

    "subnet_mask": "<ip address>",

    "gateway": "<ip address>",

    "password": "<password>",

    "ipmi_address": "<ip address>",

    "domain": "<domain name>"
  },
  {
    "id": "ContrailVM",

    "host_name": "ContrailVM", <<<<<<<< Provide a hostname for VM,
otherwise the hostname "nodeb2-contrail-vm" is created and hardcoded.

    "mac_address": "mac address",<<<<<<<< The mac_address should be
in the range 00:50:56:*:~:*

    "ip_address": "<ip address>",

    "parameters": {
      "interface_name": "eth0",

      "esx_server": "nodeh6",

      "esx_uplink_nic": "vmnic0",

      "esx_fab_vswitch": "vSwitch0",

      "esx_vm_vswitch": "vSwitch1",

      "esx_fab_port_group": "contrail-fab-pg",

      "esx_vm_port_group": "contrail-vm-pg",

      "esx_vmdk": "/home/smgr_files/json/ContrailVM-disk1.vmdk",

      "vm_deb":
"/home/smgr_files/json/contrail-install-packages_1.10-34~havana_all.deb"
    },

    "roles": [
      "compute"
    ],

    "cluster_id": "clusteresx",

    "subnet_mask": "<ip address>",

```

```
        "gateway": "<ip address>",  
        "password": "<password>",  
        "domain": "<domain name>"  
    }  
]  
}
```

4. Add the servers, using the **server.json** file:

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

5. Create the file **image.json** to add the needed images (Ubuntu, ESXi, and Contrail Ubuntu package) to the Server Manager database.

The following sample shows the parameters to include.

```
{
  "image": [
    {
      "id": "esx",
      "type": "esxi5.5",
      "version": "5.5",
      "path": "/home/smgr_files/json/esx5.5_x86_64.iso"
    },
    {
      "id": "Ubuntu-12.04.3",
      "type": "ubuntu",
      "version": "12.04.3",
      "path": "/home/smgr_files/json/Ubuntu-12.04.3-server-amd64.iso"
    },
    {
      "id": "esx",
      "type": "esxi5.5",
      "version": "5.5",
      "path": "/home/smgr_files/json/esx5.5_x86_64.iso"
    }
  ]
}
```

6. Add the images:

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

7. Reimage nodes.

Issue **server-manager reimage --server\_id nodea10 Ubuntu-12.04.3** to reimage nodea10 with Ubuntu 12.04.3.

Issue **server-manager reimage --server\_id nodeh6 esx5-5** to reimage nodeh6 with ESXi 5.5.

8. Provision roles.

Issue **server-manager provision --server\_id nodea10 contrail-uh-r110-b34** to configure and provision all the roles on nodea10.

9. Ensure that the DHCP server on Server Manager is configured to provide an IP address to the virtual machine that will be spawned on ESXi as part of the next provisioning command. The DHCP.template on the Server Manager with Cobbler machine should be modified to provide IP to VIRTUAL MACHINE with the virtual MAC that is configured on that virtual machine.

10. Provision the ESXi server.

Issue **server-manager provision --server\_id ContrailVM contrail-uh-r110-b3** to configure and provision the compute role on a virtual machine that will be created on the ESXi.



**NOTE:** Provisioning on the ESXi consists of specifying the server id corresponding to the virtual machine on ESXi and not the ESXi server itself.

Upon completion of provisioning, the two compute nodes can be seen in the OpenStack cluster. One of the compute nodes is on a physical Ubuntu server and the other compute node is on an ESXi virtual machine.

The system is now ready for users to launch virtual machine instances and virtual networks and use them to communicate with each other and with external networks.

## Using Fab Commands to Configure Contrail Compute Nodes on VMware ESXi

As of Contrail Release 2.0 and greater, you can use fab (Fabric) commands to configure the VMware ESXi hypervisor as a Contrail compute node. Refer to Contrail Installation and Provisioning Roles for details of using fab commands for installation.

### Requirements Before You Begin

The guidelines for using fab commands to configure Contrail compute node on an ESXi server have the following prerequisites:

- The **testbed.py** must be populated with both ESXi hypervisor information and Contrail virtual machine information.
- The ESXi hypervisor must be up and running with an appropriate ESXi version.



**NOTE:** ESXi cannot be installed using fab commands in Contrail.

## Fab Installation Guidelines for ESXi

Use the following guidelines when using fab commands to set up ESXi as a compute node for Contrail.

1. Issue **fab prov\_esxi** to provision the ESXi with the required vswitches, port groups, and the **contrail-compute-vm**.

The ESXi hypervisor information is provided in the **esxi\_hosts** stanza, as shown in the following..

#Following are ESXi Hypervisor details.

```
esxi_hosts = {

#Ip address of Hypervisor

    'esxi_host1' : {'ip': '<ip address>'},

#Username and password of ESXi Hypervisor

    'username': 'user',

    'password': '<password>',

#Uplink port of Hypervisor through which it is connected to external world

    'uplink_nic': 'vmnic2',

#Vswitch on which above uplink exists

    'fabric_vswitch' : 'vSwitch0',

#Port group on 'fabric_vswitch' through which ContrailVM connects to external
world

    'fabric_port_group' : 'contrail-fab-pg',

#Vswitch name to which all openstack virtual machine's are hooked to

    'vm_vswitch': 'vSwitch1',

#Port group on 'vm_vswitch', which is a member of all vlans, to which ContrailVM
is connected to all openstack VM's

    'vm_port_group' : 'contrail-vm-pg',

#links 'host2' ContrailVM to esxi_host1 hypervisor

    contrail_vm : {

        'name': 'ContrailVM2' # Name for the contrail-compute-vm,
        'mac' : '00:50:56:aa:ab:ac', # VM's eth0 mac address, same should be
        configured on DHCP server

        'host' : host2, # host string for VM, as specified in the
        env.rolesdef['compute']

        'vmdk' : 'file.vmdk' # local path of the VMDK file

    },

# Another ESXi hypervisor follows

}
```



NOTE: The VMDK for `contrail-compute-vm` (ESXi-v5.5-Contrail-host-Ubuntu-precise-12.04.3-LTS.vmdk) can be downloaded from <https://www.juniper.net/support/downloads/?p=contrail#sw>



NOTE: The `contrail-compute-vm` gets the IP address and its host name from the DHCP server.

2. The standard installation fab commands, such as `fab install_pkg_all`, `fab install_contrail` and `fab setup_all` can now be used to finish the setup of the entire cluster.

Refer to [Installing the Contrail Packages, Part Two](#) for details of using fab commands for installation.

---

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

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

All installation files are available from <http://www.juniper.net/support/downloads/?p=contrail#sw>.

**CentOS Systems** Contrail packages for CentOS are provided either as part of the Contrail ISO installation or separately in an RPM file with the format: **contrail-install-packages-1.xx-xxx~openstack\_version.el6.noarch.rpm**, where **xx-xxx~openstack\_version** represents the release number, build number, and OpenStack common version name (such as Havana or Icehouse) for the included Contrail install packages.

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

**Ubuntu Systems** Contrail packages for Ubuntu are provided only as packages in a Debian file of the format: **contrail-install-packages-1.xx-xxx~openstack\_version\_all.deb**, where **xx-xxx~openstack\_version** represents the release number, build number, and OpenStack common version name (such as Havana or Icehouse) for the included Contrail install packages.

It is expected that you already have a compatible Ubuntu operating system installed, such as Ubuntu12.04.3 LTS, kernel version 3.13.0-2934 generic, before installing the Contrail packages.



**NOTE:** the stock kernel version as part of 12.04.3 or 12.04.4 LTS is older than 3.13.0-34. In such cases, the following Fabric task can be used to upgrade the kernel version to 3.13.0-34 in all nodes.

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

#### *Installing Contrail Packages for CentOS or Ubuntu*

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

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

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

*CentOS:* **contrail-install-packages-1.xx-xxx~openstack\_version.el6.noarch.rpm**

*Ubuntu:* **contrail-install-packages-1.xx-xxx~openstack\_version\_all.deb**

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

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

*CentOS:* **scp**

```
<id@server>:/path/to/contrail-install-packages-1.xx-xxx~openstack_version.el6.noarch.rpm
/tmp
```

*Ubuntu:* `scp`

`<id@server>:/path/to/contrail-install-packages-1.xx-xxx~openstack_version_all.deb`  
`/tmp`

5. Install the Contrail packages:

*CentOS:* `yum localinstall`

`/tmp/contrail-install-packages-1.xx-xxx~openstack_version.el6.noarch.rpm`

*Ubuntu:* `dpkg -i /tmp/contrail-install-packages-1.xx-xxx~openstack_version_all.deb`

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

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

7. Populate the `testbed.py` definitions file, see [“Populating the Testbed Definitions File” on page 26](#).



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

**Related  
Documentation**

- [Populating the Testbed Definitions File on page 26](#)
- [Download Software](#)
- [Supporting Multiple Interfaces on Servers and Nodes on page 29](#)
- [Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines on page 47](#)
- [Configuring the Control Node on page 50](#)

---

## Populating the Testbed Definitions File

Populate a testbed definitions file, `/opt/contrail/utls/fabfile/testbeds/testbed.py`, with parameters specific to your system, then run the fab commands as provided in [“Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines” on page 47](#) to launch the role-based provisioning script tasks.

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

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

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



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

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

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

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

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

If there are no external routers, define

```
ext_routers = []
```

3. Provide the BGP autonomous system number.

```
router_asn = 64512
```



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

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

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

5. Define which hosts will operate with which roles.

*For multinode setups:*

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

*For single node all-in-one setups:*

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

6. Define password credentials for each of the hosts.

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



**NOTE:** Ensure *both* `env.password` and `env.passwords` are set for releases prior to Contrail Release 1.10.



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

If your system servers and nodes have multiple interfaces, refer to [“Supporting Multiple Interfaces on Servers and Nodes” on page 29](#) for information about setting up the `testbed.py` for your system.

To deploy the Contrail High Available cluster, refer to [“Juniper OpenStack High Availability” on page 193](#) for information about setting up the `testbed.py` for your system.

To deploy with an existing Openstack, refer to [“Testbed Definitions File Settings for Deploying Contrail with an Existing OpenStack Node” on page 36](#) for `testbed.py` definitions.

When finished, continue on to [“Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines” on page 47](#).

#### Related Documentation

- [Supporting Multiple Interfaces on Servers and Nodes on page 29](#)
- [Testbed Definitions File Settings for Deploying Contrail with an Existing OpenStack Node on page 36](#)
- [Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines on page 47](#)

## Supporting Multiple Interfaces on Servers and Nodes

---

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

- [Support for Multiple Interfaces on page 29](#)
- [Server Interface Examples on page 30](#)
- [Interface Naming and Configuration Management on page 31](#)
- [Setting Up Interfaces and Installing on page 31](#)
- [Sample testbed.py File With Exclusive Interfaces on page 32](#)

### 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.

For Contrail Release 1.10 and later, control and data must share the same interface, configured in **testbed.py** 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

---

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

### Support for VLAN Interfaces

---

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

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

### Support for Bonding Options

---

Contrail Release 1.10 and later provides support for all available bond interface options.

The default bond interface options are:

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

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

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

### Support for Static Route Options

---

Contrail Release 1.04 and later provides support for adding static routes on target systems. This option is ideal for use cases in which a system has servers with multiple interfaces and has control\_data or management connections that span multiple networks.

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

```
static_route = {  
  
    host2: [{ 'ip': '<ip address>', 'netmask': '<ip address>', 'gw': '<ip  
address>', 'intf': 'bond0' },  
  
            { 'ip': '<ip address>', 'netmask': '<ip address>',  
  
            'gw': '<ip address>', 'intf': 'bond0' }],  
  
    host5: [{ 'ip': '<ip address>', 'netmask': '<ip address>', 'gw': '<ip  
address>', 'intf': 'bond0' }],  
  
}
```

### Server Interface Examples

For 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.

The following server interface combinations are no longer allowed for Contrail Release 1.10 and later:

- **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 gives interface names such as **p4p0** and **p4p1** for onboard interfaces, the Contrail system generates those names as **p4p0p0** and **p4p0p1**, when using the optional package **contrail-interface-name**.

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

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

## Setting Up Interfaces and Installing

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

The **fab setup\_interface** command creates bond interface configurations, if there is a corresponding configuration in the **testbed.py** file (see sample **testbed.py** file in the next section).

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

The **fab add\_static\_route** command creates static routes in a node, if there is a corresponding configuration in the **testbed.py** file (see sample **testbed.py** file in the next section).

A typical work flow for setting up a cluster with multiple interfaces follows:

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

**fab install\_contrail** (meaning: change the **testbed.py** file with the renamed interface name)

**fab setup\_interface**

**fab add\_static\_route**

**fab setup\_all**



**NOTE:** The **fab setup\_interface** command and **fab add\_static\_route** command can be executed simultaneously by using the **fab setup\_network** command.

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

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

## Sample testbed.py File With Exclusive Interfaces

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

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

host1 = '<user@<ip address>'
host2 = '<user@<ip address>'
host3 = '<user@<ip address>'
host4 = '<user@<ip address>'
host5 = '<user@<ip address>'
host6 = '<user@<ip address>'
host7 = '<user@<ip address>'
host8 = '<user@<ip address>'

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

host_build = user@<ip address>'

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

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

```

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

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

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

env.passwords = {
  host1:'secret',
  host2:'secret',
  host3:'secret',
  host4:'secret',
  host5:'secret',
  host6:'secret',
  host7:'secret',
  host8:'secret',

  host_build: 'secret'
}

```

**Related Documentation** • [Juniper OpenStack High Availability on page 193](#)

## High Availability Support

This section describes how to set up Contrail options for high availability support.

- In Ubuntu setups, OpenStack high availability and Contrail high availability are both supported, for Contrail Release 1.10 and greater.
- In CentOS setups, only Contrail high availability is supported, and only for Contrail Release 1.20 and greater.
- [Contrail High Availability Features on page 34](#)
- [Configuration Options for Enabling Contrail High Availability on page 34](#)
- [Supported Cluster Topologies for High Availability on page 35](#)
- [Deploying OpenStack and Contrail on the Same High Available Nodes on page 35](#)
- [Deploying OpenStack and Contrail on Different High Available Nodes on page 35](#)
- [Deploying Contrail Only on High Available Nodes on page 36](#)

## Contrail High Availability Features

The Contrail OpenStack high availability design and implementation provides:

- A high availability active-active implementation for scale-out of the cloud operation and for flexibility to expand the controller nodes to service the compute fabric.
- Anytime availability of the cloud for operations, monitoring, and workload monitoring and management.
- Self-healing of the service and states.
- VIP-based access to the cloud operations API provides an easy way to introduce new controllers and an API to the cluster with zero downtime. Improved capital efficiencies compared with dedicated hardware implementations, by using nodes assigned to controllers and making them a federated node in the cluster.
- Operational load distribution across the nodes in the cluster.

For more details about high availability implementation in Contrail, see [“High Availability Support” on page 33](#).

## Configuration Options for Enabling Contrail High Availability

The following are options available to configure high availability within the Contrail configuration file (**testbed.py**).

Option	Description
<b>internal_vip</b>	The virtual IP of the OpenStack high availability nodes in the control data network. In a single interface setup, the <b>internal_vip</b> will be in the management data control network.
<b>external_vip</b>	The virtual IP of the OpenStack high availability nodes in the management network. In a single interface setup, the <b>external_vip</b> is not required.
<b>contrail_internal_vip</b>	The virtual IP of the Contrail high availability nodes in the control data network. In a single interface setup, the <b>contrail_internal_vip</b> will be in the management data control network.
<b>contrail_external_vip</b>	The virtual IP of the Contrail high availability nodes in the management network. In a single interface setup, the <b>contrail_external_vip</b> is not required.
<b>nfs_server</b>	The IP address of the NFS server that will be mounted to <b>/var/lib/glance/images</b> for the openstack node. The default is to <b>env.roledefs['compute'][0]</b> .
<b>nfs_glance_path</b>	The NFS server path to save images. The default is to <b>/var/tmp/glance-images/</b> .
<b>manage_amqp</b>	A flag to tell the <b>setup_all</b> task to provision separate <b>rabbitmq</b> setups for openstack services in openstack nodes.



## Supported Cluster Topologies for High Availability

This section describes configurations for the cluster topologies supported, including:

- OpenStack and Contrail on the same high available nodes
- OpenStack and Contrail on different high available nodes
- Contrail only on high available nodes

### Deploying OpenStack and Contrail on the Same High Available Nodes

OpenStack and Contrail services can be deployed in the same set of high available nodes by setting the **internal\_vip** parameter in the **env.ha** dictionary of the **testbed.py**.

Because the high available nodes are shared by both OpenStack and Contrail services, it is sufficient to specify only **internal\_vip**. However, if the nodes have multiple interfaces with management and data control traffic separated by provisioning multiple interfaces, then the **external\_vip** also needs to be set in the **testbed.py**.

**Example**

```
env.ha = {
    'internal_vip' : 'an-ip-in-control-data-network',
    'external_vip' : 'an-ip-in-management-network',
}
```

### Deploying OpenStack and Contrail on Different High Available Nodes

OpenStack and Contrail services can be deployed on different high available nodes by setting the **internal\_vip** and the **contrail\_internal\_vip** parameter in the **env.ha** dictionary of the **testbed.py**.

Because the OpenStack and Contrail services use different high available nodes, it is required to separately specify **internal\_vip** for OpenStack high available nodes and **contrail\_internal\_vip** for Contrail high available nodes. If the nodes have multiple interfaces, with management and data control traffic separated by provisioning multiple interfaces, then the **external\_vip** and **contrail\_external\_vip** options also must be set in the **testbed.py**.

**Example**

```
env.ha = {
    'internal_vip' : 'an-ip-in-control-data-network',
    'external_vip' : 'an-ip-in-management-network',
    'contrail_internal_vip' : 'another-ip-in-control-data-network',
    'contrail_external_vip' : 'another-ip-in-management-network',
}
```

To manage separate **rabbitmq** clusters in the OpenStack high available nodes for OpenStack services to communicate, specify **manage\_amqp** in the **env.openstack**

dictionary of **testbed.py**. If **manage\_amqp** is not specified, the default is for the OpenStack services to use the **rabbitmq** cluster available in the Contrail high available nodes for communication.

**Example:**

```
env.openstack = {  
  
    'manage_amqp' : 'yes'  
  
}
```

## Deploying Contrail Only on High Available Nodes

Contrail services can be deployed only on a set of high available nodes by setting the **contrail\_internal\_vip** parameter in the **env.ha** dictionary of the **testbed.py**.

Because the high available nodes are used by only Contrail services, it is sufficient to specify only **contrail\_internal\_vip**. If the nodes have multiple interfaces with management and data control traffic are separated by provisioning multiple interfaces, the **contrail\_external\_vip** also needs to be set in the **testbed.py**.

**Example**

```
env.ha = {  
  
    'contrail_internal_vip' : 'an-ip-in-control-data-network',  
  
    'contrail_external_vip' : 'an-ip-in-management-network',  
  
}
```

To manage separate **rabbitmq** clusters in the OpenStack node for the OpenStack services to communicate, specify **manage\_amqp** in the **env.openstack** dictionary of the **testbed.py**. If the **manage\_amqp** is not specified, the default is the OpenStack services will use the cluster available in the Contrail high available nodes for communication.

**Example:**

```
env.openstack = {  
  
    'manage_amqp' : 'yes'  
  
}
```

- Related Documentation**
- [Juniper OpenStack High Availability on page 193](#)
  - *Example: Adding New OpenStack or Contrail Roles to an Existing High Availability Cluster*

---

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

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

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

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

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

2. Update the keystone environment section as in the following.

```
env.keystone = {
  'keystone_ip' : 'x.y.z.a',    # same IP as the openstack IP address

  'auth_protocol' : 'http',    # Default is http

  'auth_port' : '35357',       # Default is 35357

  'admin_token' : '$ABC123',
                                # Uses the admin_token from /etc/keystone/keystone.conf
of the openstack node
  'admin_user' : 'admin',      # Default is admin

  'admin_password': '<password>',
  'service_tenant': 'service', # Default is service
  'admin_tenant' : 'admin',    # Default is admin

  'region_name' : 'RegionOne', # Default is RegionOne

  'insecure' : 'True',        # Default = False, however, "insecure" is applicable only when
protocol is https
  'manage_neutron': 'no',     # Default = 'yes' , Does configure neutron user/role in
keystone required.

}
```

3. Update the openstack environment section as in the following, where:

- **service\_token** is the common service token for all services like nova, neutron, glance, cinder, and so on.
- **amqp\_host** is the IP of the AMQP server to be used for openstack.
- **manage\_amqp**, the default = 'no', if set to 'yes', provisions amqp in openstack nodes and openstack services uses the amqp in openstack nodes instead of config nodes. The **amqp\_host** is neglected if **manage\_amqp** is set.

```
env.openstack = {

  'service_token': '$ABC123',    # the admin_token from keystone.conf of the
                                openstack node

  'amqp_host' : '<ip address>', # same as the IP address of the openstack node

  'manage_amqp' : 'yes',

}
```

#### Related Documentation

- [Supporting Multiple Interfaces on Servers and Nodes on page 29](#)
- [Installing the Contrail Packages, Part Two \(CentOS or Ubuntu\) — Installing on the Remaining Machines on page 47](#)

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

---

- [Introduction to the Simple Gateway on page 38](#)
- [How the Simple Gateway Works on page 38](#)
- [Setup Without Simple Gateway on page 38](#)
- [Setup With Simple Gateway on page 39](#)
- [Simple Gateway Configuration Features on page 40](#)
- [Packet Flows with the Simple Gateway on page 41](#)
- [Packet Flow Process From the Virtual Network to the Public Network on page 41](#)
- [Packet Flow Process From the Public Network to the Virtual Network on page 42](#)
- [Four Methods for Configuring the Simple Gateway on page 42](#)
- [Using Fab Provisioning to Configure the Simple Gateway on page 42](#)
- [Using the vRouter Configuration File to Configure the Simple Gateway on page 44](#)
- [Using Thrift Messages to Dynamically Configure the Simple Gateway on page 44](#)
- [Common Issues with Simple Gateway Configuration on page 47](#)

### Introduction to the Simple Gateway

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

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

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

### How the Simple Gateway Works

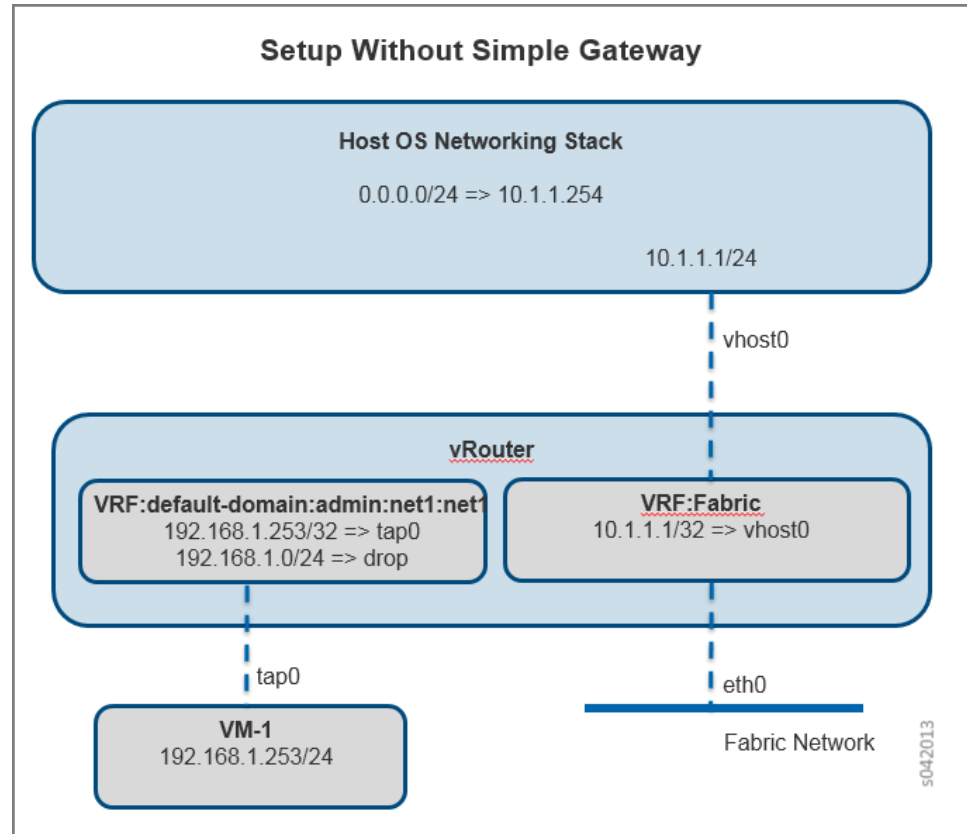
The following sections illustrate how the simple gateway works, first, by showing a virtual network setup with no simple gateway, then illustrating the same setup with a simple gateway configured.

#### Setup Without Simple Gateway

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

- A virtual network, `default-domain:admin:net1`, is configured with the subnet `192.168.1.0/24`.
- The routing instance `default-domain:admin:net1:net1` is associated with the virtual network `default-domain:admin:net1`.

- A virtual machine with the IP 192.168.1.253 is spawned in net1.
- A virtual machine is spawned on compute server 1.
- An interface, vhost0, is in the host OS of server 1 and is assigned the IP 10.1.1.1/24.
- The interface vhost0 is added to the vRouter in the routing instance fabric.
- The simple gateway is not configured.



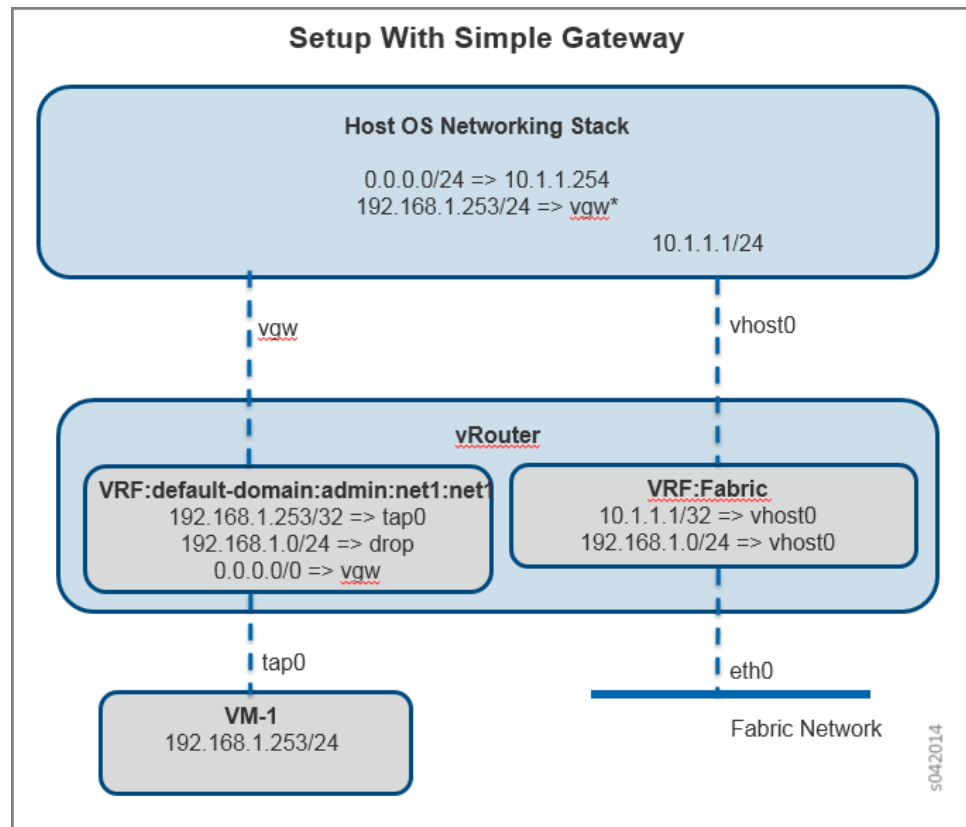
### Setup With Simple Gateway

The following diagram shows a virtual network setup with the simple gateway configured for the virtual network default-domain:admin:net1.

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

The following shows the packet flows between Fabric and the default-domain:admin:net1:net1.

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



## Simple Gateway Configuration Features

The simple gateway configuration has the following features.

- The simple gateway is configured for the virtual network **default-domain:admin:net1**.
  - The gateway interface `vgw` provides connectivity between the routing instance **default-domain:admin:net1:net1** and the fabric.
  - An IP address is not configured for the gateway interface **vgw**.
- The host OS is configured with the following:
  - Two INET interfaces are added to the host OS: **vgw** and **vhost0**
  - The host OS is not aware of the routing instances, so **vgw** and **vhost0** are part of the same routing instance in the host OS.
  - The simple gateway adds the route `192.168.1.0/24`, pointing to the **vgw** interface, and that setup is added to the host OS. This route ensures that any packet destined to the virtual machine is sent to the vrouter on the `vgw` interface.
- The vRouter is configured with the following:

- The routing instance named **Fabric** is created for the fabric network.
- The interface **vhost0** is added to the routing instance **Fabric**.
- The interface **eth0**, which is connected to the fabric network, is added to the routing instance named **Fabric**.
- The simple gateway adds the route **192.168.1.0/24 => vhost0**, consequently, packets destined to the virtual network **default-domain:admin:net1** are sent to the host OS.
- The routing instance **default-domain:admin:net1:net1** is created for the virtual network **default-domain:admin:net1**.
  - The interface **vgw** is added to the routing instance **default-domain:admin:net1:net1**.
  - The simple gateway adds a default route **0.0.0.0/0** that points to the interface **vgw**. Packets in the routing instance **default-domain:admin:net1:net** that hit this route are sent to the host OS on the **vgw** interface. The host OS routes the packets to the Fabric network over the **vhost0** interface.

#### Simple Gateway Restrictions

The following are restrictions of the simple gateway.

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

### Packet Flows with the Simple Gateway

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

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

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

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

1. A packet with **source-ip=192.168.1.253** and **destination-ip=10.1.1.253** comes from a virtual machine and is received by the vRouter on interface **tap0**.
2. The interface **tap0** is in the routing instance of **default-domain:admin:net1:net1**.
3. The route lookup for **10.1.1.253** in the routing instance **default-domain:admin:net1:net1** finds the default route pointing to the tap interface named **vgw**.
4. The vRouter transmits the packet toward **vgw** and it is received by the networking stack of the host OS.
5. The host OS performs forwarding based on its routing table and forwards the packet on the **vhost0** interface.

6. Packets transmitted on **vhost0** are received by the vRouter.
7. The vhost0 interface is added to the routing instance Fabric.
8. The routing table for 10.1.1.253 in the routing instance Fabric indicates that the packet is to be transmitted on the eth0 interface.
9. The vRouter transmits the packet on the **eth0** interface.
10. The host 10.1.1.253 on Fabric receives the packet.

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

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

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

## Four Methods for Configuring the Simple Gateway

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

### Using Fab Provisioning to Configure the Simple Gateway

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

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



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

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

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

If the vgw interfaces are defined in **env.roledefs**, when provisioning the system nodes with the command **fab setup\_all**, the vgw interfaces will be provisioned, along with all of the other nodes.

**Example: Testbed.py  
Env.roledefs for vgw**

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

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

```
        'ipam-subnets': ['10.204.220.144/29']]],  
    host5: {  
        'vgw2': {  
            'vn': 'default-domain:admin:public1:public1',  
            'ipam-subnets': ['10.204.220.144/29']  
        }  
    }  
}
```

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

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

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

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

## Using Thrift Messages to Dynamically Configure the Simple Gateway

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

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

- **AddVirtualGateway**—add a virtual gateway
- **DeleteVirtualGateway**—delete a virtual gateway
- **ConnectForVirtualGateway**—allows audit of the virtual gateway configuration by stateful clients. Upon a new **ConnectForVirtualGateway** request, one minute is allowed

for the configuration to be redone. Any older virtual gateway configuration remaining after this time is deleted.

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

### How to Dynamically Create a Virtual Gateway

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

When run, the script does the following:

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

#### Example: Dynamically Create a Virtual Gateway

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

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

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

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

2. Run the `vgw provision` command with the `oper create` option.

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

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

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

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

### How to Dynamically Delete a Virtual Gateway

To dynamically delete a virtual gateway, you run a script on the compute node where the virtual gateway was created.

When run, the script does the following:

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

#### Example: Dynamically Create a Virtual Gateway

The following procedure dynamically deletes the interface vgw1, and also deletes the subnets 20.30.40.0/24 and 30.40.50.0/24 in the vrf **default-domain:admin:vn1:vn1**.

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

```
export
PYTHONPATH=/usr/lib/python2.7/dist-packages/nova_contrail_vif/gen_py/instance_service
export
PYTHONPATH=/usr/lib/python2.6/site-packages/contrail_vrouter_api/gen_py/instance_service
```
2. Run the vgw provision command with the oper delete option.
 

```
python /opt/contrail/utils/provision_vgw_interface.py --oper delete --interface vgw1
--subnets 20.30.40.0/24 30.40.50.0/24 --routes 8.8.8.0/24 9.9.9.0/24
```
3. (optional) If using a stateful client, send the **ConnectForVirtualGateway** thrift message to the vRouter agent when the client starts.



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

### Using Devstack to Configure the Simple Gateway

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

The following parameters are available:

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

This method can only add the default route 0.0.0.0/0 into the routing instance specified in **CONTRAIL\_VGW\_PUBLIC\_NETWORK**.

#### Example: Devstack Configuration for Simple Gateway

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

```
CONTRAIL_VGW_INTERFACE=vgw1
```

```
CONTRAIL_VGW_PUBLIC_SUBNET=192.168.1.0/24
```

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



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

## Common Issues with Simple Gateway Configuration

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

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

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

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

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

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

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

### Preinstallation Checklist



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

- [Installing the Contrail Packages, Part One \(CentOS or Ubuntu\) on page 24](#)
- [Populating the Testbed Definitions File on page 26](#)

And the following system tasks are accomplished:

- All of the servers are time synced.
- All servers can ping from one to another, both on management and on data and control, if part of the system.
- All servers can `ssh` and `scp` between one another.

- All host names are resolvable.
- If using CentOS or RHEL, SELinux has been disabled (`/etc/sysconfig/selinux`).

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

To copy and install Contrail packages on the remaining machines in your cluster, you can use `scp` and `yum localinstall` as on the first server (for CentOS) or `scp` and `dpkg -i` (for Ubuntu), as in [“Installing the Contrail Packages, Part One \(CentOS or Ubuntu\)” on page 24](#), or you can use a Fabric utility to copy onto all machines at once, as follows:

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

See [“Populating the Testbed Definitions File” on page 26](#).

2. Run Fabric commands to install packages as follows:

*CentOS:* `/opt/contrail/utls/fab`

`install_pkg_all:/tmp/contrail-install-packages-1.xx-xxx~openstack_version..el6.noarch.rpm`

*Ubuntu:* `/opt/contrail/utls/fab`

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



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

3. *Ubuntu:* The recommended Kernel version for Ubuntu based system is 3.13.0-34. Nodes can be upgraded to kernel version 3.13.0-34 using below fabric-utls command :

`fab upgrade_kernel_all`



**NOTE:** This step upgrades the kernel version to 3.13.0-34 in all nodes and performs reboot. Reconnect to perform remaining tasks.

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

`fab install_contrail`



**NOTE:** To install Contrail with an existing OpenStack node:

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

or

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

5. If your installation has multiple interfaces (see [“Supporting Multiple Interfaces on Servers and Nodes” on page 29](#)), run `setup_interface`:

```
fab setup_interface
```

6. Provision the entire cluster:

```
fab setup_all
```



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

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

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

When finished, you can proceed to [“Configuring the Control Node” on page 50](#).

#### Related Documentation

- [Configuring the Control Node on page 50](#)

## Configuring the Control Node

---

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

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

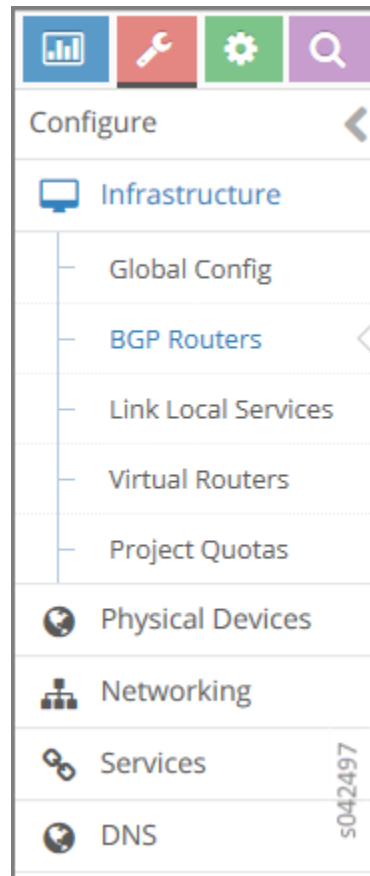
- The Contrail Controller base system image has been installed on all servers.
- The role-based services have been assigned and provisioned.
- IP connectivity has been verified between all nodes of the Contrail Controller.
- You can access the Contrail user interface at **<http://nn.nn.nn.nn:8080>**,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

To configure BGP peering in the control node:



1. From the Contrail Controller module control node (<http://nn.nn.nn.nn:8080>), select **Configure > Infrastructure > BGP Routers**; see [Figure 1 on page 51](#).

Figure 1: Configure &gt; Infrastructure &gt; BGP Routers



A summary screen of the control nodes and BGP routers appears; see [Figure 2 on page 51](#).

Figure 2: BGP Routers Summary

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

2. (Optional) The global AS number is 64512 by default. To change the global AS number, on the **BGP Router** summary screen click **Global ASN** and enter the new number.

- To configure control nodes and BGP routers, on the **BGP Routers** summary screen, click the **+** icon. The **Create BGP Router** window is displayed; see [Figure 3 on page 52](#).

**Figure 3: Create BGP Router**

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

For each node you want to add, populate the fields with values for your system. See [Table 3 on page 52](#).

**Table 3: Create BGP Router Fields**

Field	Description
Hostname	Enter a name for the node being added.
IP Address	The IP address of the node.
Autonomous System	Enter the AS number for the node. (BGP peer only)
Router ID	Enter the router ID.
Vendor ID	Required for external peers. Populate with a text identifier, for example, "MX-0". (BGP peer only)
Address Families	Enter the address family, for example, <b>inet-vpn</b>
Hold Time	BGP session hold time. The default is 90 seconds; change if needed.

Table 3: Create BGP Router Fields (*continued*)

Field	Description
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.

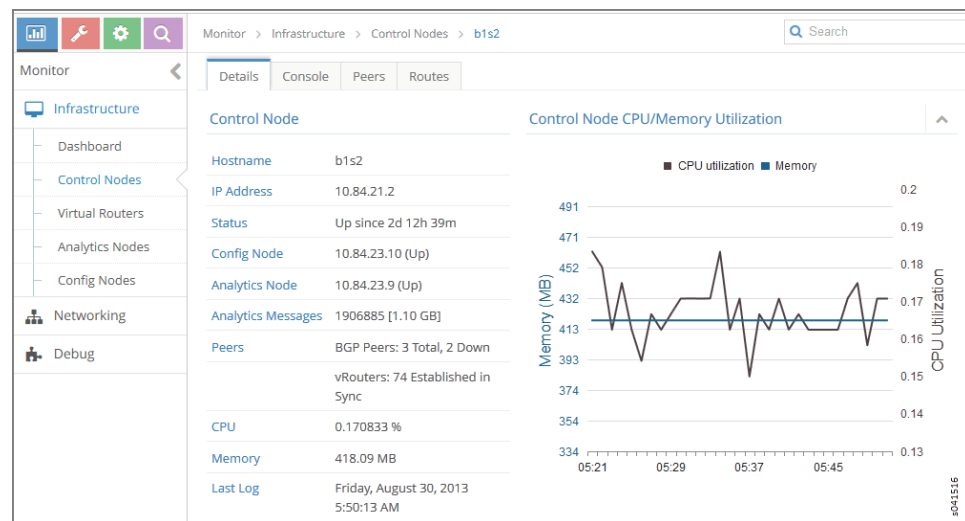
- Click **Save** to add each node that you configure.
- To configure an existing node as a peer, select it from the list in the **Available Peers** box, then click >> to move it into the **Configured Peers** box.  
Click << to remove a node from the **Configured Peers** box.
- You can check for peers by selecting **Monitor > Infrastructure > Control Nodes**; see [Figure 4 on page 53](#).

Figure 4: Control Nodes



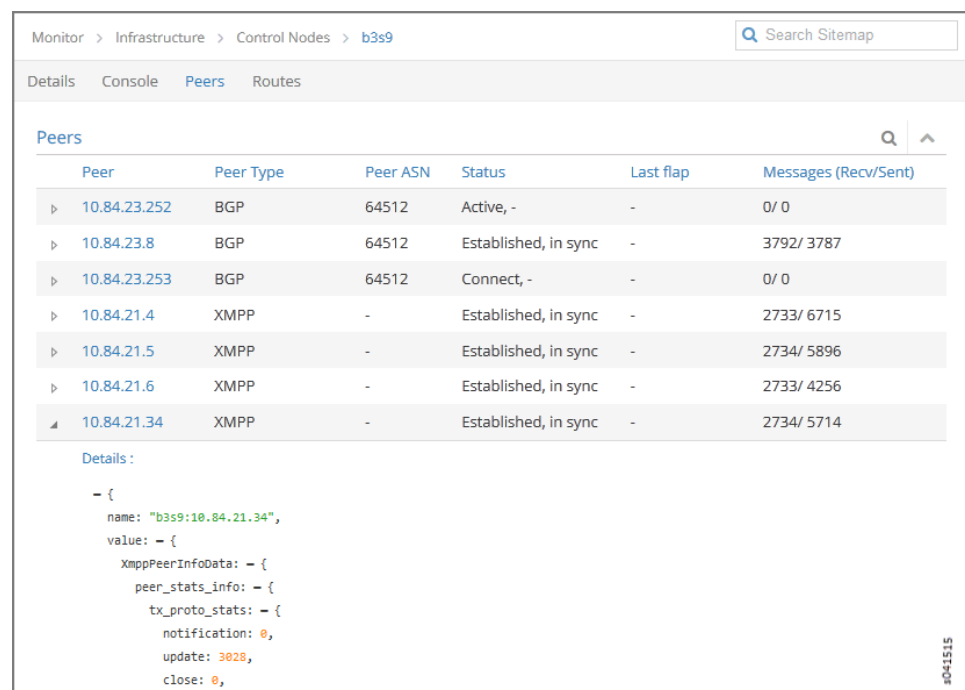
In the **Control Nodes** screen, click any hostname from the memory map to view its details; see [Figure 5 on page 54](#).

Figure 5: Control Node Details



8. Click the **Peers** tab to view the peers of a control node; see [Figure 6 on page 54](#).

Figure 6: Control Node Peers Tab



#### Related Documentation

- [Creating a Virtual Network—Juniper Networks Contrail](#)
- [Creating a Virtual Network—OpenStack Contrail on page 139](#)

## Using Server Manager to Automate Provisioning

---

- [Overview of Server Manager on page 55](#)
- [Server Manager Requirements and Assumptions on page 55](#)
- [Server Manager Component Interactions on page 57](#)
- [Configuring Server Manager on page 58](#)
- [Configuring the Cobbler DHCP Template on page 59](#)
- [User-Defined Tags for Server Manager on page 60](#)
- [Server Manager Client Configuration File on page 60](#)
- [Restart Services on page 61](#)
- [Accessing Server Manager on page 61](#)
- [Communicating with the Server Manager Client on page 62](#)
- [Server Manager Commands for Configuring Servers on page 62](#)
- [Server Manager REST API Calls on page 80](#)
- [Example: Reimaging and Provisioning a Server on page 86](#)

### Overview of Server Manager

Starting with Contrail Release 1.10, the Contrail Server Manager can be used to provision, configure, and reimage a Contrail virtual network system of servers, clusters, and nodes. Server Manager is an alternative to using Fabric commands to provision a Contrail system.

This section describes the functions and usage guidelines for the Contrail server manager.

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

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

The server manager:

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

### Server Manager Requirements and Assumptions

The following requirements are assumed 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.
  - Contrail 1.10 server manager assumes that DNS and DHCP servers embedded with Cobbler provide IP addresses and names to the servers being managed, although it is possible to use external DNS and DHCP servers.
- The server manager uses Puppet software, an open source configuration management tool, to accomplish the configuration management of target servers, including the installation and configuration of different software packages and the launching of various services.
- SQLite3 database management software is used to maintain and manage server configurations and it runs on the same machine where the server manager daemon is running.

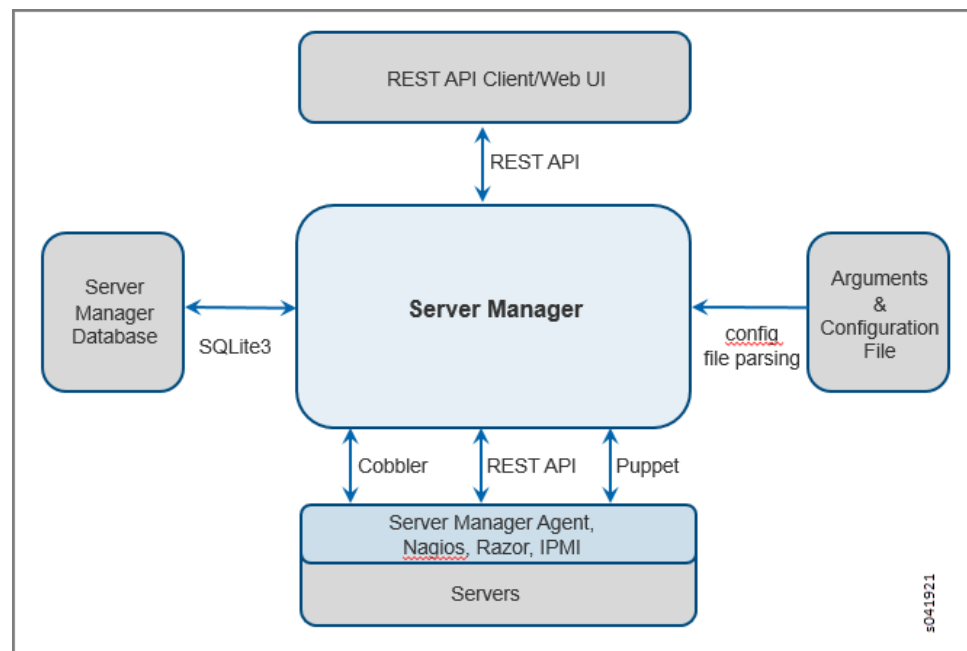
## Server Manager Component Interactions

The server manager runs as a daemon and provides REST APIs for interaction with the client. The server manager accepts user input in the form of REST API requests, performs the requested function on the resources, and responds to the user 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 7 on page 57 illustrates several high-level components with which the server manager interacts.

**Figure 7: Server Manager Component Interactions**



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

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

A server manager agent runs on each of the servers and communicates with the server manager, providing the information needed to monitor the operation of the servers. The server manager agent also uses REST APIs to communicate with the server manager, and it can use other software tools to fetch other information, such as Nagios infrastructure monitoring, Razor spam filtering, Intelligent Platform Interface (IPMI), and

so on. Monitoring functionality is not part of server manager for the Contrail release. It will be available at a later date.

## 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\_user**, and **cobbler\_passwd** values need to be modified.

```
[SERVER-MANAGER]

listen_ip_addr = <IP-Address-of-SM>

listen_port    = <port-number>

database_name  = <database-file-name>

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

html_root_dir  = <html-root-dir>

cobbler_ip_address = <cobbler-ip-address>

cobbler_port    = <cobbler-port-number>

cobbler_username = <cobbler-username>

cobbler_password = <cobbler-password>

puppet_dir     = <puppet-directory>

ipmi_username   = <IPMI username>

ipmi_password   = <IPMI password>

ipmi_type       = <IPMI type>
```

[Table 4 on page 59](#) provides details for each of the parameters in **SERVER-MANAGER** section.



Table 4: Server Manager Parameters

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

## Configuring the Cobbler DHCP Template

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

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



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

```
subnet <ip address> netmask <ip address> {  
  
    option routers          <ip address>;  
  
    option subnet-mask      <ip address>;  
  
    option domain-name-servers $next_server, <ip address> ;  
  
    option domain-search    "domain name 1", "domain name 2";  
  
    option domain-name      "domain name" ;  
  
    option ntp-servers      $next_server;  
  
    default-lease-time      21600;  
  
    max-lease-time          43200;  
  
    next-server              $next_server;  
  
    filename                 "/pxelinux.0";  
  
}
```

## User-Defined Tags for Server Manager

Server manager provides the user the ability to define tags that can be used to group servers for performing a particular operation such as show information, reimage, provision, and so on. The server manager supports up to seven different tags that can be configured and used for grouping servers.

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

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

The following is a sample **tags.ini** file that is copied on installation. In the sample file, the user has defined five tags – **datacenter**, **floor**, **hall**, **rack**, and **user\_tag**. The tags can be used 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 a sample configuration file, **/opt/contrail/server\_manager/client/sm-client-config.ini**, that contains parameter values such as the IP address to reach the server manager, the port used by server manager, default values for cluster table entries, default values for server table entries, and so on. You must modify the values in the **sm-client-config.ini** file to match your environment.

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

The following is a sample client configuration file.

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

## Restart Services

When all user changes have been made to the configuration files, restart the server manager to run with the modifications by issuing the following:

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

## Accessing Server Manager

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

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

1. Configure elements such as servers, clusters, and images in the server manager.  
Before managing servers, the server manager needs to have configuration details of the servers that the server manager is managing.
2. Specify the name and location of boot images, packages, and repositories used to bring up the servers with needed software.  
Currently, the servers can be imaged with CentOS, Ubuntu Linux, or VMWare ESXi distributions.

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

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

4. Set up API calls for monitoring servers.

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

## Communicating with the Server Manager Client

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

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

Each of the commands described in this section takes a set of parameters from the user, constructs a REST API request to the server manager, and provides the server's response to the user.

The following describes each server manager client CLI command in detail.

## Server Manager Commands for Configuring Servers

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

- [Create New Servers or Update Existing Servers on page 63](#)
- [Delete Servers on page 64](#)
- [Show Server Configuration on page 65](#)
- [Server Manager Commands for Managing Clusters on page 66](#)
- [Server Manager Commands for Managing Tags on page 69](#)
- [Server Manager Commands for Managing Images on page 71](#)
- [Server Manager Operational Commands for Managing Servers on page 74](#)
- [Reimaging Server\(s\) on page 75](#)

- [Provisioning and Configuring Roles on Servers on page 76](#)
- [Restarting Server\(s\) on page 78](#)
- [Show Status of Server\(s\) on page 79](#)

Create New Servers or Update Existing Servers

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

Usage:

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

[Table 5 on page 63](#) lists the optional arguments.

Table 5: Server Manager Add Server Command Options

Option	Description
-h, --help	Show the options available for the current command and exit.
--config_file CONFIG_FILE, -c CONFIG_FILE	The name of the server manager client configuration file. The default file is: /opt/contrail/server_manager/client/sm-client-config.ini
--file_name FILE_NAME, -f FILE_NAME	The JSON file that contains the server parameter values.

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

```
{
  "server": [
    {
      "id": "demo2-server",
      "mac_address": "<mac address>",
      "ip_address": "<ip address>",
      "parameters": {
        "interface_name": "eth1",
        "partition": ""
      },
      "roles": [
        "config", "openstack", "control", "compute", "collector", "webui", "database"
      ]
    }
  ]
}
```

```

"cluster_id": "demo-cluster",

"subnet_mask": "<ip address>",

"gateway": "<ip address>",

"password": "<password>r",

"domain": "<domain name>",

"ipmi_address": "<ip address>",

"tag" : {

    "datacenter" : "demo-dc",

    "floor" : "demo-floor",

    "hall" : "demo-hall",

    "rack" : "demo-rack",

    "user_tag" : "demo-user"

}

}

]

}

```

Most of the parameters in the JSON sample file are self-explanatory. **Cluster\_id** defines the cluster to which the server belongs. The **interface\_name** is the Ethernet interface name on the server used to configure the server and **roles** define the roles that can be configured for the server. The sample **roles** array in the example lists all valid role values. **Tag** defines the list of tag values for grouping and classifying the server.

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

### Delete Servers

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

Table 6 on page 64 lists the optional arguments.

**Table 6: Server Manager Delete Server Command Options**

Option	Description
<code>-h, --help</code>	Show the options available for the current command and exit.

Table 6: Server Manager Delete Server Command Options (*continued*)

Option	Description
<code>--config_file CONFIG_FILE, -c CONFIG_FILE</code>	The name of the server manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>
<code>--server_id SERVER_ID</code>	The server ID for the server or servers to be deleted.
<code>--mac MAC</code>	The MAC address for the server or servers to be deleted.
<code>--ip IP</code>	The IP address for the server or servers to be deleted.
<code>--cluster_id CLUSTER_ID</code>	The cluster ID for the server or servers to be deleted.
<code>--tag TagName=TagValue</code>	The TagName that is to be matched with the Tagvalue. Up to seven TagName and Tagvalue pairs separated by commas can be provided.

The criteria to identify servers to be deleted can be specified by providing the **server\_id** or the server: **mac address**, **ip**, **cluster\_id**, or the **TagName = TagValue**.

Provide one of the server matching criteria to display a list of servers available to be deleted.

### Show Server Configuration

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

Usage:

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

Table 7 on page 65 lists the optional arguments.

Table 7: Server Manager Show Server Command Options

Option	Description
<code>-h, --help</code>	Show the options available for the current command and exit.
<code>--config_file CONFIG_FILE, -c CONFIG_FILE</code>	The name of the server manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>
<code>--server_id SERVER_ID</code>	The server ID for the server or servers to be deleted.
<code>--mac MAC</code>	The MAC address for the server or servers to be displayed.
<code>--ip IP</code>	The IP address for the server or servers to be displayed.

Table 7: Server Manager Show 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 to identify servers to be displayed can be specified by providing the **server\_id** or the server: **mac address**, **ip**, **VNS\_id**, **cluster\_id**, or **TagName=TagValue**.

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

### Server Manager Commands for Managing Clusters

A cluster is used to store parameter values that are common to all servers belonging to that cluster. The commands in this section facilitate managing clusters in the server manager database, enabling the user 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 66](#)
- [Delete a Cluster on page 68](#)
- [Show Cluster Configuration on page 68](#)

#### Create a New Cluster or Update an Existing Cluster

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

Usage:

```
server-manager [-h] [--config_file CONFIG_FILE]
cluster [--file_name FILE_NAME]
```

[Table 8 on page 66](#) lists the optional arguments.

Table 8: Server Manager Add Cluster Command Options

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



Table 8: Server Manager Add Cluster Command Options (*continued*)

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

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

```
{
  "cluster" : [
    {
      "id" : "demo-cluster",
      "parameters" : {
        "router_asn": "<asn number>",
        "database_dir": "/home/cassandra",
        "database_token": "",
        "use_certificates": "False",
        "multi_tenancy": "False",
        "encapsulation_priority": "MPLSoUDP,MPLSoGRE,VXLAN",
        "service_token": "<password>",
        "keystone_username": "admin",
        "keystone_password": "<password>",
        "keystone_tenant": "admin",
        "analytics_data_ttl": "168",
        "haproxy": "disable",
        "subnet_mask": "<ip address>",
        "gateway": "<ip address>",
        "password": "<password>",
        "external_bgp": "",
        "domain": "<domain name>"
      }
    }
  ]
}
```

```

    }
  }
]
}

```

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

### Delete a Cluster

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

Usage:

```

server-manager delete [-h] [--config_file CONFIG_FILE]

cluster [--cluster_id CLUSTER_ID]

```

[Table 9 on page 68](#) lists the optional arguments.

**Table 9: Server Manager Delete Cluster Command Options**

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

### Show Cluster Configuration

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

Usage:

```

server-manager show [-h] [ --config_file CONFIG_FILE]

cluster [--cluster_id CLUSTER_ID] [--detail]

```

[Table 10 on page 68](#) lists the optional arguments.

**Table 10: Server Manager Show Cluster Command Options**

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

Table 10: Server Manager Show Cluster Command Options (*continued*)

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 information about a particular cluster. If the optional parameter is not specified, information about all clusters in the system is returned.

### Server Manager Commands for Managing Tags

Tags are used for grouping servers together so that an operation such as get, reimage, provision, status, and so on can be easily performed on servers that have matching tags. The server manager provides a flexible way for users to define their own tags, then use those tags to assign values to servers. Servers with matching tag values can be easily grouped together. The server manager can store a maximum of seven tag values. At initialization, the server manager reads the tag names from the configuration file. The tag names can be retrieved or modified using CLI commands. When modifying tag names, the server manager ensures that the tag name being modified is not used by any of the server entries.

- [Create a New Tag or Update an Existing Tag on page 69](#)
- [Show Tag Configuration on page 70](#)

#### Create a New Tag or Update an Existing Tag

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

Usage:

```
server-manager add [-h] [--config_file CONFIG_FILE]
tag [--file_name FILE_NAME]
```

Optional arguments include the following:

Option	Description
<code>-h, --help</code>	Show the options available for the current command and exit.
<code>--config_file CONFIG_FILE, -c CONFIG_FILE</code>	The name of the server manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>
<code>--file_name FILE_NAME, -f FILE_NAME</code>	The JSON file that contains the tag names.

If no JSON file is specified, the client program prompts the user for tag names, then builds a JSON file and makes a REST API call to the server manager. The JSON file contains a number of tag entries, in the format shown in the following example:

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

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

### Show Tag Configuration

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

Usage:

```
server-manager show [-h] [ --config_file CONFIG_FILE] tag
```

Optional arguments include the following:

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

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

```
{
  "tag1": "datacenter",
  "tag2": "floor",
  "tag3": "hall",
  "tag4": "rack",
  "tag5": "user_tag"
}
```

### Server Manager Commands for Managing Images

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

The image types currently supported are summarized in the following table:

Image Type	Description
<b>centos</b>	Manages the CentOS stock ISO, and does not include the Contrail packages repository packaged with the ISO.
<b>contrail-centos-package</b>	Maintains a repository of the package to be installed on the CentOS system image.
<b>ubuntu</b>	Manages the base Ubuntu ISO.
<b>contrail-ubuntu-package</b>	Maintains a repository of packages that contain Contrail and dependent packages to be installed on an Ubuntu base system.
<b>ESXi5.1/ESXi5.5</b>	Manages VMware ESXi 5.1 or 5.5 ISO.

- [Creating New Images or Updating Existing Images on page 71](#)
- [Add an Image on page 71](#)
- [Upload an Image on page 73](#)
- [Delete an Image on page 73](#)
- [Show Image Configuration on page 74](#)

#### ***Creating New Images or Updating Existing Images***

The server manager maintains five types of images – CISO, Ubuntu ISO, ESXi hypervisor ISO, Contrail CentOS package, and Contrail Ubuntu package.

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

- Use **add** when the new image is present locally on the server manager machine. The path provided is the image path on the server manager machine.
- Use **upload\_image** when the new image is present on the machine where the client program is being invoked. The path provided is the image path on the client machine.

#### ***Add an Image***

Usage:

```
server-manager add [-h] [ --config_file CONFIG_FILE]
                  image [--file_name FILE_NAME]
```

Optional arguments include the following:

Option	Description
<code>-h, --help</code>	Show the options available for the current command and exit.
<code>--config_file CONFIG_FILE, -c CONFIG_FILE</code>	The name of the server manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>
<code>--file_name FILE_NAME, -f FILE_NAME</code>	The name of the JSON file that contains the image parameter values.

If no JSON file is specified, the client program accepts parameter values from the user interactively, then builds a JSON file and makes a REST API call to the server manager.

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

```
{
  "image": [
    {
      "id": "ubuntu-12.04.3",
      "type": "ubuntu",
      "version": "ubuntu-12.04.3",
      "path": "/iso/ubuntu-12.04.3-server-amd64.iso"
    },
    {
      "id": "centos-6.4",
      "type": "centos",
      "version": "centos-6.4",
      "path": "/iso/CentOS-6.4-x86_64-minimal.iso"
    },
    {
      "id": "contrail-ubuntu-r11-b33",
      "type": "contrail-ubuntu-package",
```

```

        "version": "contrail-ubuntu-r11-b33",

        "path": "/iso/contrail-install-packages_1.11-33_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.

Usage:

```

server-manager upload_image [-h]

--config_file CONFIG_FILE]

image_id image_version image_type file_name

```

Positional arguments include the following:

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

Optional arguments include the following:

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

### Delete an Image

Use the **server-manager delete** command to delete an image from the server manager database. When an image is deleted from server manager database, the corresponding

distribution, profile, or repository that corresponds to the image is also deleted from the Cobbler database.

Usage:

```
server-manager delete [-h] [ --config_file CONFIG_FILE]
image image_id
```

Positional arguments include the following:

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

Optional arguments include the following:

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

### *Show Image Configuration*

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

Usage:

```
server-manager [-h] [--config_file CONFIG_FILE] image [--image_id IMAGE_ID] [--detail]
```

Optional arguments include the following:

Option	Description
<code>-h, --help</code>	Show the options available for the current command and exit.
<code>--config_file CONFIG_FILE, -c CONFIG_FILE</code>	The name of the server manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>
<code>image_id</code>	The image ID for the image or images.
<code>--detail, -d</code>	Flag to indicate if details are requested.

### Server Manager Operational Commands for Managing Servers

The server manager commands in the following sections are operational commands for performing a specific operation on a server or a group of servers. These commands



assume that the base configuration of entities required to execute the operational commands is already completed using configuration CLI commands.

### Reimaging Server(s)

Use the **server-manager reimage** command to reimage an identified server or servers with a provided base ISO and package. Servers are specified by providing match conditions to select servers from the database.

Before issuing the **reimage** command, the images must be added to the server manager using the **create image** command, which also adds the images to Cobbler. The set of servers to be reimaged can be specified by providing match criteria for servers already added to the server manager database, using the **server\_id** or server: **vns\_id**, **cluster\_id**, **pod\_id**, or **rack\_id**.

You must identify the base image ID to be used to reimage, plus any optional Contrail package to be used. When a Contrail package is provided, a local repository is created that can be used for subsequent provisioning of reimaged servers.

The command asks 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.

Usage:

```
server-manager reimage [-h]
                        [ --config_file CONFIG_FILE]
                        [--package_image_id PACKAGE_IMAGE_ID]
                        [--no_reboot]
                        (--server_id SERVER_ID | --cluster_id CLUSTER_ID [--tag <tag_name=tag_value>])
                        [--no_confirm]
                        base_image_id
```

Positional arguments include the following:

Option	Description
<b>base_image_id</b>	The image ID of the base image to be used.

Optional arguments include the following:

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

Option	Description
<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 by the user.

From the configuration of servers in the database, the server manager determines which roles to configure on which servers and uses this information along with other server and VNS parameters from the database to achieve the task of configuring the servers with specific roles.

When the **server-manager provision** command is used, the server manager builds the manifest files corresponding to each of the servers and pushes them to the Puppet agent for execution upon the servers.

Usage:

```
server-manager provision [-h]
```

```
    [--config_file CONFIG_FILE]
    (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag <tag_name=tag_value> |
    --provision_params_file PROVISION_PARAMS_FILE | --interactive)
    [--no_confirm]
    package_image_id
```

Positional arguments include the following:

Option	Description
<code>package_image_id</code>	The Contrail package image ID to be used for provisioning.

Optional arguments include the following:

Option	Description
<code>-h, --help</code>	Show the options available for the current command and exit.
<code>--config_file CONFIG_FILE, -c CONFIG_FILE</code>	The name of the server manager client configuration file. The default file is: <code>/opt/contrail/server_manager/client/sm-client-config.ini</code>
<code>--server_id SERVER_ID</code>	The server ID for the server or servers to be provisioned.
<code>--cluster_id CLUSTER_ID</code>	The cluster ID for the server or servers to be provisioned.
<code>--tag TagName=TagValue</code>	TagName to be matched with Tagvalue.
<code>--provision_params_file PROVISION_PARAMS_FILE, -f PROVISION_PARAMS_FILE</code>	Optional JSON file containing the parameters for provisioning the server(s).
<code>--interactive, -l</code>	Flag indicating that the user will manually enter the server parameters for provisioning.
<code>--no_confirm, -F</code>	Flag to bypass confirmation message, default = do NOT bypass.

You can specify roles different from what is configured in the database by using the JSON file option parameter. When using the file option, the rest of the server parameters, the cluster parameters, and the list of servers must be configured before using the provision command. The following is a sample format for the file option:

```
{
  "roles" : {
    "database" : ["demo2-server"],
    "openstack" : ["demo2-server"],
    "config" : ["demo2-server"],
    "control" : ["demo2-server"],
    "collector" : ["demo2-server"],
    "webui" : ["demo2-server"],
    "compute" : ["demo2-server"]
  }
}
```

The final option for specifying roles for provisioning servers is to specify the **--interactive** option flag. When the provision command is used, the user is prompted to enter role definitions interactively.

### 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.

Usage:

```
server-manager restart [-h]
                        [ --config_file CONFIG_FILE]
                        (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag <tag_name=tag_value>)
                        [--net_boot]
                        [--no_confirm]
```

Optional arguments include the following:

Option	Description
-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: <b>/opt/contrail/server_manager/client/sm-client-config.ini.</b>
--server_id SERVER_ID	The server ID for the server or servers to be restarted.
--cluster_id CLUSTER_ID	The cluster ID for the server or servers to be restarted.
--tag TagName=TagValue	TagName to be matched with Tagvalue.
--net_boot, -n	Optional parameter to indicate if the server should be netbooted.
--no_confirm, -F	Flag to bypass confirmation message, default = do NOT bypass.

### Show Status of Server(s)

Use the **server-manager status** command to view the reimaging or provisioning status of server(s).

Usage:

```
server-manager status server [-h]
                        [--config_file CONFIG_FILE]
                        (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag
                        <tag_name=tag_value>)
```

Optional arguments include the following:

Option	Description
-h, --help	Show the options available for the current command and exit.
--config_file CONFIG_FILE, -c CONFIG_FILE	The name of the server manager client configuration file. The default file is: /opt/contrail/server_manager/client/sm-client-config.ini
--server_id SERVER_ID	The server ID for the server 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 CLI provides a way to fetch the current status of a server.

Status outputs include the following:

restart\_issued  
reimage\_started  
provision\_started  
provision\_completed  
database\_started  
database\_completed  
openstack\_started  
openstack\_completed  
config\_started  
config\_completed  
control\_started  
control\_completed  
collector\_started  
collector\_completed  
webui\_started  
webui\_completed  
compute\_started  
compute\_completed

## Server Manager REST API Calls

This section describes all of the REST API calls to the server manager. Each description includes an example configuration.

- [REST APIs for Server Manager Configuration Database Entries on page 81](#)
- [API: Add a Server on page 81](#)
- [API: Delete Servers on page 82](#)
- [API: Retrieve Server Configuration on page 82](#)
- [API: Add an Image on page 82](#)
- [API: Upload an Image on page 83](#)
- [API: Get Image Information on page 83](#)
- [API: Delete an Image on page 84](#)
- [API: Add or Modify a Cluster on page 84](#)
- [API: Delete a Cluster on page 85](#)
- [API: Get Cluster Configuration on page 85](#)
- [API: Get All Server Manager Configurations on page 85](#)
- [API: Reimage Servers on page 85](#)

- [API: Provision Servers on page 85](#)
- [API: Restart Servers on page 86](#)

### 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 array of servers to be added. For each server in the array, all the parameters are specified as JSON fields. The fields mask, gateway, password, and domain 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 format for adding a server.

```
{
  "server": [
    {
      "id": "demo2-server",
      "mac_address": "<mac address>",
      "ip_address": "<ip address>",
      "parameters": {
        "interface_name": "eth1"
      },
      "roles": [
        "config",
        "openstack",
        "control",
        "compute",
        "collector",
        "webui",
        "database"
      ],
      "cluster_id": "demo-cluster",
      "mask": "<ip address>",
      "gateway": "<ip address>",
      "password": "juniper",
      "domain": "<domain name>",
      "email": "id@company.net",
      "tag": {
        "datacenter": "demo-dc",
```

```
        "rack": "demo-rack"
    },
}
]
```

---

### API: Delete Servers

Use one of the following formats to delete a server.

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

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

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

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

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

Method : DELETE

Payload : None

---

### API: Retrieve Server Configuration

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

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

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

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

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

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

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

Method : GET

Payload : None

---

### API: Add an Image

Use the following to add a new image to the server manager configuration database from the server manager machine.

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

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



Method: **PUT**

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

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

### API: Upload an Image

Use the following to upload a new image from a client to the server manager configuration database.

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

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

Method: **PUT**

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

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

### API: Get Image Information

Use the following to get image information.

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

Method: **GET**

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

### API: Delete an Image

---

Use the following to delete an image.

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

Method: **DELETE**

Payload: Specifies criteria for the image being deleted.

### API: Add or Modify a Cluster

---

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

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

Method: **PUT**

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

```
{
  "cluster" : [
    {
      "id" : "demo-cluster",
      "parameters" : {
        "router_asn" : "<asn number>",
        "database_dir" : "/home/cassandra",
        "database_token" : "",
        "use_certificates" : "False",
        "multi_tenancy" : "False",
        "encapsulation_priority" : "MPLSoUDP,MPLSoGRE,VXLAN",
        "service_token" : "<password>",
        "keystone_user" : "admin",
        "keystone_password" : "<password>",
        "keystone_tenant" : "admin",
        "analytics_data_ttl" : "168",
        "subnet_mask" : "<ip address>",
        "gateway" : "<ip address>",
        "password" : "juniper",
        "haproxy" : "disable",
        "external_bgp" : "",
        "domain" : "<domain name>"
      }
    }
  ]
}
```

### API: Delete a Cluster

---

Use this API to delete a cluster from the server manager database.

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

Method: **DELETE**

Payload: None

### API: Get Cluster Configuration

---

Use this API to get a cluster configuration.

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

Method: **GET**

Payload: None

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

### API: Get All Server Manager Configurations

---

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

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

Method: **GET**

Payload: None

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

### API: Reimage Servers

---

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

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

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

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

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

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

Method: **POST**

Payload: None

### API: Provision Servers

---

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

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

Method: **POST**

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

```
{
  server_id : <server_id> OR
  mac : <server_mac_address> OR
  cluster_id : <cluster_id> OR
  tag : {"data-center" : "dc1"} OR
  provision_parameters = {
    "roles" : {
      "database" : ["demo2-server"],
      "openstack" : ["demo2-server"],
      "config" : ["demo2-server"],
      "control" : ["demo2-server"],
      "collector" : ["demo2-server"],
      "webui" : ["demo2-server"],
      "compute" : ["demo2-server"]
    }
  }
}
```

---

### API: Restart Servers

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

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

If servers are only being reprovisioned, the **net-boot** flag is not needed, however, the Puppet agent must be running on the target systems with the correct puppet configuration to talk 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 = <ip address> (Server Manager IP address)**

**listen\_port = 9001**

The steps to be followed include:

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

1. Configure a cluster.

**server-manager add cluster -f cluster.json**

Where cluster.json contains :

```
{
  "cluster" : [
    {
      "id" : "demo-cluster",
      "parameters" : {
        "router_asn": "<asn number>",
        "database_dir": "/home/cassandra",
        "database_token": "",
        "use_certificates": "False",
        "multi_tenancy": "False",
        "encapsulation_priority": "MPLSoUDP,MPLSoGRE,VXLAN",
        "service_token": "<password>",
        "keystone_user": "admin",
        "keystone_password": "<password>",
        "keystone_tenant": "admin",
        "analytics_data_ttl": "168",
        "subnet_mask": "<ip address>",
        "gateway": "<ip address>",
        "password": "<password>",
        "haproxy": "disable",
        "external_bgp": ""
      }
    }
  ]
}
```

```

        "domain": "demo.company.net"
    }
}
]
}

```

## 2. Configure the server.

**server-manager add server -f server.json**

Where server.json contains :

```

{
  "server": [
    {
      "id": "demo2-server",
      "mac_address": "<mac address>",
      "ip_address": "10.84.51.13",
      "parameters": {
        "interface_name": "eth1",
      },
      "roles": ["config", "openstack", "control", "compute", "collector", "webui", "database"],
      "cluster_id": "demo-cluster",
      "subnet_mask": "<ip address>",
      "gateway": "<ip address>",
      "password": "<password>",
      "domain": "demo.company.net",
      "ipmi_address": "<ip address>"
    }
  ]
}

```

## 3. Configure images.

In the example, the image files for ubuntu 12.04.3 and contrail-ubuntu-164 are located at the corresponding image path specified on the server manager.

**server-manager add -c smgr\_client\_config.ini image -f image.json**

Where image.json contains:

```

{
  "image": [
    {
      "id": "ubuntu-12.04.3",
      "type": "ubuntu",
      "version": "ubuntu-12.04.3",
      "path": "/iso/ubuntu-12.04.3-server-amd64.iso"
    },
    {
      "id": "contrail-ubuntu-164",
      "type": "contrail-ubuntu-package",
      "version": "contrail-ubuntu-164",
      "path": "/iso/contrail-install-packages_1.05-164~havana_all.deb"
    }
  ]
}

```

#### 4. Reimage servers.

This step can be performed after the configuration in the previous steps is in the server manager database.

```
server-manager reimage --server_id demo2-server -r contrail-ubuntu-164 ubuntu-12.04.3
```

#### 5. Provision servers.

```
server-manager provision --server_id demo2-server contrail-ubuntu-164
```



**NOTE:** Optionally, the Contrail package to be used can be specified with the reimage command, in which case the repository with the Contrail packages is created and made available to the target nodes as part of the reimage process. It is a mandatory parameter of the provision command.

---

---

## Installing Server Manager

- [Overview: Installation Requirements for Server Manager on page 89](#)
- [Installing Server Manager on page 90](#)
- [Upgrading Server Manager Software on page 91](#)
- [Server Manager Installation Completion Checks on page 92](#)
- [Sample Configurations for Server Manager Templates on page 93](#)

### Overview: Installation Requirements for Server Manager

This document provides details for installing Server Manager, starting with Contrail Release 2.10.

---

#### Platform Support

Server Manager can be installed on the following platform operating systems:

- Ubuntu 12.04.3
- Ubuntu 14.04
- Ubuntu 14.04.1

Server Manager can be used to reimage and provision the following target platform operating systems:

- Ubuntu 12.04.3
- Ubuntu 14.04
- Ubuntu 14.04.1
- VMware ESXi 5.5

## Installation Prerequisites

Before installing Server Manager ensure the following prerequisites have been met.

- Internet access is required 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 these versions of dependent packages: Puppet 3.7.3-1 and Cobbler 2.6.3. As part of the installation, these versions will be installed.

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

## Installing Server Manager

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

Ubuntu: **contrail-server-manager-installer\_<version~sku>.deb**

The user 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 the Server Manager 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. You 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
```

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. The user must finish the provisioning by modifying the following templates. Refer to the sample configuration section included in this topic for details on configuring these files.

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

---

### Starting the Server Manager Service

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

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

## Upgrading Server Manager Software

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

---

### Prerequisite to Upgrading

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

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

### Use Steps for New Installation

---

Once the existing server manager installer package has been removed, use the installation steps for a new installation, see details in previous section:

1. **`dpkg -i <contrail-server-manager-installer*.deb>`**
2. **`cd /opt/contrail/contrail-server-manager`**
3. **`./setup.sh -all`**

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.

- The following services should all be running.

**`service contrail-server-manager status`**

**`service cobblerd status`**

**`service bind9 status`**

**`service isc-dhcp-server status`**

- Also check:

**`ps auwx | grep Passenger`**

### Server Manager Client Checks

---

- Check

**`which server-manager`**

- Check the client configuration at

**`/opt/contrail/server-manager/client/sm-client-config.ini`**

- Make sure `listen_ip_addr` is configured with the server manager IP address.

### Server Manager Webui Checks

---

- Use the following to check the status of the server manager webui.

**`service supervisor-webui status`**

- Check the webui access from the browser `http:<server manager> :8080`.

## Sample Configurations for Server Manager Templates

The following are sample parameters for the server manager templates. Use settings specific for your environment. Typically you will pay attention to parameters for DHCP, bind, and email services.

### Sample Settings

```
bind_master: <ip address>

manage_forward_zones: ['contrail.company.net']

manage_reverse_zones: ['<ip address>']

next_server: <ip address>

server: <ip address>
```

### dhcp.template

Add server manager hooks into the dhcp template, then when dhcp commit, release, or expire actions occur, the server manager will be notified. The DHCP servers are detected at the server manager and kept with status as 'Discovered'.

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

### named.conf.options

Be sure to configure

```
forwarders {
    x.x.x.x;
};

allow-query { any; };

recursion yes;
```

### named.template

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

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

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

### sendmail.cf

The `sendmail.cf` template is present with a juniper.net configuration. Populate with configuration specific to your environment. The Server manager uses the template to generate emails when reimage or provision is completed.

#### Related Documentation

- [Using Server Manager to Automate Provisioning on page 55](#)
- [Using the Server Manager Web User Interface on page 94](#)

## 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 94](#)
- [Create a Cluster for Server Manager on page 94](#)
- [Working with Servers in the Server Manager User Interface on page 101](#)
- [Add a Server on page 101](#)
- [Edit Tags for Servers on page 103](#)
- [Using the Edit Config Option for Multiple Servers on page 103](#)
- [Filter Servers by Tag on page 104](#)
- [Viewing Server Details on page 104](#)
- [Configuring Images and Packages on page 104](#)
- [Add New Image or Package on page 105](#)
- [Selecting Server Manager Actions for Clusters on page 105](#)
- [Reimage a Cluster on page 105](#)
- [Provision a Cluster on page 106](#)

### Log In to Server Manager

The server manager user interface can be accessed using:

**http://<server-manager-user-interface-ip>:8080**

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.

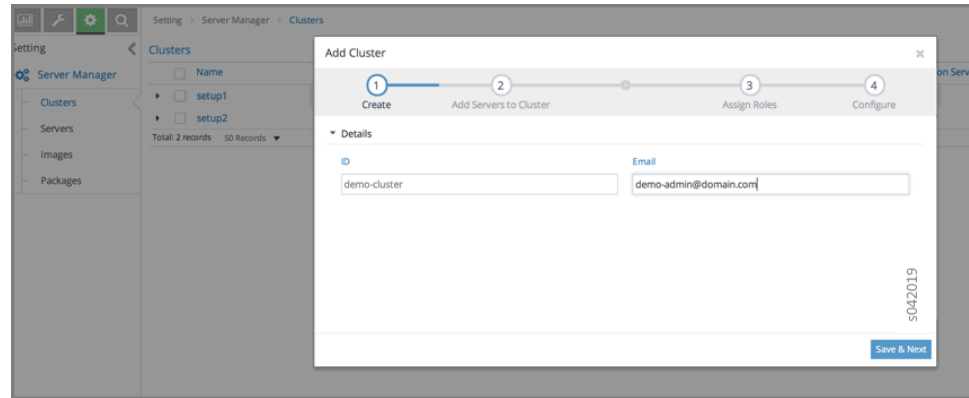
### Create a Cluster for Server Manager

Select **Add Cluster** to identify a cluster to be managed by the server manager. From **Setting > Server Manager > Clusters**, to access the **Clusters** page, see the following:

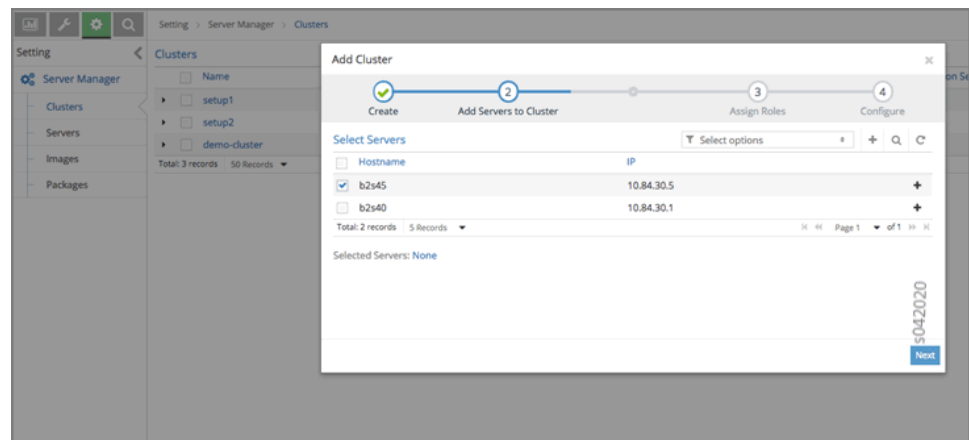
Name	Email	New Servers	Configured Servers	In-Reimage Servers	Reimage Servers	In-Provision Servers	Provisioned Servers	Total Servers
setup1	cluster-admin@domain.com	0	0	0	0	0	1	1
setup2	cluster-admin@domain.com	0	0	0	0	0	1	1

Total: 2 records | 50 Records Page 1 of 1

To create a new cluster, click the plus icon in the upper right of the **Clusters** page. The **Add Cluster** window appears, where you can add a new cluster ID and the domain email address of the cluster.

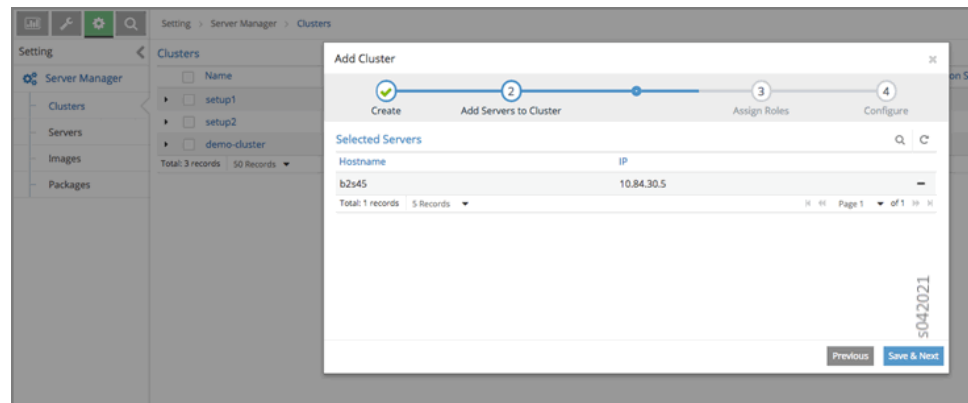


When you are finished adding information about the new cluster in the **Add Clusters** page, click **Save & Next**. Now you can add servers to the cluster, as shown in the following image.

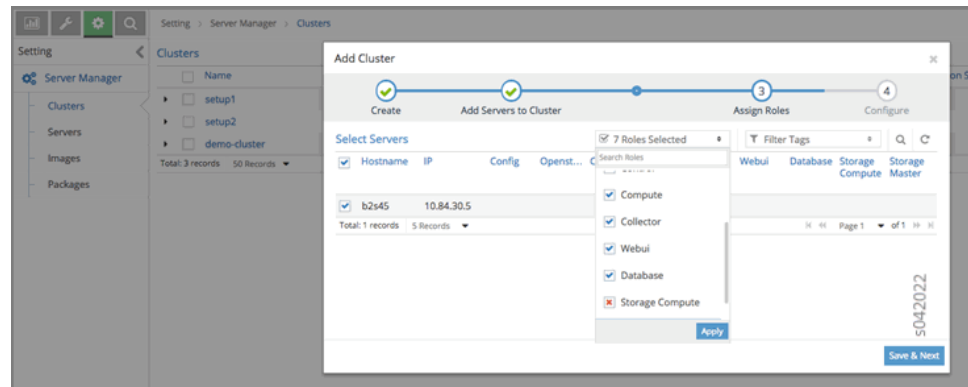


Click the check box of each server to be added to the cluster.

When finished, click the **Next** button, and the selected servers are added to the cluster.

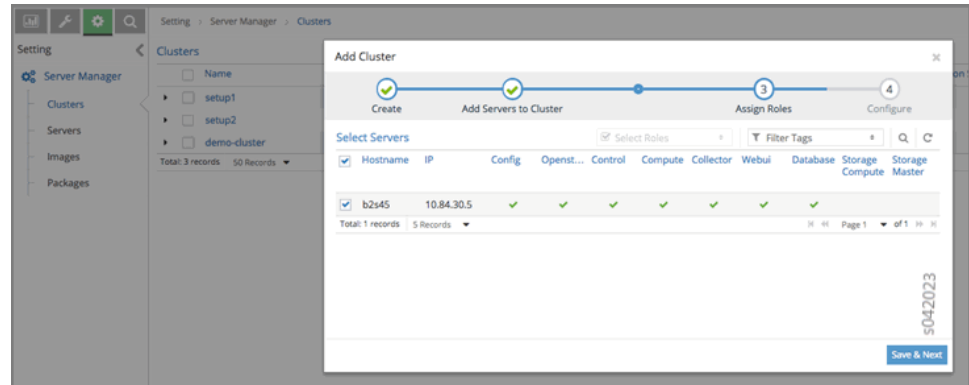


When finished adding servers, click the **Save & Next** button. Now you can assign roles to servers that you select in the cluster. Roles available are the Contrail roles of Config, Openstack, Control, Compute, and Collector. Click the check box for each role assignment for the selected server. You can also click a check box to remove the check mark for any assigned role. The assigned roles correspond to the role functions in operation on the server.

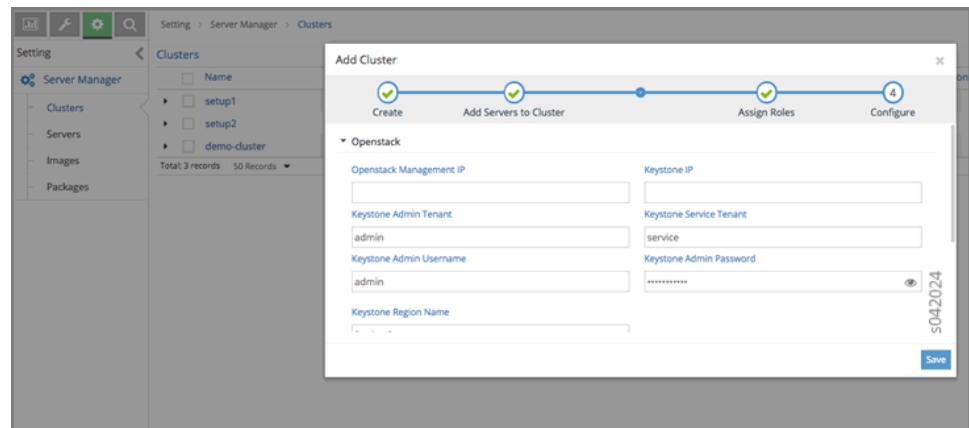


When finished selecting roles for the selected server in the **Roles** window, click the **Apply** button to save your choices.

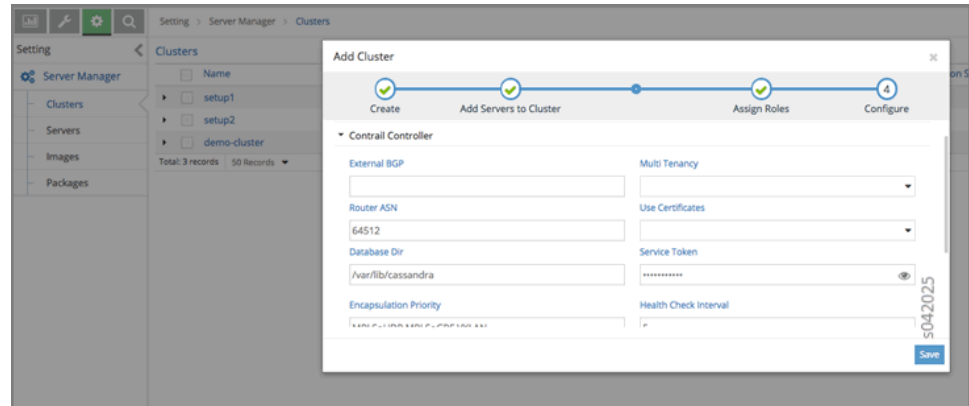
Click the **Save & Next** button to view your selections, which display as check marks in the columns on the **Add Cluster** window, see the following image.



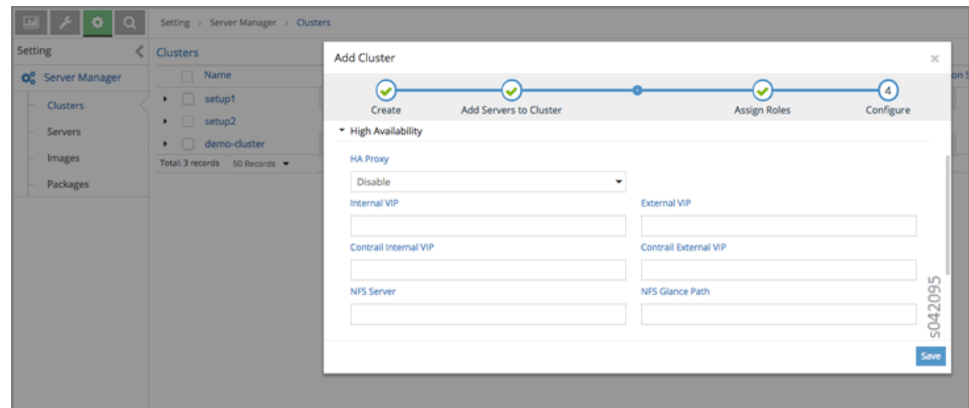
The next step after roles are assigned is to enter the cluster configuration information for OpenStack. After viewing the assigned roles, click the **Save & Next** button. A window opens where you can click an icon that opens a set of fields where you can enter OpenStack or Contrail configuration information for the cluster. In the following image, the Openstack icon is open, with fields where you can enter configuration information, such as Openstack or Keystone passwords and usernames.



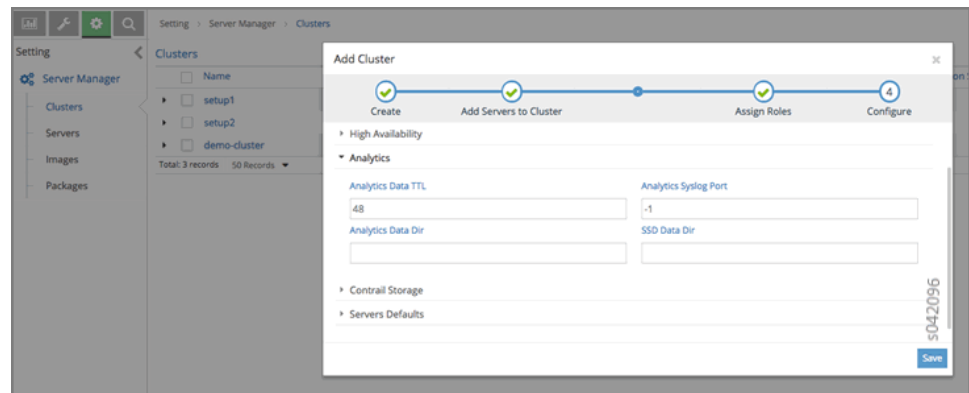
In the following image, the Contrail controller icon is open, where the user can enter configuration information for Contrail, such as **External BGP**, **Multi Tenancy**, **Router ASN**, **HA Proxy**, and so on. See the following image.



In following image, the High Availability (HA) icon is open, where the user can configure high availability parameters such as **HA Proxy**, **Internal** and **External VIP**, and so on.

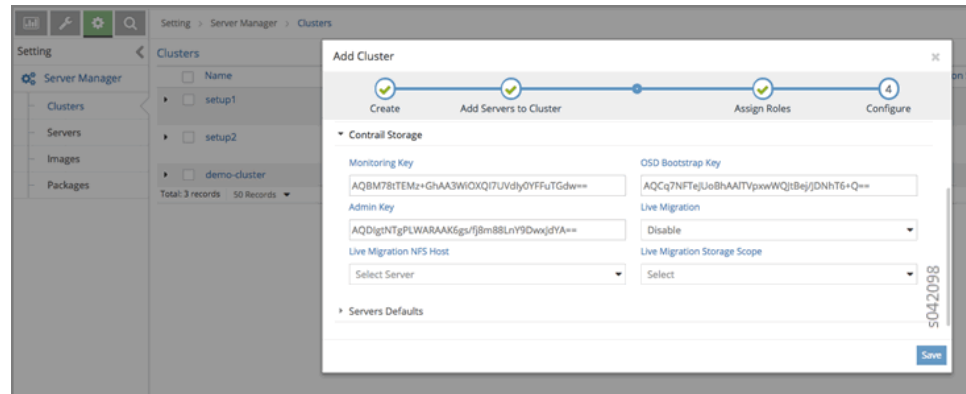


In following image, the **Analytics** icon is open, where the user can configure parameters for Contrail Analytics, including **TTL**, **Syslog Port**, **Data Dir**, and so on.





In following image, the **Contrail Storage** icon is open, where the user can configure parameters for Contrail Storage, including **Monitoring Key**, **OSD Bootstrap Key**, **Admin Key**, and so on.



When finished entering all of the cluster configuration information, click **Save** to submit the configurations. You can view all configured clusters at the **Clusters** screen (**Setting > Server Manager > Clusters**).

Name	Email	New Servers	Configured Servers	In-Reimage Servers	Reimage Servers	In-Provision Servers	Provision Servers
setup1	cluster-admin@domain.com	0	0	0	0	0	0
setup2	cluster-admin@domain.com	0	0	0	0	0	1
demo-cluster	demo-admin@domain.com	0	0	0	0	0	1

Total: 3 records 50 Records

To perform an action on one of the configured clusters, click the cog icon at the right to pick from a list of actions available for that cluster, including **Add Servers**, **Remove Servers**, **Assign Roles**, **Edit Config**, **Reimage**, **Provision**, and **Delete**, as shown in the following.

Name	Email	New Servers	Servers	Provisioned Servers	Total Servers
setup1	cluster-admin@domain.com	0	0	0	0
setup2	cluster-admin@domain.com	0	1	1	1
demo-cluster	demo-admin@domain.com	0	1	1	1

Total: 3 records 50 Records

- + Add Servers
- Remove Servers
- Assign Roles
- Edit Config
- Reimage
- Provision
- Delete

You can also click the expansion icon on the left side of the cluster name to display the details of that cluster in an area below the name line, as in the following.

The screenshot shows the 'Clusters' page in the Server Manager. A table lists three clusters: 'setup1', 'setup2', and 'demo-cluster'. The 'demo-cluster' is selected, and its details are displayed below. The details are organized into sections: Details, Openstack, Contrail Controller, High Availability, Status, Analytics, Contrail Storage, and Servers Defaults.

Name	Email	New Servers	Configured Servers	In-Reimage Servers	Reimage Servers	In-Provision Servers	Provisioned Servers
setup1	cluster-admin@domain.com	0	0	0	0	0	0
setup2	cluster-admin@domain.com	0	0	0	0	0	1
demo-cluster	demo-admin@domain.com	0	0	0	0	0	1

**demo-cluster Details:**

- ID:** demo-cluster
- Email:** demo-admin@domain.com
- Openstack:**
  - Openstack Management IP: -
  - Keystone Admin Tenant: admin
  - Keystone Service Tenant: service
  - Keystone Admin Username: admin
  - Keystone Region Name: RegionOne
- Contrail Controller:**
  - Encapsulation Priority: MPLSoUDP,MPLSoGRE,VXLAN
  - Router ASN: 64512
  - Database Dir: /var/lib/cassandra
  - Health Check Interval: 5
- High Availability:**
  - HA Proxy: disable
- Status:**
  - Total Servers: 1
  - New Servers: 0
  - Configured Servers: 0
  - In-Reimage Servers: 0
  - Reimage Servers: 0
  - In-Provision Servers: 0
  - Provisioned Servers: 1
- Analytics:**
  - Analytics Data TTL: 48
  - Analytics Syslog Port: -1
- Contrail Storage:**
  - Storage Virsh UUID: c542fac0-0c28-4ed8-82a8-abe476c7ba2d
  - Storage FSID: 266b08f2-7746-42b4-8ba8-b56b3977a54e
- Servers Defaults:**
  - Domain: englab.juniper.net
  - Subnet Mask: 255.255.255.0

Click the upper right icon to switch to the JSON view to see the contents of the JSON file for the cluster.

The screenshot shows the 'Clusters' page in the Server Manager. The 'demo-cluster' is selected, and its details are displayed in JSON format. The JSON content is as follows:

```
{
  "parameters": {
    "domain": "englab.juniper.net",
    "keystone_ip": "10.10.10.10",
    "analytics_data_dir": "/var/lib/cassandra",
    "keystone_region_name": "RegionOne",
    "encapsulation_priority": "MPLSoUDP,MPLSoGRE,VXLAN",
    "keystone_username": "admin",
    "analytics_data_ttl": 48,
    "subnet_mask": "255.255.255.0",
    "nfs_glance_path": null,
    "admin_key": null,
    "keystone_password": "contrail123",
    "router_asn": 64512,
    "database_dir": "/var/lib/cassandra",
    "analytics_syslog_port": -1,
    "keystone_tenant": "admin",
    "gateway": null,
    "database_token": null,
    "haproxy": "disable",
    "storage_virsh_uuid": "c542fac0-0c28-4ed8-82a8-abe476c7ba2d",
    "password": "c@ntrai123",
    "uuid": "4fa141b2-1eed-4b69-b2a3-e8293492da2f",
    "service_token": "contrail123",
    "external_bgp": null,
    "storage_fsid": "266b08f2-7746-42b4-8ba8-b56b3977a54e",
    "internal_vip": null,
    "ssd_data_dir": null,
    "hc_interval": 5,
    "osd_bootstrap_key": null,
    "openstack_passwd": "c@ntrai123",
    "multi_tenancy": false,
    "storage_mon_secret": null,
    "use_certificates": false,
    "openstack_mgmt_ip": null,
    "keystone_service_tenant": "service",
    "nfs_server": null,
    "external_vip": null
  }
}
```

The cluster name is a link, click the cluster name to go to the cluster **Details** page, as in the following.

The screenshot shows the 'demo-cluster' details page in the OpenStack Server Manager interface. The left sidebar shows the navigation menu with 'Clusters' selected. The main content area is divided into several sections:

- Details:** ID: demo-cluster, Email: demo-admin@domain.com
- Openstack:** Openstack Management IP, Keystone Admin Tenant: admin, Keystone Service Tenant: service, Keystone Admin Username: admin, Keystone Region Name: RegionOne
- Control Controller:** Encapsulation Priority: MPLSoUDP,MPLSoGRE,VXLAN, Router ASN: 64512, Database Dir: /var/lib/cassandra, Health Check Interval: 5
- High Availability:** HA Proxy: disable
- Status:** Total Servers: 1, New Servers: 0, Configured Servers: 0, In-Reimage Servers: 0, Reimage Servers: 0, In-Provision Servers: 0, Provisioned Servers: 1
- Analytics:** Analytics Data TTL: 48, Analytics Syslog Port: -1
- Control Storage:** Storage Virsh UUID: c542fac0-0c28-4ed8-82a8-abe476c7ba2d, Storage FSID: 266b0872-7746-42b4-8ba8-b56b3977a54e
- Servers Defaults:** Domain: englab.juniper.net, Subnet Mask: 255.255.255.0

At the bottom, there is a table of servers:

ID	Tags	IP	IPMI	Config	Openstack	Control	Compute	Collector	Webui	Database	Storage	Compute
b2s45	control:lab rack:b2 floor:6	10.84.30.5	10.84.60.148	✓	✓	✓	✓	✓	✓	✓	✓	✓

Total: 1 records 50 Records

## Working with Servers in the Server Manager User Interface

Click on the **Servers** link in the left sidebar at **Setting > Server Manager** to view a list of all servers.

The screenshot shows the 'Servers' page in the OpenStack Server Manager interface. The left sidebar shows the navigation menu with 'Servers' selected. The main content area displays a table of servers:

ID	Cluster	Tags	IP	IPMI	Status
b2s45	demo-cluster	control:lab rack:b2 floor:6	10.84.30.5	10.84.60.148	provision_completed
b2s40	setup2	control:lab rack:b2 floor:6	10.84.30.1	10.84.60.146	provision_completed

Total: 2 records 50 Records

## Add a Server

To add a new server, from **Setting > Server Manager > Servers**, click the plus (+) icon at the upper right side in the header line. The **Add Server** window pops up, as in the following.

The screenshot shows the 'Servers' page with the 'Add Server' dialog box open. The dialog box has two main sections: 'System Management' and 'Interfaces'.

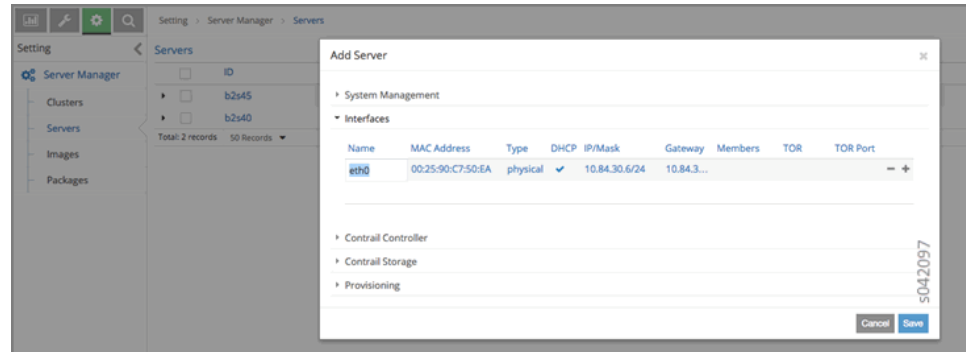
**System Management:**

- 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)

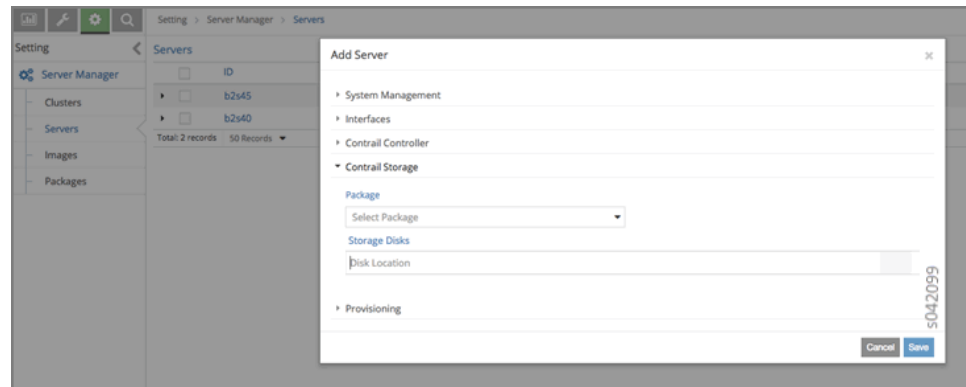
**Interfaces:** (empty section)

Buttons: Cancel, Save

In following image, the **Interfaces** icon is open, where the user can add new or edit existing interfaces. To enable editing for any field, hover the cursor on any selected field to open it.



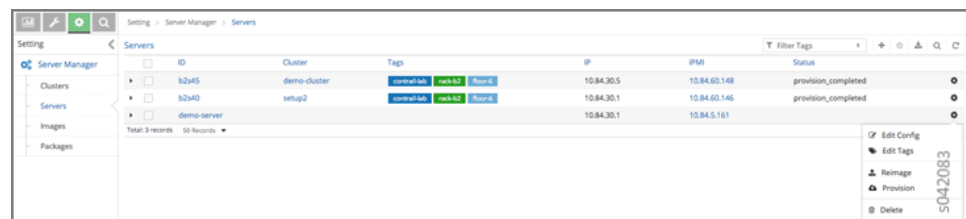
In following image, the **Contrail Storage** icon is open, where the user can configure parameters for Contrail Storage, including selecting a package and adding storage disks locations.



When finished entering new server details at the **Add Server** window, click **Save** to add the new server configuration to the list of servers at **Setting > Server Manager > Servers**.

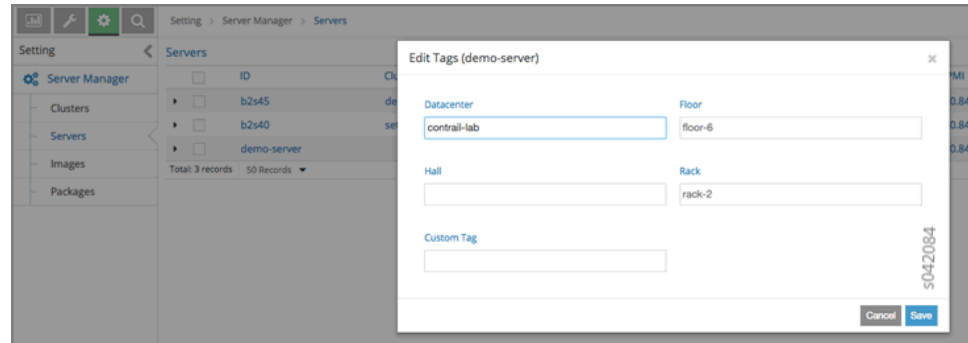
You can change details of the new server by clicking the cog icon to the right side to get a list of actions available, including **Edit Config**, **Edit Tags**, **Reimage**, **Provision**, and **Delete**.

In the following, the user is preparing to edit tags for the new server.



## Edit Tags for Servers

The following shows the **Edit Tags** window that pops up when you select **Edit Tags** from the list at the cog icon for a server. Enter any user-defined tags that are to be associated with the selected server, then click **Save** to add the tags to the server configuration.

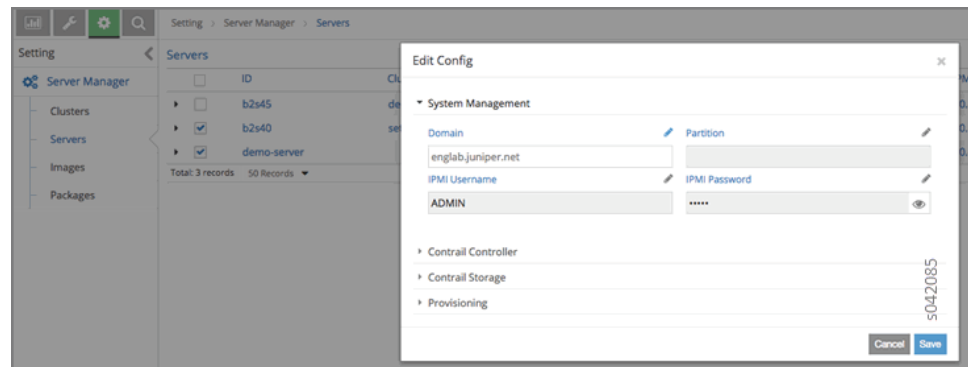


## Using the Edit Config Option for Multiple Servers

You can also edit the configuration of multiple servers at one time. From the **Servers** page at **Setting > Server Manager > Servers**, click the check box of each of the servers you will edit, then click a cog icon at the right to open the action list, and select **Edit Config**.

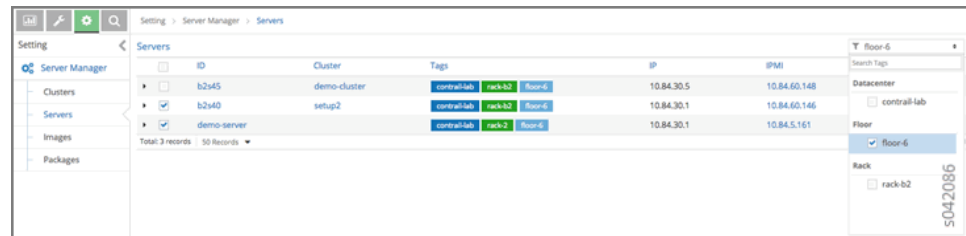
The **Edit Config** window pops up, as in the following.

Click a pencil icon to open fields of configuration areas that can be edited, including **System Management**, **Contrail Controller**, **Contrail Storage**, and so on.



## Filter Servers by Tag

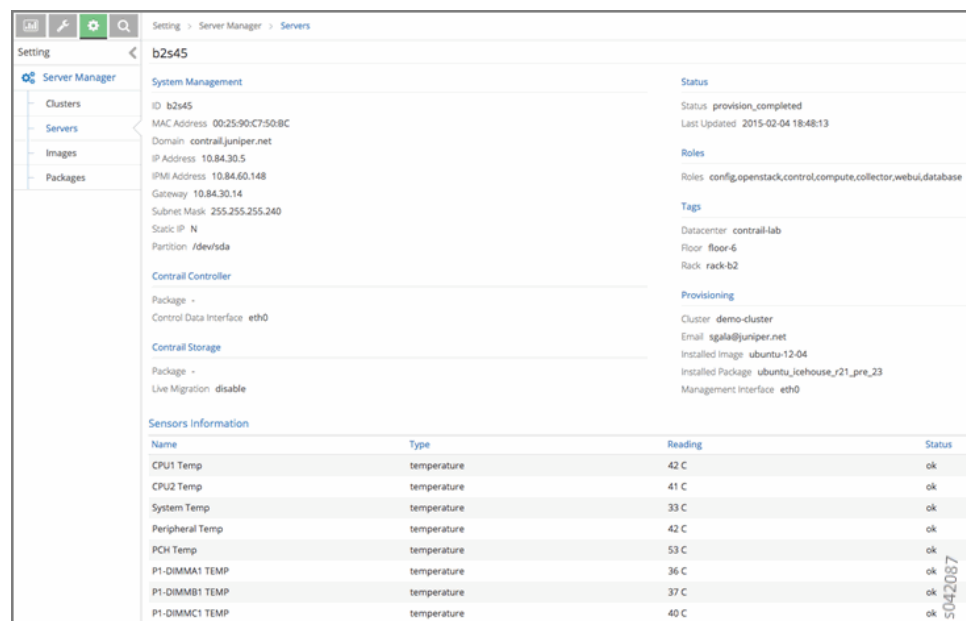
You can filter servers according to the tags defined for them. At the **Servers** page, click the **Filter Tags** field at the upper right heading, and a list of configured tags appears. Click the check box to select a tag by which to filter the list of servers.



## Viewing Server Details

Each server name on the **Servers** page is a link to the details page for that server. Click any server name to open the details for that server, as in the following image.

For each server, the **Sensors Information** area shows the **Name**, **Type**, **Reading**, and **Status** for the following sensor types: **temperature** (degrees Celsius), **fan** (rpm), and **power** (watts).




## Configuring Images and Packages

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

Both areas of the server manager user interface operate in similar fashion. Examples given here are for the **Images** section, and the **Packages** section has similar options.

Click on **Images** in the sidebar to open the list of images configured on the Images page, as in the following.



Name	Category	Type	Version	Path
ubuntu-12.04.3	image	ubuntu	12.04.3	/root/ubuntu-12.04.3-server-amd64.iso
centos-6.4	image	centos	6.4	/root/CentOS-6.4-x86_64-minimal.iso

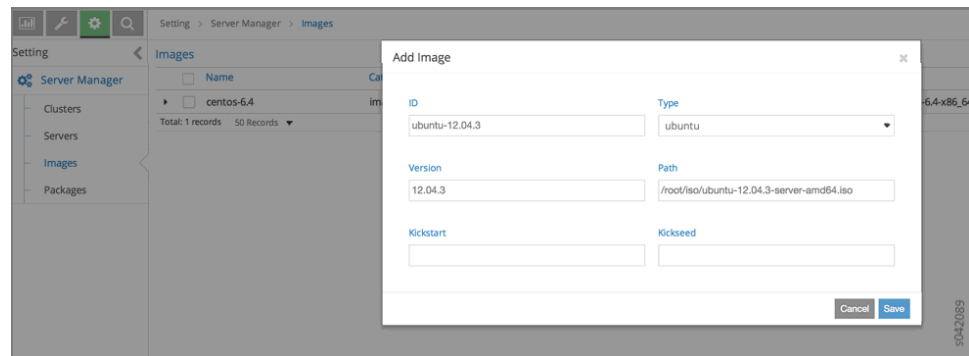
Total: 2 records 50 Records

## 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 to open the **Add** dialog box. The **Add Image** dialog box is shown in the following. Enter the information for the new image (or package) and click **Save** to add the new item to the list of configured items.



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



ID	Type	Version	Path	Kickstart	Kickseed
ubuntu-12.04.3	ubuntu	12.04.3	/root/iso/ubuntu-12.04.3-server-amd64.iso		

Cancel Save

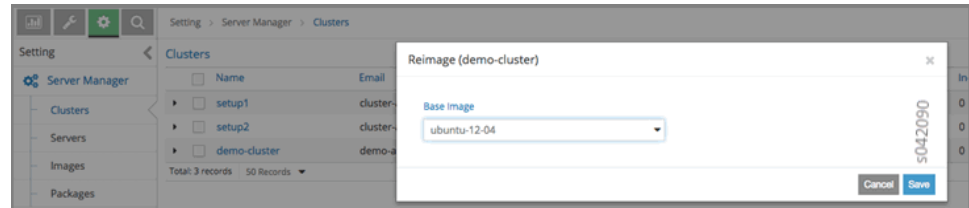
## Selecting Server Manager Actions for Clusters

Once 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

From the **Clusters** page (click **Clusters** in the sidebar or navigate to **Setting > Servers > Clusters**), click the right side cog icon of the cluster to be reimaged, then select **Reimage** from the action list.

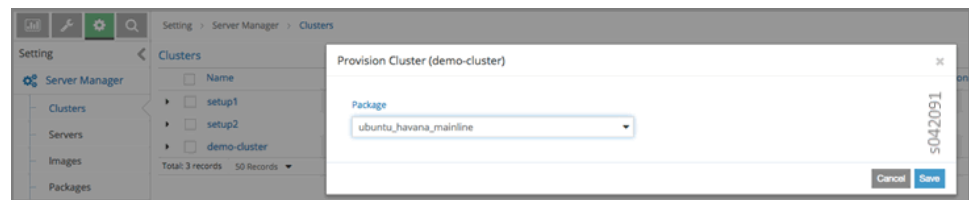
The **Reimage** dialog box pops up, as in the following. Ensure that the correct image for reimagining is selected in the **Default Image** field, then click **Save** to initiate the reimage action.



## Provision a Cluster

The process to provision a cluster is similar to the process to reimage a cluster. From the **Clusters** page (click **Clusters** in the sidebar or navigate to **Setting > Servers > Clusters**), click the right side cog icon of the cluster to be provisioned, then select **Provision** from the action list.

The **Provision Cluster** dialog box pops up, as in the following. Ensure that the correct package for provisioning is selected in the **Default Package** field, then click **Save** to initiate the provision action.



## Installing Contrail with Red Hat OpenStack

- [Overview: Contrail with Red Hat OpenStack on page 106](#)
- [Procedure for Installing RHOSP5 on page 107](#)
- [Install and Configure Contrail on page 108](#)
- [Download and Install Contrail-Install-Packages to the First Config Node on page 108](#)
- [Update the Testbed.py on page 109](#)
- [Complete the Installation on Remaining Nodes on page 111](#)
- [Appendix: Installing with RDO on page 112](#)
- [Install All-in-One OpenStack on page 112](#)

### Overview: Contrail with Red Hat OpenStack

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



## Procedure for Installing RHOSP5

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

1. Install OpenStack.

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

*DEPLOYING OPENSTACK: LEARNING ENVIRONMENTS (MANUAL SETUP)*



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

2. Update the OpenStack password in the `testbed.py`. Copy the OpenStack Keystone password to the `testbed.py`. Refer to the Testbed section in the following.
3. Stop the **nova-compute** and Neutron services in the OpenStack node:

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

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

```
# openstack-config --set /etc/nova/nova.conf DEFAULT network_api_class
nova.network.neutronv2.api.API
# openstack-config --set /etc/nova/nova.conf DEFAULT neutron_url
http://<FIRST_CFGM_IP>:9696
(FIRST_CFGM_IP is the IP address of the first node in the CFGM role defined in the
testbed.py.)
# openstack-config --set /etc/nova/nova.conf DEFAULT neutron_admin_auth_url
http://<KEYSTONE_IP_ADDRESS>:35357/v2.0
# openstack-config --set /etc/nova/nova.conf DEFAULT compute_driver
nova.virt.libvirt.LibvirtDriver
```

5. Restart the Nova services:

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

6. (optional) Configure the **novncproxy\_port**.

Contrail uses port 5999 for the **novncproxy\_port**. If the same port is preferred for any openstack node, update the **novncproxy\_port** as in the following:

```
# openstack-config --set /etc/nova/nova.conf DEFAULT novncproxy_port 5999
# service openstack-nova-novncproxy restart
```

## Install and Configure Contrail

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

### *Repositories for Third Party Packages*

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

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

Ensure that the **contrail\_install\_repo** has the highest priority.

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

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

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

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

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

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

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

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

To change priority of repos using **yum-config-manager**:

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

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

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

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

1. Copy the **contrail-install-packages** to the **/root/** of the first config node. The first config node is the first **cfgm** node defined in the section **env.roledefs['cfgm']** roles in the **testbed.py**.
2. Install the **contrail-install-packages**:

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

3. Set up **contrail\_install\_repo** and install **fabric-utils**:

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

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

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

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

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

## Update the Testbed.py

The OpenStack node is not provisioned by using the Contrail **fabric-utils**, it is the **testbed.py** that carries node information.

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

1. Update the Openstack admin password in the **testbed.py**:

```
#Openstack admin password
```

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

```
and
```

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

2. Update the **keystone\_ip** in the **testbed.py**. The Keystone IP address is the same as the OpenStack IP address.

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

3. Update the **admin\_token** in the **testbed.py**.

The **admin\_token** is available in the **/etc/keystone/keystone.conf** of the OpenStack node.

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

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

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

5. Update the **service\_token** in the **testbed.py**.

Copy the **admin\_token** from the **/etc/keystone/keystone.conf** of the OpenStack node, and enter it as the value for the **service\_token**, as in the following:

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

6. Update the **amqp\_host** in the **testbed.py**.

Enter the value for the **amqp\_host** to be the same as the IP address of the OpenStack node, as in the following:

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

7. Precheck: Issue a precheck to make sure all nodes are reachable and properly updated in the **testbed.py**.

One easy way to precheck is to issue the following command and see if it passes.

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

## Complete the Installation on Remaining Nodes

Use the following procedure to complete the installation.

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

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

2. Disable iptables. It is currently required to permanently disable iptables. This is a known issue regarding Contrail install and setup and deletion is the current resolution.

Iptables can be disabled by issuing the following fab commands. The **fab all\_command**, as used in the following, executes the given command in all nodes configured in the **testbed.py**.

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

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

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

3. Install Contrail without OpenStack.

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

```
# fab install_without_openstack
```

4. Set up Contrail.

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

```
# fab setup_without_openstack
```

5. Verify the setup.

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

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

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



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

## Appendix: Installing with RDO

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

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

### *Prerequisites*

The following are the prerequisite steps.

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

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

3. Install Packstack.

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

## Install All-in-One OpenStack

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

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

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



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

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

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

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

## Installing and Using Contrail Storage

---

- [Overview of the Contrail Storage Solution on page 113](#)
- [Basic Storage Functionality with Contrail on page 114](#)
- [Ceph Block and Object Storage Functionality on page 114](#)
- [Using the Contrail Storage User Interface on page 114](#)
- [Hardware Specifications on page 115](#)
- [Software Files for Compute Storage Nodes on page 116](#)
- [Contrail OpenStack Nova Modifications on page 116](#)
- [Installing the Contrail Storage Solution on page 116](#)
- [Using Fabric Commands to Install and Configure Storage on page 117](#)
- [Fabric Installation Procedure on page 117](#)
- [Using Server Manager to Install and Configure Storage on page 119](#)
- [Server Manager Installation Procedure for Storage on page 120](#)
- [Example Configurations for Storage for Reimaging and Provisioning a Server on page 121](#)
- [Storage Installation Limits on page 126](#)

### Overview of the Contrail Storage Solution

Starting with Contrail Release 2.00, 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 solution provides a validated Network File System (NFS) storage service, however, it is not the Ceph FS distributed file system.

The Contrail storage solution has the following features:

- Provides storage class features to Contrail clusters, including replication, reliability, and robustness.
- Uses open source components.
- Uses Ceph block and object storage functionality.
- Integrates with OpenStack Cinder functionality.
- Does not require virtual machines (VMs) to configure mirrors for replication.
- Allows nodes to provide both compute and storage services.
- Provides easy installation of basic storage functionality based on Contrail roles.
- Provides services necessary to perform virtual machine migrations between compute nodes, and supports both migratable and non-migratable virtual machines.
- Provides a Contrail-integrated user interface from which the user can monitor Ceph components and drill down for more information about components.

## Basic Storage Functionality with Contrail

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

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

## Ceph Block and Object Storage Functionality

Installing the Contrail storage solution creates the following Ceph configurations.

- Each disk is configured as a standalone storage device, enhancing optimal performance and creating proper failure boundaries. Ceph allocates and assigns a process called object storage daemon (OSD) to each disk.
- A replication factor of 2 is configured, consisting of one original instance plus one replica copy. Ceph ensures that each replica is on a different storage node.
- A Ceph monitor process (mon) is configured on each storage node.
- The correct number of placement groups are automatically configured, based on the number of disk drives in the cluster.
- Properly identified SSD drives are set up for use as Ceph OSD journals to reduce write latencies.
- An NFS server is created in a virtual machine within the cluster to support virtual machine migration. The NFS file system is mounted on all storage nodes, and every storage node has a shared Nova directory under `/var/lib/nova/instances`. By default, this NFS file system is configured to utilize 30% of the total initial Contrail storage capacity.

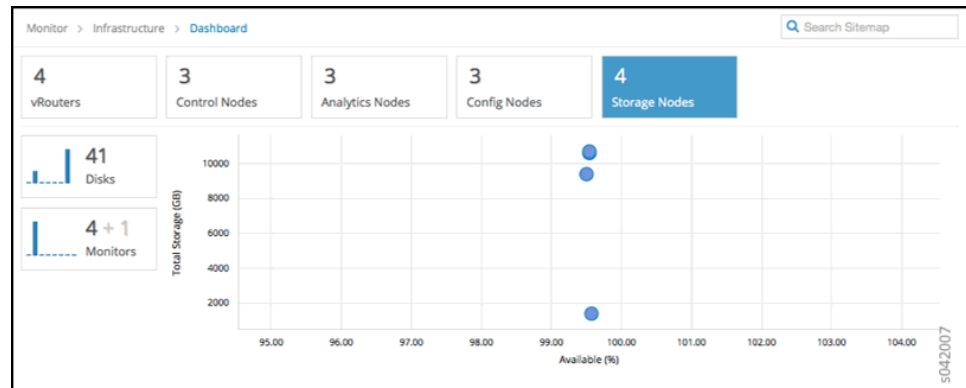
## Using the Contrail Storage User Interface

The Contrail storage solution provides a user interface integrated into the Contrail user interface. The storage solution user interface displays the following:

- Customer usable space, which is different from Ceph total space. The displayed usable space does not display the space used by replication and other Ceph functions.
- Monitor OSDs (disks), monitoring processes (MON), and state changes, enabling quick identification of resource failures within storage components.
- Total cluster I/O statistics and individual drive statistics.
- Ceph-specific information about each OSD (disk).
- Ceph logs, Ceph nodes, and Ceph alerts.



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, 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.

## 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 will define additional storage information in the **testbed.py**, 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 to all nodes:  
**fab install\_storage\_pkg\_all:/YYYY/contrail-storage-package-XXX.deb**
  2. After issuing **fab install\_contrail**, issue **fab install\_storage**.
  3. After issuing **fab setup\_all**, issue **fab setup\_storage**.
  4. If Contrail-based live virtual machine migration needs to be enabled, issue **fab setup\_nfs\_livem** or **fab setup\_nfs\_livem\_global**, as described in the following



**NOTE:** If virtual machine migration is not needed, do not issue either command.

- Use **fab setup\_nfs\_livem** to store the virtual machine's ephemeral storage on local drives.
  - Use **fab setup\_nfs\_livem\_global** to store the virtual machine's ephemeral storage within Contrail's storage (using Ceph). This command sets the cluster storage scope to global.
5. Add two new Contrail storage roles: **compute-storage** and **storage-master**.
    - Define the **storage-master** role on all nodes running OpenStack. Although Ceph has no notion of a master, define this role because Ceph must be run on the node that runs the OpenStack software. OpenStack nodes typically do not have any cluster storage defined, only local storage.
    - The **storage-compute** role is an add-on role, which means that compute nodes have the option of providing storage functionality. Standalone storage nodes are not supported.
  6. Change the **testbed.py** details as needed for your environment.

In the base configuration, define the **storage\_node\_config**, 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 which are used as OSD journals for other HDD drives.
  - `local-ssd-disks` is a list of disks used for local SSD LVM storage.
7. Add virtual machine migration as needed, using the following parameters.
- ```
live_migration = True

ceph_nfs_livevm = True

ceph_nfs_livem_subnet = '192.168.10.0/24' # Private subnet to be provided for live migration VM

ceph_nfs_livem_image = '/Ubuntu/libmnfs.qcow2' # path of live migration qcow2 image. This image is provided by Juniper.

ceph_nfs_livem_host = host3 # host in which the NFS VM will run
```
- For external NFS server based live migration, use the following configuration, valid for Contrail Release 2.20 and greater.
- ```
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/gids correctly and provides read/write access for all the uids. If there is any issue related to the permission, either the VM launch will error out or the live migration will fail 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, starting with Contrail Release 2.10.

Before installing the base operating system with Server Manager, ensure that the compute storage nodes have been configured with single operating system device installs.

### Cautions

- Virtual machine migration support uses a fixed IP (192.168.101.3) for the `livemnfs` virtual machine, starting with Contrail Release 2.10.
- 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. Run **ceph-authtool** if you need to generate unique keys for administration, monitor, and OSD.

- a. To install **ceph-authtool** on CentOS:

```
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. Run this command once for each key:

```
ceph-authtool --gen-print-key
```

- d. Add the generated keys to the **cluster.json** file:

```
cluster.json:
```

```
"storage_mon_secret":
```

```
"AQBDCdpTsB5FChAAOzI2++uosfmtj7tjmhPuOg==",
```

```
"osd_bootstrap_key":
```

```
"AQBKCdpTmN+HGRAAl6rmStq5iYoPnANzSXLcXA=="
```

```
"admin_key":
```

```
"AQBLCdpTuOS6FhAAfDW0SsdzyDAUeuwOr/h61A=="
```

- e. Starting with Release 2.10, add live-migration configuration to the **cluster.json** file, if you are using live migration.

```
"live_migration": "enable",
```

```
"live_migration_nfs_vm_host": "compute-node-01",
```

```
"live_migration_storage_scope": "global",
```

## Example Configurations for Storage for Reimaging and Provisioning a Server

Use the following example configurations as guidelines for reimaging and provisioning a server for storage. Examples are given for configurations for releases prior to Release 2.10 and for configurations for Release 2.10 and greater.

1. Define storage in the cluster. The following example configurations show new key-value pairs added to the configuration. The “cluster” section should appear similar to the following when storage is defined in a cluster.

**Example: Storage and key-value pairs defined in releases prior to 2.10:**

```
{
  "cluster" : [
    {
      "id" : "demo-cluster",
      "parameters" : {
        "router_asn": "<asn number>",
        "database_dir": "/home/cassandra",
        "database_token": "",
        "use_certificates": "False",
        "multi_tenancy": "False",
        "encapsulation_priority": "MPLSoUDP,MPLSoGRE,VXLAN",
        "service_token": "<password>",
        "keystone_user": "admin",
        "keystone_password": "<password>",
```

```
    "keystone_tenant": "admin",
    "analytics_data_ttl": "168",
    "subnet_mask": "<ip address>",
    "gateway": "<ip address>",
    "password": "<password>",
    "haproxy": "disable",
    "external_bgp": "",
    "domain": "demo.company.net",
    "storage_mon_secret": "$ABC123",
    "osd_bootstrap_key": "$ABC123",
    "admin_key": "$ABC123"
  }
}
]
```

**Example: Storage and key-value pairs defined in releases 2.10 and greater:**

```
{
  "cluster" : [
    {
      "id": "demo-cluster",
      "parameters" : {
        "router_asn": "<asn number>",
        "database_dir": "/home/cassandra",
        "database_token": "",
        "use_certificates": "False",
        "multi_tenancy": "False",
        "encapsulation_priority": "MPLSoUDP,MPLSoGRE,VXLAN",
```



```

    "service_token": "<password>",
    "keystone_user": "admin",
    "keystone_password": "<password>",
    "keystone_tenant": "admin",
    "analytics_data_ttl": "168",
    "subnet_mask": "<ip address>",
    "gateway": "<ip address>",
    "password": "<password>",
    "haproxy": "disable",
    "external_bgp": "",
    "domain": "demo.company.net",
    "storage_mon_secret": "$ABC123",
    "osd_bootstrap_key": "$ABC123",
    "admin_key": "$ABC123",
    "live_migration": "enable",
    "live_migration_nfs_vm_host": "compute-host-01",
    "live_migration_storage_scope": "global",
  }
}
]
}

```

2. Add the **disks** key, the **storage-compute** role value, and the **storage\_repo\_id**.
  - 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 which have the **storage-compute** role.

The following server section is an example, showing the new keys **storage\_repo\_id** and **disks**, and the new values **storage-compute** and **storage-master**.

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 will correctly partition **/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 prior to 2.10:**

```
{ "server": [
  {
    "id": "demo2-server",
    "mac_address": "<mac address>",
    "ip_address": "<ip address>",
    "parameters": {
      "interface_name": "eth1",
      "compute_non_mgmt_ip": "",
      "compute_non_mgmt_gway": "",
      "storage_repo_id": "contrail-storage-packages",
      "disks": ["/dev/sdb:/dev/sde", "/dev/sdc:/dev/sde", "/dev/sdd:/dev/sde"]
    },
    "roles":
    ["config","openstack","control","compute","collector","webui","database","storage-compute","storage-master"],
    "cluster_id": "demo-cluster",
    "subnet_mask": "<ip address>",
    "gateway": "<ip address>",
    "password": "<password>",
    "domain": "demo.company.net",
    "email": "id@company.net"
  }
]
```

```
}
```

Example: Server.json updates defined in releases 2.10 and greater:

Server.json :

```
{
  "server": [
    {
      "id": "demo2-server",
      "mac_address": "<mac address>",
      "ip_address": "<ip address>",
      "parameters": {
        "interface_name": "eth1",
        "compute_non_mgmt_ip": "",
        "compute_non_mgmt_gway": "",
        "storage_repo_id": "contrail-storage-packages",
        "disks": ["/dev/sdb:/dev/sde", "/dev/sdc:/dev/sde", "/dev/sdd:/dev/sde"]
      },
      "roles":
        ["config","openstack","control","compute","collector","webui","database","storage-compute","storage-master"],
      "contrail": {
        "control_data_interface": "p3p2"
      },
      "network": {
        "interfaces": [
          {
            "default_gateway": "<ip address>",
            "dhcp": true,
            "ip_address": "<ip address>",
            "mac_address": "<mac address>",
            "member_interfaces": "",
            "name": "eth1",
            "tor": "",
            "tor_port": "",
            "type": "physical"
          },
          {
            "default_gateway": "<ip address>",
            "dhcp": ""
          }
        ]
      }
    }
  ]
}
```

```
        "ip_address": "<ip address>",
        "mac_address": "<mac address>",
        "member_interfaces": "",
        "name": "p3p2",
        "tor": "",
        "tor_port": "",
        "type": "physical"
    }
],
"management_interface": "eth1"
},

"cluster_id": "demo-cluster",

"subnet_mask": "<ip address>",

"gateway": "<ip address>",

"password": "<password>",

"domain": "demo.company.net",

"email": "id@company.net"

} ]

}
```

3. Use the following commands to provision the entire cluster:

```
# /opt/contrail/server_manager/client/server-manager -c

# /opt/contrail/server_manager/smgr_config.ini provision --cluster_id test-cluster
contrail_test_pkg
```

## Storage Installation Limits

### General Limitations

- Minimum number of storage nodes to configure: 2
- The number of storage nodes should always be an even number (2, 4, 12, 22, etc.).

### Fab Storage Install Limitations

There are no 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 3

# Upgrading Contrail Software

- [Upgrading Contrail Software from Release 1.20 or Greater to Release 2.10 on page 130](#)
- [Adding or Removing a Compute Node in an Existing Contrail Cluster on page 133](#)
- [DKMS for vRouter Kernel Module on page 134](#)

## Upgrading Contrail Software from Release 1.20 or Greater to Release 2.10

Use the following procedure to upgrade an installation of Contrail software from one release to a more recent release. This procedure is valid starting from Contrail Release 1.20 and greater.



**NOTE:** If you are installing Contrail for the first time, refer to the full documentation and installation instructions in “[Installing the Operating System and Contrail Packages](#)” on page 12.

Instructions are given for both CentOS and Ubuntu versions. The only Ubuntu versions supported for upgrading are Ubuntu 12.04 and 14.04.

To upgrade Contrail software from Contrail Release 1.20 or greater to Contrail Release 2.10 or greater:

1. Download the file **contrail-install-packages-x.xx-xxx.xxx.noarch.rpm | deb** from <http://www.juniper.net/support/downloads/?p=contrail#sw> and copy it to the **/tmp** directory on the config node, as follows:

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

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



**NOTE:** The variables **xxx-xxx** and so on represent the release and build numbers that are present in the name of the installation packages that you download.

2. Install the **contrail-install-packages**, using the correct command for your operating system:

*CentOS:* `yum localinstall /tmp/contrail-install-packages-x.xx-xxx.xxx..noarch.rpm`

*Ubuntu:* `dpkg -i /tmp/ contrail-install-packages_x.xx-xxx~havana_all.deb`

3. Set up the local repository by running the **setup.sh**:

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

4. Ensure that the **testbed.py** that was used to set up the cluster with Contrail is intact at **/opt/contrail/utils/fabfile/testbeds/**.

- Ensure that **testbed.py** has been set up with a combined **control\_data** section (required as of Contrail Release 1.10).
- Ensure that the **do\_parallel** flag is set to **True** in **testbed.py**, see bug [1426522](#) in Launchpad.net.

See “[Populating the Testbed Definitions File](#)” on page 26.



5. Upgrade the software, using the correct set of commands to match your operating system and vrouter, as described in the following:

Change to the utils folder:

```
cd /opt/contrail/utils; \
```

and select the correct upgrade procedure from the following to match your operating system and vrouter. In the following, *<from>* refers to the currently installed release number, such as 1.20, 1.21, 2.0:

*CentOS Upgrade Procedure:*

```
fab upgrade_contrail:<from>,/tmp/contrail-install-packages-x.xx-xxx.xxx.noarch.rpm;
```

*Ubuntu 12.04 Procedure:*

```
fab upgrade_contrail:<from>,/tmp/contrail-install-packages-x.xx-xxx~havana_all.deb;
```

*Ubuntu 14.04 Upgrade, Two Procedures:*

There are two different upgrade procedures for Ubuntu 14.04 upgrade to Contrail Release 2.10, depending on which vrouter (**contrail-vrouter-3.13.0-35-generic** or **contrail-vrouter-dkms**) is installed in your current setup.

As of Contrail Release 2.10, the recommended kernel version for an Ubuntu 14.04-based system is 3.13.0-40. Both procedures can use the command **fab upgrade\_kernel\_all** to upgrade the kernel.

#### **Ubuntu 14.04 Upgrade Procedure For System With contrail-vrouter-3.13.0-35-generic:**

Use the following upgrade procedure for Contrail Release 1.20 systems based on Ubuntu 14.04 with the **contrail-vrouter-3.13.0-35-generic** installed. The command sequence upgrades the kernel version and also reboots the compute nodes when finished.

```
fab install_pkg_all:/tmp/contrail-install-packages-x.xx-xxx~havana_all.deb;
```

```
fab migrate_compute_kernel;
```

```
fab upgrade_contrail:<from>,/tmp/contrail-install-packages-x.xx-xxx~havana_all.deb;
```

```
fab upgrade_kernel_all;
```

```
fab restart_openstack_compute;
```

#### **Ubuntu 14.04 Upgrade Procedure For System with contrail-vrouter-dkms:**

Use the following upgrade procedure for Contrail R1.20 systems based on Ubuntu 14.04 with the **contrail-vrouter-dkms**. The command sequence upgrades the kernel version and also reboots the compute nodes when finished.

```
fab upgrade_contrail:<from>,/tmp/contrail-install-packages-x.xx-xxx~havana_all.deb;
```

All nodes in the cluster can be upgraded to kernel version 3.13.0-40 by using the following fabric-utils command:

```
fab upgrade_kernel_all
```

6. On the OpenStack node, soft reboot all of the virtual machines.

You can do this in the OpenStack dashboard or log into the node that uses the **openstack** role and issue the following commands:

```
source /etc/contrail/openstackrc ; nova reboot <vm-name>
```

You can also use the following fab command to reboot all virtual machines:

```
fab reboot_vm
```

7. Check to ensure that the **openstack-nova-novncproxy** is still running:

```
service openstack-nova-novncproxy status
```

If necessary, restart the service:

```
service openstack-nova-novncproxy restart
```

8. (For Contrail Storage option, only.)

Contrail Storage has its own packages.

To upgrade Contrail Storage, download the file:

```
contrail-storage-packages_x.x-xx*.deb
```

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

and copy it to the **/tmp** directory on the config node, as follows:

```
Ubuntu: scp <id@server>:/path/to/contrail-storage-packages_x.x-xx*.deb /tmp
```



**NOTE:** Use only Icehouse packages (for example, **contrail-storage-packages\_2.0-22~icehouse\_all.deb**) because OpenStack Havana is no longer supported.

Use the following statement to upgrade the software:

```
cd /opt/contrail/utils; \
```

```
Ubuntu: fab
```

```
upgrade_storage:<from>,/tmp/contrail-storage-packages_2.0-22~icehouse_all.deb;
```

When upgrading to Contrail Release 2.10, add the following steps if you have live migration configured. Upgrades to Release 2.0 do not require these steps. Select the command that matches your live migration configuration.

```
fab setup_nfs_livem
```

or

```
fab setup_nfs_livem_global
```

**Related Documentation**

- *Contrail Getting Started Guide*

## Adding or Removing a Compute Node in an Existing Contrail Cluster

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

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



**NOTE:** For convenience, this procedure assumes you are adding a node @1.1.1.1, however, replace the 1.1.1.1 with the correct IP for the node or nodes that you are adding.

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

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

Ubuntu: **scp <id@server>:/path/to/contrail-install-packages\_xxx-xxx-havana\_all.deb /tmp**

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

**cd /opt/contrail/utlis; fab upgrade\_kernel\_node:root@1.1.1.1**

where 1.1.1.1 should be replaced with the server's actual IP address.

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

CentOS: **fab**

**install\_pkg\_node:/tmp/contrail-install-packages\_x.xx-xxx.xxx.noarch.rpm,root@1.1.1.1**

Ubuntu: **fab**

**install\_pkg\_node:/tmp/contrail-install-packages\_x.xx-xxx-havana\_all.deb,root@1.1.1.1**

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

**fab add\_vrouter\_node:root@1.1.1.1**

### Removing a Node

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



**NOTE:** For convenience, this procedure assumes you are adding a node @1.1.1.1, however, replace the 1.1.1.1 with the correct IP for the node or nodes that you are adding.

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

**fab detach\_vrouter\_node:root@1.1.1.1**

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

## DKMS for vRouter Kernel Module

---

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

In Contrail, the vRouter kernel module is an out-of-tree, high performance packet forwarding module that provides advanced packet forwarding functionality in a reliable and stable manner. Contrail provides a DKMS-compatible source package for Ubuntu so that a customer who deploys an Ubuntu-based Contrail system does 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, both the DKMS package and the **contrail-vrouter-utils** package must be installed, because the **contrail-vrouter-dkms** package is dependent on both. Installing the **contrail-vrouter-dkms** package adds the vrouter sources to the DKMS database, builds the vrouter module, and installs it in the existing kernel modules tree. When a kernel upgrade occurs, DKMS ensures that the module is compiled for the newer kernel and installed in the proper location so that upon reboot, the newer module can be used with the upgraded kernel.

This feature is supported as of Contrail Release 1.10 on Ubuntu distributions. Support for CentOS is in the product roadmap.

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>

**Related** •  
**Documentation**

## PART 3

# Configuration

- [Configuring Virtual Networks on page 137](#)
- [Examples of Tiered Web Configurations on page 181](#)
- [High Availability on page 193](#)



## CHAPTER 4

# Configuring Virtual Networks

- [Creating Projects in OpenStack for Configuring Tenants in Contrail on page 138](#)
- [Creating a Virtual Network—Juniper Networks Contrail on page 139](#)
- [Deleting a Virtual Network—Juniper Networks Contrail on page 142](#)
- [Creating a Virtual Network—OpenStack Contrail on page 143](#)
- [Deleting a Virtual Network—OpenStack Contrail on page 144](#)
- [Creating an Image on page 146](#)
- [Launching a Virtual Machine \(Instance\) on page 148](#)
- [Creating a Network Policy—Juniper Networks Contrail on page 151](#)
- [Associating a Network to a Policy—Juniper Networks Contrail on page 153](#)
- [Creating a Network Policy—OpenStack Contrail on page 157](#)
- [Associating a Network to a Policy—OpenStack Contrail on page 160](#)
- [Creating a Floating IP Address Pool on page 162](#)
- [Allocating a Floating IP Address to a Virtual Machine on page 164](#)
- [Using Security Groups with Virtual Machines \(Instances\) on page 165](#)
- [Support for IPv6 Networks in Contrail on page 168](#)
- [Configuring EVPN and VXLAN on page 171](#)
- [Configuring Network QoS Parameters on page 177](#)

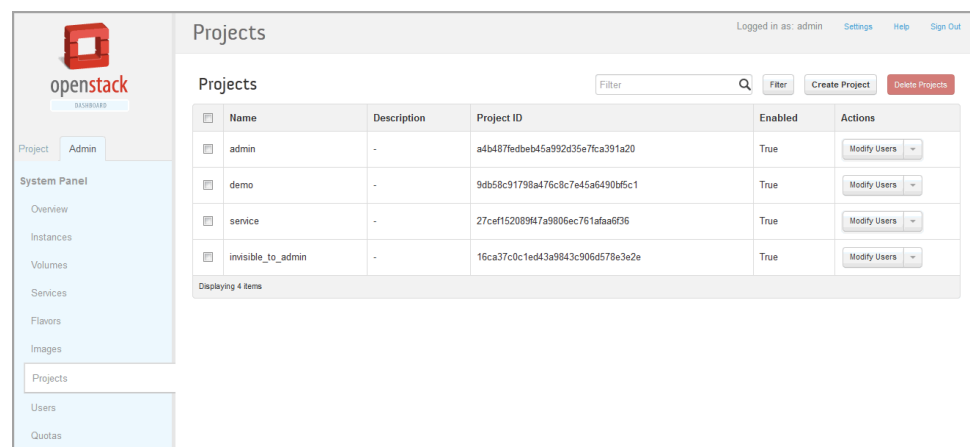
## Creating Projects in OpenStack for Configuring Tenants in Contrail

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** screen.

1. Click the **Admin** tab on the OpenStack dashboard, then click the **Projects** link to access the **Projects** screen; see [Figure 8 on page 138](#).

**Figure 8: OpenStack Projects**



2. In the upper right, click the **Create Project** button to access the **Add Project** screen; see [Figure 9 on page 138](#).

**Figure 9: 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.



- 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.

- Click **Finish** to create the project.
- Refer to OpenStack documentation for more information about creating and managing projects.

#### Related Documentation

- [Creating a Virtual Network—Juniper Networks Contrail](#)
- [Creating a Virtual Network—OpenStack Contrail on page 139](#)
- [OpenStack documentation](#)

## Creating a Virtual Network—Juniper Networks Contrail

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

- Before creating a virtual network, create an IP Address Management (IPAM) for your project. Select **Configure > Networking > IP Address Management**, then click the **Create** button.

The **Add IP Address Management** window appears, see [Figure 10 on page 139](#).

**Figure 10: Add IP Address Management**

- Complete the fields in **Add IP Address Management**: see field descriptions in [Table 11 on page 139](#).

**Table 11: Add IP Address Management Fields**

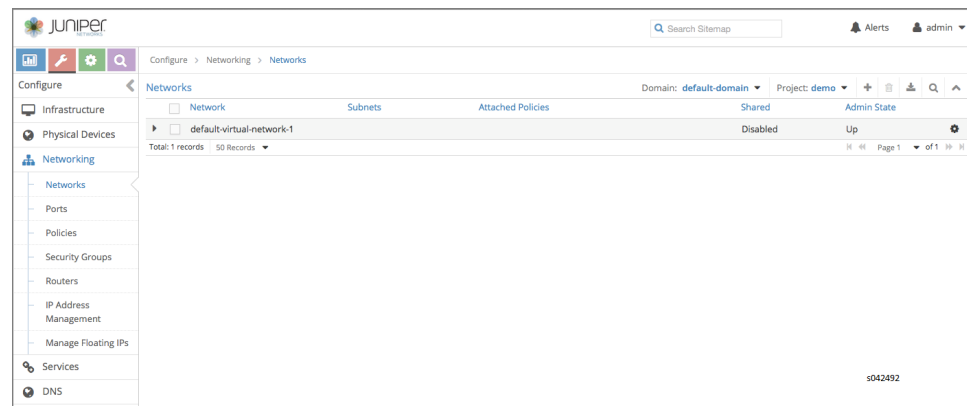
Field	Description
<b>Name</b>	Enter a name for the IPAM you are creating.

Table 11: Add IP Address Management Fields (*continued*)

Field	Description
DNS Method	Select from a drop-down list the domain name server method for this IPAM: <b>Default</b> , <b>Virtual DNS</b> , <b>Tenant</b> , or <b>None</b> .
NTP Server IP	Enter the IP address of an NTP server to be used for this IPAM.
Domain Name	Enter a domain name to be used for this IPAM.

3. Click **Configure > Networking > Networks** to access the **Configure Networks** screen; see [Figure 11 on page 140](#).

Figure 11: Configure Networks



4. Verify that your project is displayed as active in the upper right field, then click the **Create** button to access the **Create Network** window; see [Figure 12 on page 140](#). Use the scroll bar to access all sections of this window.

Figure 12: Create Network

5. Complete the fields in the **Create Network** screen with values that identify the network name, network policy, and IP options as needed. See field descriptions in [Table 12 on page 141](#).

**Table 12: Create Network Fields**

Field	Description
<b>Network Name</b>	Enter a name for the virtual network you are creating.
<b>Network Policy(s)</b>	Select from a drop-down list of available policies the policy to be applied to this network. You can select more than one policy by clicking each one needed.
<b>Subnets</b>	Use this area to identify and manage subnets for this virtual network. Click the + icon to open fields for IPAM, CIDR, Allocation Pools, Gateway, DNS, and DHCP. Select the subnet to be added from a drop down list in the IPAM field. Complete the remaining fields as necessary. You can add multiple subnets to a network. When finished, click the + icon to add the selections into the columns below the fields. Or click the - icon to remove the selections.
<b>Host Routes</b>	Use this area to add or remove host routes for this network. Click the + icon to open fields where you can enter the Route Prefix and the Next Hop. Click the + icon to add the information, or click the - icon to remove the information.
<b>Advanced Options</b>	Use this area to add or remove advanced options, including identifying the Admin State to be Up or Down, to identify the network as Shared or External, to add DNS servers, or to define a VxLAN Identifier.
<b>Floating IP Pools</b>	Use this area to identify and manage the Floating IP address pools for this virtual network. Click the + icon to open fields where you can enter the Pool Name and Projects. Click the + icon to add the information, or click the - icon to remove the information.
<b>Route Target(s)</b>	Move the scroll bar down to access this area, then specify one or more route targets for this virtual network. Click the + icon to open fields where you can enter route target identifiers. Click the + icon to add the information, or click the - icon to remove the information.

6. To save your network, click the **Save** button, or click **Cancel** to discard your work and start over.

Now you can create a network policy, see [“Creating a Network Policy—Juniper Networks Contrail” on page 151](#).

**Related Documentation**

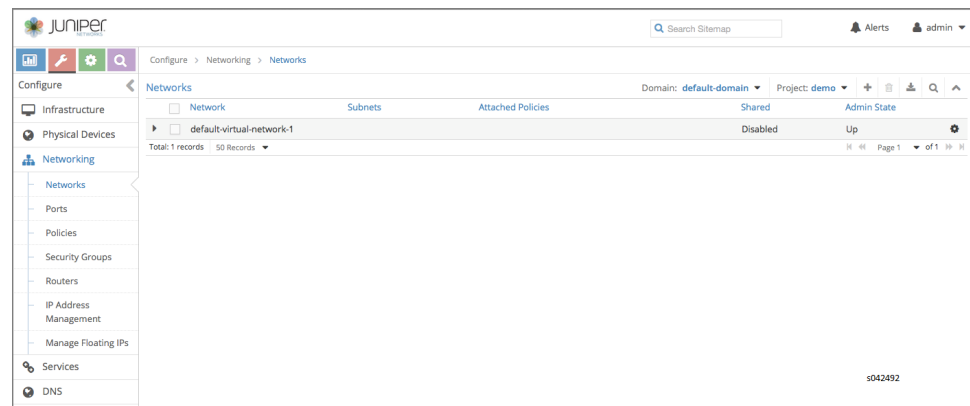
- [Creating an Image on page 146](#)
- [Launching a Virtual Machine \(Instance\) on page 148](#)
- [Creating a Network Policy—Juniper Networks Contrail on page 151](#)
- [Deleting a Virtual Network—Juniper Networks Contrail](#)

## Deleting a Virtual Network—Juniper Networks Contrail

You can delete any of the virtual networks in your system. However, you must first disassociate any virtual machines (instances) that are associated with that network. Use OpenStack to view and delete the virtual machines associated with a virtual network, see [“Deleting a Virtual Network—OpenStack Contrail” on page 142](#). When you are finished deleting the virtual machines associated with a virtual network, you can delete the network in OpenStack, or you can delete the network in Juniper Networks Contrail, using the following procedure.

1. To view the virtual networks in the current project, click **Configure > Networks** to access the **Configure Networks** screen in Juniper Networks Contrail; see [Figure 13 on page 142](#).

**Figure 13: Configure Networks**



2. Click the network you want to delete, then click the Delete (trashcan) icon at the top right. A confirm window is displayed.
3. Click **Confirm** to delete the network, or click **Cancel** to quit the delete activity.

**Related Documentation**

- [Creating a Virtual Network—Juniper Networks Contrail](#)

## Creating a Virtual Network—OpenStack 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 OpenStack.

1. Click **Project > Networking** to access the **Networks** screen; see [Figure 14 on page 143](#).

**Figure 14: Networks**

2. Verify that the correct project is displayed in the **Current Project** box, then click the **Create Network** button to access the **Create Network** window; see [Figure 15 on page 143](#).

**Figure 15: Create Network**

3. Complete the fields in the **Create Network** window with values that identify the network name, IP block, policy, and IP options as needed. See field descriptions in [Table 13 on page 143](#).

**Table 13: Create Network Fields**

Field	Description
<b>Name</b>	Enter a name for the network.
<b>Description</b>	Enter a description for the network.
<b>IP Block (optional)</b>	Enter the IP address of the address block assigned to this project.

Table 13: Create Network Fields (*continued*)

Field	Description
IPAM (optional)	For new projects, an IPAM can be added while creating the virtual network. VM instances created in this virtual network will be assigned an address from this address block automatically by the system when a VM is launched.
Gateway (optional)	Optionally, enter an explicit gateway IP for the IP address block.
Network Policy	Any policies already created are listed. To select a policy, click the check box for the policy.

- To save your network, click the **Create Network** button, or click **Cancel** to discard your work and start over.

#### Related Documentation

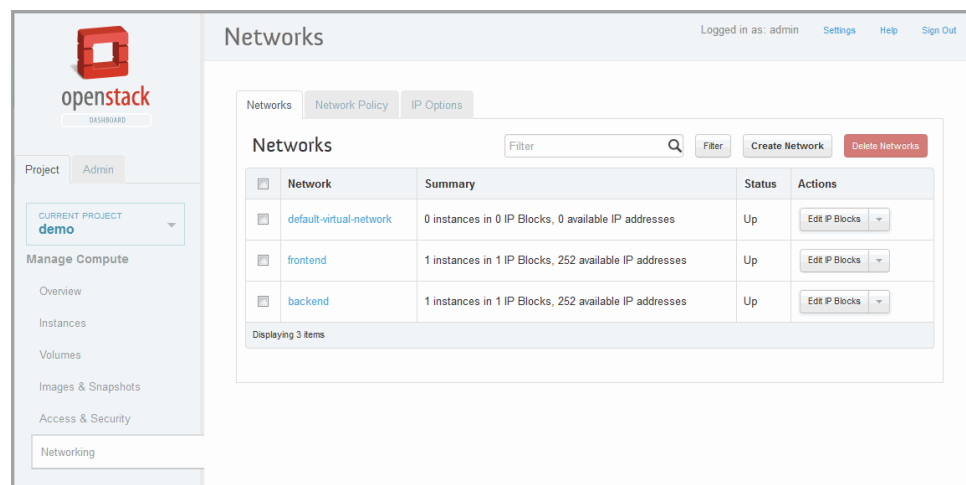
- [Deleting a Virtual Network—OpenStack Contrail on page 142](#)

## Deleting a Virtual Network—OpenStack Contrail

You can delete any of the virtual networks in your system. However, you must first disassociate any virtual machines (instances) that are associated with that network. The following procedure shows how to delete a virtual network when using OpenStack.

- To view virtual machines that are associated with a virtual network, in the OpenStack module, access the **Project** tab and click **Networking**. The **Networks** screen appears; see [Figure 16 on page 144](#).

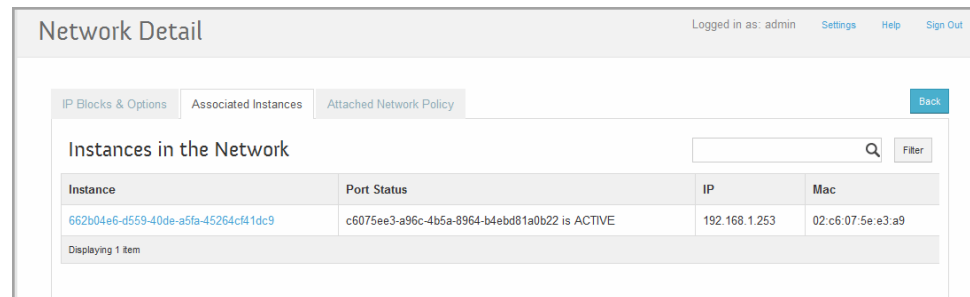
Figure 16: OpenStack Networks



- At the **Networks** screen, click the network to be deleted.

The **Network Detail** screen appears; see [Figure 17 on page 145](#).

Figure 17: OpenStack Network Detail , Associated Instances Tab

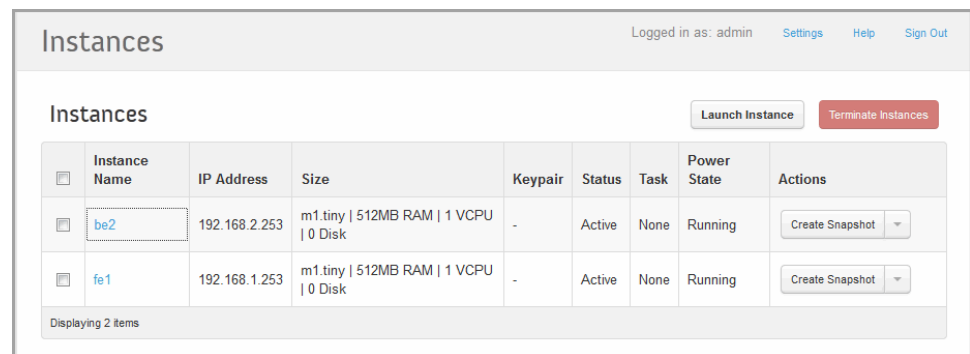


- Click the **Associated Instances** tab to see the instances associated with this network. Make note of the IP addresses of any instances that are associated with this network.

- In the **Project** tab, select **Instances**.

The **Instances** screen appears, displaying the instances associated with the current project; see [Figure 18 on page 145](#).

Figure 18: Instances



- On the **Instances** screen, click the check box for any instance that is associated with the network that you want to delete, then click the **Terminate Instances** button to delete the instance.
- When all instances that are associated with the network to be deleted have been terminated, delete the network.

To delete a network, return to the **Networks** screen (see [Figure 16 on page 144](#)), select the network to be deleted, then click the **Delete Networks** button in the upper right.

#### Related Documentation

- [Creating a Virtual Network—OpenStack Contrail on page 139](#)

## Creating an Image

You can use the OpenStack dashboard to specify an image to upload to the Image Service for a project in your system.

1. Make sure you have selected the correct project to which you will associate an image.
  - a. In OpenStack, in the left column, select the **Project** tab.
  - b. Make sure your project is selected as the project in the **Current Project** box, then click **Images & Snapshots** in the left column.

The **Images & Snapshots** screen appears; see [Figure 19 on page 146](#)

**Figure 19: Images & Snapshots**

Images & Snapshots

Logged in as: admin [Settings](#) [Help](#) [Sign Out](#)

**Images** [Create Image](#) [Delete Images](#)

<input type="checkbox"/>	Image Name	Type	Status	Public	Format	Actions
<input type="checkbox"/>	ubuntu	Image	Active	Yes	QCOW2	<a href="#">Launch</a>
<input type="checkbox"/>	redmine-db	Image	Active	Yes	VMDK	<a href="#">Launch</a>
<input type="checkbox"/>	redmine-web	Image	Active	Yes	VMDK	<a href="#">Launch</a>

Displaying 3 items

**Instance Snapshots**

<input type="checkbox"/>	Image Name	Type	Status	Public	Format	Actions
No items to display.						

Displaying 1 item

2. Click **Create Image**.

The **Create An Image** window appears; see [Figure 20 on page 147](#).



Figure 20: Create An Image

**Create An Image**

**Name**

**Image Location**

**Format**

**Minimum Disk (GB)**

**Minimum Ram (MB)**

**Public**

**Description:**  
Specify an image to upload to the Image Service.  
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.

Cancel Create Image

3. Complete the fields in this window to specify your image. [Table 14 on page 147](#) describes each of the fields on this screen.



**NOTE:** Only images available via 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 14: Create An Image Fields

Field	Description
<b>Name</b>	Required field. Enter a name for this image.
<b>Image Location</b>	Required field. 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 will result in unusable images.
<b>Format</b>	Required field. Select the format of the image from a list: AKI- Amazon Kernel Image AMI- Amazon Machine Image ARI- Amazon Ramdisk Image ISO- Optical Disk Image QCOW2-QEMU Emulator Raw VDI VHD VMDK

Table 14: Create An Image Fields (*continued*)

Field	Description
Minimum Disk (GB)	Enter the minimum disk size required to boot the image. If you do not specify a size, the default is 0 (no minimum).
Minimum Ram (MB)	Enter the minimum RAM required to boot the image. If you do not specify a size, the default is 0 (no minimum).
Public	Check the box if this is a public image. Leave unchecked for a private image.

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

#### Related Documentation

- [Launching a Virtual Machine \(Instance\) on page 148](#)

## Launching a Virtual Machine (Instance)

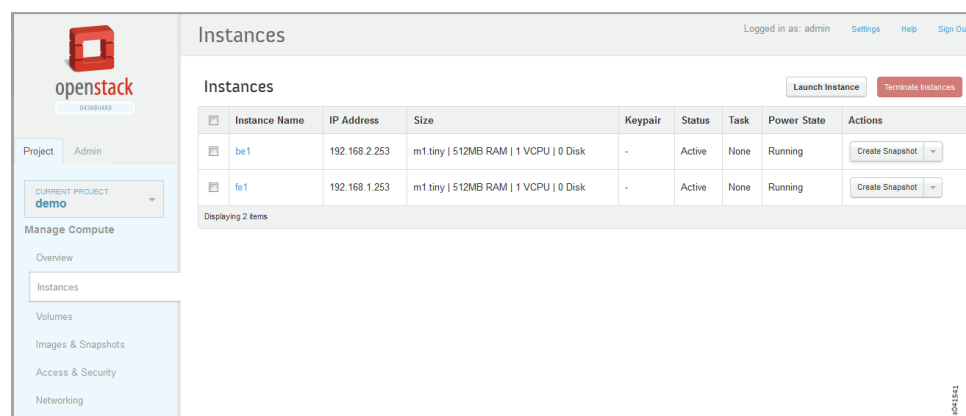
After you have created virtual networks for your project, you can create and launch virtual machines. Virtual networks (VNs) are populated with virtual machines (VMs), also called instances. A VM is a simulation of a physical machine, such as a workstation or a server, that runs on a host that supports virtualization. Many VMs can run on the same host, sharing its resources. A VM has its own operating system image that can be different from that of other VMs running on the same host.

You use the OpenStack module to define and launch VMs (instances).

- On the OpenStack dashboard **Project** tab, make sure your project is selected in the left column in the **Current Project** box, then click **Instances**.

The **Instances** screen appears, displaying all instances (VMs) currently in the selected project; see [Figure 21 on page 148](#).

Figure 21: OpenStack Instances



- To create and launch a new instance, click the **Launch Instance** button in the upper right corner.

The **Launch Instance** window appears, where you can define and launch a new instance.

Figure 22: Launch Instance , Details Tab

**Launch Instance**

Details Access & Security Networking Volume Options Post-Creation

**Instance Source**  
Image

**Image**  
redmine-db

**Instance Name**  
redmine-db-10

**Flavor**  
m1.tiny

**Instance Count**  
1

**Compute Hostname**  
compute-nodea33

Specify the details for launching an instance.  
The chart below shows the resources used by this project in relation to the project's quotas.

**Flavor Details**

Name	m1.tiny
VCPUs	1
Root Disk	0 GB
Ephemeral Disk	0 GB
Total Disk	0 GB
RAM	512 MB

**Project Quotas**

Number of Instances (4) 99,996 Available

Compute Host to launch Instance on 99,994 Available

Total RAM (9,216 MB) 9,990,784 MB Available

Cancel Launch

3. Make sure the **Details** tab is active; see [Figure 22 on page 149](#), then define your instance using the fields shown in [Table 15 on page 149](#)

Table 15: Launch Instance Details Tab Fields

Field	Description
<b>Instance Source</b>	Select from the list the source type: <b>Image</b> or <b>Snapshot</b> .
<b>Image</b>	Select from a list the image to use for this instance. The images represent the operating systems and applications available for this project.
<b>Instance Name</b>	Enter a name for this instance. .
<b>Flavor</b>	Select from a list the OpenStack Flavor for this instance. Flavors provide general definitions for sizing VMs. The Flavor Details of the Flavor you select displays on the right column of this window.
<b>Instance Count</b>	Enter the number of instances you want to launch using the details defined in this screen. On the right side column, <b>Project Quotas</b> displays the number of instances currently active and the number still available for this project.

Table 15: Launch Instance Details Tab Fields (*continued*)

Field	Description
<b>Compute Hostname</b>	To launch a VM on a specific compute node, enter the name of the compute node. This functionality is only available to administrators.

- Click the **Networking** tab on the **Launch Instance** screen to identify one or more networks to associate with this instance; see [Figure 23 on page 150](#).

Figure 23: Launch Instance, Networking Tab

- When finished defining this instance, click the **Launch** button at the lower right. Your new VM instance is launched as part of your project.

**Related Documentation**

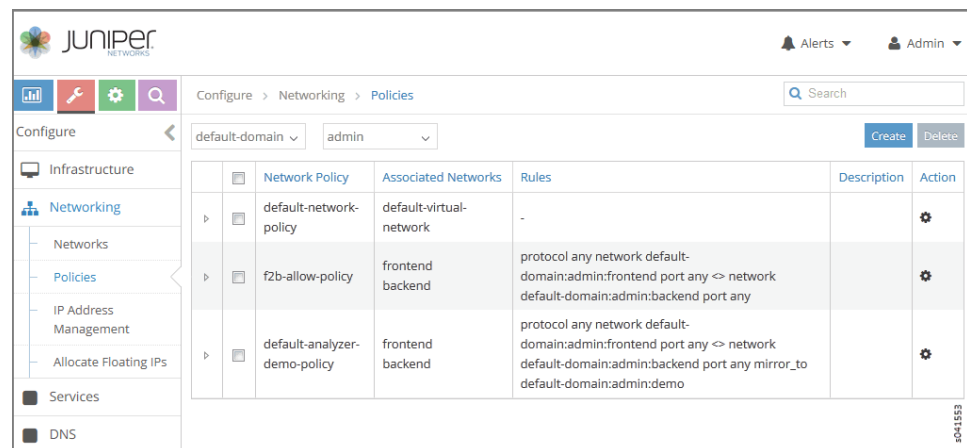
- [OpenStack documentation](#)

## Creating a Network Policy—Juniper Networks Contrail

The Contrail Controller makes creating network traffic policies very simple. You work from the self-service user interface to define a policy, then define a rule or rules to be applied in that policy. You can define such things as the type and direction of traffic for the rule, the source and destination of that traffic, traffic originating from or destined for specific ports, the sequence in which to apply a rule, and so on. The following procedure shows how to create a network policy when using Juniper Networks Contrail.

1. In the Contrail module, start at **Configure > Networking > Policies** to display the **Network Policy** screen; see [Figure 24 on page 151](#).

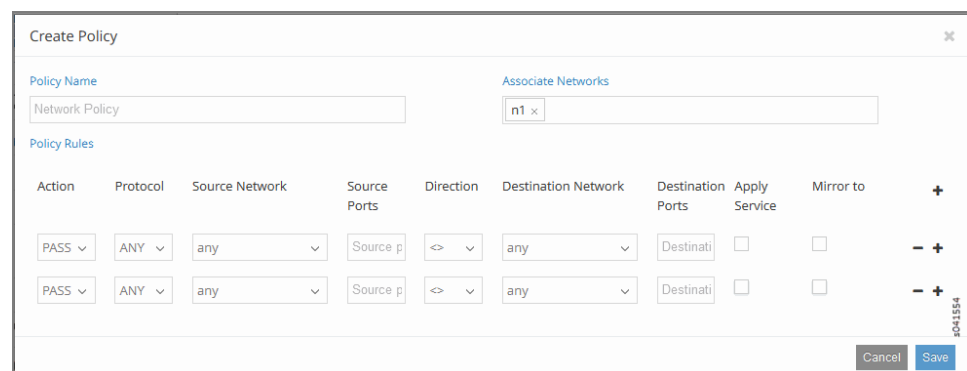
**Figure 24: Network Policy**



2. Click the **Create** button at the upper right.

The **Create Policy** window appears; see [Figure 25 on page 151](#).

**Figure 25: Create Policy**



3. Complete the fields in the **Create Policy** window, using the guidelines in [Table 16 on page 152](#).

Table 16: Create Policy Fields

Field	Description
<b>Policy Name</b>	Enter a name for the policy you are creating.
<b>Associate Networks</b>	Click this field to select from a list of available networks the networks to be associated with this policy. Click one network at a time to add one or more networks to the field. The selected networks are listed in the field. To remove any selected network, click the X to the right of the network.
<b>Policy Rules</b>	Use this area to define the rules for the policy you are creating. Click the + (plus sign) to open up the fields for defining the rules. Click the - (minus sign) to delete any rule. Multiple rules can be added to a policy. Each policy rule field is described in the following table rows.
<b>Action</b>	Define the action to take with traffic that matches the current rule. Select from a list: <b>Pass, Deny</b> .
<b>Protocol</b>	Define the protocol associated with traffic for this policy rule. Select from a list of available protocols (or <b>ANY</b> ): <b>ANY, TCP, UDP, ICMP</b> .
<b>Source Network</b>	Select the source network for traffic associated with this policy rule. Choose <b>ANY</b> or select from a list of all sources available displayed in the drop-down, in the form: <i>domain-name:project-name:network-name</i> .
<b>Source Ports</b>	Use this field to specify that traffic from particular source port(s) are associated with this policy rule. Identify traffic from <b>any</b> port or enter a specific port, a list of ports separated with commas, or a range of ports in the form <i>nnnn-nnnnn</i> .
<b>Direction</b>	Define the direction of traffic to match the rule, for example, to traffic moving in and out, or only to traffic moving in one direction. Select from a list: <> (bidirectional), > (unidirectional).
<b>Destination Network</b>	Select the destination network for traffic to match this rule. Choose <b>ANY</b> or select from a list of all destinations available displayed in the drop-down, in the form: <i>domain-name:project-name:network-name</i> .
<b>Destination Ports</b>	Define the destination port for traffic to match this rule. Enter <b>any</b> for any destination port, or enter a specific port, a list of ports separated with commas, or a range of ports in the form <i>nnnn-nnnnn</i> .
<b>Apply Service</b>	Check the box to open a field where you can select from a list of available services the services to apply to this policy. The services will be applied in the order in which they are selected. There is a restricted set of options that can be selected when applying services. For more information about services, see <a href="#">“Service Chaining” on page 205</a> .
<b>Mirror to</b>	Check the box to open a field where you can select from the list of configured services the services that you want to mirror in this policy. You can select a maximum of two services to mirror. For more information about mirroring, see <a href="#">“Configuring Traffic Analyzers and Packet Capture for Mirroring” on page 257</a> .

- When you are finished selecting the rules for this policy, click the **Save** button.

The policy you just defined displays in the **Network Policy** column.

Next you can associate the policy to a network, see [“Associating a Network to a Policy—Juniper Networks Contrail” on page 153](#).

- Related Documentation**
- [Associating a Network to a Policy—Juniper Networks Contrail on page 153](#)
  - [Creating a Virtual Network—Juniper Networks Contrail](#)

---

## Associating a Network to a Policy—Juniper Networks Contrail

- [Associating Network Policies Overview on page 153](#)
- [Associating a Network Policy to a Network from the Edit Network on page 153](#)
- [Associating Networks with Network Policies from the Edit Policy on page 156](#)

### Associating Network Policies Overview

Contrail helps you create and manage virtual networks (VNs). By default, all traffic in a VN is isolated to that VN. Traffic can only leave a VN by means of network policies that are defined for the VN.

This procedure shows how to associate a network policy with a network, using the Juniper Networks Contrail interface.

If you did not associate an existing network policy when you created your virtual network, you can use the **Network Policy(s)** field from the **Edit Network** window, or you can use the **Associate Networks** field from the **Edit Policy** window to associate or disassociate network policies with networks. The following procedures demonstrate both methods.

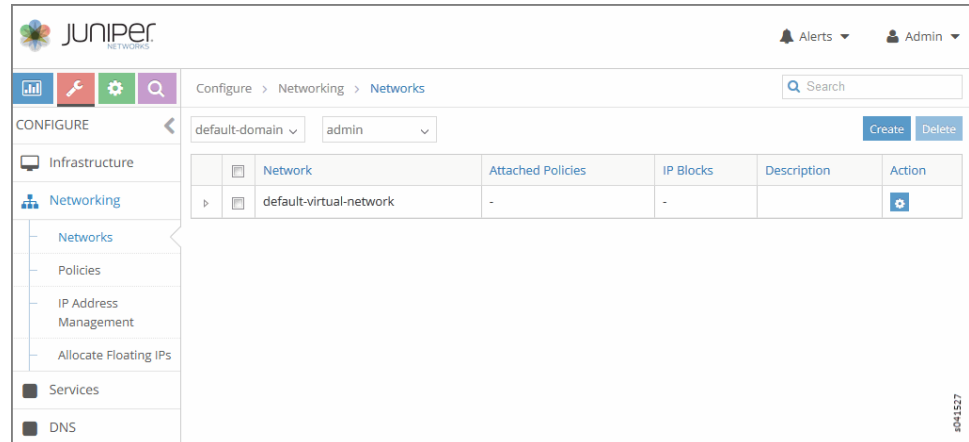
### Associating a Network Policy to a Network from the Edit Network

This procedure shows how to attach (associate) a network policy to a network when starting from the **Edit Network** window.

1. Start at **Configure > Networking > Networks**; see [Figure 26 on page 154](#).

Make sure your project is the active project in the upper right.

**Figure 26: Configure > Networking > Networks**



2. Click the network you will associate with a policy, then in the **Action** column, click the action icon and select **Edit**.

The **Edit Network** window for the selected network appears; see [Figure 27 on page 155](#).

Any policies already associated with the selected network appear in the **Network Policy(s)** field.



Figure 27: Edit Network

Edit Network default-virtual-network

**Network Name** default-virtual-netwc

**Network Policy(s)** default-network-policy x

**Address Management** default-network... v xxx.xxx.xxx.xxx/xx + -

IPAM	IP Block
default-network-ipam	10.84.23.9/24

**Floating IP Pools** Pool Name Select Projects... + -

Pool Name	Projects

Cancel Save

3. Click the **Network Policy(s)** field to show a list of existing policies, and click to select a policy to associate with the selected network.

You can also disassociate a selected policy by clicking the X next to its name when it appears configured in the **Network Policy(s)** field.

4. When finished, click the **Save** button, or click the **Cancel** button to undo your selections.

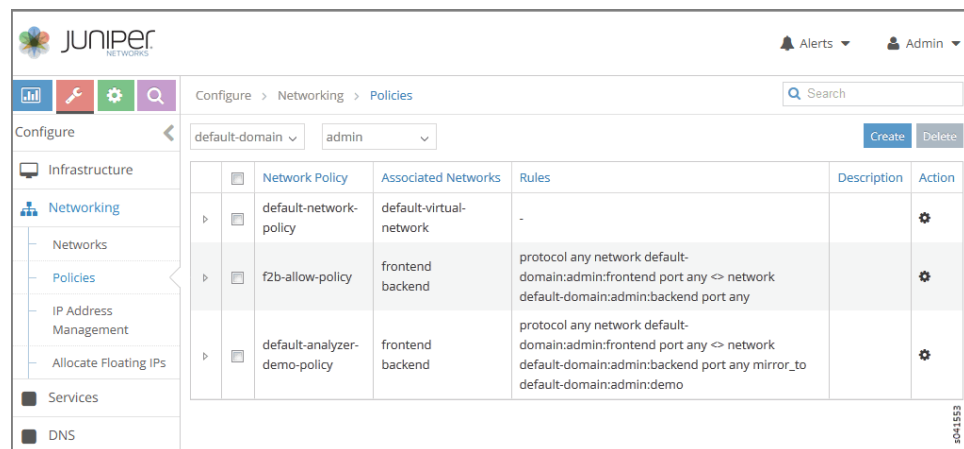
## Associating Networks with Network Policies from the Edit Policy

If you did not associate a network when you created a network policy, you can use **Edit Policy** to associate a selected policy with a network. This procedure shows how to associate network policies when starting from the **Edit Policy** window.

1. Start at **Configure > Networking > Policies**.

Make sure your project is selected in the field in the upper right; see [Figure 28 on page 156](#).

**Figure 28: Configure > Networking > Policies**

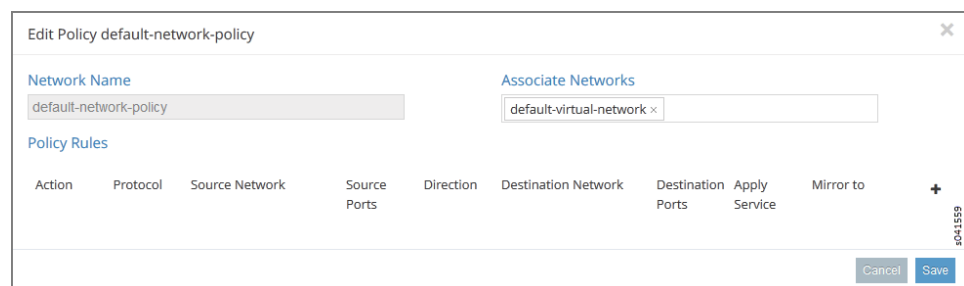


2. Click the policy you will associate with a network, then in the **Action** column, click the action icon and select **Edit**.

The **Edit Policy** window for the selected policy appears; see [Figure 29 on page 156](#).

Any networks already associated with the selected policy appear in the **Associate Networks** field.

**Figure 29: Edit Policy**



3. Click in the **Associate Networks** field to show a list of existing networks, and click to select a network to associate with the selected policy.

You can also disassociate a selected network by clicking the X next to its name when it appears configured in the **Associate Networks** field.

4. When finished, click the **Save** button, or click the **Cancel** button to undo your selections.

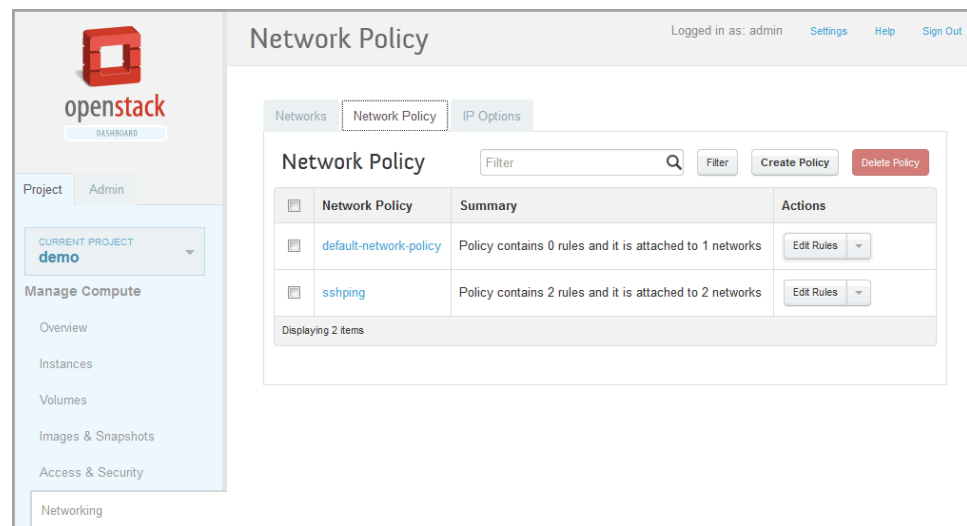
- Related Documentation**
- [Creating a Floating IP Address Pool on page 162](#)
  - [Allocating a Floating IP Address to a Virtual Machine on page 164](#)

## Creating a Network Policy—OpenStack Contrail

Contrail makes creating network traffic policies very simple. You work from the self-service user interface to define a policy, then define a rule or rules to be applied in that policy. You can define such things as the type and direction of traffic for the rule, the source and destination of that traffic, traffic originating from or destined for specific ports, the sequence in which to apply a rule, and so on. The following procedure shows how to create a network policy when using OpenStack.

1. On the OpenStack dashboard, make sure your project is displayed in the **Current Project** box, click **Networking**, and then click the **Network Policy** tab to display the **Network Policy** screen; see [Figure 30 on page 157](#).

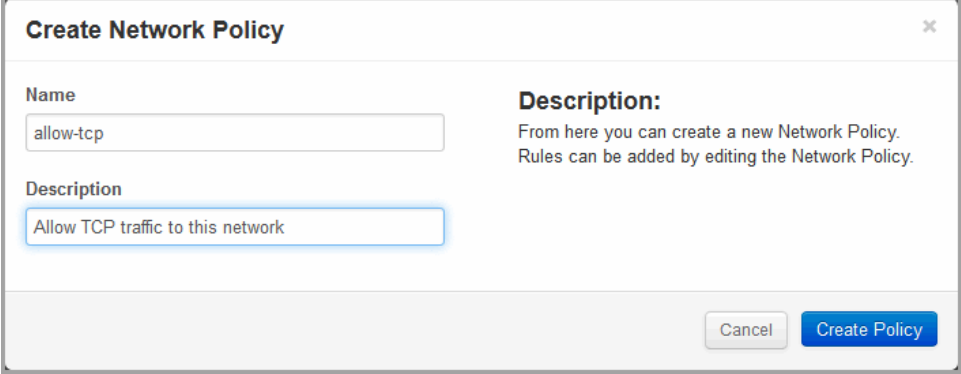
**Figure 30: Network Policy**



2. Click the **Create Policy** button at the upper right.

The **Create Network Policy** window appears; see [Figure 31 on page 158](#).

Figure 31: Create Network Policy



**Create Network Policy** [X]

**Name**

**Description**

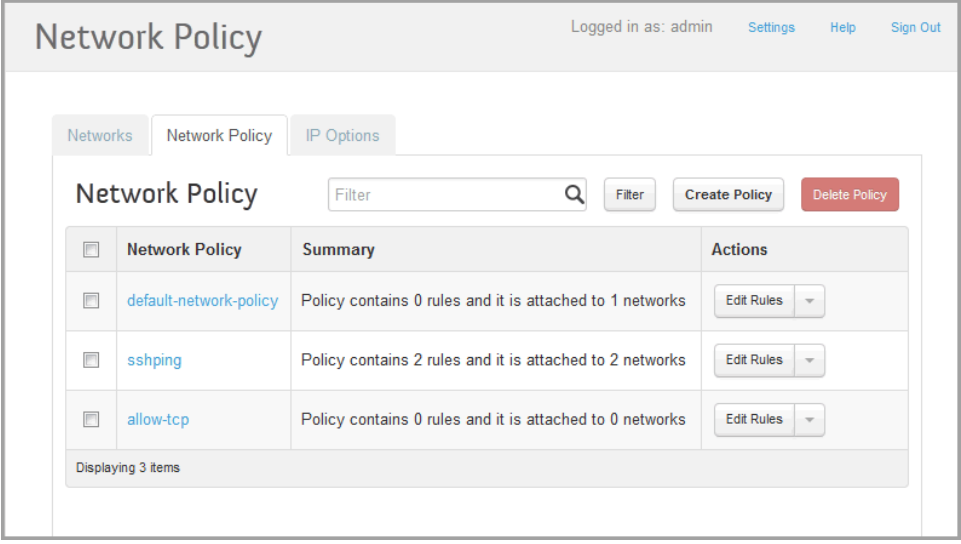
**Description:**  
 From here you can create a new Network Policy. Rules can be added by editing the Network Policy.

Cancel Create Policy

3. Enter a name and a description for this policy. Names cannot include spaces.
4. When finished, click the **Create Policy** button on the lower right.

Your policy is created and it appears on the **Network Policy** screen; see [Figure 32 on page 158](#).

Figure 32: Network Policy



Network Policy

Logged in as: admin Settings Help Sign Out

Networks Network Policy IP Options

Network Policy Filter [X] Filter Create Policy Delete Policy

<input type="checkbox"/>	Network Policy	Summary	Actions
<input type="checkbox"/>	default-network-policy	Policy contains 0 rules and it is attached to 1 networks	Edit Rules
<input type="checkbox"/>	sshping	Policy contains 2 rules and it is attached to 2 networks	Edit Rules
<input type="checkbox"/>	allow-tcp	Policy contains 0 rules and it is attached to 0 networks	Edit Rules

Displaying 3 items

5. On the **Network Policy** window, click the check box for your new policy, then click the **Edit Rules** button for that policy.

The **Edit Policy Rules** window appears; see [Figure 33 on page 159](#).

Figure 33: Edit Policy Rules

6. Define the rules for your policy, using the guidelines in [Table 17](#) on page 159.

Table 17: Edit Policy Rules Fields

Field	Description
<b>Policy Rules Details</b>	This section of the window displays any rules that have already been created for this policy.
<b>Id</b>	Displays a sequential number identifier for each rule within a policy.
<b>Rule Details</b>	Displays a description of the rule on this line.
<b>Actions</b>	Available actions for the rule on this line appear in this column. Currently you can use the <b>Delete</b> button in this column to delete a rule.
<b>Sequence Id</b>	This field lets you define the order in which to apply the current rule. Select from a list: <b>Last Rule</b> , <b>First Rule</b> , <b>After Rule</b> .
<b>Action</b>	Define the action to take with traffic that matches the current rule. Select from a list: <b>Pass</b> , <b>Deny</b> .
<b>Direction</b>	Define the direction in which to apply the rule, for example, to traffic moving in and out, or only to traffic moving in one direction. Select from a list: <b>Bidirectional</b> , <b>Unidirectional</b> .
<b>IP Protocol</b>	Select from a list of available protocols (or <b>ANY</b> ): <b>ANY</b> , <b>TCP</b> , <b>UDP</b> , <b>ICMP</b> ,

Table 17: Edit Policy Rules Fields (*continued*)

Field	Description
Source Net	Select the source network for this rule. Choose <b>Local</b> (any network to which this policy is associated), <b>Any</b> (all networks created under the current project) or select from a list of all sources available displayed in the drop-down list, in the form: <i>domain-name:project-name:network-name</i> .
Source Ports	Accept traffic from <b>any</b> port or enter a specific port, a list of ports separated with commas, or a range of ports in the form <i>nnnn-nnnnn</i> .
Destination Net	Select the destination network for this rule. Choose <b>Local</b> (any network to which this policy is associated), <b>Any</b> (all networks created under the current project) or select from a list of all destinations available displayed in the drop-down list, in the form: <i>domain-name:project-name:network-name</i> .
Destination Ports	Send traffic to <b>any</b> port or enter a specific port, a list of ports separated with commas, or a range of ports in the form <i>nnnn-nnnnn</i> .

- When you are finished selecting the rules for this policy, click the **Add Rule** button on the lower right of the **Edit Policy Rules** window.

Next you can associate the policy to a network, see “[Associating a Network to a Policy—OpenStack Contrail](#)” on page 160.

#### Related Documentation

- [Associating a Network to a Policy—OpenStack Contrail](#) on page 160

## Associating a Network to a Policy—OpenStack Contrail

- [Associating Network Policies Overview](#) on page 161
- [Associating a Network Policy to a Network](#) on page 161

## Associating Network Policies Overview

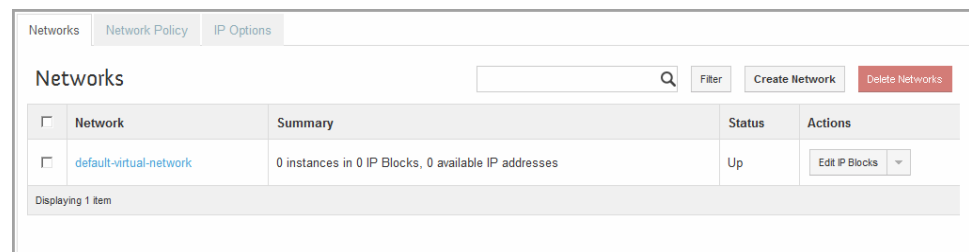
Contrail helps you create and manage virtual networks (VNs). By default, all traffic in a VN is isolated to that VN. Traffic can only leave a VN by means of network policies that are defined for the VN.

This procedure shows how to associate a network policy with a network when using OpenStack.

## Associating a Network Policy to a Network

1. Using the OpenStack Networking module, start at the **Project** tab and click **Networking**. The **Networks** screen appears; see [Figure 34 on page 161](#).

Figure 34: Networks Screen

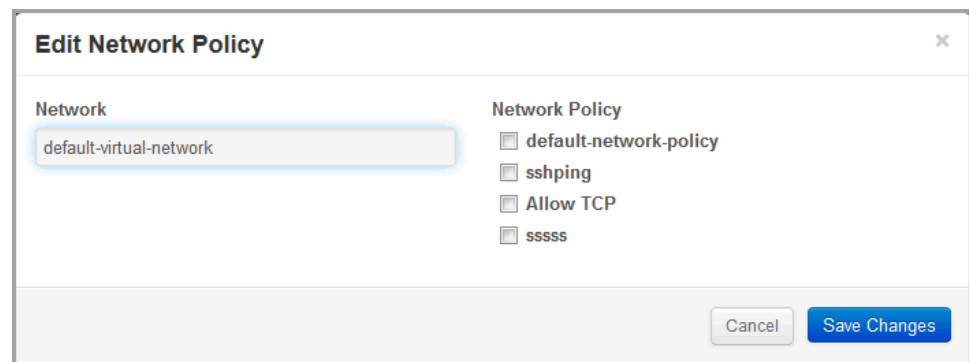


2. Click the check box to select the network you will associate with a policy, then click the drop-down box in the **Actions** column and select **Edit Policy**.

The **Edit Network Policy** window appears; see [Figure 35 on page 161](#).

Available network policies are listed in the **Edit Network Policy** window.

Figure 35: Edit Network Policy



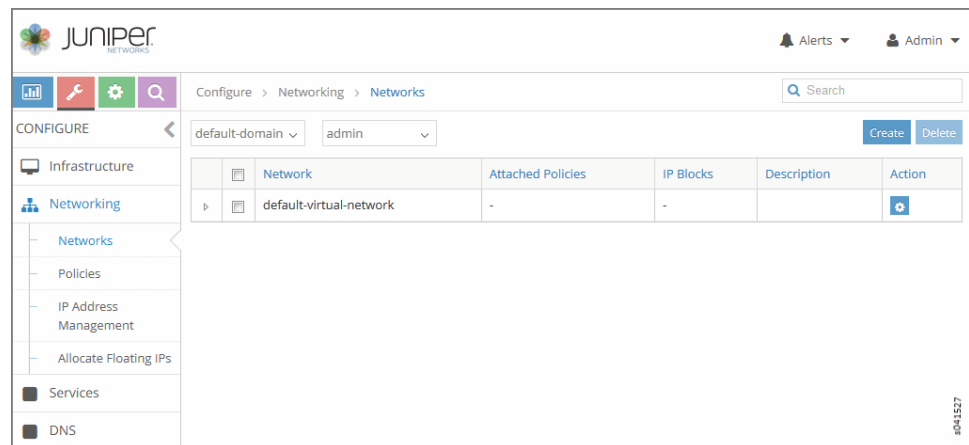
3. Click the check box of any policies to be associated with the selected network.
4. When finished, click the **Save Changes** button.

## Creating a Floating IP Address Pool

A floating IP address is an IP address (typically public) that can be dynamically assigned to a running virtual instance. You can configure floating IP address pools in project networks in Contrail, then allocate floating IP addresses from the pool to virtual machine instances in other virtual networks.

1. Start at **Configure > Networking > Networks**; see [Figure 36 on page 162](#). Make sure your project is the active project in the upper right.

**Figure 36: Configure > Networking > Networks**



2. Click the network you will 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 appears; see [Figure 37 on page 163](#).



Figure 37: Edit Network

Edit Network public

Network Name: public

Network Policy(s): Select Policies...

Address Management: default-network...  + -

IPAM	IP Block
default-network-ipam	10.84.41.0/24

Floating IP Pools: public\_pool  + -

Pool Name
admin

Cancel Save

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

**Related Documentation**

- [Allocating a Floating IP Address to a Virtual Machine on page 164](#)

## Allocating a Floating IP Address to a Virtual Machine

If you have configured a floating IP address pool, you can use the following procedure to allocate the pool to a VM instance.

1. In the Contrail controller module, start at **Configure > Networking > Allocate Floating IPs**.

Make sure your project is displayed (active) in the upper right. Click to select the virtual network that has the floating IP pool; see [Figure 38 on page 164](#).

**Figure 38: Allocate Floating IPs**

IP Address	Instance	Floating IP and Pool	UUID	Action
172.27.56.124				

2. In **Allocated Floating IPs**, click the **Allocate** button.

The **Allocate Floating IP** window appears; see [Figure 39 on page 164](#).

**Figure 39: Allocate Floating IP**

3. Select from a drop-down list the name of the floating IP pool. The floating IP pool is shared among multiple projects. Click **Save**.
4. Once the floating IP pool has been allocated, you can associate or disassociate it from instance addresses. In **Allocate Floating IPs**, click to select the floating IP pool you want to use, then in the **Actions** column, click and select the **Associate** option.

The **Associate Floating IP to Instance** window appears; see [Figure 40 on page 165](#)

Figure 40: Associate Floating IP to Instance

5. In the **Instance** field, select from a drop-down list the UUID of the VM instance to associate with the selected floating IP, and click **Save** to finish.

**Related Documentation**

- [Creating a Floating IP Address Pool on page 162](#)

## Using Security Groups with Virtual Machines (Instances)

- [Security Groups Overview on page 165](#)
- [Creating Security Groups and Adding Rules on page 165](#)

### 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 will be 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

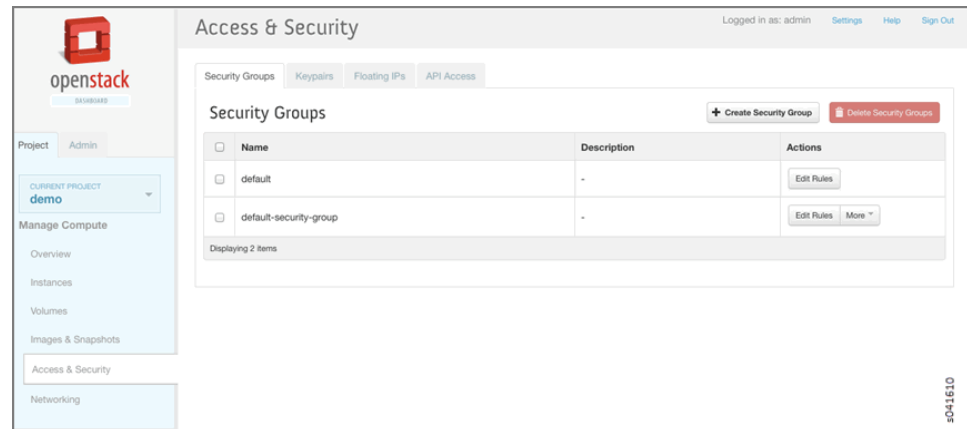
A default security group is created for each project. You can add security rules to the default security group and you can create additional security groups and add rules to them. The security groups are then associated with a VM, when the VM is launched or at a later date.

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 41 on page 166](#).

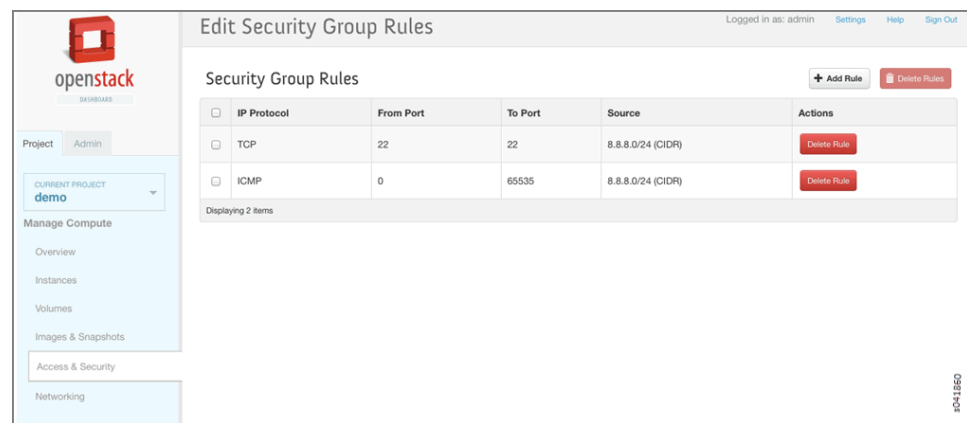
**Figure 41: Security Groups**



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

The **Edit Security Group Rules** screen appears; see [Figure 42 on page 166](#). Any rules already associated with the security group are listed.

**Figure 42: Edit Security Group Rules**



3. Click the **Add Rule** button to add a new rule; see [Figure 43 on page 167](#).

Figure 43: 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

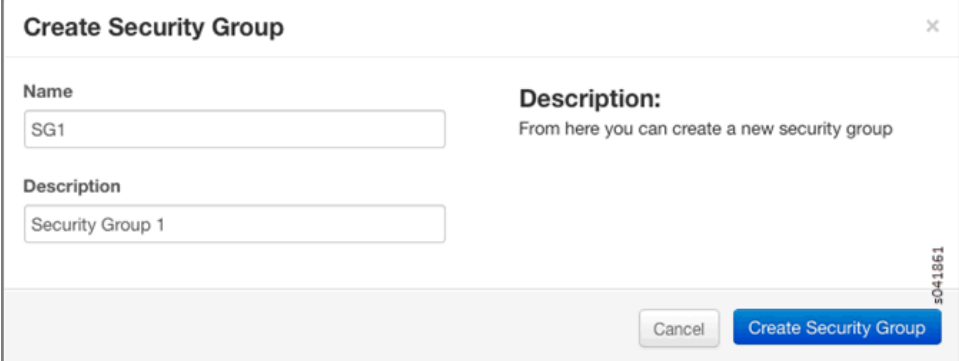
Add

Table 18: 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 the **Create Security Group** button to create additional security groups.
- The **Create Security Group** window appears; see [Figure 44 on page 168](#).
- Each new security group has a unique 32-bit security group ID and an ACL is associated with the configured rules.

Figure 44: Create Security Group



**Create Security Group**

**Name**

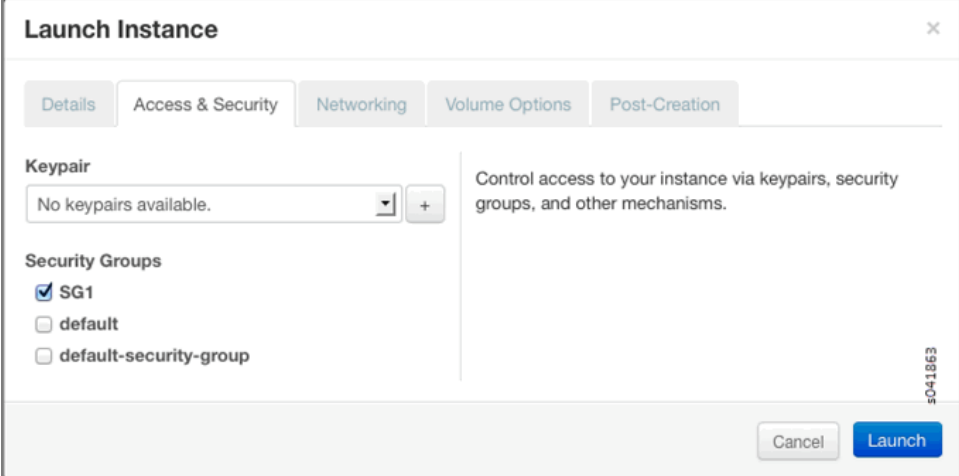
**Description**

**Description:**  
 From here you can create a new security group

5. When an instance is launched, there is an opportunity to associate a security group; see [Figure 45 on page 168](#).

In the **Security Groups** list, click the check box next to a security group name to associate with the instance.

Figure 45: Associate Security Group at Launch Instance



**Launch Instance**

**Access & Security**

**Keypair**

**Security Groups**

☒ SG1  
☐ default  
☐ default-security-group

Control access to your instance via keypairs, security groups, and other mechanisms.

6. You can verify that security groups are attached by viewing the **SgListReq** and **IntfReq** associated with the **agent.xml**.

## Support for IPv6 Networks in Contrail

As of Contrail Release 2.0, support for IPv6 overlay networks is provided.

- [Overview: IPv6 Networks in Contrail on page 169](#)
- [Creating IPv6 Virtual Networks in Contrail on page 169](#)
- [Adding IPv6 Peers on page 170](#)

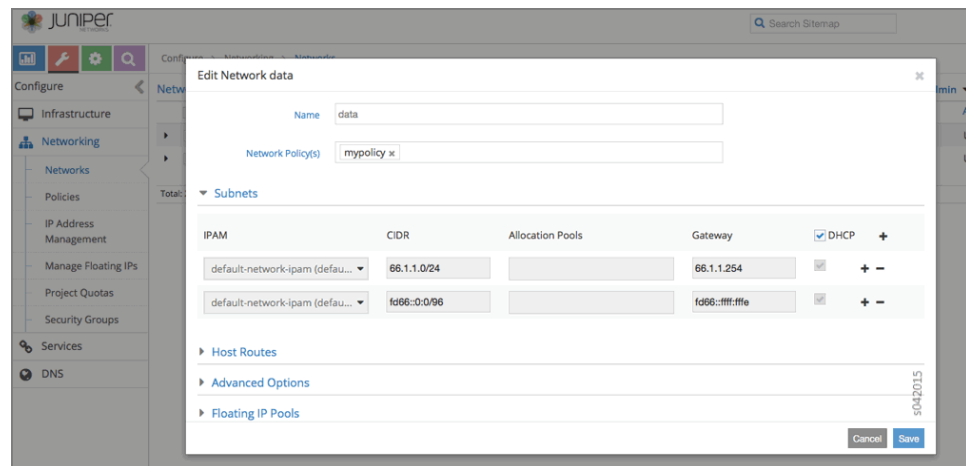
## Overview: IPv6 Networks in Contrail

As of Contrail Release 2.0, support for IPv6 overlay networks is provided, including:

- Configuring IPv6 subnets from the Contrail user interface or by using Neutron APIs
- IPv6 address assignment to virtual machine interfaces over DHCPv6
- IPv6 forwarding in overlay networks between virtual machines, and between virtual machines and BGP peers
- IPv6 to-VPN peering with other BGP peers
- IPv6 forwarding in Layer 2-only networks
- IPv6 interface static routes

## Creating IPv6 Virtual Networks in Contrail


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



openstack  
accelerate

Project Admin

Current Project: **admin**

Manage Compute

Overview

Instances

Volumes

Images & Snapshots

Access & Security

Other

Routers

Network Topology

Load Balancers

Networking

Instances

Logged in as: admin [Settings](#) [Help](#) [Sign Out](#)

Filter  [Filter](#) [+ Launch Instance](#) [+ Edit Network Instance](#) [+ Terminate Instance](#)

	Instance Name	Image Name	IP Address	Size	Keypair	Status	Task	Power State	Uptime	Actions
<input type="checkbox"/>	Test-6d8a3261-ac9d-41fe-8659-bcd89d78e63	ubuntu-jd4f	data 66.1.1.251 f060-###8f7 vn-jd4f 76.1.1.252	m1.medium   4GB RAM   2 VCPU   40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<a href="#">Create Snapshot</a> <a href="#">More</a>
<input type="checkbox"/>	Test-7a767c0b-e6a5-493d-934d-29079a1ab0ba	ubuntu-jd4f	data 66.1.1.250 f060-###8f7 vn-jd4f 76.1.1.250	m1.medium   4GB RAM   2 VCPU   40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<a href="#">Create Snapshot</a> <a href="#">More</a>
<input type="checkbox"/>	Test-663309b7-1765-4c04-9edc-90256cd4ee65	ubuntu-jd4f	data 66.1.1.245 f060-###8f7 vn-jd4f 76.1.1.244	m1.medium   4GB RAM   2 VCPU   40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<a href="#">Create Snapshot</a> <a href="#">More</a>
<input type="checkbox"/>	Test-a03bdc07-3c20-447e-8854-d794eead20ab	ubuntu-jd4f	data 66.1.1.252 f060-###8f7 vn-jd4f 76.1.1.251	m1.medium   4GB RAM   2 VCPU   40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<a href="#">Create Snapshot</a> <a href="#">More</a>
<input type="checkbox"/>	Test-43345608-456f-47e6-934d-5cd185ac1917	ubuntu-jd4f	data 66.1.1.247 f060-###8f7 vn-jd4f 76.1.1.247	m1.medium   4GB RAM   2 VCPU   40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<a href="#">Create Snapshot</a> <a href="#">More</a>

## 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 `/etc/network/interfaces`:

```
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. At **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.

The screenshot shows the Juniper Contrail configuration interface. A modal dialog titled 'Edit BGP Peer' is open. The dialog contains the following fields and values:

- Hostname: MX
- IP Address: 10.84.18.252
- Router ID: 10.84.18.252
- Autonomous System: 64512
- Address Family: inet-vpn, inet6-vpn, route-target, e-vpn
- Hold Time: 90
- BGP Port: 179
- Peer Type: BGP Peer (selected)
- Vendor ID: juniper

At the bottom of the dialog, there are buttons for 'Cancel' and 'Save'. The background shows the 'Configure > Infrastructure > BGP Peers' navigation path.



**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 addresses learned, which cause churn at node failures. EVPN is designed to address these issues without disturbing flat MAC connectivity.

In EVPN, 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 EVPN*

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 are all handled the same way, and get 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*

With Contrail Release 1.03, EVPN is enabled by default. The supported forwarding modes include:

- Fallback bridging—IPv4 traffic lookup is performed using IP FIB. All non-IPv4 traffic is directed to a MAC FIB.
- Layer 2-only— All traffic is forwarded using MAC FIB lookup.

The forwarding mode can be configured individually on each virtual network.

EVPN is used to share MACs across different control planes in both forwarding models. The result of a MAC 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 Forwarding on page 173](#)
- [Configuring the VXLAN Identifier Mode on page 174](#)
- [Configuring the VXLAN Identifier on page 175](#)
- [Configuring Encapsulation Methods on page 175](#)

## Configuring Forwarding

With Contrail 1.03, 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.

From the Contrail web UI, select the virtual network for which you will change the forwarding mode, select **Edit Network**, then select **Advanced Options** to change the forwarding mode. In the following, the network named **vn** is being edited. Under **Advanced Options->Forwarding Mode**, two options are available:

- Select **L2 and L3** to enable IP and MAC FIB (fallback bridging).
- Select **L2** to enable only MAC FIB.

The screenshot shows the 'Edit Network vn' interface. At the top, there is a table with three columns: IPAM, IP Block, and Gateway. The first row shows 'default', '1.1.1.0/24', and '1.1.1.254'. Below this table are several expandable sections: 'Route Targets', 'Floating IP Pools', 'Host Routes', and 'Advanced Options'. The 'Advanced Options' section is expanded, showing 'Forwarding Mode' set to 'L2 Only' and 'VxLAN Identifier' set to 'L2 Only'. At the bottom right, there are 'Cancel' and 'Save' buttons.

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 Mode

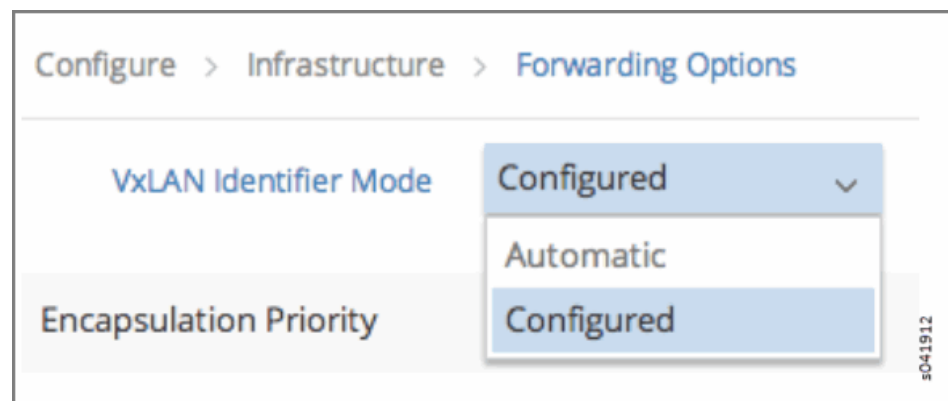
The VXLAN identifier mode can be configured to select an auto-generated VNID or a user-generated VXLAN ID, either through the Contrail web UI or by modifying a python file.

The following Contrail web UI shows the location for configuring the VXLAN identifier mode at **Configure > Infrastructure > Forwarding Options**. The user can select one of these options:

- **Automatic**— The VXLAN identifier is automatically assigned for the virtual network.
- **Configured**— The VXLAN identifier must be provided by the user for the virtual network.



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



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

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

## Configuring the VXLAN Identifier

The VXLAN identifier can be set only if the VXLAN network identifier mode has been set to configured. You can then set the VXLAN ID either by using the Contrail web UI or by using python commands.

The following shows the web UI location of the VXLAN identifier field in **Edit Network, Advanced Options**.

The screenshot shows the 'Edit Network vn' window. At the top, there is a table with three columns: IPAM, IP Block, and Gateway. The first row has values 'default', '1.1.1.0/24', and '1.1.1.254'. Below the table are expandable sections: 'Route Targets', 'Floating IP Pools', 'Host Routes', and 'Advanced Options'. The 'Advanced Options' section is expanded, showing 'Forwarding Mode' set to 'L2 Only' and 'VxLAN Identifier' set to '101'. At the bottom right, there are 'Cancel' and 'Save' buttons.

IPAM	IP Block	Gateway
default	1.1.1.0/24	1.1.1.254

Route Targets

Floating IP Pools

Host Routes

Advanced Options

Forwarding Mode: L2 Only

VxLAN Identifier: 101

Cancel 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 to 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 web UI or in the file `encap.py`.

The following shows the web UI location for setting the encapsulation method priorities from **Configure > Infrastructure > Forwarding Options**, in which the user has selected the priority order to be VXLAN first, then MPLS over GRE, and last priority is MPLS over UDP.

Use the following procedure to change the default encapsulation method to VXLAN.



**NOTE:** VXLAN is *only* supported for EVPN unicast. It is not supported for IP traffic or multicast traffic. VXLAN priority and presence in `encap.py` or configured in web UI is ignored for traffic not supported by VXLAN.

To set the priority of encapsulation methods to VXLAN:

1. Modify the file `encap.py` found in `/opt/contrail/utils/`.

The default encapsulation line is:

```
encap_obj=EncapsulationPrioritiesType(encapsulation=['MPLSoUDP','MPLSoGRE'])
```

Modify the line to:

```
encap_obj=EncapsulationPrioritiesType(encapsulation=['VXLAN',
'MPLSoUDP','MPLSoGRE'])
```

2. Once 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.

## Configuring Network QoS Parameters

- [Overview on page 177](#)
- [QoS Configuration Examples on page 177](#)
- [Limitations on page 178](#)

### Overview

You can use the OpenStack Nova command-line interface (CLI) to specify a quality of service (QoS) setting for a virtual machine's network interface, by setting the **quota** of a Nova flavor. Any virtual machine created with that Nova flavor will inherit all of the specified QoS settings. Additionally, if the virtual machine that was created with the QoS settings has multiple interfaces in different virtual networks, the same QoS settings will be applied to all of the network interfaces associated with the virtual machine. The QoS settings can be specified in unidirectional or bidirectional mode.

The **quota** driver in Neutron converts QoS parameters into **libvirt** network settings of the virtual machine.

The QoS parameters available in the quota driver only cover rate limiting the network interface. There are no specifications available for policy-based QoS at this time.

### QoS Configuration Examples

Although the QoS setting can be specified in quota by using either Horizon or CLI, quota creation using CLI is more robust and stable, therefore, creating by CLI is the recommended method.

**Example** CLI for Nova flavor has the following format:

```
nova flavor-key <flavor_name> set quota:vif_<direction>_<param_name> = value
```

where:

**<flavor\_name>** is the name of an existing Nova flavor.

**vif\_<direction>\_<param\_name>** is the inbound or outbound QoS data name.

QoS **vif** types include the following:

- **vif\_inbound\_average** lets you specify the average rate of inbound (receive) traffic, in kilobytes/sec.
- **vif\_outbound\_average** lets you specify the average rate of outbound (transmit) traffic, in kilobytes/sec.

- Optional: **vif\_inbound\_peak** and **vif\_outbound\_peak** specify the maximum rate of inbound and outbound traffic, respectively, in kilobytes/sec.
- Optional: **vif\_inbound\_burst** and **vif\_outbound\_peak** specify the amount of kilobytes that can be received or transmitted, respectively, in a single burst at the peak rate.

Details for various QoS parameters for **libvirt** can be found at <http://libvirt.org/formatnetwork.html>.

The following example shows an inbound average of 800 kilobytes/sec, a peak of 1000 kilobytes/sec, and a burst amount of 30 kilobytes.

```
nova flavor-key m1.small set quota:vif_inbound_average=800
nova flavor-key m1.small set quota:vif_inbound_peak=1000
nova flavor-key m1.small set quota:vif_inbound_burst=30
```

The following is an example of specified outbound parameters:

```
nova flavor-key m1.small set quota:vif_outbound_average=800
nova flavor-key m1.small set quota:vif_outbound_peak=1000
nova flavor-key m1.small set quota:vif_outbound_burst=30
```

After the Nova flavor is configured for QoS, a virtual machine instance can be created, using either Horizon or CLI. The instance will have network settings corresponding to the nova flavor-key, as in the following:

```
<interface type="ethernet">
  <mac address="02:a3:a0:87:7f:61"/>
  <model type="virtio"/>
  <script path=""/>
  <target dev="tapa3a0877f-61"/>
  <bandwidth>
    <inbound average="800" peak="1000" burst="30"/>
    <outbound average="800" peak="1000" burst="30"/>
  </bandwidth>
</interface>
```

## Limitations

- The stock **libvirt** does not support rate limiting of **ethernet** interface types. Consequently, settings like those in the example for the guest interface will not result in any **tc qdisc** settings for the corresponding tap device in the host. For more details, refer to issue [#1367095](#) in [Launchpad.net](#), where you can find patches and instructions to make **libvirt** work for network rate limiting of virtual machine interfaces.
- The **nova flavor-key rxtx\_factor** takes a float as an input and acts as a scaling factor for receive (inbound) and transmit (outbound) throughputs. This key is only available to Neutron extensions (private extensions). The Contrail Neutron plugin doesn't implement this private extension. Consequently, setting the **nova flavor-key rxtx\_factor** will not have any effect on the QoS setting of the network interface(s) of any virtual machine created with that nova flavor.
- The outbound rate limits of a virtual machine interface are not strictly achieved. The outbound throughput of a virtual machine network interface is always less than the



average outbound limit specified in the virtual machine's libvirt configuration file. The same behavior is also seen when using a Linux bridge.



## CHAPTER 5

# Examples of Tiered Web Configurations

- [Example: Deploying a Multi-Tier Web Application on page 181](#)
- [Sample Network Configuration for Devices for Simple Tiered Web Application on page 187](#)

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

---

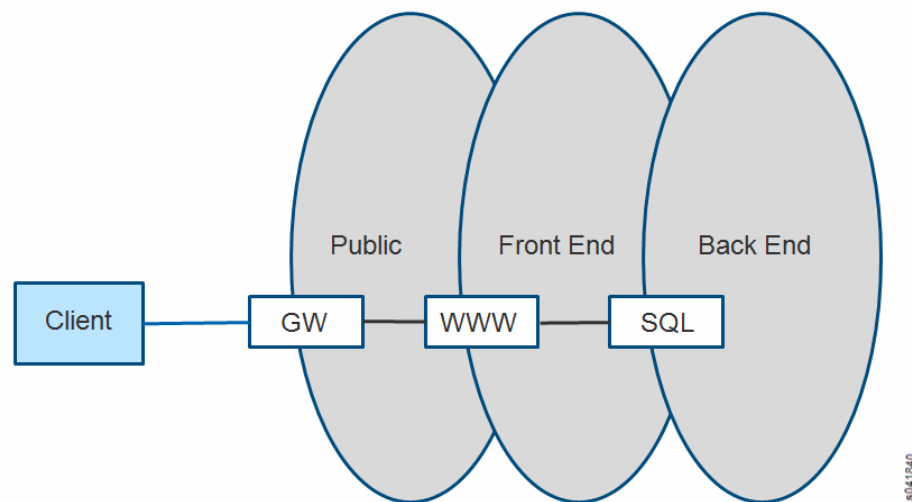
- [Multi-Tier Web Application Overview on page 181](#)
- [Example: Setting Up Virtual Networks for a Simple Tiered Web Application on page 182](#)
- [Verifying the Multi-Tier Web Application on page 184](#)
- [Sample Addressing Scheme for Simple Tiered Web Application on page 184](#)
- [Sample Physical Topology for Simple Tiered Web Application on page 185](#)
- [Sample Physical Topology Addressing on page 186](#)

### 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 46 on page 182](#).

Figure 46: 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 138](#).

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—OpenStack Contrail” on page 139 or *Creating a Virtual Network—Juniper Networks Contrail*.

3. Create a floating IP pool named **public\_pool** for the **public** network within the **demo** project; see Figure 47 on page 183.

Figure 47: Create Floating IP Pool

The screenshot shows the 'Edit Network public' dialog box. It has a title bar with a close button. The main content area is divided into several sections:

- Network Name:** A text field containing 'public'.
- Network Policy(s):** A button labeled 'Select Policies...'.
- Address Management:** A dropdown menu showing 'default-network...' and a text field containing 'xxx.xxx.xxx.xxx/xx'. To the right are '+' and '-' buttons.
- IPAM:** A table with two columns: 'IPAM' and 'IP Block'. The first row contains 'default-network-ipam' and '10.84.41.0/24'.
- Floating IP Pools:** A text field containing 'public\_pool' and a dropdown menu showing 'demo'. To the right are '+' and '-' buttons.
- Pool Name:** A text field containing 'admin'.

At the bottom right, there are 'Cancel' and 'Save' buttons. A small vertical label 's041841' is visible on the right side of the dialog.

4. Allocate the floating IP pool **public\_pool** to the **demo** project; see Figure 48 on page 183.

Figure 48: Allocate Floating IP

The screenshot shows the 'Allocate Floating IP' dialog box. It has a title bar with a close button. The main content area is divided into two sections:

- Floating IP Pool:** A dropdown menu showing 'public:public\_pool'.

At the bottom right, there are 'Cancel' and 'Save' buttons. A small vertical label 's041842' is visible on the right side of the dialog.

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 162 and “Allocating a Floating IP Address to a Virtual Machine” on page 164.

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” on page 151](#), [“Associating a Network to a Policy—Juniper Networks Contrail” on page 153](#), or [“Creating a Network Policy—OpenStack Contrail” on page 157](#), and [“Associating a Network to a Policy—OpenStack Contrail” on page 160](#).

7. Launch the virtual machine instances that represent the web server and the database server.



**NOTE:** Your installation might not include the virtual machines needed for the web server and the database server. Contact your account team if you need to download the VMs for this setup.

On the **Instances** tab for this project, select **Launch Instance** and for each instance that you launch, complete the fields to make the following associations:

- Web server VM: select **frontend** network and the policy created to allow communication between **frontend** and **backend** networks. Apply the floating IP address pool to the web server.
- Database server VM: select **backend** network and the policy created to allow communication between **frontend** and **backend** networks.

For more information; see [“Launching a Virtual Machine \(Instance\)” on page 148](#).

## 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 19 on page 185](#) as a guide for addressing devices in the simple tiered web example.

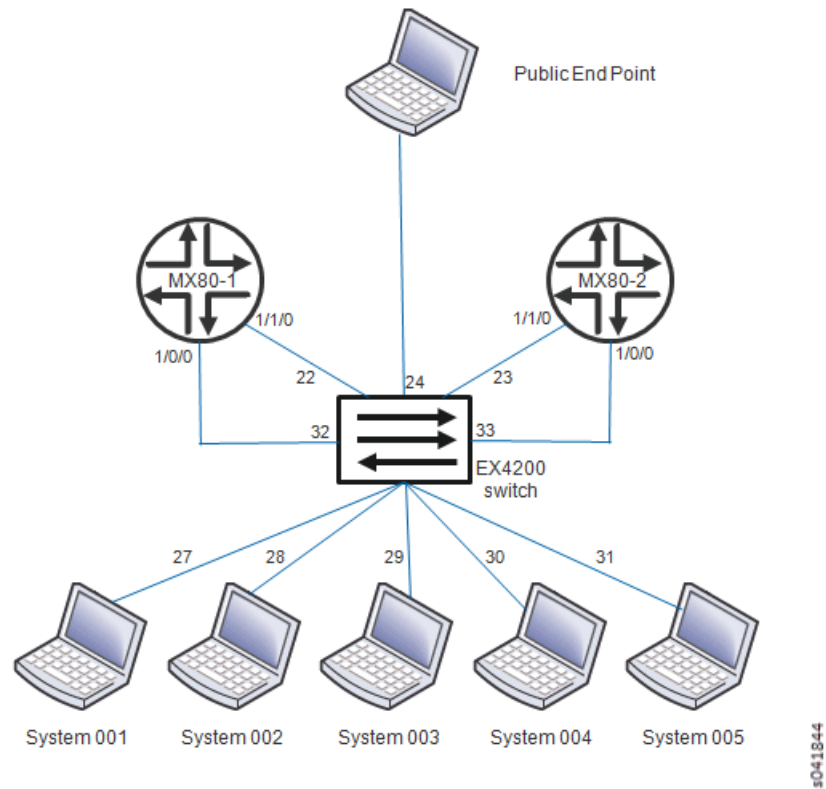
Table 19: Sample Addressing Scheme for Example

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 49 on page 186](#) provides a guideline diagram for the physical topology for the simple tiered web application example.

Figure 49: Sample Physical Topology for Simple Tiered Web Application

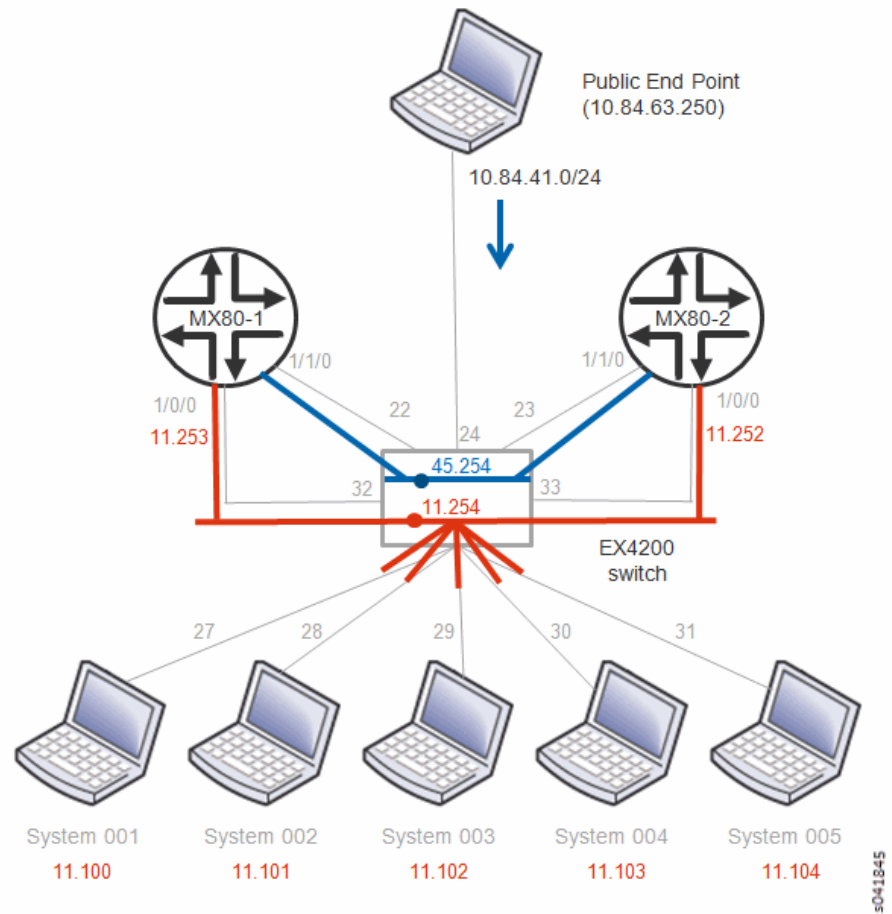


### Sample Physical Topology Addressing

Figure 50 on page 187 provides a guideline diagram for addressing the physical topology for the simple tiered web application example.



Figure 50: 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 181. Configurations are shown for Juniper Networks devices: two MX80s and one EX4200.

### MX80-1 Configuration

```
version 12.2R1.3;
system {
  root-authentication {
    encrypted-password "xxxxxxxxx"; ## SECRET-DATA
  }
  services {
    ssh {
      root-login allow;
    }
  }
  syslog {
    user * {
```

```
        any emergency;
    }
    file messages {
        any notice;
        authorization info;
    }
}
chassis {
    fpc 1 {
        pic 0 {
            tunnel-services;
        }
    }
}
interfaces {
    ge-1/0/0 {
        unit 0 {
            family inet {
                address 10.84.11.253/24;
            }
        }
    }
    ge-1/1/0 {
        description "IP Fabric interface";
        unit 0 {
            family inet {
                address 10.84.45.1/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 127.0.0.1/32;
            }
        }
    }
}
routing-options {
    static {
        route 0.0.0.0/0 next-hop 10.84.45.254;
    }
    route-distinguisher-id 10.84.11.253;
    autonomous-system 64512;
    dynamic-tunnels {
        setup1 {
            source-address 10.84.11.253;
            gre;
            destination-networks {
                10.84.11.0/24;
            }
        }
    }
}
protocols {
```

```

bgp {
  group mx {
    type internal;
    local-address 10.84.11.253;
    family inet-vpn {
      unicast;
    }
    neighbor 10.84.11.252;
  }
  group contrail-controller {
    type internal;
    local-address 10.84.11.253;
    family inet-vpn {
      unicast;
    }
    neighbor 10.84.11.101;
    neighbor 10.84.11.102;
  }
}
routing-instances {
  customer-public {
    instance-type vrf;
    interface ge-1/1/0.0;
    vrf-target target:64512:10000;
    routing-options {
      static {
        route 0.0.0.0/0 next-hop 10.84.45.254;
      }
    }
  }
}
}

```

#### *MX80-2 Configuration*

```

version 12.2R1.3;
system {
  root-authentication {
    encrypted-password "xxxxxxxxx"; ## SECRET-DATA
  }
  services {
    ssh {
      root-login allow;
    }
  }
  syslog {
    user * {
      any emergency;
    }
    file messages {
      any notice;
      authorization info;
    }
  }
}
chassis {

```

```
fpc 1 {  
  pic 0 {  
    tunnel-services;  
  }  
}  
}  
interfaces {  
  ge-1/0/0 {  
    unit 0 {  
      family inet {  
        address 10.84.11.252/24;  
      }  
    }  
  }  
  ge-1/1/0 {  
    description "IP Fabric interface";  
    unit 0 {  
      family inet {  
        address 10.84.45.2/24;  
      }  
    }  
  }  
  lo0 {  
    unit 0 {  
      family inet {  
        address 127.0.0.1/32;  
      }  
    }  
  }  
}  
routing-options {  
  static {  
    route 0.0.0.0/0 next-hop 10.84.45.254;  
  }  
  route-distinguisher-id 10.84.11.252;  
  autonomous-system 64512;  
  dynamic-tunnels {  
    setup1 {  
      source-address 10.84.11.252;  
      gre;  
      destination-networks {  
        10.84.11.0/24;  
      }  
    }  
  }  
}  
protocols {  
  bgp {  
    group mx {  
      type internal;  
      local-address 10.84.11.252;  
      family inet-vpn {  
        unicast;  
      }  
      neighbor 10.84.11.253;  
    }  
  }  
}
```

```

    group contrail-controller {
        type internal;
        local-address 10.84.11.252;
        family inet-vpn {
            unicast;
        }
        neighbor 10.84.11.101;
        neighbor 10.84.11.102;
    }
}
routing-instances {
    customer-public {
        instance-type vrf;
        interface ge-1/1/0.0;
        vrf-target target:64512:10000;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 10.84.45.254;
            }
        }
    }
}
}

```

#### *EX4200 Configuration*

```

system {
    host-name EX4200;
    time-zone America/Los_Angeles;
    root-authentication {
        encrypted-password "xxxxxxxxxxxx"; ## SECRET-DATA
    }
    login {
        class read {
            permissions [ clear interface view view-configuration ];
        }
        user admin {
            uid 2000;
            class super-user;
            authentication {
                encrypted-password "xxxxxxxxxxxx"; ## SECRET-DATA
            }
        }
        user regress {
            uid 2002;
            class read;
            authentication {
                encrypted-password "xxxxxxxxxxxx"; ## SECRET-DATA
            }
        }
    }
}
services {
    ssh {
        root-login allow;
    }
    telnet;
}

```

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

## CHAPTER 6

# High Availability

- [Juniper OpenStack High Availability on page 193](#)

## Juniper OpenStack High Availability

---

- [Introduction on page 193](#)
- [Contrail High Availability on page 194](#)
- [OpenStack High Availability on page 194](#)
- [Supported Platforms on page 194](#)
- [Juniper OpenStack High Availability Architecture on page 194](#)
- [Juniper OpenStack Objectives on page 195](#)
- [Limitations on page 195](#)
- [Solution Components on page 196](#)
- [Virtual IP with Load Balancing on page 196](#)
- [Failure Handling on page 196](#)
- [Deployment on page 197](#)
- [Minimum Hardware Requirement on page 197](#)
- [Compute on page 197](#)
- [Network on page 198](#)
- [Installation on page 198](#)
- [Testbed File for Fab on page 199](#)
- [Using Headless vRouter to Improve Redundancy on page 202](#)
- [Configuring the Headless vRouter on page 202](#)

## Introduction

The Juniper Networks software-defined network (SDN) controller has two major components: OpenStack and Contrail. High availability (HA) of the controller requires that both OpenStack and Contrail are resistant to failures. Failures can range from a service instance failure, node failure, link failure, to all nodes down due to a power outage. The basic expectation from a highly available SDN controller is that when failures occur, already provisioned workloads continue to work as expected without any traffic drop, and the controller is available to perform operations on the cluster. Juniper Networks

OpenStack is a distribution from Juniper Networks that combines OpenStack and Contrail into one product.

### Contrail High Availability

Contrail has high availability already built into various components, including support for the Active-Active model of high availability, which works by deploying the Contrail node component with an appropriate required level of redundancy.

The Contrail control node runs BGP and maintains adjacency with the vRouter module in the compute nodes. Additionally, every vRouter maintains a connection with all available control nodes.

Contrail uses Cassandra as the database. Cassandra inherently supports fault tolerance and replicates data across the nodes participating in the cluster. A highly available deployment of Contrail requires at least two control nodes, three config nodes (including analytics and webui) and three database nodes.

### OpenStack High Availability

High availability of OpenStack is supported by deploying the OpenStack controller nodes in a redundant manner on multiple nodes. Previous releases of Contrail supported only a single instance of the OpenStack controller, and multiple instances of OpenStack posed new problems that needed to be solved, including: State synchronization of stateful services (e.g. MySQL) across multiple instances. Load-balancing of requests across the multiple instances of services. As of Contrail Release 1.10, the existing Active-Active high availability solution has been extended to include the OpenStack controller as well.

### Supported Platforms

Juniper OpenStack Controller has tested high availability on the following platforms:

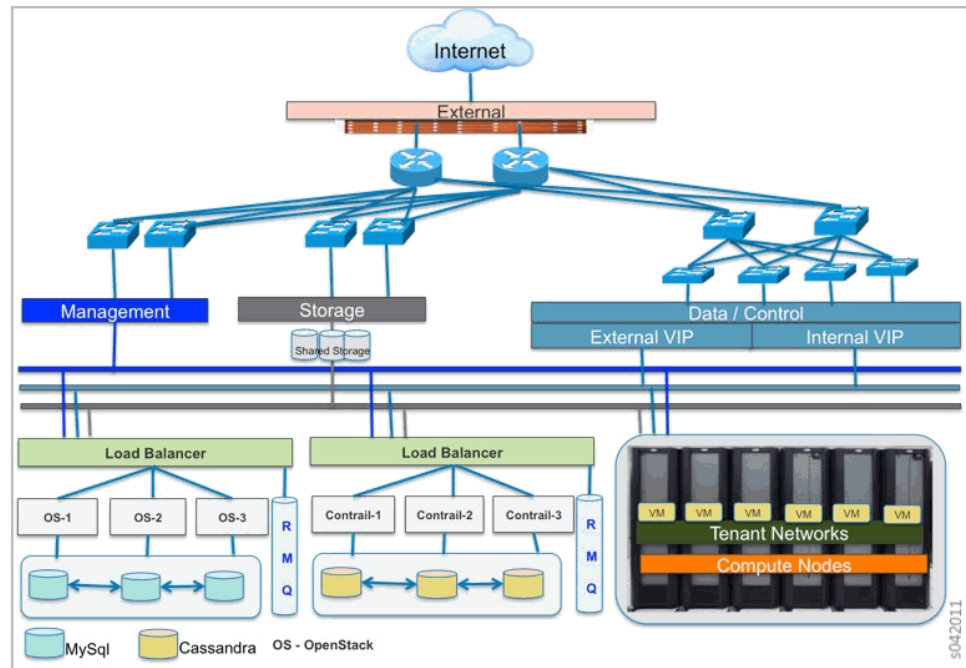
- Linux - Ubuntu 12.04 with kernel version 3.13.0-34
- OpenStack Havana

### Juniper OpenStack High Availability Architecture

A typical cloud infrastructure deployment consists of a pool of resources of compute, storage, and networking infrastructure, all managed by a cluster of controller nodes.



The following figure illustrates a high-level reference architecture of a high availability deployment using Juniper OpenStack deployed as a cluster of controller nodes.



## Juniper OpenStack Objectives

The main objectives and requirements for Juniper OpenStack high availability are:

- 99.999% availability for tenant traffic.
- Anytime availability for cloud operations.
- Provide VIP-based access to the API and UI services.
- Load balance network operations across the cluster.
- Management and orchestration elasticity.
- Failure detection and recovery.

## Limitations

The following are limitations of Juniper OpenStack high availability:

- Only one failure is supported.
- During failover, a REST API call may fail. The application or user must reattempt the call.
- Although zero packet drop is the objective, in a distributed system such as Contrail, a few packets may drop during ungraceful failures.
- Juniper OpenStack high availability is not tested with any third party load balancing solution other than HAProxy.

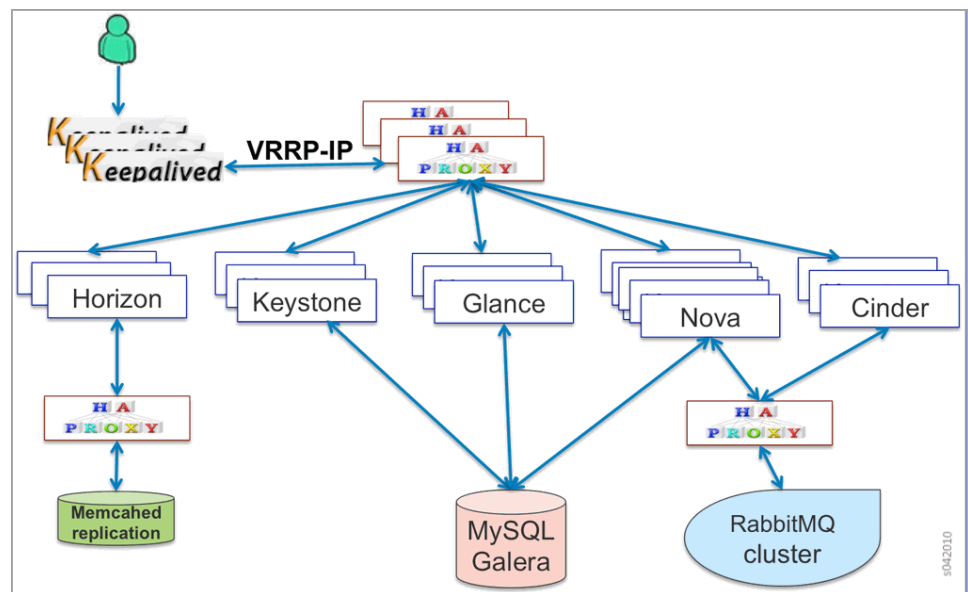
## Solution Components

Juniper Openstack's high availability active-active model provides scale out of the infrastructure and orchestration services. The model makes it very easy to introduce new services in the controller and in the orchestration layer.

### Virtual IP with Load Balancing

HAProxy is run on all nodes to load balance the connections across multiple instances of the services. To provide a Virtual IP (VIP), Keepalived (open source health check framework and hot standby protocol) runs and elects a master based on VRRP protocol. The VRRP master owns the VIP. If the master node fails, the VIP moves to a new master elected by VRRP.

The following figure shows OpenStack services provisioned to work with HAProxy and Keepalived, with HAProxy at the front of OpenStack services in a multiple operating system node deployment. The OpenStack database is deployed in clustered mode and uses Galera for replicating data across the cluster. RabbitMQ has clustering enabled as part of a multinode Contrail deployment. The RabbitMQ configuration is further tuned to support high availability.



## Failure Handling

This section describes how various types of failures are handled, including:

- Service failures
- Node failures
- Networking failures

### *Service Failures*

When an instance of a service fails, HAProxy detects the failure and load-balances any subsequent requests across other active instances of the service. The supervisor process monitors for service failures and brings up the failed instances. As long as there is one instance of a service operational, the Juniper OpenStack controller continues to operate. This is true for both stateful and stateless services across Contrail and OpenStack.

#### *Node Failures*

The Juniper OpenStack controller supports single node failures involving both graceful shutdown or reboots and ungraceful power failures. When a node that is the VIP master fails, the VIP moves to the next active node as it is elected to be the VRRP master. HAProxy on the new VIP master sprays the connections over to the active service instances as before, while the failed down node is brought back online. Stateful services (MySQL/Galera, Zookeeper, and so on) require a quorum to be maintained when a node fails. As long as a quorum is maintained, the controller cluster continues to work without problems. Data integrity is also inherently preserved by Galera, Rabbit, and other stateful components in use.

#### *Network Failures*

A connectivity break esp. in the control/data network causes the Controller cluster to partition into two. As long as the caveat around minimum number of nodes is maintained for one of the partitions, Controller cluster continues to work fine. Stateful services including MySQL Galera and RabbitMQ detect the partitioning and reorganize their cluster around the reachable nodes. Existing workloads continue to function and pass traffic and new workloads can be provisioned. When the connectivity is restored, the joining node becomes part of the working cluster and system gets restored to its original state.

## Deployment

### Minimum Hardware Requirement

A minimum of 3 servers (physical or virtual machines) are required to deploy a highly available Juniper OpenStack Controller. In Active-Active mode, the Controller cluster uses Quorum-based consistency management for guaranteeing transaction integrity across its distributed nodes. This translates to the requirement of deploying  $2n+1$  nodes to tolerate  $n$  failures.

Juniper OpenStack Controller offers variety of deployment choices. Depending on the use case, the roles can be deployed either independently or in some combined manner. The type of deployment determines the sizing of the infrastructure. The numbers below present minimum requirement across compute, storage, and network.

### Compute

- Quad core Intel(R) Xeon 2.5 Gz or higher.
- 32 GB or higher RAM for the controller hosts (increases with number of hypervisors being supported)
- Minimum 1 TB disk. SSD/ HDD.

## Network

A typical deployment separates control/data traffic from the management traffic.

- Dual 10 GE that is bonded (using LAG 802.3ad) for redundant control/data connection
- Dual 1 GE bonded (using LAG 802.3 ad) for redundant management connection.
- Single 10G and 1G will also work if link redundancy is not desired.

Need Virtual IP (VIP) addresses carved from the networks the above NICs participate in. External VIP on the management network and Internal VIP on control/data network. External facing services get load-balanced using external VIP and internal VIP is used for communication between other services.

### *Packaging*

High availability support brought in new components to the Contrail OpenStack deployment, which are packaged in a new package called **contrail-openstack-ha**. It primarily contains HAProxy, Keepalived, Galera, and their requisite dependencies.

## Installation

Installation is supported through fabric (fab) scripts. External facing, there is very little change mostly to incorporate multiple OpenStack roles and VIP configuration. `Testbed.py` has new sections to incorporate external & internal VIPs. If OpenStack and Contrail roles are co-located on the nodes, only one set of external and internal VIP is enough.

Install also supports separating OpenStack and Contrail roles on physically different servers. In this case, external and internal VIPs specified are used for OpenStack controller and a separate set of VIPs, `contrail_external_vip` and `contrail_internal_vip` are used for the Contrail controller nodes. It is also possible specify separate RabbitMQ for OpenStack and Contrail controllers.

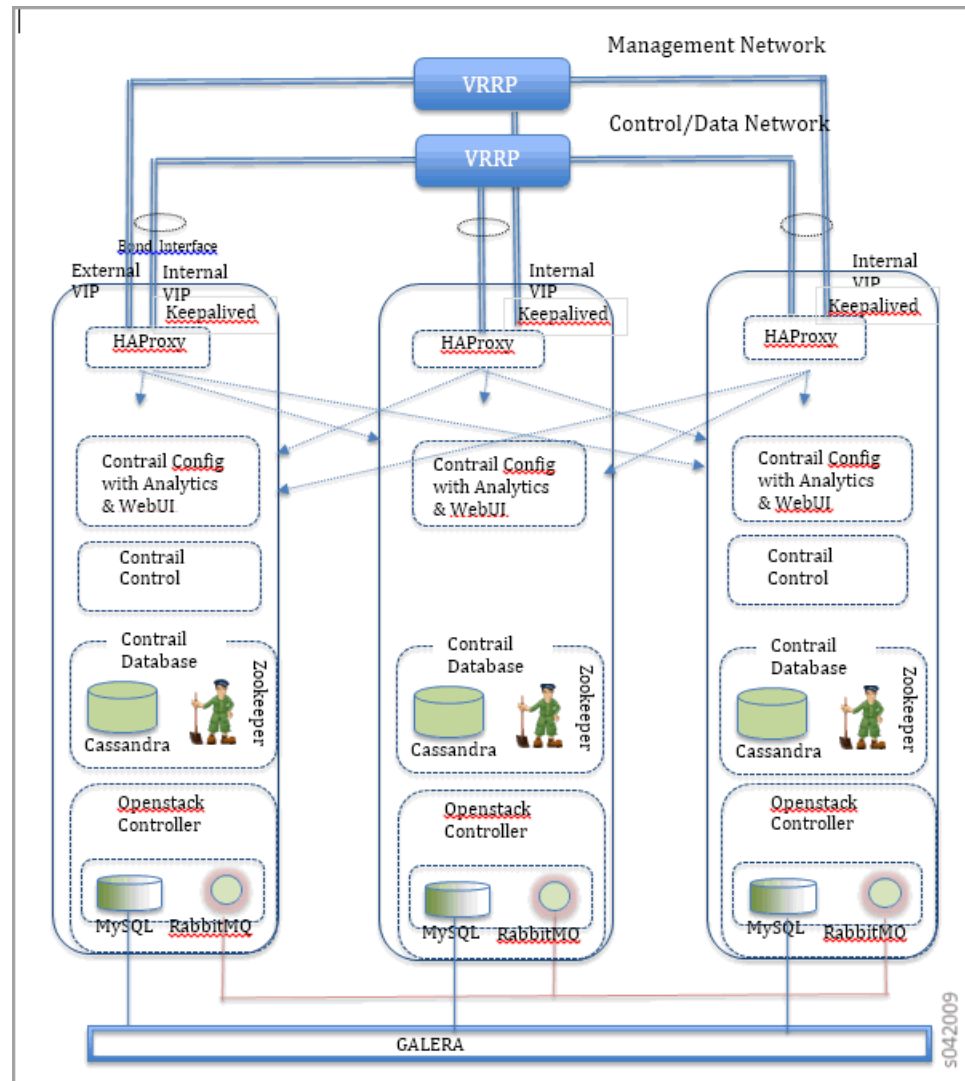
When multiple OpenStack roles are specified along with VIPs, the 'install-contrail' target treats the installation as the High Availability install and additionally adds the new 'contrail-openstack-ha' package.

Similarly, **setup\_all** treats the setup as Contrail High Availability setup and provisions the following services using the respective newly-created fab tasks.

- Keepalived - fab setup\_keepalived
- high availability proxy- fab fixup\_restart\_haproxy\_in\_openstack
- Galera- fab setup\_galera\_cluster, fab fix\_wsrep\_cluster\_address
- Glance- fab setup\_glance\_images\_loc
- Keystone - fab sync\_keystone\_ssl\_certs

Also all the provisioning scripts are changed to use VIPs instead of the physical IP of the node in all OpenStack and Contrail related configuration files. The following figure shows

a typical three node deployment where Openstack and Contrail roles are co-located on three servers.



## Testbed File for Fab

A sample file is available at:

[https://github.com/Juniper/contrail-fabric-utils/blob/R1.10/fabfile/testbeds/testbed\\_multibox\\_example.py](https://github.com/Juniper/contrail-fabric-utils/blob/R1.10/fabfile/testbeds/testbed_multibox_example.py)

You can use the sample file by uncommenting and changing the high availability section to match your deployment.

The contents of the sample testbed.py file for the minimum high availability configuration is the following.

```
from fabric.api import env

#Management ip addresses of hosts in the cluster
```

```
host1 = 'user@<ip address>'
host2 = 'user@<ip address>'
host3 = 'user@<ip address>'
host4 = 'user@<ip address>'
host5 = 'user@<ip address>'

#External routers if any
#for eg.
#ext_routers = [('mx1', 'user@<ip address>')]
ext_routers = [('mx1', 'user@<ip address>')]

public_vn_rtgt = 20000
public_vn_subnet = "user@<ip address>"

#Autonomous system number
router_asn = <asn number>

#Host from which the fab commands are triggered to install and provision
host_build = user@<ip address>'

#Role definition of the hosts.
env.roledefs = {
    'all': [host1, host2, host3, host4, host5],
    'cfgm': [host1, host2, host3],
    'openstack': [host1, host2, host3],
    'control': [host2, host3 ],
    'compute': [host4, host5 ],
    'collector': [host1, host2, host3],
    'webui': [host1, host2, host3],
    'database': [host1, host2, host3],
    'build': [host_build],
}

env.hostnames = {
    'all': ['vse2100-2', 'vse2100-3', 'vse2100-4', 'vse2100-5', 'vse2100-6']
}

#Openstack admin password
env.openstack_admin_password = '<password>'

env.password = '<password>'
#Passwords of each host
env.passwords = {
    host1: '<password>',
    host2: '<password>',
    host3: '<password>',
    host4: '<password>',
    host5: '<password>',
    host_build: '<password>',
}

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

```

    host4: 'ubuntu',
    host5: 'ubuntu',
}

#OPTIONAL BONDING CONFIGURATION
#=====
#Interface Bonding
#OPTIONAL BONDING CONFIGURATION
#=====
#Interface Bonding
bond= {
    host1 : { 'name': 'bond0', 'member': ['eth1','eth2'], 'mode':'802.3ad' },
    host2 : { 'name': 'bond0', 'member': ['eth1','eth2'], 'mode':'802.3ad' },
    host3 : { 'name': 'bond0', 'member': ['eth1','eth2'], 'mode':'802.3ad' },
    host4 : { 'name': 'bond0', 'member': ['eth1','eth2'], 'mode':'802.3ad' },
    host5 : { 'name': 'bond0', 'member': ['eth1','eth2'], 'mode':'802.3ad' },
}

#OPTIONAL SEPARATION OF MANAGEMENT AND CONTROL + DATA
#=====
#Control Interface
control_data = {
    host1 : { 'ip': '<password>', 'gw' : '<password>', 'device':'bond0' },
    host2 : { 'ip': '<password>', 'gw' : '<password>', 'device':'bond0' },
    host3 : { 'ip': '<password>', 'gw' : '<password>', 'device':'bond0' },
    host4 : { 'ip': '<password>', 'gw' : '<password>', 'device':'bond0' },
    host5 : { 'ip': '<password>', 'gw' : '<password>', 'device':'bond0' },
}

# VIP
env.ha = {
    'internal_vip' : '<password>',
    'external_vip' : '<password>'
}

#To disable installing contrail interface rename package
env.interface_rename = False

#To enable multi-tenancy feature
#multi_tenancy = True

#To Enable parallel execution of task in multiple nodes
do_parallel = True

# To configure the encapsulation priority. Default: MPLSoGRE
#env.encap_priority = "'MPLSoUDP','MPLSoGRE','VXLAN'"

```



**NOTE:** The management interface configuration happens outside of fab, so if the user needs a bond interface, the user needs to create a bond and assign the management NICs to it.

The management network must be a routable network.

## Using Headless vRouter to Improve Redundancy

### *Overview: vRouter Agent Redundancy*

The Contrail vRouter agent downloads routes, configurations, and the multicast tree from the control node. For redundancy, the vRouter agent connects to two control nodes.

However, in some circumstances, it is possible for the vRouter agent to lose the connection to the two control nodes, in this case, the information provided by the control nodes can get flushed. If the vRouter agent loses connection to the control nodes, the multicast and unicast are flushed immediately, however, the configuration agent waits awhile for the control node to come up, and after that time, the configuration information gets flushed.

<b>Headless vRouter Function</b>	When the headless vRouter feature is enabled, if the agent's connection to the control nodes is lost, the last known information about routes, configurations, and the multicast tree is retained and marked as stale entries. In the meantime, the system can remain in a working state. When the control node comes up again, the agent waits awhile to flush out the stale entries, and the newly-connected control node sends information about the routes, configurations, and multicast tree to the agent. If the control node is in an unstable state, coming up and going down again, the agent retains the stale information until the control node becomes stable again.
----------------------------------	--

## Configuring the Headless vRouter

By default, the vRouter agent runs in non-headless mode. You can enable the headless router feature by using the command-line interface or by configuring the agent configuration file. In headless mode, the vRouter agent retains the last known good configuration from the control node if all control nodes are lost.

<b>Using CLI to Configure Headless vRouter</b>	Use the following command-line parameters to enable the headless vRouter. The following argument will run the compute node in headless mode.
--	--

**--DEFAULT.headless arg**

<b>Using contrail-vrouter-agent.conf to Configure Headless vRouter</b>	Use the following line in the DEFAULT section of the <b>contrail-vrouter-agent.conf</b> to enable the headless vRouter.  <b>headless_mode=true</b>
--	--

Possible values are **true** (enable) and **false** (disable).

<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">High Availability Support on page 33</a></li><li>• <i>Example: Adding New OpenStack or Contrail Roles to an Existing High Availability Cluster</i></li></ul>
------------------------------	--



## PART 4

# Administration

- [Service Chaining on page 205](#)
- [Configuring Services on page 229](#)
- [Load Balancing-as-a-Service on page 247](#)
- [Multitenancy Support on page 251](#)



## CHAPTER 7

# Service Chaining

- [Service Chaining on page 205](#)
- [Service Chaining MX Series Configuration on page 209](#)
- [Example: Creating an In-Network or In-Network-NAT Service Chain on page 210](#)
- [Example: Creating a Transparent Service Chain on page 217](#)
- [ECMP Load Balancing in the Service Chain on page 221](#)
- [Example: Creating a Service Chain With the CLI on page 222](#)
- [Using the Juniper Networks Heat Template with Contrail on page 225](#)

## Service Chaining

---

Contrail Controller supports chaining of various Layer 2 through Layer 7 services such as firewall, NAT, IDP, and so on.

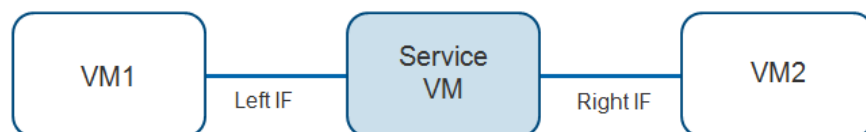
- [Service Chaining Basics on page 205](#)
- [Service Chaining Configuration Elements on page 206](#)

### 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.

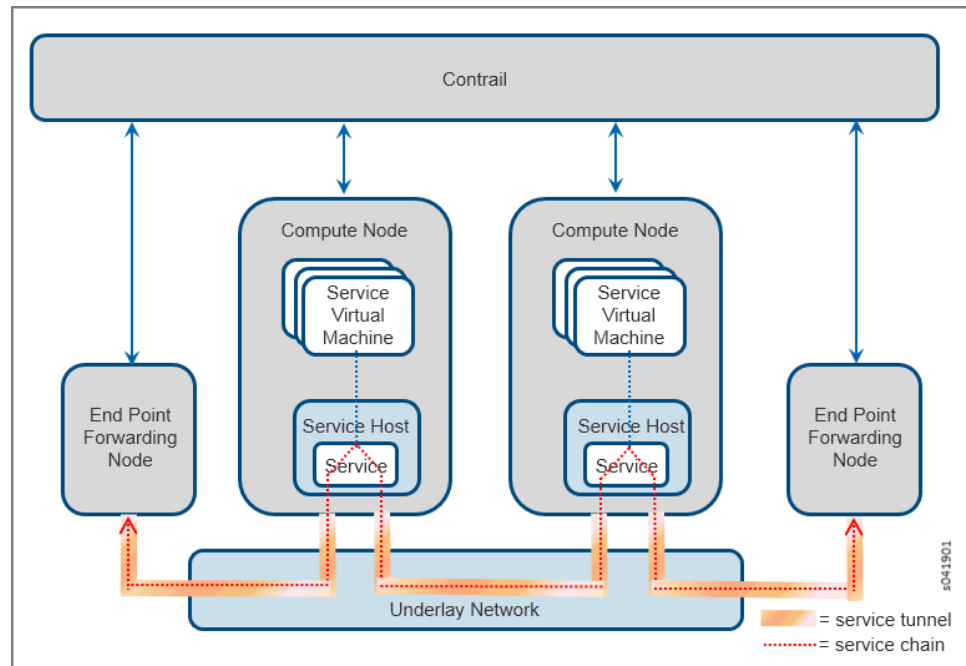
[Figure 51 on page 205](#) shows the basic service chain schema, with a single service. The service VM spawns the service, using the convention of left interface (left IF) and right interface (right IF). Multiple services can also be chained together.

**Figure 51: 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 52 on page 206](#) 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 52: 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 210](#)
- [Example: Creating a Service Chain With the CLI on page 222](#)
- [ECMP Load Balancing in the Service Chain on page 221](#)

## 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 187.

## 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 Contrail Juniper Networks user interface. This service chain example also shows scaling of service instances.

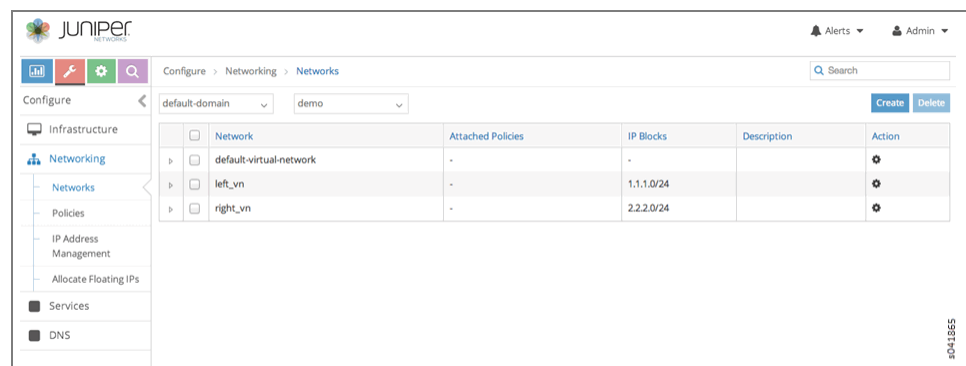
- [Creating an In-Network or In-Network-NAT Service Chain on page 210](#)

### Creating an In-Network or In-Network-NAT Service Chain

To create an **in-network** or **in-network-nat** service chain:

- Create a left and a right virtual network. Select **Configure > Networking > Networks** and create **left\_vn** and **right\_vn**; see [Figure 53 on page 210](#).

**Figure 53: Create Networks**



- 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 54 on page 211](#).



Figure 54: Add Service Template

Table 20: Add Service Template Fields

Field	Description
<b>Name</b>	Enter a name for the service template.
<b>Service Mode</b>	Select the service mode: <b>In-Network</b> (for firewall service), <b>In-Network-NAT</b> (for NAT service), or <b>Transparent</b> .
<b>Service Scaling</b>	If you will be using multiple virtual machines for a single service instance to scale out the service, select the <b>Service Scaling</b> 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.
<b>Image Name</b>	Select from a list of available images the image for the service.
<b>Interface Types</b>	<p>Select the interface type or types for this service:</p> <ul style="list-style-type: none"> <li>For firewall or NAT services, both <b>Left Interface</b> and <b>Right Interface</b> are required.</li> <li>For an analyzer service, only a <b>Left Interface</b> is required.</li> <li>For Juniper Networks virtual images, <b>Management Interface</b> is also required, in addition to any left or right requirement.</li> </ul>

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 55 on page 212](#) shows the **Left** interface selected, with the **Shared IP** check box selected, so the left interface will share the IP address.

**Figure 55: Add Service Template Shared IP**

**Add Service Template**

**Name**

**Service Mode**  **Service Type**

**Service Scaling** ☒

**Image Name**

**Interface Types**  **Shared IP** ☒ + -

Service Interface	Shared IP
Management	Disabled
Right	Disabled
Left	Enabled

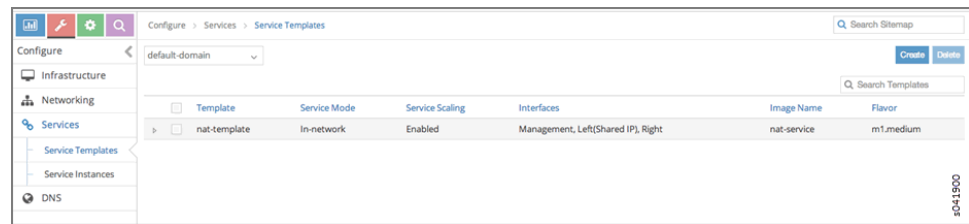
**Cancel** **Save**

s041903

5. When finished, click **Save**.

The service template is created and appears on the **Service Templates** screen, see [Figure 56 on page 213](#).

Figure 56: Service Templates



6. Now 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 57 on page 213](#).

Figure 57: Create Service Instances

Table 21: Create Service Instances Fields

Field	Description
<b>Instance Name</b>	Enter a name for the service instance.
<b>Services Template</b>	Select from a list of available service templates the service template to use for this instance.
<b>Number of Instances</b>	If scaling is enabled, enter a value in the <b>Number of Instances</b> field to define the number of instances of service virtual machines to launch.
<b>Interface List and Virtual Networks</b>	An ordered list of interfaces as defined in the Service Template. If you are using the <b>Management Interface</b> , select <b>Auto Configured</b> . The software will use an internally-created virtual network. For <b>Left Interface</b> , select <b>left_vn</b> and for <b>Right Interface</b> , select <b>right_vn</b> .

7. 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 58 on page 214](#).

Figure 58: Create Service Instances

8. 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 59 on page 214](#) and [Figure 60 on page 215](#).

Figure 59: Service Instance Details

Virtual Machine	Status	Power State	Networks
fw-instance_1	ACTIVE	RUNNING	svc-vn-mgmt-250.250.1.252 svc-vn-left-250.250.2.253 svc-vn-right-250.250.3.253

Figure 60: Service Instance Console

0.204.216.36:5999/vnc\_auto.html?token=9eada783-24e7-4808-9325-4ad257bf3762

Connected (unencrypted) to: QEMU (instance-0000000b)

Interface	Admin	Link	Proto	Local	Remote
ge-0/0/0	up	up			
ge-0/0/0.0	up	up	inet	250.250.1.253/24	
ge-0/0/0	up	up			
ip-0/0/0	up	up			
lsq-0/0/0	up	up			
lt-0/0/0	up	up			
mt-0/0/0	up	up			
sp-0/0/0	up	up			
sp-0/0/0.0	up	up	inet	10.0.0.1	--> 10.0.0.16
sp-0/0/0.16383	up	up	inet	10.0.0.6	--> 0/0
				128.0.0.1	--> 128.0.1.16
				128.0.0.6	--> 0/0
ge-0/0/1	up	up			
ge-0/0/1.0	up	up	inet	1.1.1.253/24	
ge-0/0/2	up	up			
ge-0/0/2.0	up	up	inet	2.2.2.253/24	
dsc	up	up			
gre	up	up			
ipip	up	up			
lo0	up	up			
lo0.16384	up	up	inet	127.0.0.1	--> 0/0
lo0.16385	up	up	inet	10.0.0.1	--> 0/0
---	---	---	---	---	---

5041919

9. Next, configure the network policy. Navigate to **Configure > Networking > Policies**.

- Name the policy and associate it to the networks created earlier – **left\_vn** and **right\_vn**.
- Set source network as **left\_vn** and destination network as **right\_vn**.
- Check **Apply Service** and select the service (**nat-ecmp**).

Figure 61: 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	Destination	<input checked="" type="checkbox"/>	<input type="checkbox"/>

fw-instance x

Cancel Save

10. Next, associate the policy to 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 62: Edit Network

Network Name:

Network Policy(s):

Address Management:  IP Block:  Gateway:

IPAM	IP Block	Gateway
default-domain:default-project:default-network-ipam	1.1.1.0/24	1.1.1.254

Route Targets

Floating IP Pools

Host Routes

Advanced Options

Cancel Save

11. 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 63 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 63: 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 205](#)
  - [Example: Creating a Transparent Service Chain on page 217](#)
  - [ECMP Load Balancing in the Service Chain on page 221](#)

## Example: Creating a Transparent Service Chain

This section provides an example of creating a transparent mode service chain using the Contrail Controller Juniper Networks 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.

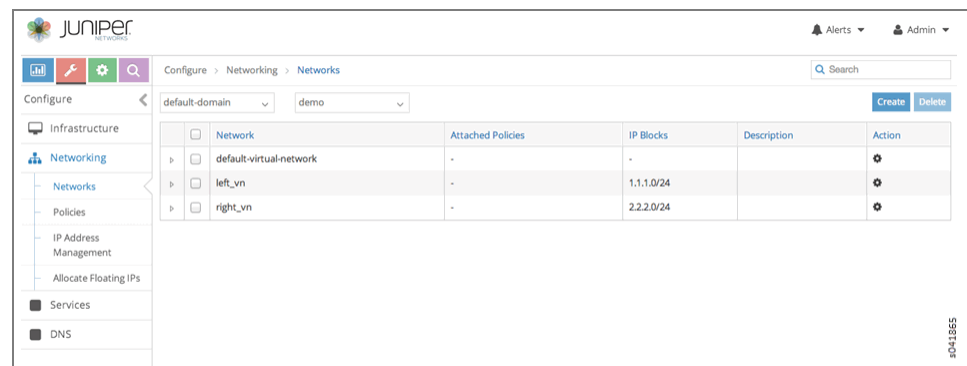
- [Creating a Transparent Mode Service Chain on page 217](#)

### Creating a Transparent Mode Service Chain

To create a transparent mode service chain:

1. First create a left and a right virtual network. Select **Configure > Networking > Networks** and create **left\_vn** and **right\_vn**; see [Figure 64 on page 217](#).

**Figure 64: Create Networks**



2. Next, 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 65 on page 218](#).

Figure 65: Add Service Template

**Add Service Template**

Name:

Service Mode:

Image Name:

Interface Types	Shared IP	Static Routes	+
Management	<input type="checkbox"/>	<input type="checkbox"/>	+ -
Left	<input checked="" type="checkbox"/>	<input type="checkbox"/>	+ -
Right	<input checked="" type="checkbox"/>	<input type="checkbox"/>	+ -

**Advanced options**

Service Scaling: ☒

Instance Flavor:

Cancel Save

Table 22: Add Service Template Fields

Field	Description
Name	Enter a name for the service template.
Service Mode	Select the service mode: <b>In-Network</b> or <b>Transparent</b>
Service Scaling	If you will be using multiple virtual machines for a single service instance to scale out the service, select the <b>Service Scaling</b> check box. When scaling is selected, you can choose to use the same IP address for a particular interface on each virtual machine interface or to allocate new addresses for each virtual machine. For a NAT service, the left (inner) interface should have the same IP address, and the right (outer) interface should have a different IP address.
Image Name	Select from a list of available images the image for the service.
Interface Types	Select the interface type or types for this service: <ul style="list-style-type: none"> <li>For firewall or NAT services, both <b>Left Interface</b> and <b>Right Interface</b> are required.</li> <li>For an analyzer service, only <b>Left Interface</b> is required.</li> <li>For Juniper Networks virtual images, <b>Management Interface</b> is also required, in addition to any left or right requirement.</li> </ul>



3. On **Add Service Template**, complete the following for the transparent mode service template:

- **Name:** firewall-template
- **Service Mode:** Transparent
- **Service Scaling:** select
- **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.

5. When finished, click **Save**.

6. Now 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 66 on page 219](#).

Figure 66: Create Service Instances

Table 23: 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 <b>Auto Configured</b> .
Right Network	Select from a list of available virtual networks the network to use for the right interface. For transparent mode, select <b>Auto Configured</b>

Table 23: Create Service Instances Fields (*continued*)

**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**

7. 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 67 on page 220](#).

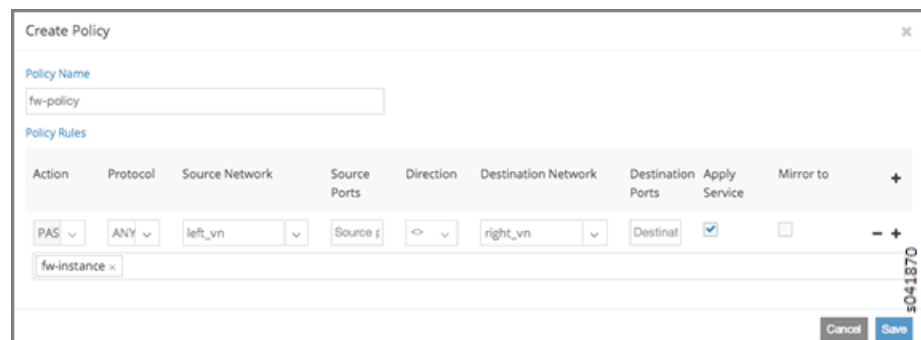
Figure 67: Service Instance Details



8. 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**).

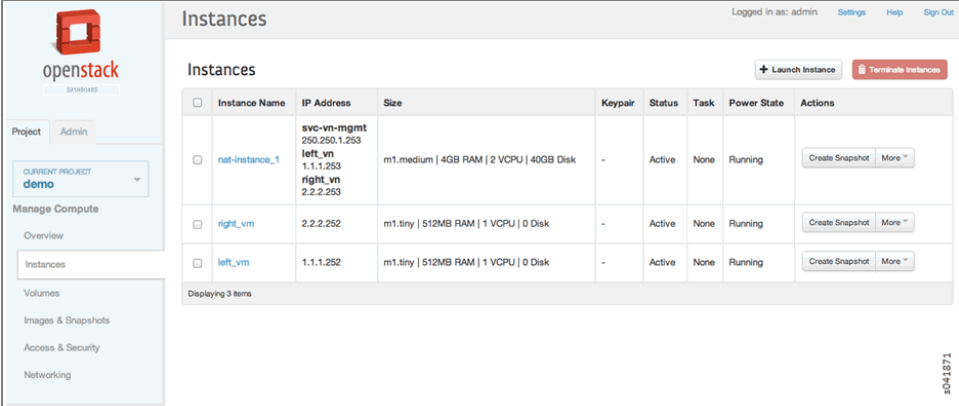
Figure 68: Create Policy



9. 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**.
10. 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 69 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 69: Launch 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

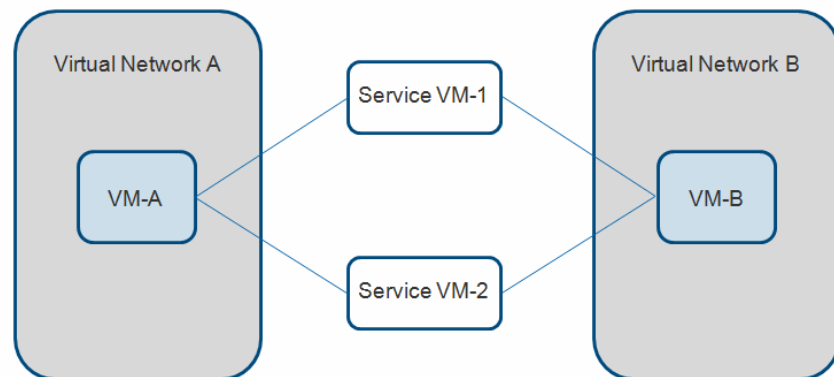
Related Documentation

- [Service Chaining on page 205](#)

## 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 70 on page 221](#), 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 70: Load Balancing a Service Chain



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 205](#)
- [Example: Creating a Service Chain With the CLI on page 222](#)

---

## 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 222](#)
- [CLI for Creating a Service Template on page 222](#)
- [CLI for Creating a Service Instance on page 223](#)
- [CLI for Creating a Service Policy on page 223](#)
- [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/utils`.

### 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 205](#)

## Using the Juniper Networks Heat Template with Contrail

Heat is the orchestration engine of the OpenStack Orchestration program. Heat enables launching multiple cloud applications based on templates that are comprised of text files.

- [Introduction to Heat on page 225](#)
- [Heat Architecture on page 225](#)
- [Juniper Heat Plugin on page 225](#)
- [Example: Creating a Service Template Using Heat on page 226](#)

### 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 also can delete all application resources if the system is finished with an application.

Heat templates can also 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, that is 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 (RPC).
- **heat-engine** --- Responsible for orchestrating the launch of templates and providing events back to the API consumer.

### Juniper Heat Plugin

The Juniper Heat plugin enables the use of some resources not currently included in the OpenStack Heat orchestration engine, including network policy, service template, and service instances. Resources are the specific objects that Heat creates or modifies as part of its operation. The Heat plugin resources are loaded into the `/usr/lib/heat/resources` directory by the heat-engine service as it starts up. The names of the resource types in the Juniper Heat plugin include:

- **OS::Contrail::NetworkPolicy**
- **OS::Contrail::ServiceTemplate**

- OS::Contrail::AttachPolicy
- OS::Contrail::ServiceInstance

## 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::Contrail::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
```



## CHAPTER 8

# Configuring Services

- [Configuring DNS Servers on page 229](#)
- [Support for Multicast on page 238](#)
- [Using Static Routes with Services on page 240](#)
- [Configuring Metadata Service on page 244](#)

### Configuring DNS Servers

---

- [DNS Overview on page 229](#)
- [Defining Multiple Virtual Domain Name Servers on page 230](#)
- [IPAM and Virtual DNS on page 230](#)
- [DNS Record Types on page 231](#)
- [Configuring DNS Using the Interface on page 232](#)
- [Configuring DNS Using Scripts on page 237](#)

### 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:

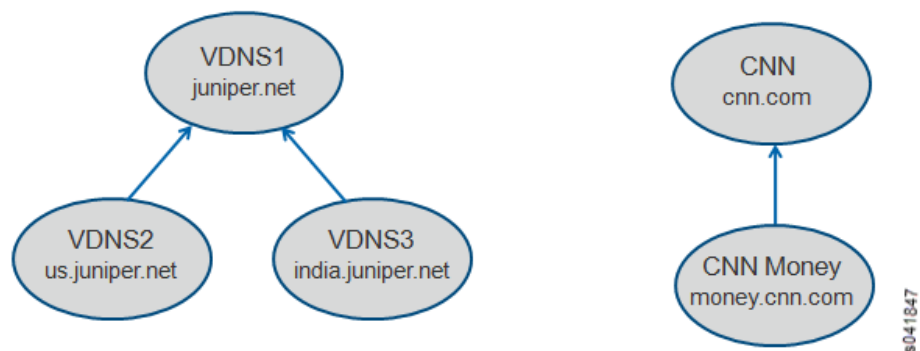
- It should be possible to configure multiple domain name servers to provide name resolution service for the virtual machines spawned in the system.
- It should be possible to configure the domain name servers to form DNS server hierarchies required by each tenant.

- 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 71 on page 230](#) shows examples of virtual DNS servers defined in **default-domain**, providing the name service for the DNS domains indicated.

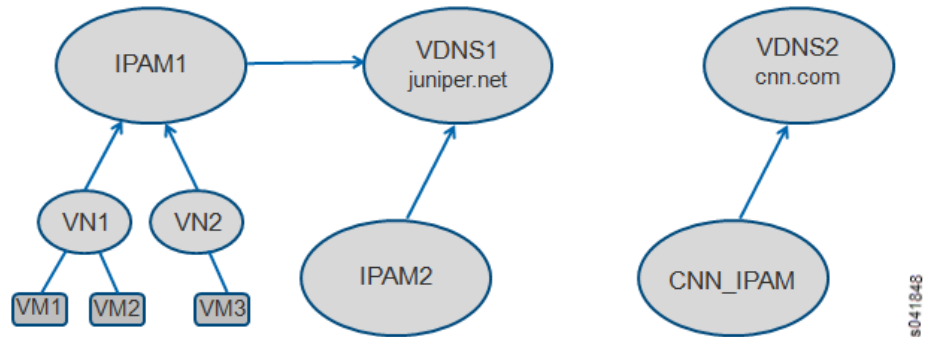
Figure 71: DNS Servers Examples



## 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 72 on page 231](#)

Figure 72: IPAM and Virtual DNS



## DNS Record Types

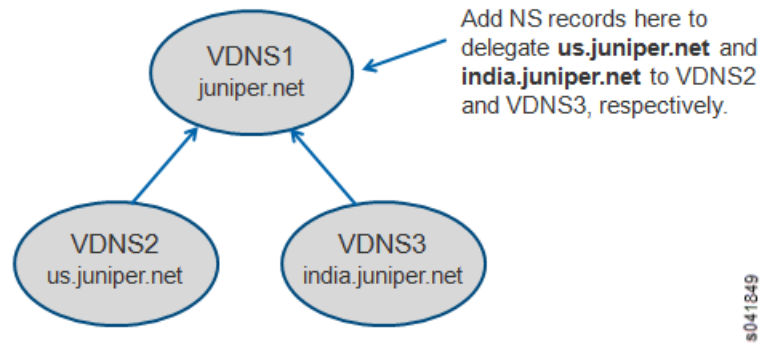
DNS records can be added statically. DNS record types **A**, **CNAME**, **PTR**, and **NS** are currently supported in the system. Each record includes the type, class (IN), name, data, and TTL values. See [Table 24 on page 231](#) for descriptions of the record types.

Table 24: DNS Record Types Supported

DNS Record Type	Description
<b>A</b>	Used for mapping hostnames to IPv4 addresses. Name refers to the name of the virtual machine, and data is the IPv4 address of the virtual machine.
<b>CNAME</b>	Provides an alias to a name. Name refers to the name of the virtual machine, and data is the new name (alias) for the virtual machine.
<b>PTR</b>	A pointer to a record, it provides reverse mapping from an IP address to a name. Name refers to the IP address, and data is the name for the virtual machine. The address in the PTR record should be part of a subnet configured for a VN within one of the IPAMs referring to this virtual DNS server.
<b>NS</b>	Used to delegate a subdomain to another DNS server. The DNS server could be another virtual DNS server defined in the system or the IP address of an external DNS server reachable via the infrastructure. Name refers to the subdomain being delegated, and data is the name of the virtual DNS server or IP address of an external server.

[Figure 73 on page 232](#) shows an example usage for the DNS record type of **NS**.

Figure 73: 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** screen appears; see [Figure 74 on page 232](#).

Figure 74: 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

s041850

2. To add a new DNS server, click the **Create** button.

Enter DNS server information in the **Add DNS** window; see [Figure 75 on page 233](#)

**Figure 75: Add DNS**

Complete the fields for the new server; see [Table 25 on page 233](#).

**Table 25: Add DNS Fields**

Field	Description
Server Name	Enter a name for this server.
Domain Name	Enter the name of the domain for this server.
Time To Live	Enter the <b>TTL</b> in seconds.
Next DNS Server	Select from a list the name of the next DNS server to process DNS requests if they cannot be processed at this server, or <b>None</b> .
Load Balancing Order	Select the load-balancing order from a drop-down list— <b>Random</b> , <b>Fixed</b> , <b>Round Robin</b> . When a name has multiple records matching, the configured record order determines the order in which the records are sent in the response. Select <b>Random</b> to have the records sent in random order. Select <b>Fixed</b> to have records sent in the order of creation. Select <b>Round Robin</b> to have the record order cycled for each request to the record.
OK	Click <b>OK</b> to create the record.
Cancel	Click <b>Cancel</b> to clear the fields and start over.

- To add a new DNS record, from the **Configure DNS Records** screen, click the **Add Record** button in the lower right portion of the screen.

The **Add DNS Record** window appears; see [Figure 76 on page 234](#).

Figure 76: Add DNS Record

**Add DNS Record**

Type: A (IP Address Record)

Host Name: Host Name to be resolved

IP Address: Enter an IP Address

Class: IN (Internet)

Time To Live: TTL(86400 secs)

Cancel Save

5041853

- Complete the fields for the new record; see [Table 26 on page 234](#).

Table 26: Add DNS Record Fields

Field	Description
Record Name	Enter a name for this record.
Type	Select the record type from a drop-down list—A, CNAME, PTR, NS.
IP Address	Enter the IP address for the location for this record.
Class	Select the record class from a drop-down list—IN is the default.
Time To Live	Enter the TTL in seconds.
OK	Click <b>OK</b> to create the record.
Cancel	Click <b>Cancel</b> to clear the fields and start over.

- To associate an IPAM to a virtual DNS server, from the **Configure DNS Records** screen, 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 77 on page 235](#).



Figure 77: Associate IPAMs to DNS

Complete the IPAM associations, using the field descriptions in [Table 27 on page 235](#).

Table 27: Associate IPAMs to DNS Fields

Field	Description
<b>Associate to All IPAMs</b>	Select this box to associate the selected DNS server to all available IPAMs.
<b>Available IPAMs</b>	This column displays the currently available IPAMs.
<b>Associated IPAMs</b>	This column displays the IPAMs currently associated with the selected DNS server.
>>	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.
<b>OK</b>	Click <b>OK</b> to commit the changes indicated in the window.
<b>Cancel</b>	Click <b>Cancel</b> to clear all entries and start over.

- Use the **IP Address Management** screen (**Configure > Networking > IP Address Management**; see [Figure 78 on page 236](#)) 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 78: Configure IP Address Management

- To associate an IPAM to a virtual DNS server or to tenant's IP addresses, at the **IP Address Management** screen, 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 79 on page 236](#).

Figure 79: DNS Server

- In the first field, select the **DNS Method** from a drop-down list (**None**, **Default DNS**, **Tenant DNS**, **Virtual DNS**; see [Table 28 on page 236](#).

Table 28: DNS Modes

DNS Mode	Description
<b>None</b>	Select <b>None</b> when no DNS support is required for the VMs.
<b>Default</b>	In default mode, DNS resolution for VMs is performed based on the name server configuration in the server infrastructure. The subnet default gateway is configured as the DNS server for the VM, and the DHCP response to the VM has this DNS server option. DNS requests sent by a VM to the default gateway are sent to the name servers configured on the respective compute nodes. The responses are sent back to the VM.

Table 28: DNS Modes (*continued*)

DNS Mode	Description
<b>Tenant</b>	Configure this mode when a tenant wants to use its own DNS servers. Configure the list of servers in the IPAM. The server list is sent in the DHCP response to the VM as DNS servers. DNS requests sent by the VMs are routed the same as any other data packet based on the available routing information.
<b>Virtual DNS</b>	Configure this mode to support virtual DNS servers (VDNS) to resolve the DNS requests from the VMs. Each IPAM can have a virtual DNS server configured in this mode.

9. Complete the remaining fields on this screen, 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 29 on page 237](#).



**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 29: 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:vdns1</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 29: 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 238](#)
- [All-Broadcast/Limited-Broadcast and Link-Local Multicast on page 239](#)
- [Host Broadcast on page 240](#)

### 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.

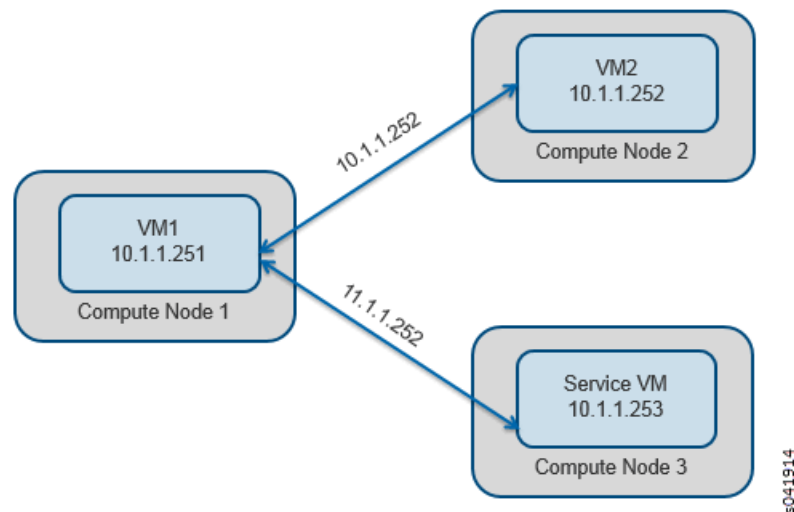
## Using Static Routes with Services

- [Static Routes for Service Instances on page 240](#)
- [Configuring Static Routes on a Service Instance on page 241](#)
- [Configuring Static Routes on Service Instance Interfaces on page 242](#)
- [Configuring Static Routes as Host Routes on page 243](#)

## 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
nat

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

s041915

## Configuring Static Routes on Service Instance Interfaces

To configure static routes on a service instance interface:

1. Go to **Configure > Services > Service Instances** and click **Create**.
2. At **Create Service Instances**, complete the fields for **Instance Name** and **Services Template**.
3. Select the virtual network for each of the interfaces
4. Click the **Static Routes** dropdown menu under each interface field for which the static routes option is enabled to open the **Static Routes** menu and configure the static routes in the fields provided.



**NOTE:** If the **Auto Configured** option is selected, traffic destined to the static route subnet is load balanced across service instances.

The following figure shows a configuration to apply a service instance between VN1 (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

5041316



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

The screenshot shows the 'Create Service Instances' window with the following configuration:

Interface	Direction	Virtual Network	Static Routes						
Interface 2	Left	vn1	<table border="1"> <thead> <tr> <th>Prefix</th> <th>Next hop</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>11.1.1.0/24</td> <td>Interface 2</td> <td>+ -</td> </tr> </tbody> </table>	Prefix	Next hop	Action	11.1.1.0/24	Interface 2	+ -
Prefix	Next hop	Action							
11.1.1.0/24	Interface 2	+ -							
Interface 3	Right	vn2	<table border="1"> <thead> <tr> <th>Prefix</th> <th>Next hop</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>10.1.1.0/24</td> <td>Interface 3</td> <td>+ -</td> </tr> </tbody> </table>	Prefix	Next hop	Action	10.1.1.0/24	Interface 3	+ -
Prefix	Next hop	Action							
10.1.1.0/24	Interface 3	+ -							

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

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

s041318

Related •  
Documentation

## 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 9

# Load Balancing-as-a-Service

- [Configuring Load-Balancing-as-a-Service in Contrail on page 247](#)

## Configuring Load-Balancing-as-a-Service in Contrail

---

- [Overview: Load-Balancing-as-a-Service on page 247](#)
- [Contrail LBaaS Implementation on page 248](#)

### Overview: Load-Balancing-as-a-Service

Load-Balancing-as-a-Service (LBaaS) is a feature available through OpenStack Neutron. Contrail Release 1.20 and greater allows the use of the Neutron API for LBaaS to apply open source load balancing technologies to provision a load balancer in the Contrail system.

The LBaaS load balancer enables the creation of a pool of virtual machines serving applications, all front-ended by a virtual-ip. The LBaaS implementation has the following features:

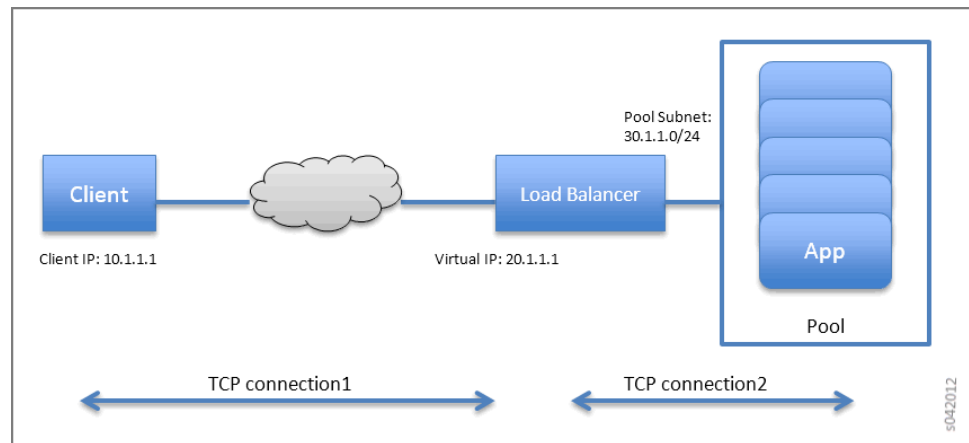
- Load balancing of traffic from clients to a pool of backend servers. The load balancer proxies all connections to its virtual IP.
- Provides load balancing for HTTP, TCP, and HTTPS.
- Provides health monitoring capabilities for applications, including HTTP, TCP, and ping.
- Enables floating IP association to **virtual-ip** for public access to the backend pool.

In the following figure, the load balancer is launched with the virtual IP address 20.1.1.1. The backend pool of virtual machine applications (App Pool) is on the subnet 30.1.1.0/24. Each of the application virtual machines gets an IP address (virtual-ip) from the pool subnet. When a client connects to the **virtual-ip** for accessing the application, the load balancer proxies the TCP connection on its **virtual-ip**, then creates a new TCP connection to one of the virtual machines in the pool.

The pool member is selected using one of following methods:

- weighted round robin (WRR), based on the weight assignment
- least connection, selects the member with the fewest connections

- source IP selects based on the **source-ip** of the packet



Additionally, the load balancer monitors the health of each pool member using the following methods:

- Monitors TCP by creating a TCP connection at intervals.
- Monitors HTTP by creating a TCP connection and issues an HTTP request at intervals.
- Monitors ping by checking if a member can be reached by pinging.

## Contrail LBaaS Implementation

Contrail supports the OpenStack LBaaS Neutron APIs and creates relevant objects for LBaaS, including **virtual-ip**, **loadbalancer-pool**, **loadbalancer-member**, and **loadbalancer-healthmonitor**. Contrail creates a service instance when a **loadbalancer-pool** is associated with a **virtual-ip** object. The service scheduler then launches a namespace on a randomly selected virtual router and spawns HAProxy into that namespace. The configuration for HAProxy is picked up from the load balancer objects. Contrail supports high availability of namespaces and HAProxy by spawning active and standby on two different routers.

**Example: Configuring LBaaS** This feature is enabled on Contrail through Neutron API calls. The following is an example of creating a pool network and a VIP network. The VIP network is created in the public network and members are added in the pool network.

**Creating a Load Balancer** Use the following steps to create a load balancer in Contrail.

1. Create a VIP network.  

```
neutron net-create vipnet
```

```
neutron subnet-create --name vipsubnet vipnet 20.1.1.0/24
```
2. Create a pool network.  

```
neutron net-create poolnet
```

```
neutron subnet-create --name poolsubnet poolnet 10.1.1.0/24
```
3. Create a pool for HTTP.  

```
neutron lb-pool-create --lb-method ROUND_ROBIN --name mypool --protocol HTTP --subnet-id poolsubnet
```
4. Add members to the pool.  

```
neutron lb-member-create --address 10.1.1.2 --protocol-port 80 mypool
```

```
neutron lb-member-create --address 10.1.1.3 --protocol-port 80 mypool
```
5. Create a VIP for HTTP and associate it to the pool.  

```
neutron lb-vip-create --name myvip --protocol-port 80 --protocol HTTP --subnet-id vipsubnet mypool
```

**Deleting a Load Balancer** Use the following steps to delete a load balancer in Contrail.

1. Delete the VIP.  

```
neutron lb-vip-delete <vip-uuid>
```
2. Delete members from the pool.  

```
neutron lb-member-delete <member-uuid>
```
3. Delete the pool.  

```
neutron lb-pool-delete <pool-uuid>
```

**Managing Healthmonitor for Load Balancer** Use the following commands to create a healthmonitor, associate a healthmonitor to a pool, disassociate a healthmonitor, and delete a healthmonitor.

- Create a healthmonitor.  

```
neutron lb-healthmonitor-create --delay 20 --timeout 10 --max-retries 3 --type HTTP
```
- Associate a healthmonitor to a pool.  

```
neutron lb-healthmonitor-associate <healthmonitor-uuid> mypool
```
- Disassociate a healthmonitor from a pool.  

```
neutron lb-healthmonitor-disassociate <healthmonitor-uuid> mypool
```

**Configuring an SSL VIP  
with an HTTP Backend  
Pool**

Use the following steps to configure an SSL VIP with an HTTP backend pool.

1. Copy an SSL certificate to all compute nodes.  

```
scp ssl_certificate.pem <compute-node-ip> <certificate-path>
```
2. Update the information in `/etc/contrail/contrail-vrouter-agent.conf`.  

```
# SSL certificate path haproxy  
haproxy_ssl_cert_path=<certificate-path>
```
3. Restart `contrail-vrouter-agent`.  

```
service contrail-vrouter-agent restart
```
4. Create a VIP for port 443 (SSL).  

```
neutron lb-vip-create --name myvip --protocol-port 443 --protocol HTTP --subnet-id  
vipsubnet mypool
```

**A Note on Installation**

To use the LBaaS feature, HAProxy, version 1.5 or greater and `iproute2`, version 3.10.0 or greater must both be installed on the Contrail compute nodes.

If you are using `fab` commands for installation, the `haproxy` and `iproute2` packages will be installed automatically with LBaaS if you set the following:

```
env.enable_lbaas=True
```

Use the following to check the version of the `iproute2` package on your system:

```
root@nodeh5:/var/log# ip -V  
ip utility, iproute2-ss130716  
root@nodeh5:/var/log#
```

**Limitations**

LBaaS currently has these limitations:

- A pool should not be deleted before deleting the VIP.
- Multiple VIPs cannot be associated with the same pool. If pool needs to be reused, create another pool with the same members and bind it to the second VIP.
- Members cannot be moved from one pool to another. If needed, first delete the members from one pool, then add to a different pool.
- In case of active-standby failover, namespaces might not get cleaned up when the agent restarts.
- The floating-ip association needs to select the VIP port and not the service ports.



## CHAPTER 10

# Multitenancy Support

- [Configuring Multitenancy Support on page 251](#)

## Configuring Multitenancy Support

---

The following sections describe enabling and viewing multitenancy support.

- [Multitenancy Permissions on page 251](#)
- [API Server on page 252](#)
- [API Library Keystone Integration on page 252](#)
- [Supporting Utilities on page 252](#)

## Multitenancy Permissions

The multi tenancy feature of the API server enables multiple tenants to coexist on the system without interfering with each other. This is achieved by encoding ownership information and permissions with each resource, allowing fine-grained control over create, read, update, and delete (CRUD) operations on those resources.

The Contrail **api-server** enforces resources permissions in a manner similar to Unix files. Each resource has an owner and group. Permissions associated with owner, group, and "others" are:

**R** - reading resource

**W** - create/update resource

**X** - link (refer to) object

CRUD permission requirements for resources managed by **api-server** are as follows:

**C** - write on parent object

For example, to create a virtual network requires write permission on the project.

**R** - read on object (parent if a collection)

**U** - write on object

**D** - write on parent

**ref(link)** - execute on object

For example, on a virtual network using **network-ipam**, **network-ipam** should have X permissions for owner, group, or "others".

## API Server

If multitenancy is enabled, **api-server** deploys keystone middleware in its pipeline. The keystone middleware architecture supports a common authentication protocol in use between OpenStack projects.

The keystone middleware works in conjunction with **api-server** to derive the user name and role for each incoming request. Once obtained, the user name and role are matched against resource ownership and permissions. If the ownership matches or the permissions allow access, access is granted.

For example, assume Tenant A has the following attributes:

- owner = Bob
- group = Staff
- permissions = 750

In this example, only Bob can create a virtual network in Tenant A. Other staff members can view the virtual networks in Tenant A. No others can create or view any virtual networks in Tenant A.

Clients can obtain an **auth\_token** by posting credentials to the keystone admin API (**/v2.0/tokens**). The **VncApi** client library does this automatically. If an **auth\_token** is present in an incoming request, **api-server** validates credentials derived from the token against object permissions. If an incoming request has an invalid or missing **auth\_token**, a 401 error is returned.

Notes:

- Multitenancy is enabled by the flag **multi\_tenancy** in **/etc/contrail/api-server.conf**
- If multitenancy is enabled, **memcaching** is automatically enabled, to improve token validation response time.

## API Library Keystone Integration

**VncApi** has been updated to check for any 401 error that **api-server** returns as a result of a missing or invalid token. This forces **VncApi** to connect with the keystone middleware and fetch an **auth\_token**. All subsequent requests to **api-server** include the **auth\_token**.

## Supporting Utilities

- **/opt/contrail/utls/chmod.py**— To change permissions and ownership (user or group membership) of a resource. Requires the resource type (for example, **virtual-network**) and the resource FQN (for example, **default-domain:default-project:default-virtual-network**).

Invoke **python /opt/contrail/utls/chmod.py -h** to see usage information

Example 1 - See current permissions:

```
[user@host]# python /opt/contrail/utils/chmod.py <ip address>:8082 project
default-domain:default-project
Type = project
Name = default-domain:default-project
API Server = <ip address>:8082
Keystone credentials admin/contrail123/admin
Obj uuid = $ABC123
Obj perms = cloud-admin/cloud-admin-group 777
```

```
[root@host]# python /opt/contrail/utils/chmod.py <ip address>:8082 --owner foo
--group bar --perms 555 project default-domain:default-project
Type = project Name = default-domain:default-project
API Server = <ip address>
Owner = foo
Group = bar
Perms = 555
Keystone credentials admin/contrail123/admin
Obj uuid = $ABC123
Obj perms = cloud-admin/cloud-admin-group 777
New perms = foo/bar 555
```

- **/opt/contrail/utils/multi\_tenancy.py** — Show if multitenancy is enabled or disabled. Also used to turn multitenancy on or off. Requires admin credentials.

Invoke **python /opt/contrail/utils/multi\_tenancy.py -h** to see usage information

Example 1: View multitenancy status

```
[root@host]# python /opt/contrail/utils/multi_tenancy.py <ip address>:8082
API Server = <ip address>:8082
Keystone credentials admin/contrail123/admin
```

Multi Tenancy is enabled

Example 2: Turn multitenancy off

```
[root@host]# python /opt/contrail/utils/multi_tenancy.py <ip address>:8082 --off
API Server = <ip address>:8082
Keystone credentials admin/contrail123/admin
```

Multi Tenancy is disabled



## PART 5

# Optimizing and Troubleshooting

- [Traffic Mirroring on page 257](#)
- [Monitoring and Troubleshooting on page 271](#)
- [Application Programming Interfaces \(APIs\) on page 333](#)
- [Optimizing on page 371](#)
- [Common Support Answers on page 393](#)
- [Contrail Commands on page 451](#)



# Traffic Mirroring

- [Configuring Traffic Analyzers and Packet Capture for Mirroring on page 257](#)
- [Configuring Interface Monitoring and Mirroring on page 266](#)
- [Analyzer Service Virtual Machine \(analyzer-vm-console.qcow2\) on page 267](#)

## Configuring Traffic Analyzers and Packet Capture for Mirroring

---

Contrail provides traffic mirroring so you can mirror specified traffic to a traffic analyzer where you can perform deep traffic inspection. Traffic mirroring enables you to designate certain traffic flows to be mirrored to a traffic analyzer, where you can view traffic flows in great detail.

Use **Monitor > Debug > Packet Capture** to configure packets to be captured and “mirrored” to a virtual machine configured as a traffic analyzer. The packet activity can then be inspected for monitoring and troubleshooting purposes. This section demonstrates how to set up packet capture to mirror traffic packets to an analyzer.

- [Traffic Analyzer Images on page 257](#)
- [Configuring Traffic Analyzers on page 258](#)
- [Setting Up Traffic Mirroring Using Monitor > Debug > Packet Capture on page 258](#)
- [Setting Up Traffic Mirroring Using Configure > Networking > Services on page 261](#)

## Traffic Analyzer Images

Before using the Contrail interface to configure traffic analyzers and packet capture for mirroring, make sure that the following analyzer images are available in the VM image list for your system. The traffic analyzer images are enhanced for viewing details of captured packets in Wireshark. When creating a policy for the traffic analyzer, the traffic analyzer instance should always have the **Mirror to** field selected in the policy, do not select the **Apply Service** field for a traffic analyzer.

- **analyzer-vm-console-qcow2**—Standard traffic analyzer; should be named **analyzer** in the image list. This type of traffic analyzer is always configured with a single interface, and the interface should be a **Left** interface.
- **analyzer-vm-console-two-if qcow2**—This type of traffic analyzer has two interfaces, **Left** and **Management**. This traffic analyzer can have any name except the name **analyzer**, which is reserved for the single interface analyzer.

## Configuring Traffic Analyzers

In Contrail Controller, you use a two-part configuration to mirror captured packet traffic to a traffic analyzer, where the traffic details can be inspected. The configuration has the following steps:

1. Configure analyzer(s) on the host.
2. Set up rules for packet capture.

Additionally, there are two ways to configure the packet capture for the analyzers from within the Contrail interface:

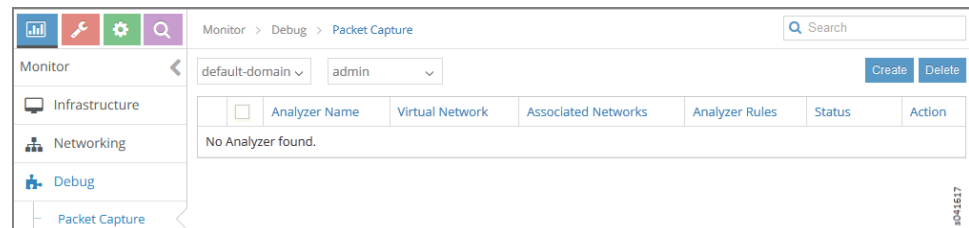
- Configure from **Monitor > Debug > Packet Capture**
- Configure from **Configure > Networking > Services**

## Setting Up Traffic Mirroring Using Monitor > Debug > Packet Capture

The following are the steps needed to set up packet capture in order to “mirror” the traffic to an analyzer VM for the purpose of reviewing various aspects of packet traffic moving through the system.

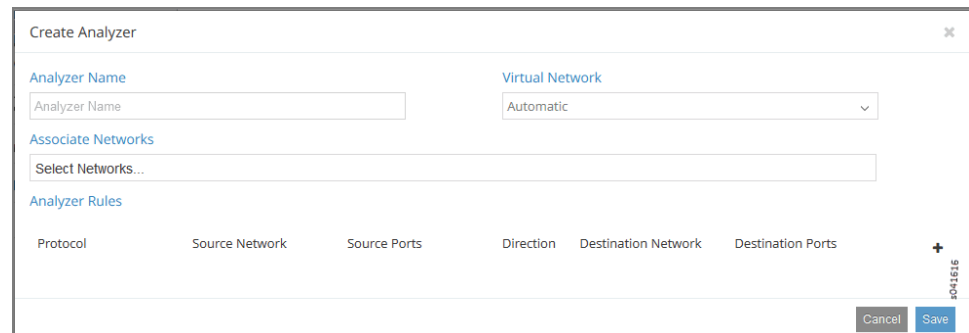
1. Select **Monitor > Debug > Packet Capture**. The **Packet Capture** screen appears; see [Figure 80 on page 258](#).

**Figure 80: Packet Capture**



2. Click **Create** to add an analyzer; see [Figure 81 on page 258](#).

**Figure 81: Create Analyzer**



3. In the **Analyzer Name** field, enter a name for the analyzer and in the **Virtual Network** field, select **Automatic** or select a specific virtual network from the drop-down list of available networks; click **Save** when finished.



- 4. To create rules for the analyzer, in the lower portion of the **Create Analyzer** screen, click the + button to add a rule.

The **Analyzer Rules** fields appear; see [Figure 82 on page 259](#).

Figure 82: Analyzer Rules

- 5. Select the rules to apply to determine which packets should be “mirrored”—sent to the analyzer for monitoring.

See [Table 30 on page 259](#) for guidelines for completing the rule fields.

Table 30: Analyzer Rule Fields

Field	Description
IP Protocol	Select from a list to define from which protocol packets are to be captured: <ul style="list-style-type: none"><li>• ANY</li><li>• TCP</li><li>• UDP</li><li>• ICMP</li></ul>
Source Network	Select from a list the source network from which packets are to be captured: <ul style="list-style-type: none"><li>• any</li><li>• local</li><li>• domain:network 1</li><li>• domain:network 2</li><li>• domain:network .....</li></ul>
Source Ports	If you want to capture only those packets that originate from a specific port number, enter the port number.
Direction	Select the direction of flow for the packets to be captured: <ul style="list-style-type: none"><li>• Bidirectional</li><li>• Unidirectional</li></ul>

Table 30: Analyzer Rule Fields (*continued*)

Field	Description
<b>Destination Network</b>	<p>Select from a list the destination network for the packets to be captured:</p> <ul style="list-style-type: none"> <li>any</li> <li>local</li> <li>domain:network 1</li> <li>domain:network 2</li> <li>domain:network .....</li> </ul>
<b>Destination Ports</b>	If you want to capture only those packets that are destined to a specific port number, enter the port number.
Cancel, Save	When finished, click <b>Save</b> to commit your selections, or click <b>Cancel</b> to clear the entries and start over.

- To associate virtual networks with the analyzer, click the **Associate Networks** field in the center portion of the screen. Select from a drop-down list of available networks the networks to associate with this analyzer; see [Figure 83 on page 260](#).

Figure 83: Create Analyzer Associate Networks

The screenshot shows the 'Create Analyzer' window. The 'Analyzer Name' is 'demo'. The 'Virtual Network' is set to 'Automatic'. Under 'Associate Networks', a list shows 'frontend', 'default-virtual-network', and 'backend' (highlighted). Below the list is a table with the following data:

Protocol	Source Network	Source Ports	Direction	Destination Network	Destination Ports
Any	any	Source ports	<=>	any	Destination ports

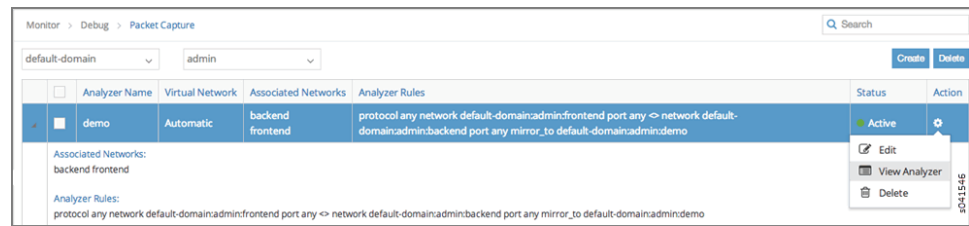
Buttons for 'Cancel' and 'Save' are at the bottom right.



**NOTE:** If there is already a network policy attached to the virtual network selected, any conflicting rules configured for the analyzer will not take effect.

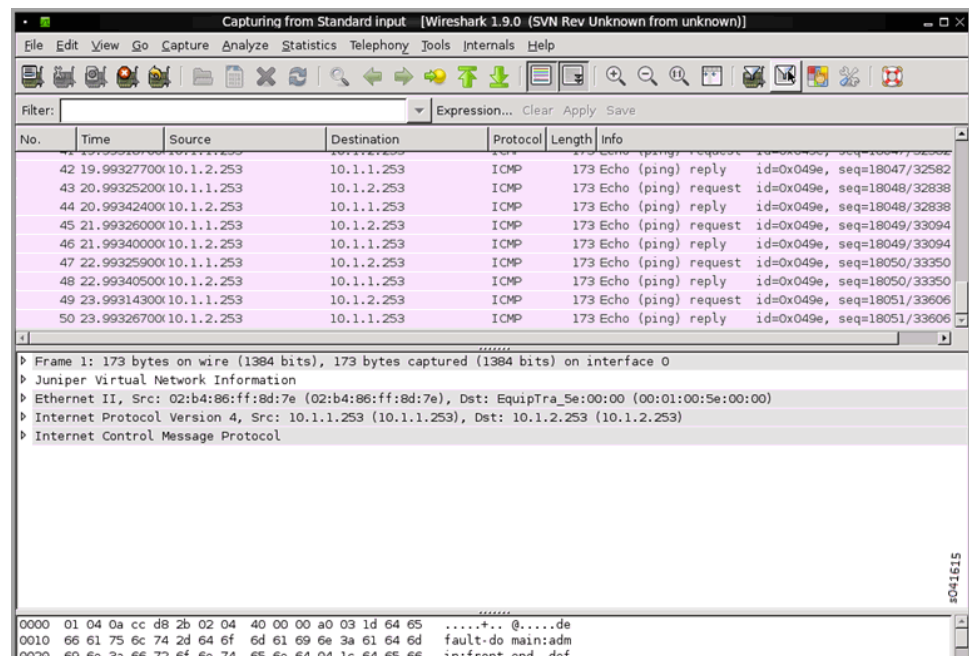
- View the analyzer activity from **Monitor > Debug > Packet Capture**. For the selected analyzer, click in the **Action** column and select **View Analyzer**; see [Figure 84 on page 261](#).

Figure 84: Launch Analyzer VM



8. The Wireshark **Packet Capture Display** appears; see [Figure 85 on page 261](#).

Figure 85: Packet Capture Display



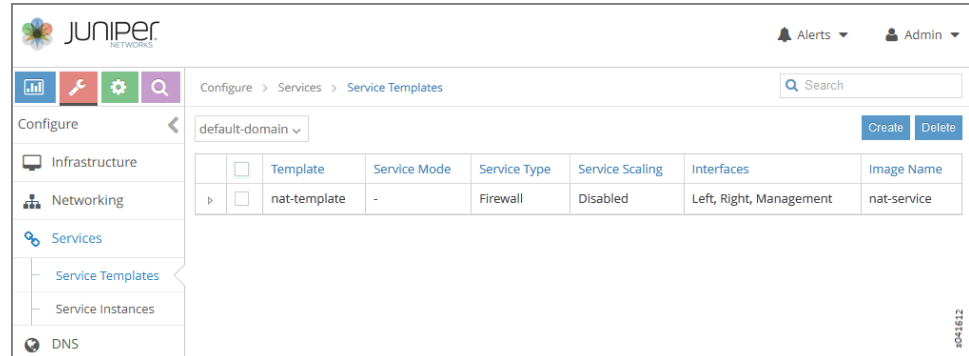
## Setting Up Traffic Mirroring Using Configure > Networking > Services

You can set up packet capture for mirroring to an analyzer within a service chain utilizing more than one interface by starting with a service template. The following procedure provides the steps needed.

1. Access **Configure > Networking > Services > Service Templates**.

The **Service Templates** screen appears; see [Figure 86 on page 262](#).

**Figure 86: Service Templates**



2. To create a new service template, click the **Create** button.

The **Add Service Template** window appears; see [Figure 87 on page 262](#).

**Figure 87: Add Service Template**

3. Complete the fields by using the guidelines in [Table 31 on page 262](#).

**Table 31: Add Service Template Fields**

Field	Description
<b>Name</b>	Enter a descriptive text name for this service template.

Table 31: Add Service Template Fields (*continued*)

Field	Description
<b>Service Mode</b>	Select <b>Transparent</b> from the drop-down list to indicate that this service template is for purposes of mirroring.
<b>Service</b>	Select <b>Analyzer</b> from the drop-down list to indicate that this service template is for a traffic analyzer.
<b>Image Name</b>	Select from a drop-down list of available images the analyzer image to use for this analyzer service template. You should select the analyzer named <b>analyzer two interfaces</b> if you used the recommended naming for the image <b>analyzer-vm-console-two-if qcow2</b> in the image list.
<b>Interface Types</b>	From the drop-down list, click the check boxes to indicate which two interface types are used for this analyzer service template: <ul style="list-style-type: none"> <li>• Left</li> <li>• Right</li> <li>• Management</li> </ul>
<b>Save</b>	When finished, click <b>OK</b> to commit the changes
<b>Cancel</b>	Click <b>Cancel</b> to clear the fields and start over.

4. Create a service instance by clicking the **Service Instances** link and clicking the **Create** button.

The **Create Service Instances** window appears; see [Figure 88 on page 263](#).

Figure 88: Create Service Instances

5. Complete the fields by using the guidelines in [Table 32 on page 263](#).

Table 32: Create Service Instances Fields

Field	Description
<b>Services Template</b>	Select from a drop-down list of available service templates the template to use for this service instance (e.g. <b>AnalyzerTemplate</b> ).

Table 32: Create Service Instances Fields (*continued*)

Field	Description
<b>Instance Name</b>	Enter a text name for this service instance.
<b>Left Virtual Network</b>	Select from a drop-down list of available networks the network for the left interface, or select <b>Automatic</b> .
<b>Right Virtual Network</b>	Select from a drop-down list of available networks the network for the right interface, or select <b>Automatic</b> .
<b>Management Virtual Network</b>	Select from a drop-down list of available networks the network for the management interface, or select <b>Automatic</b> .
<b>Save</b>	Click <b>Save</b> to commit your changes.
<b>Cancel</b>	Click <b>Cancel</b> to clear your changes and start over.

- To create a network policy rule for this service instance, click **Configure > Networking > Policies**.

The **Policies** window appears.

- Click **Create** to get to the **Create Policy** window; see [Figure 89 on page 264](#).

Figure 89: Create Policy

- Click the **+** button in the lower portion of the screen to open the **Policy Rules** fields; see [Figure 90 on page 264](#).

Figure 90: Policy Rules

9. To add policy rules, complete the fields, using the guidelines in [Table 33 on page 265](#).



**NOTE:** When there is a network policy attached to the virtual network, any conflicting rules configured for the analyzer will not take effect.

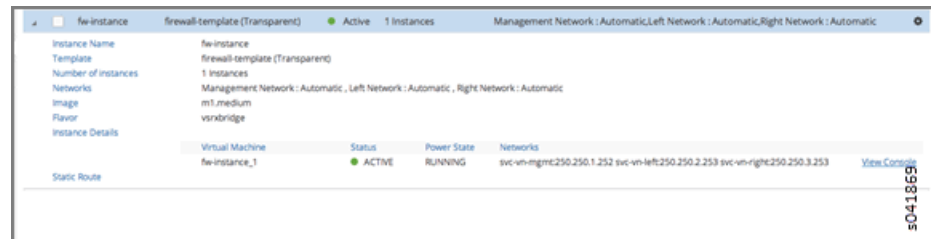
**Table 33: Add Rule Fields**

Field	Description
<b>Action</b>	Enter a text name for this service instance.
<b>Protocol</b>	Select from a drop-down list of available networks the network for the left interface, or select <b>Automatic</b> .
<b>Source Network</b>	Select from a drop-down list of available networks the network for the right interface, or select <b>Automatic</b> .
<b>Source Ports</b>	Select from a drop-down list of available networks the network for the management interface, or select <b>Automatic</b> .
<b>Direction</b>	Select the direction of flow for the packets to be captured: <ul style="list-style-type: none"> <li>• Bidirectional</li> <li>• Unidirectional</li> </ul>
<b>Destination Network</b>	Select from a list the destination network for the packets to be captured: <ul style="list-style-type: none"> <li>• any</li> <li>• local</li> <li>• <i>domain:network 1</i></li> <li>• <i>domain:network 2</i></li> <li>• <i>domain:network .....</i></li> </ul>
<b>Destination Ports</b>	Select from a list the destination network for the packets to be captured: <ul style="list-style-type: none"> <li>• any</li> <li>• local</li> <li>• <i>domain:network 1</i></li> <li>• <i>domain:network 2</i></li> <li>• <i>domain:network .....</i></li> </ul>
<b>Apply Service</b>	Check this box to open a field where you can select a service to apply.
<b>Mirror to</b>	Check this box to open a field where you can select a service to accept the mirrored packets.
<b>Save</b>	Click <b>Save</b> to commit your changes.
<b>Cancel</b>	Click <b>Cancel</b> to clear your changes and start over.

10. Click the **Mirror to** box and select the available analyzer service instance, then click **Save**.
11. To verify packet capture, at **Configure > Services > Service Instances**, select the analyzer service instance and click **View Console**.

The packet capture displays; see [Figure 91 on page 266](#). The analyzer service VM launches the Contrail enhanced Wireshark as it starts and captures the mirrored packets destined to this service.

**Figure 91: Service Instances View Console**



**NOTE:** When using the Firefox web browser, you may have difficulty viewing the mixed content presented by the View Console enhanced Wireshark option. To fix this, please enable mixed content in Firefox. Alternatively, you can select [Click here to show only console](#) to view the console information in a separate window.

## Configuring Interface Monitoring and Mirroring

Contrail supports user monitoring of traffic on any guest virtual machine interface when using the Juniper Contrail user interface.

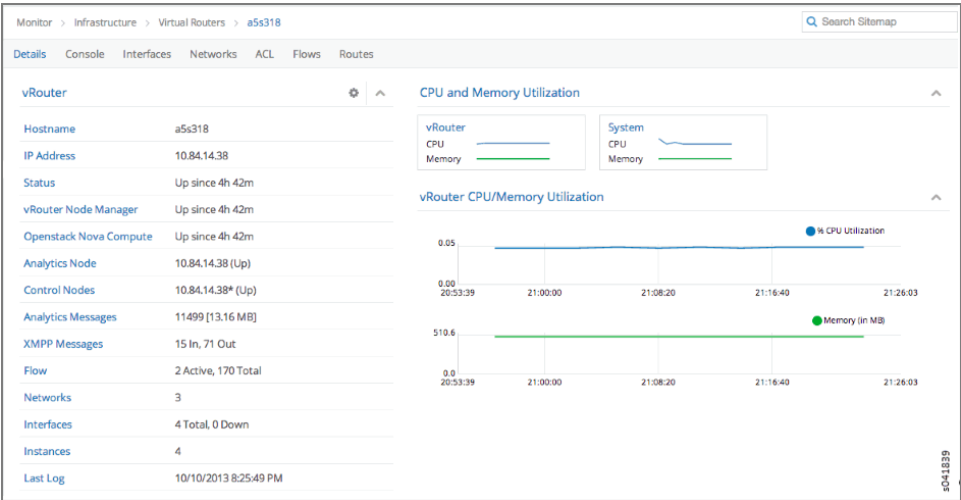
When interface monitoring (packet capture) is selected, a default analyzer is created and all traffic from the selected interface is mirrored and sent to the default analyzer. If a mirroring instance is already launched, the traffic will be redirected to the selected instance. The interface traffic is only mirrored during the time that the monitor packet capture interface is in use. When the capture screen is closed, interface mirroring stops.

To configure interface mirroring:

1. Select **Monitor > Infrastructure > Virtual Routers**, then select the vRouter that has the interface to mirror.
2. In the list of attributes for the vRouter, select Interfaces; see [Figure 92 on page 267](#).



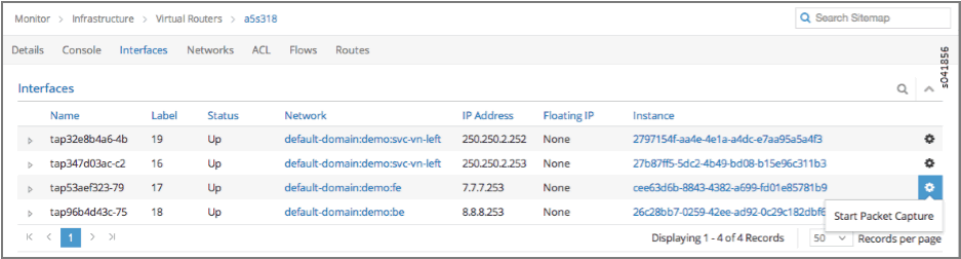
Figure 92: Individual vRouter



A list of interfaces for that vRouter appears.

3. For the interface to mirror, click the Action icon in the last column and select the option Packet Capture; see [Figure 93 on page 267](#).

Figure 93: Interfaces



The mirror packet capture starts and displays at this screen.

The mirror packet capture stops when you exit this screen.

Related  
Documentation

Analyzer Service Virtual Machine (analyzer-vm-console.qcow2)

The analyzer service virtual machine launches a Contrail-enhanced version of the network protocol analyzer Wireshark as the analyzer starts capturing mirror packets destined to the analyzer service.

- [Packet Format for Analyzer on page 268](#)
- [Metadata Format on page 268](#)
- [Wireshark Changes on page 269](#)
- [Troubleshooting Packet Display on page 269](#)

## Packet Format for Analyzer

The analyzer uses the PCAP format, which has these parts:

- Global header
- PCAP packet header
- Packet data (original packet data)

The global header is added by the analyzer service by means of the Wireshark instance. The vRouter DP uses the configured UDP session to send mirrored packets to the analyzer, adding the PCAP packet header to the packet data as it sends it over the UDP socket to the analyzer.

The following additional information is also added to the packet data as metadata:

- Captured host (IP address)
- Ingress or egress
- Action (Pass/Deny/...)
- Source VN (fully qualified name)
- Destination VN (fully qualified name)

In the existing PCAP, a network ID is added in the global header. The metadata (additional flow information) is added in front of the existing packet as follows.

```
+-----+
| Global header | Packet header| Meta data |Packet data| Packet header| Meta data
|Packet data|
+-----+
```

## Metadata Format

The metadata is in type-length-value (TLV) format as follows.

Type: 1 Byte

Length: 1 Byte

Value: up to length

Type

- 1 – Captured host IPv4 address
- 2 - Action field
- 3 – Source VN
- 4 – Destination VN
- 255 – TLV end

*Captured host address*

Length is 4 or 16 bytes based on IP address type

*Action field*

Length is 2 bytes. Multiple bits might be turned on, if there are more actions. Ingress or egress bit will be present in the Action field.

*Source VN or Destination VN*

Length is variable and up to 256 characters

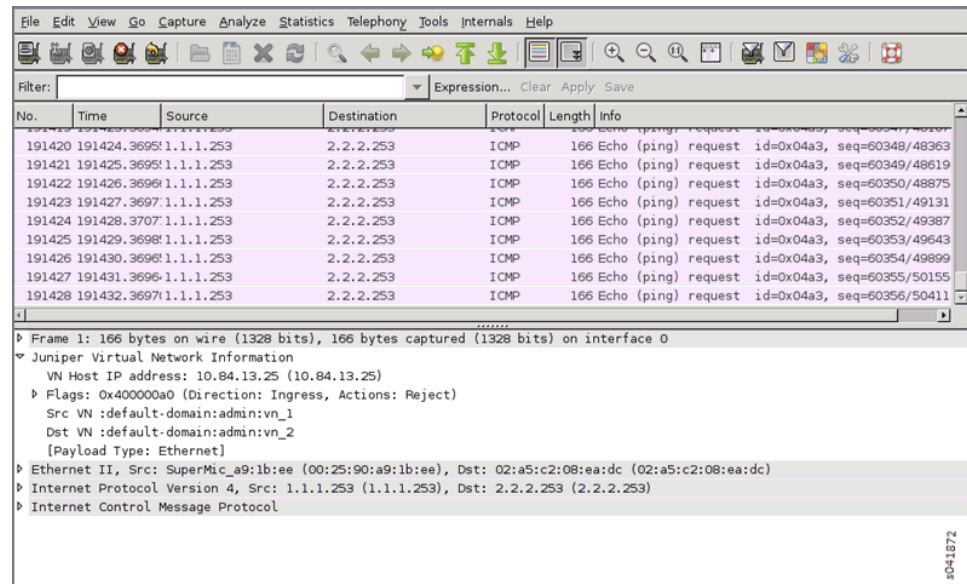
*TLV end*

A special type **255 (0xFF)** is used to identify the end of TLV entries. The TLV end must be last, at the end of the metadata.

## Wireshark Changes

A plugin is added to the Wireshark code. The plugin parses the metadata and displays the packet fields; see example in [Figure 94 on page 269](#).

**Figure 94: Wireshark Packet Display**



## Troubleshooting Packet Display

Follow these steps if the packets are not displaying:

1. Use **tcpdump** on the tap interfaces to see if packets are going towards the analyzer VM.
2. Check introspect to see whether the flow action has mirror activity in it or not.



## CHAPTER 12

# Monitoring and Troubleshooting

- [Contrail Analytics Overview on page 271](#)
- [Monitoring the System on page 273](#)
- [Monitor > Infrastructure > Dashboard on page 275](#)
- [Monitor > Infrastructure > Control Nodes on page 278](#)
- [Monitor > Infrastructure > Virtual Routers on page 285](#)
- [Monitor > Infrastructure > Analytics Nodes on page 296](#)
- [Monitor > Infrastructure > Config Nodes on page 301](#)
- [Monitor > Networking on page 304](#)
- [Query > Flows on page 312](#)
- [Query > Logs on page 319](#)
- [System Log Receiver in Contrail Analytics on page 324](#)
- [Example: Debugging Connectivity Using Monitoring for Troubleshooting on page 325](#)
- [Analytics Scalability on page 329](#)
- [High Availability for Analytics on page 330](#)

## Contrail Analytics Overview

---

Contrail is a distributed system of compute nodes, control nodes, configuration nodes, database nodes, web UI nodes, and analytics nodes.

The analytics nodes are responsible for the collection of system state information, usage statistics, and debug information from all of the software modules across all of the nodes of the system. The analytics nodes store the data gathered across the system in a database that is based on the Apache Cassandra open source distributed database management system. The database is queried by means of an SQL-like language and representational state transfer (REST) APIs.

System state information collected by the analytics nodes is aggregated across all of the nodes, and comprehensive graphical views allow the user to get up-to-date system usage information easily.

Debug information collected by the analytics nodes includes the following types:

- System log (syslog) messages—informational and debug messages generated by system software components.
- Object log messages—records of changes made to system objects such as virtual machines, virtual networks, service instances, virtual routers, BGP peers, routing instances, and the like.
- Trace messages—records of activities collected locally by software components and sent to analytics nodes only on demand.

Statistics information related to flows, CPU and memory usage, and the like is also collected by the analytics nodes and can be queried at the user interface to provide historical analytics and time-series information. The queries are performed using REST APIs.

Analytics data is written to a database in Contrail. The data expires after the default time-to-live (TTL) period of 48 hours. This default TTL time can be changed as needed by changing the value of the **database\_ttl** value in the file **testbed.py**.

**Related  
Documentation**

- [Monitoring the System on page 273](#)

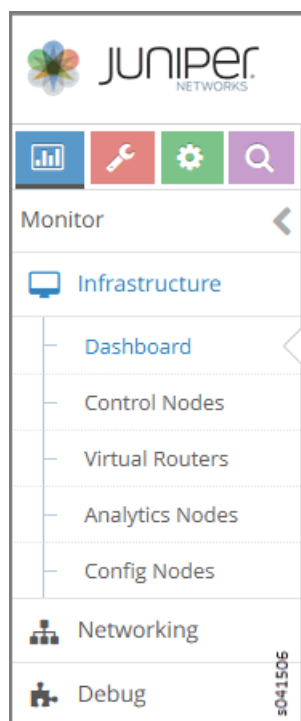
## Monitoring the System

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 95 on page 273](#)

Figure 95: Monitor Menu



See [Table 34 on page 273](#) for descriptions of the items available under each of the menu options from the **Monitor** icon.

Table 34: 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 <a href="#">“Monitor &gt; Infrastructure &gt; Dashboard” on page 275</a> .

Table 34: Monitor Menu Options (*continued*)

Option	Description
<b>Infrastructure &gt; Control Nodes</b>	<p>View a summary for all control nodes in the system, and for each control node, view:</p> <ul style="list-style-type: none"> <li>Graphical reports of memory usage and average CPU load.</li> <li>Console information for a specified time period.</li> <li>A list of all peers with details about type, ASN, and the like.</li> <li>A list of all routes, including next hop, source, local preference, and the like.</li> </ul> <p>See <a href="#">"Monitor &gt; Infrastructure &gt; Control Nodes"</a> on page 278.</p>
<b>Infrastructure &gt; Virtual Routers</b>	<p>View a summary of all vRouters in the system, and for each vRouter, view:</p> <ul style="list-style-type: none"> <li>Graphical reports of memory usage and average CPU load.</li> <li>Console information for a specified time period.</li> <li>A list of all interfaces with details such as label, status, associated network, IP address, and the like.</li> <li>A list of all associated networks with their ACLs and VRFs.</li> <li>A list of all active flows with source and destination details, size, and time.</li> </ul> <p>See <a href="#">"Monitor &gt; Infrastructure &gt; Virtual Routers"</a> on page 285.</p>
<b>Infrastructure &gt; Analytics Nodes</b>	<p>View activity for the analytics nodes, including memory and CPU usage, analytics host names, IP address, status, and more. See <a href="#">"Monitor &gt; Infrastructure &gt; Analytics Nodes"</a> on page 296.</p>
<b>Infrastructure &gt; Config Nodes</b>	<p>View activity for the config nodes, including memory and CPU usage, config host names, IP address, status, and more. See <a href="#">"Monitor &gt; Infrastructure &gt; Config Nodes"</a> on page 301.</p>
<b>Networking &gt; Networks</b>	<p>For all virtual networks for all projects in the system, view graphical traffic statistics, including:</p> <ul style="list-style-type: none"> <li>Total traffic in and out.</li> <li>Inter VN traffic in and out.</li> <li>The most active ports, peers, and flows for a specified duration.</li> <li>All traffic ingress and egress from connected networks, including their attached policies.</li> </ul> <p>See <a href="#">"Monitor &gt; Networking"</a> on page 304.</p>
<b>Networking &gt; Dashboard</b>	<p>For all virtual networks for all projects in the system, view graphical traffic statistics, including:</p> <ul style="list-style-type: none"> <li>Total traffic in and out.</li> <li>Inter VN traffic in and out.</li> </ul> <p>You can view the statistics in varying levels of granularity, for example, for a whole project, or for a single network. See <a href="#">"Monitor &gt; Networking"</a> on page 304.</p>
<b>Networking &gt; Projects</b>	<p>View essential information about projects in the system including name, associated networks, and traffic in and out.</p>



Table 34: Monitor Menu Options (*continued*)

Option	Description
<b>Networking &gt; Networks</b>	View essential information about networks in the system including name and traffic in and out.
<b>Networking &gt; Instances</b>	View essential information about instances in the system including name, associated networks, interfaces, vRouters, and traffic in and out.
<b>Debug &gt; Packet Capture</b>	<ul style="list-style-type: none"> <li>• Add and manage packet analyzers.</li> <li>• Attach packet captures and configure their details.</li> <li>• View a list of all packet analyzers in the system and the details of their configurations, including source and destination networks, ports, and IP addresses.</li> </ul>

**Related Documentation**

- [Monitor > Infrastructure > Dashboard on page 275](#)
- [Monitor > Infrastructure > Control Nodes on page 278](#)
- [Monitor > Infrastructure > Virtual Routers on page 285](#)
- [Monitor > Networking on page 304](#)
- [Query > Logs on page 319](#)
- [Query > Flows on page 312](#)

## 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 275](#)
- [Monitor Individual Details from the Dashboard on page 276](#)
- [Using Bubble Charts on page 277](#)
- [Color-Coding of Bubble Charts on page 277](#)

## Monitor Dashboard

Click **Monitor > Infrastructure > Dashboard** on the left to view the **Dashboard**. See [Figure 96 on page 276](#).

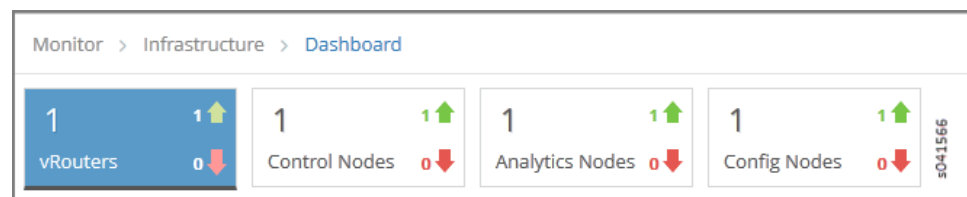
Figure 96: Monitor &gt; Infrastructure &gt; Dashboard



## 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 97 on page 276](#). 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 97: Dashboard Summary Boxes



Detailed information about monitoring each of the areas represented by the boxes is provided in the links in [Table 35 on page 276](#).

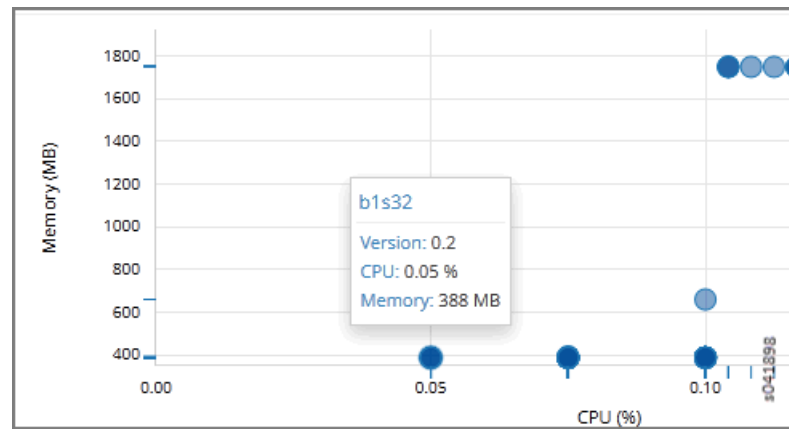
Table 35: Dashboard Summary Boxes

Box	For More Information
vRouters	<a href="#">“Monitor &gt; Infrastructure &gt; Virtual Routers” on page 285</a>
Control Nodes	<a href="#">“Monitor &gt; Infrastructure &gt; Control Nodes” on page 278</a>
Analytics Nodes	<a href="#">“Monitor &gt; Infrastructure &gt; Analytics Nodes” on page 296</a>
Config Nodes	<a href="#">“Monitor &gt; Infrastructure &gt; Config Nodes” on page 301</a>

## 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 98 on page 277](#). You can click through the summary information to get more details about the component.

**Figure 98: 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 285](#)
- [Monitor > Infrastructure > Control Nodes on page 278](#)
- [Monitor > Infrastructure > Analytics Nodes on page 296](#)
- [Monitor > Infrastructure > Config Nodes on page 301](#)

## Monitor > Infrastructure > Control Nodes

Use **Monitor > Infrastructure > Control Nodes** to gain insight into usage statistics for control nodes.

- [Monitor Control Nodes Summary on page 278](#)
- [Monitor Individual Control Node Details on page 279](#)
- [Monitor Individual Control Node Console on page 280](#)
- [Monitor Individual Control Node Peers on page 282](#)
- [Monitor Individual Control Node Routes on page 283](#)

### 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 99 on page 278](#). See [Table 36 on page 278](#) for descriptions of the fields on this screen.

Figure 99: Control Nodes Summary



Table 36: 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.

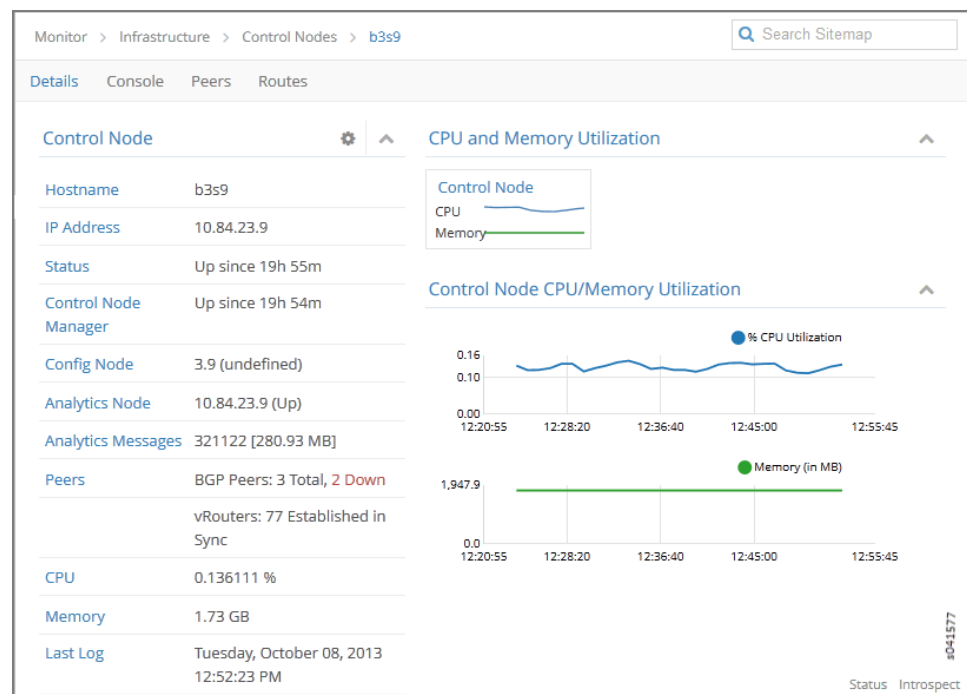
Table 36: Control Nodes Summary Fields (*continued*)

Field	Description
<b>Memory</b>	The memory in MB currently in use and the total memory available for this control node.
<b>Total Peers</b>	The total number of peers for this control node.
<b>Established in Sync Peers</b>	The total number of peers in sync for this control node.
<b>Established in Sync vRouters</b>	The total number of vRouters in sync for this control node.

## Monitor Individual Control Node Details

Click the name of any control nodes listed under the **Control Nodes** title to view an array of graphical reports of usage and numerous details about that node. There are several tabs available to help you probe into more details about the selected control node. The first tab is the **Details** tab; see [Figure 100 on page 279](#).

Figure 100: 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 37 on page 280](#) for descriptions of the fields on this tab.

Table 37: 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.
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 101 on page 281](#).

Figure 101: 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 38 on page 281](#) for descriptions of the fields on the **Console** tab screen.

Table 38: Control Node: Console Tab Fields

Field	Description
<b>Time Range</b>	Select a timeframe for which to review logging information as sent to the console. There are 11 options, ranging from the <b>Last 5 mins</b> through to the <b>Last 24 hrs</b> . The default display is for the <b>Last 5 mins</b> .
<b>Log Category</b>	Select a log category to display: <ul style="list-style-type: none"> <li>All</li> <li>_default_</li> <li>XMPP</li> <li>IFMap</li> <li>TCP</li> </ul>
<b>Log Type</b>	Select a log type to display.

Table 38: Control Node: Console Tab Fields (*continued*)

Field	Description
<b>Log Level</b>	Select a log severity level to display:  SYS_EMERG SYS_ALERT SYS_CRIT SYS_ERR SYS_WARN SYS_NOTICE SYS_INFO SYS_DEBUG
<b>Search</b>	Enter any text string to search and display logs containing that string.
<b>Limit</b>	Select from a list an amount to limit the number of messages displayed:  No Limit Limit 10 messages Limit 50 messages Limit 100 messages Limit 200 messages Limit 500 messages
<b>Auto Refresh</b>	Click the check box to automatically refresh the display if more messages occur.
<b>Display Logs</b>	Click this button to refresh the display if you change the display criteria.
<b>Reset</b>	Click this button to clear any selected display criteria and reset all criteria to their default settings.
<b>Time</b>	This column lists the time received for each log message displayed.
<b>Category</b>	This column lists the log category for each log message displayed.
<b>Log Type</b>	This column lists the log type for each log message displayed.
<b>Log</b>	This column lists the log message for each log displayed.

## Monitor 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 102 on page 283](#).



Figure 102: Individual Control Node—Peers Tab

Monitor > Infrastructure > Control Nodes > b3s9

Search Sitemap

Details Console **Peers** Routes

Peers

Peer	Peer Type	Peer ASN	Status	Last flap	Messages (Recv/Sent)
10.84.23.252	BGP	64512	Active, -	-	0/ 0
10.84.23.8	BGP	64512	Established, in sync	-	3754/ 3758
10.84.23.253	BGP	64512	Connect, -	-	0/ 0
10.84.21.4	XMPP	-	Established, in sync	-	2751/ 5189
10.84.21.5	XMPP	-	Established, in sync	-	2753/ 5802
10.84.21.6	XMPP	-	Established, in sync	-	2752/ 4264
10.84.21.34	XMPP	-	Established, in sync	-	2753/ 5659

Details:

```

- {
  name: "b3s9:10.84.21.34",
  value: - {
    XmppPeerInfoData: - {
      state_info: - {
        last_state: "Active",
        state: "Established",
        last_state_at: 1381190447915913
      },
    },
    peer_stats_info: - {

```

See Table 39 on page 283 for descriptions of the fields on the **Peers** tab screen.

Table 39: 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 103 on page 284.

Figure 103: Individual Control Node—Routes Tab

Details Console Peers **Routes**

Routing Instance: All Address Family: All Limit 50 Routes

Peer Source: All Prefix: Protocol: All

Display Routes Reset

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 40 on page 284](#) for descriptions of the fields on the **Routes** tab screen.

Table 40: Control Node: Routes Tab Fields

Field	Description
<b>Routing Instance</b>	You can select a single routing instance from a list of all instances for which to display the active routes.
<b>Address Family</b>	Select an address family for which to display the active routes: <ul style="list-style-type: none"> <li>All (default)</li> <li>l3vpn</li> <li>inet</li> <li>inetmcast</li> </ul>
<b>(Limit Field)</b>	Select to limit the display of active routes: <ul style="list-style-type: none"> <li>Limit 10 Routes</li> <li>Limit 50 Routes</li> <li>Limit 100 Routes</li> <li>Limit 200 Routes</li> </ul>
<b>Peer Source</b>	Select from a list of available peers the peer for which to display the active routes, or select All.
<b>Prefix</b>	Enter a route prefix to limit the display of active routes to only those with the designated prefix.

Table 40: Control Node: Routes Tab Fields (*continued*)

Field	Description
<b>Protocol</b>	Select a protocol for which to display the active routes:  All (default) XMPP BGP ServiceChain Static
<b>Display Routes</b>	Click this button to refresh the display of routes after selecting different display criteria.
<b>Reset</b>	Click this button to clear any selected criteria and return the display to default values.
<i>Column</i>	<i>Description</i>
<b>Routing Table</b>	The name of the routing table that stores this route.
<b>Prefix</b>	The route prefix for each active route displayed.
<b>Protocol</b>	The protocol used by the route.
<b>Source</b>	The host source for each active route displayed.
<b>Next hop</b>	The IP address of the next hop for each active route displayed.
<b>Label</b>	The label for each active route displayed.
<b>Security</b>	The security value for each active route displayed.
<b>Origin VN</b>	The virtual network from which the route originates.
<b>AS Path</b>	The AS path for each active route displayed.

## Monitor > Infrastructure > Virtual Routers

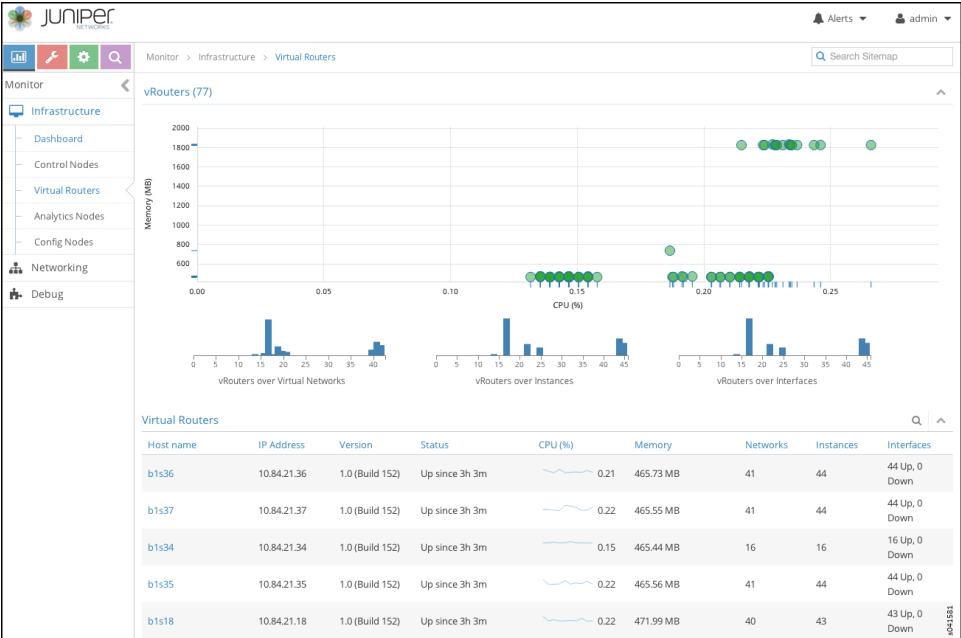
- [Monitor vRouters Summary on page 286](#)
- [Monitor Individual vRouters Tabs on page 287](#)
- [Monitor Individual vRouter Details Tab on page 287](#)
- [Monitor Individual vRouters Interfaces Tab on page 288](#)
- [Configuring Interface Monitoring and Mirroring on page 289](#)
- [Monitor Individual vRouters Networks Tab on page 290](#)
- [Monitor Individual vRouters ACL Tab on page 291](#)
- [Monitor Individual vRouters Flows Tab on page 292](#)

- [Monitor Individual vRouters Routes Tab on page 293](#)
- [Monitor Individual vRouter Console Tab on page 294](#)

Monitor vRouters Summary

Click **Monitor > Infrastructure > Virtual Routers** to view the **vRouters** summary screen. See [Figure 104 on page 286](#).

Figure 104: vRouters Summary



See [Table 41 on page 286](#) for descriptions of the fields on the **vRouters Summary** screen.

Table 41: 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 41: vRouters Summary Fields (*continued*)

Field	Description
<b>Interfaces</b>	The total number of interfaces for this vRouter.

## Monitor Individual vRouters Tabs

Click the name of any vRouter to view details about performance and activities for that vRouter. Each individual vRouters screen has the following tabs.

- **Details**—similar display of information as on individual control nodes **Details** tab. See [Figure 105 on page 287](#).
- **Console**—similar display of information as on individual control nodes **Console** tab. See [Figure 113 on page 295](#).
- **Interfaces**—details about associated interfaces. See [Figure 106 on page 289](#).
- **Networks**—details about associated networks. See [Figure 109 on page 291](#).
- **ACL**—details about access control lists. See [Figure 110 on page 292](#).
- **Flows**—details about associated traffic flows. See [Figure 111 on page 293](#).
- **Routes**—details about associated routes. See [Figure 112 on page 294](#).

## 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 105 on page 287](#). See [Table 42 on page 287](#) for descriptions of the fields on this tab.

Figure 105: Individual vRouters—Details Tab

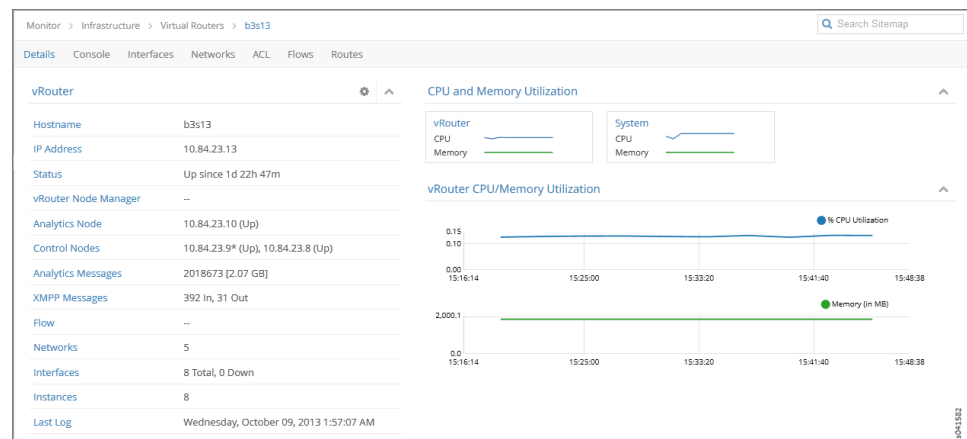


Table 42: vRouters Details Tab Fields

Field	Description
<b>Hostname</b>	The hostname of the vRouter.

Table 42: 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 106 on page 289](#). See [Table 43 on page 289](#) for descriptions of the fields on the **Interfaces** tab screen.

Figure 106: Individual vRouters—Interfaces Tab

Name	Label	Status	Network	IP Address	Floating IP	Instance
tap25e5cee3-07	18	Up	default-domain:demo:vn30	192.168.30.247	None	005132fd-0d83-4db7-88c8-bd49d68e9480
tap4d91aab1-f1	25	Up	default-domain:demo:vn26	192.168.26.247	None	65d6c6e9-7a82-43d8-a706-f74d81715920
tap5a8cd9dd-5b	27	Up	default-domain:demo:vn23	192.168.23.249	None	a159c518-4fb6-402a-ae0d-eb5b4457b551
tap603a5e0b-8b	16	Up	default-domain:demo:vn19	192.168.19.247	None	fe622580-b0cf-4c6d-89e5-d2065e7e87e4
tap68ad232c-76	19	Up	default-domain:demo:vn28	192.168.28.247	None	91089d89-76b5-46c2-abc9-b9693bcb37ac

Details :

```

- {
  index: "6",
  name: "tap68ad232c-76",
  uuid: "68ad232c-76d1-4fe2-a200-42182497545e",
  vrf_name: "default-domain:demo:vn28:vn28",
  active: "Active",
  dhcp_service: "Enable",

```

Table 43: vRouters: Interfaces Tab Fields

Field	Description
Name	The name of the interface.
Label	The label for the interface.
Status	The current status of the interface.
Network	The network associated with the interface.
IP Address	The IP address of the interface.
Floating IP	Displays any floating IP addresses associated with the interface.
Instance	The name of any instance associated with the interface.

## Configuring Interface Monitoring and Mirroring

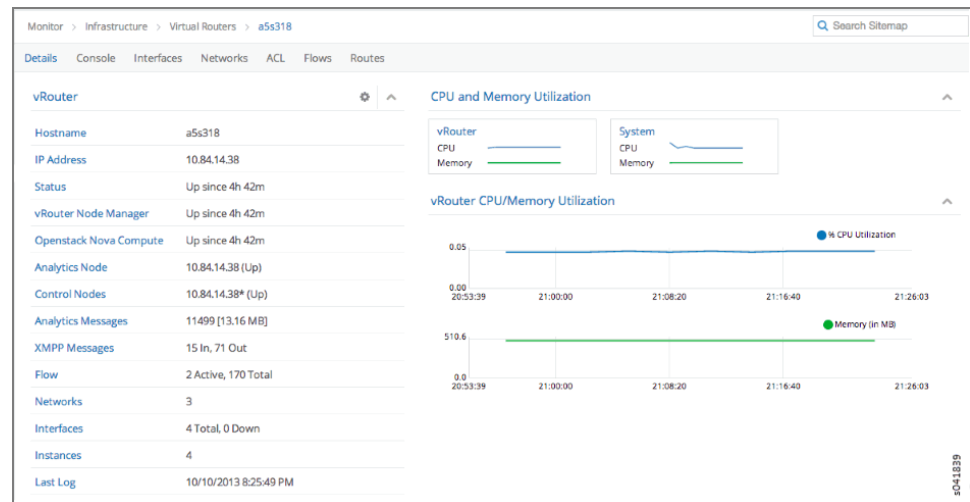
Contrail supports user monitoring of traffic on any guest virtual machine interface when using the Juniper Contrail user interface.

When interface monitoring (packet capture) is selected, a default analyzer is created and all traffic from the selected interface is mirrored and sent to the default analyzer. If a mirroring instance is already launched, the traffic will be redirected to the selected instance. The interface traffic is only mirrored during the time that the monitor packet capture interface is in use. When the capture screen is closed, interface mirroring stops.

To configure interface mirroring:

1. Select **Monitor > Infrastructure > Virtual Routers**, then select the vRouter that has the interface to mirror.
2. In the list of attributes for the vRouter, select **Interfaces**; see [Figure 92 on page 267](#).

**Figure 107: Individual vRouter**



A list of interfaces for that vRouter appears.

3. For the interface to mirror, click the Action icon in the last column and select the option **Packet Capture**; see [Figure 93 on page 267](#).

**Figure 108: Interfaces**

Name	Label	Status	Network	IP Address	Floating IP	Instance
tap32e8b4a6-4b	19	Up	default-domain:demo:svc-vn-left	250.250.2.252	None	2797154f-a4fe-4e1a-a4dc-e7aa95a5a4f3
tap347d03ac-c2	16	Up	default-domain:demo:svc-vn-left	250.250.2.253	None	27b87ff5-5dc2-4b49-bd08-b15e96c311b3
tap53ae323-79	17	Up	default-domain:demo:fe	7.7.7.253	None	cee63d6b-8843-4382-a699-fd01e85781b9
tap96b4d43c-75	18	Up	default-domain:demo:be	8.8.8.253	None	26c28bb7-0259-42ee-ad92-0c29c182dbfe

The mirror packet capture starts and displays at this screen.

The mirror packet capture stops when you exit this screen.

## 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 109 on page 291](#). See [Table 44 on page 291](#) for descriptions of the fields on the **Networks** tab screen.



Figure 109: Individual vRouters—Networks Tab

The screenshot shows the 'Networks' tab in the vRouter configuration interface. The breadcrumb trail is 'Monitor > Infrastructure > Virtual Routers > b1s36'. A search bar is present. The 'Networks' tab is selected, with other tabs like 'Details', 'Console', 'Interfaces', 'ACL', 'Flows', and 'Routes' visible. The main content area displays a table of networks with columns for Name, ACLs, and VRF. Each row has an expansion arrow and a gear icon. Below the table, the 'Details' for the selected network 'default-domain:demo:vn2' are shown in a JSON format.

Name	ACLs	VRF
default-domain:demo:vn24	a372751f-6497-41e9-b409-fa4ab5ce6b7f	default-domain:demo:vn24:vn24
default-domain:demo:vn22	195af177-0a28-49a1-9cf0-2ceac22af5a1	default-domain:demo:vn22:vn22
default-domain:demo:vn30	362cce6e-2894-42d6-ba03-3ee98cac8809	default-domain:demo:vn30:vn30
default-domain:demo:vn21	5918a068-1cd5-4993-9cff-386a807940ca	default-domain:demo:vn21:vn21
default-domain:demo:vn28	dd87c461-97c0-4d47-bff0-89040e7d6ab0	default-domain:demo:vn28:vn28
default-domain:demo:vn19	f0465432-6fc0-4fb3-967c-392100617408	default-domain:demo:vn19:vn19
default-domain:demo:vn2	1c46e7e0-f799-4bc6-ae09-e4654c263aa6	default-domain:demo:vn2:vn2

Details:

```

- {
  name: "default-domain:demo:vn2",
  uuid: "63d08f7a-b342-4892-9171-edab9f4c397f",
  acl_uuid: "1c46e7e0-f799-4bc6-ae09-e4654c263aa6",
  mirror_acl_uuid: - {},
  mirror_cfg_acl_uuid: - {},
  vrf_name: "default-domain:demo:vn2:vn2",
  ipam_data: - {
    list: - {

```

Table 44: 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 110 on page 292](#). See [Table 45 on page 292](#) for descriptions of the fields on the **ACL** tab screen.

Figure 110: 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: - {
      AcEntrySandeshData: - [
        - {
          ace_id: "1",

```

Table 45: 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 111 on page 293. See Table 46 on page 293 for descriptions of the fields on the **Flows** tab screen.

Figure 111: Individual vRouters—Flows Tab

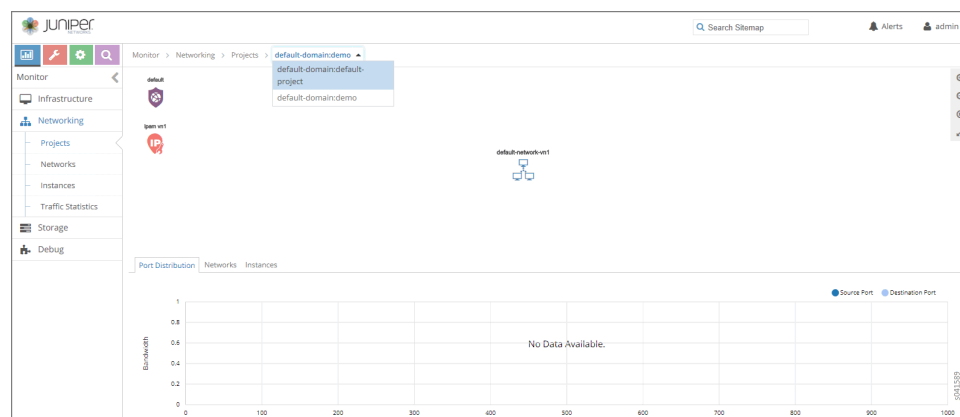


Table 46: vRouters: Flows Tab Fields

Field	Description
ACL UUID	The default is to show <b>All</b> flows, however, you can select from a drop down list any single flow to view its details.
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 112 on page 294. See Table 47 on page 294 for descriptions of the fields on the **Routes** tab screen.

Figure 112: Individual vRouters—Routes Tab

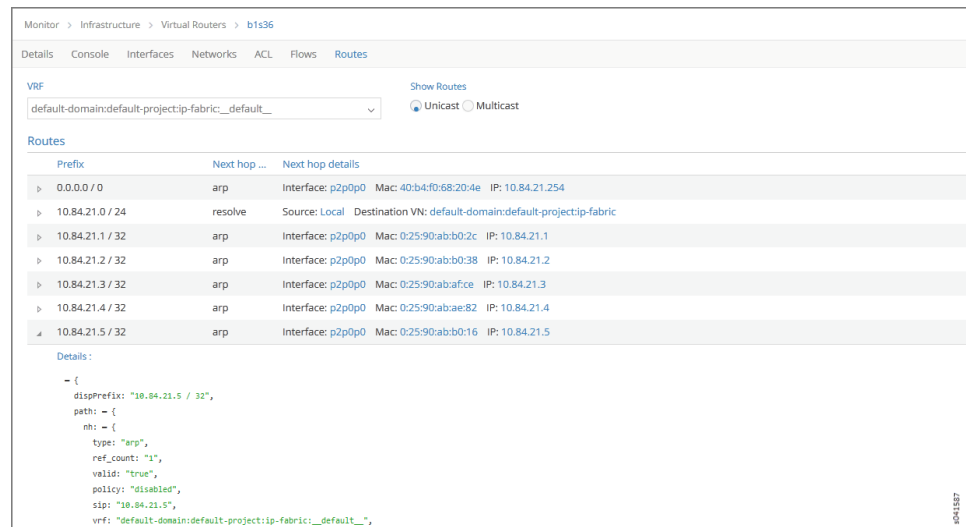


Table 47: vRouters: Routes Tab Fields

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: <b>Unicast</b> or <b>Multicast</b> .
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 113 on page 295](#). See [Table 48 on page 295](#) for descriptions of the fields on the **Console** tab screen.

Figure 113: Individual vRouter—Console Tab

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 48: Control Node: Console Tab Fields

Field	Description
<b>Time Range</b>	Select a timeframe for which to review logging information as sent to the console. There are several options, ranging from <b>Last 5 mins</b> through to the <b>Last 24 hrs</b> , plus a <b>Custom</b> time range.
<b>From Time</b>	If you select <b>Custom</b> in <b>Time Range</b> , enter the start time.
<b>To Time</b>	If you select <b>Custom</b> in <b>Time Range</b> , enter the end time.
<b>Log Category</b>	Select a log category to display: <ul style="list-style-type: none"> <li>• All</li> <li>• _default_</li> <li>• XMPP</li> <li>• IFMap</li> <li>• TCP</li> </ul>
<b>Log Type</b>	Select a log type to display.
<b>Log Level</b>	Select a log severity level to display: <ul style="list-style-type: none"> <li>• SYS_EMERG</li> <li>• SYS_ALERT</li> <li>• SYS_CRIT</li> <li>• SYS_ERR</li> <li>• SYS_WARN</li> <li>• SYS_NOTICE</li> <li>• SYS_INFO</li> <li>• SYS_DEBUG</li> </ul>

Table 48: Control Node: Console Tab Fields (*continued*)

Field	Description
<b>Limit</b>	Select from a list an amount to limit the number of messages displayed: <ul style="list-style-type: none"> <li>• No Limit</li> <li>• Limit 10 messages</li> <li>• Limit 50 messages</li> <li>• Limit 100 messages</li> <li>• Limit 200 messages</li> <li>• Limit 500 messages</li> </ul>
<b>Auto Refresh</b>	Click the check box to automatically refresh the display if more messages occur.
<b>Display Logs</b>	Click this button to refresh the display if you change the display criteria.
<b>Reset</b>	Click this button to clear any selected display criteria and reset all criteria to their default settings.
<i>Columns</i>	
<b>Time</b>	This column lists the time received for each log message displayed.
<b>Category</b>	This column lists the log category for each log message displayed.
<b>Log Type</b>	This column lists the log type for each log message displayed.
<b>Log</b>	This column lists the log message for each log displayed.

## Monitor > Infrastructure > Analytics Nodes

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 296](#)
- [Monitor Analytics Individual Node Details Tab on page 297](#)
- [Monitor Analytics Individual Node Generators Tab on page 298](#)
- [Monitor Analytics Individual Node QE Queries Tab on page 299](#)
- [Monitor Analytics Individual Node Console Tab on page 300](#)

## Monitor Analytics Nodes

Select **Monitor > Infrastructure > Analytics Nodes** to view a summary of activities for the analytics nodes; see [Figure 114 on page 297](#). See [Table 49 on page 297](#) for descriptions of the fields on the analytics summary.

Figure 114: Analytics Nodes Summary



Table 49: 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 115 on page 298](#).

See [Table 50 on page 298](#) for descriptions of the fields on this screen.

Figure 115: Monitor Analytics Individual Node Details Tab

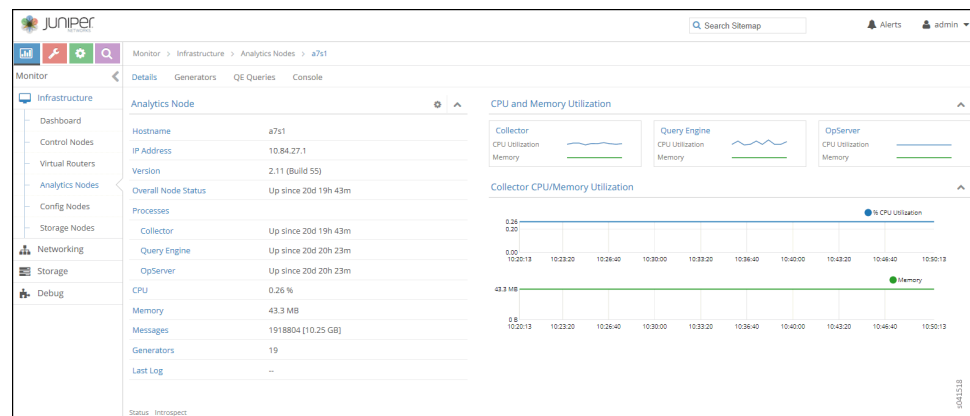


Table 50: Monitor Analytics Individual Node Details Tab Fields

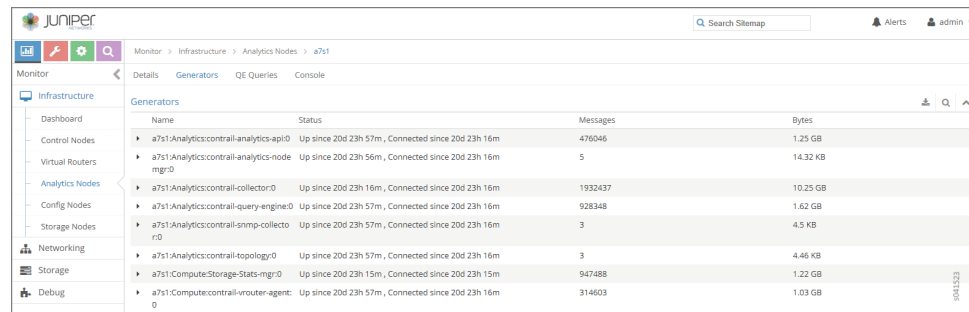
Field	Description
<b>Hostname</b>	The name of this node.
<b>IP Address</b>	The IP address of this node.
<b>Version</b>	The installed version of the software.
<b>Overall Node Status</b>	The current operational status of the node — Up or Down — and the length of time in this state.
<b>Processes</b>	The current status of each analytics process, including Collector, Query Engine, and OpServer.
<b>CPU (%)</b>	The average CPU percentage usage for this node.
<b>Memory</b>	The average memory usage of this node.
<b>Messages</b>	The total number of messages for this node.
<b>Generators</b>	The total number of generators associated with this node.
<b>Last Log</b>	The date and time of the last log message issued about this node.

### Monitor Analytics Individual Node Generators Tab

The **Generators** tab displays information about the generators for an individual analytics node; see [Figure 116 on page 299](#). Click the expansion arrow next to any generator name to reveal more details. See [Table 51 on page 299](#) for descriptions of the fields on the **Peers** tab screen.



Figure 116: 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 51: 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 117 on page 299](#).

See [Table 52 on page 299](#) for descriptions of the fields on the **QE Queries** tab screen.

Figure 117: Individual Analytics Node—QE Queries Tab



Enqueue Time	Query	Progress
No QE Queries to display		

Table 52: 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 118 on page 300](#). See [Table 53 on page 300](#) for descriptions of the fields on the **Console** tab screen.

**Figure 118: Analytics Individual Node—Console Tab**

**Table 53: Monitor Analytics Individual Node Console Tab Fields**

Field	Description
<b>Time Range</b>	Select a timeframe for which to review logging information as sent to the console. There are 11 options, ranging from the <b>Last 5 mins</b> through to the <b>Last 24 hrs</b> . The default display is for the <b>Last 5 mins</b> .
<b>Log Category</b>	Select a log category to display: <ul style="list-style-type: none"> <li>All</li> <li>_default_</li> <li>XMPP</li> <li>IFMap</li> <li>TCP</li> </ul>
<b>Log Type</b>	Select a log type to display.
<b>Log Level</b>	Select a log severity level to display: <ul style="list-style-type: none"> <li>SYS_EMERG</li> <li>SYS_ALERT</li> <li>SYS_CRIT</li> <li>SYS_ERR</li> <li>SYS_WARN</li> <li>SYS_NOTICE</li> <li>SYS_INFO</li> <li>SYS_DEBUG</li> </ul>
<b>Keywords</b>	Enter any text string to search for and display logs containing that string.

Table 53: 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
<b>Auto Refresh</b>	Click the check box to automatically refresh the display if more messages occur.
<b>Display Logs</b>	Click this button to refresh the display if you change the display criteria.
<b>Reset</b>	Click this button to clear any selected display criteria and reset all criteria to their default settings.
<b>Time</b>	This column lists the time received for each log message displayed.
<b>Category</b>	This column lists the log category for each log message displayed.
<b>Log Type</b>	This column lists the log type for each log message displayed.
<b>Log</b>	This column lists the log message for each log displayed.

## Monitor > Infrastructure > Config Nodes

Select **Monitor > Infrastructure > Config Nodes** to view the information about the system config nodes.

- [Monitor Config Nodes on page 301](#)
- [Monitor Individual Config Node Details on page 302](#)
- [Monitor Individual Config Node Console on page 303](#)

## Monitor Config Nodes

Select **Monitor > Infrastructure > Config Nodes** to view a summary of activities for the analytics nodes. See [Figure 119 on page 302](#).

Figure 119: Config Nodes Summary

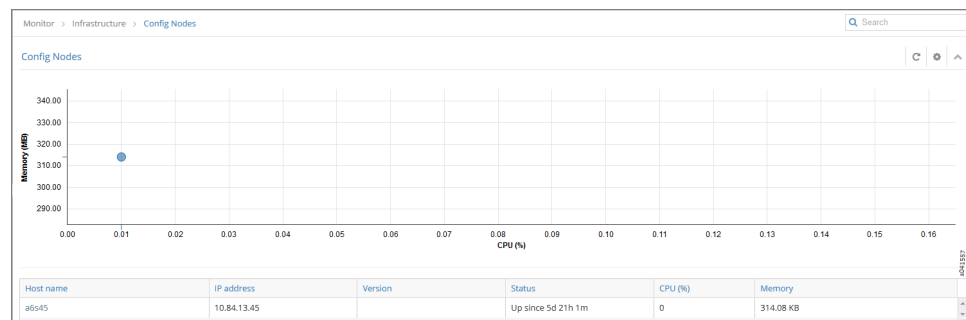


Table 54 on page 302 describes the fields in the Config Nodes summary.

Table 54: 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 120 on page 302](#).

Figure 120: Individual Config Nodes— Details Tab

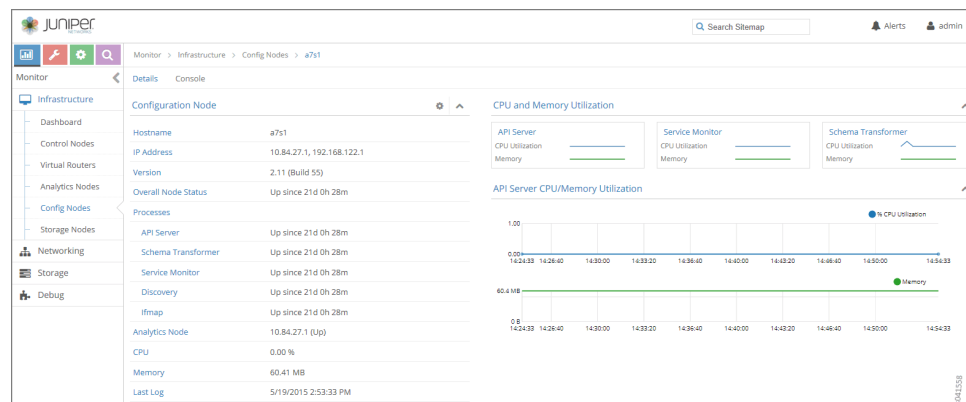


Table 55 on page 303 describes the fields on the Details screen.

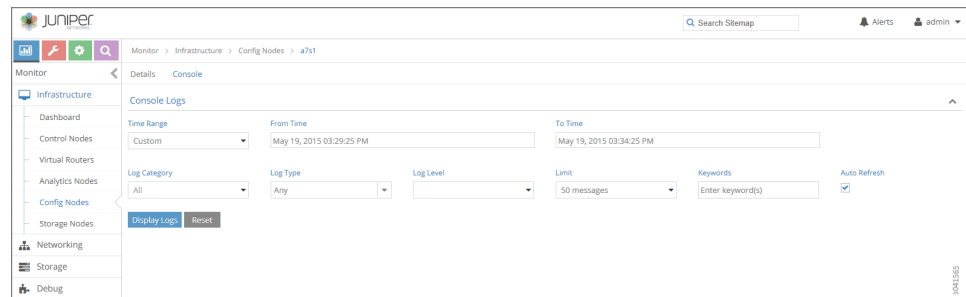
Table 55: Individual Config Nodes— Details Tab Fields

Field	Description
Hostname	The name of the config node.
IP Address	The IP address of this node.
Version	The installed version of the software.
Overall Node Status	The current operational status of the node — Up or Down — and the length of time it is in this state.
Processes	The current operational status of the processes associated with the config node, including AI Server, Schema Transformer, Service Monitor, Discovery, and Ifmap.
Analytics Node	The analytics node associated with this node.
CPU (%)	The average CPU percentage usage for this node.
Memory	The average memory usage by this node.

## Monitor Individual Config Node Console

Click the **Console** tab for an individual config node to display system logging information for a defined time period. See [Figure 121 on page 303](#).

Figure 121: Individual Config Node—Console Tab



See [Table 56 on page 303](#) for descriptions of the fields on the **Console** tab screen.

Table 56: 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 56: 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

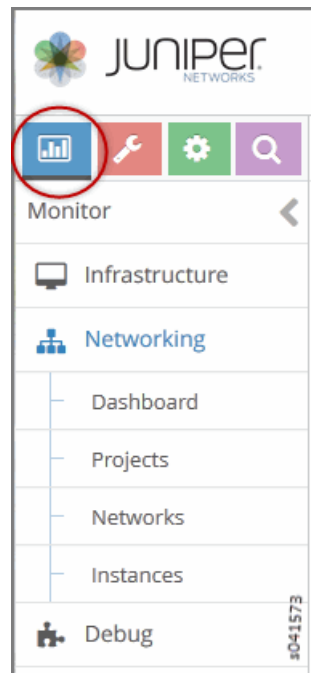
The **Monitor -> Networking** pages give an overview of the networking traffic statistics and health of domains, projects within domains, virtual networks within projects, and virtual machines within virtual networks.

- [Monitor > Networking Menu Options on page 304](#)
- [Monitor -> Networking -> Dashboard on page 305](#)
- [Monitor > Networking > Projects on page 306](#)
- [Monitor Projects Detail on page 307](#)
- [Monitor > Networking > Networks on page 309](#)

### Monitor > Networking Menu Options

Figure 122 on page 305 shows the menu options available under **Monitor > Networking**.

Figure 122: 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 123 on page 305](#).

Figure 123: Traffic Statistics for Domain Window

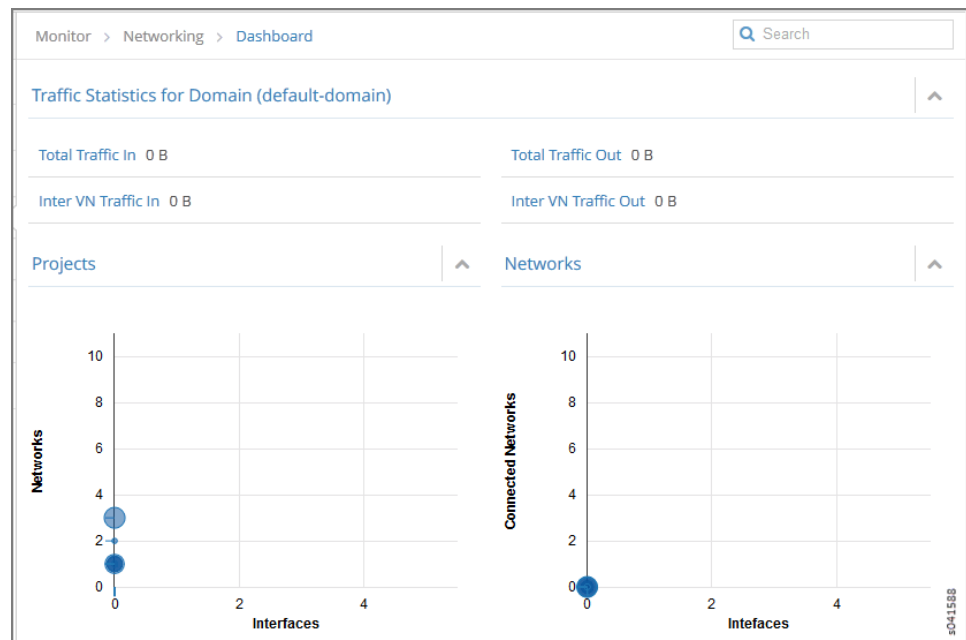


Table 57 on page 306 describes the fields in the Traffic Statistics for Domain window.

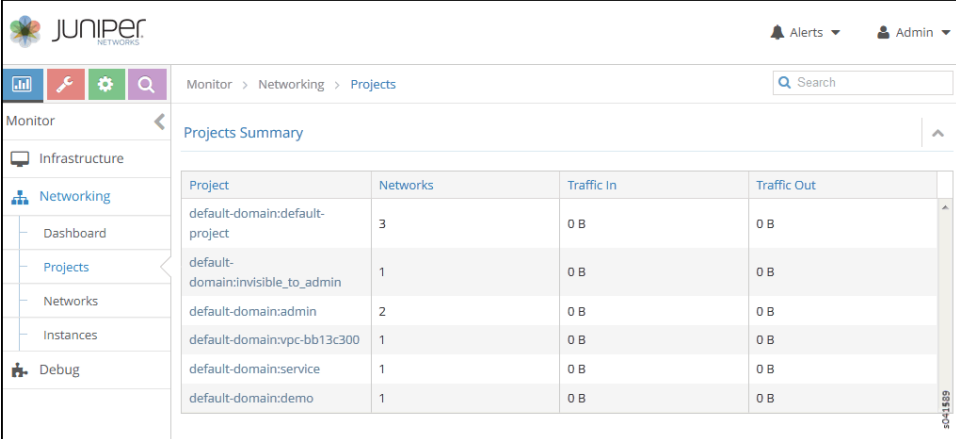
Table 57: Projects Summary Fields

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 <b>Projects</b> then select <b>Monitor &gt; Networking &gt; Projects</b> , to display more detailed statistics.
Networks	This chart displays the networks for projects with the most throughput over the past 30 minutes. Click <b>Networks</b> then select <b>Monitor &gt; Networking &gt; Networks</b> , to display more detailed statistics.

## Monitor > Networking > Projects

Select **Monitor > Networking > Projects** to see information about projects in the system. See Figure 124 on page 306.

Figure 124: 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 58 on page 306 for descriptions of the fields on this screen.

Table 58: 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.



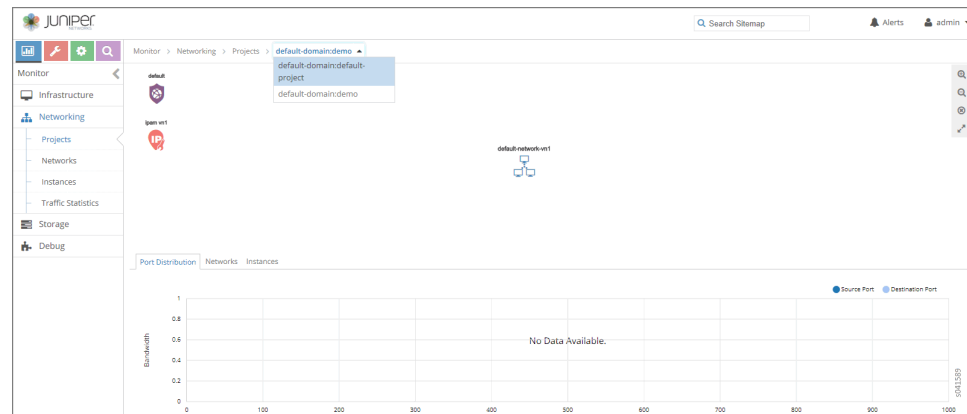
Table 58: Projects Summary Fields (*continued*)

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 125 on page 307](#). Hover over any of the connections to get more details.

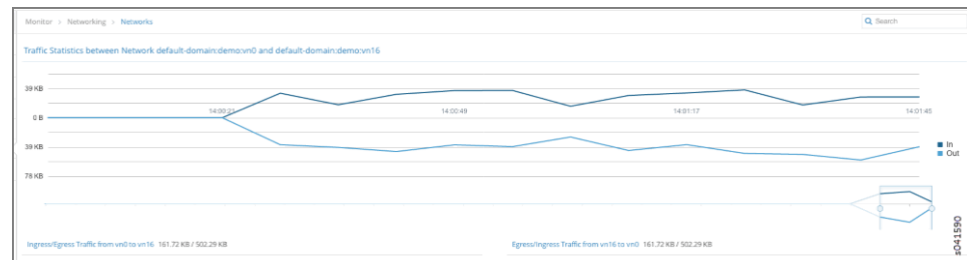
Figure 125: Monitor Projects Connectivity Details



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 126 on page 307](#).

Figure 126: 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 127: Projects Instances Summary

Instance	Virtual Network	Interfaces	vRouter	IP Address	Floating IP	Traffic (In/Out)
out	default-domain:admin:right	1	hp1	2.2.2.252		129.87 KB / 119.83 KB
NAT1_1	default-domain:admin:right	1	hp1	2.2.2.253 250.250.1.253 (1 more)		3.69 MB / 1.15 MB
in	default-domain:admin:left	1	hp1	1.1.1.252		132.75 KB / 122.02 KB

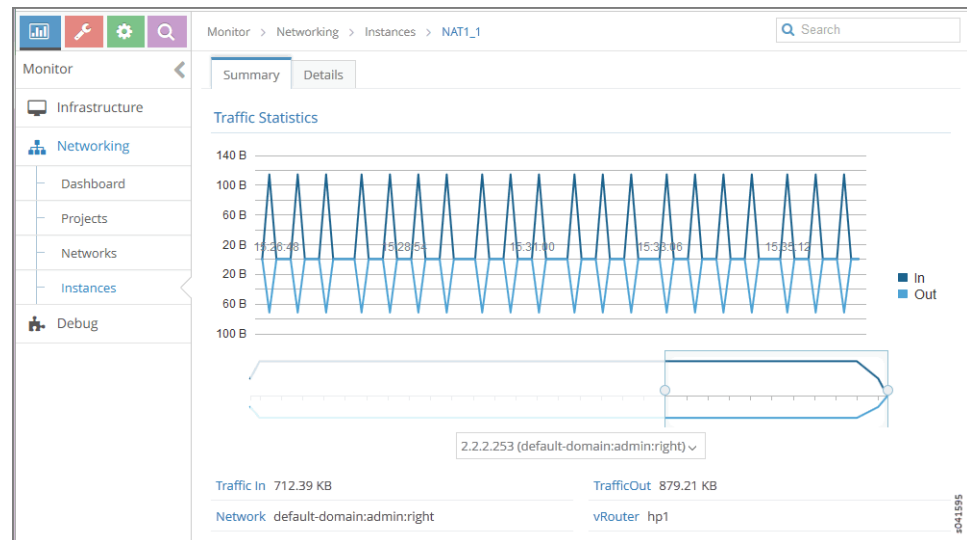
See Table 3 for a description of the fields on this screen.

Table 59: Projects Instances Summary Fields

Field	Description
Instance	The name of the instance. Click the name then select <b>Monitor &gt; Networking &gt; Instances</b> to display details about the traffic statistics for this instance.
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 128 on page 309](#).

Figure 128: Instance Traffic Statistics



## Monitor > Networking > Networks

Select **Monitor > Networking > Networks** to view a summary of the virtual networks in your system. See [Figure 129 on page 309](#).

Figure 129: Network Summary

The screenshot displays the 'Networks Summary' page. The left sidebar shows the navigation menu with 'Monitor' selected. The main content area has a 'Networks Summary' section with a table of virtual networks. The table has columns for 'Network', 'Instances', 'Traffic (In/Out) (Last 1 hr)', and 'Throughput (In/Out)'. The table lists four networks: 'default-domain:default-project-link-local...', 'default-domain:default-project:default-virtual-network', 'default-domain:default-project:ip-fabric', and 'default-domain:demo:default-network-vn1'. All networks show 0 instances and 0 B / 0 B traffic.

Network	Instances	Traffic (In/Out) (Last 1 hr)	Throughput (In/Out)
default-domain:default-project-link-local...	0	0 B / 0 B	0 bps / 0 bps
default-domain:default-project:default-virtual-network	0	0 B / 0 B	0 bps / 0 bps
default-domain:default-project:ip-fabric	0	0 B / 0 B	0 bps / 0 bps
default-domain:demo:default-network-vn1	0	0 B / 0 B	0 bps / 0 bps

Table 60: 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 130 on page 310 shows the **Summary** tab for an individual network, which displays connectivity details and traffic statistics for the selected network.

**Figure 130: Individual Network Connectivity Details—Summary Tab**

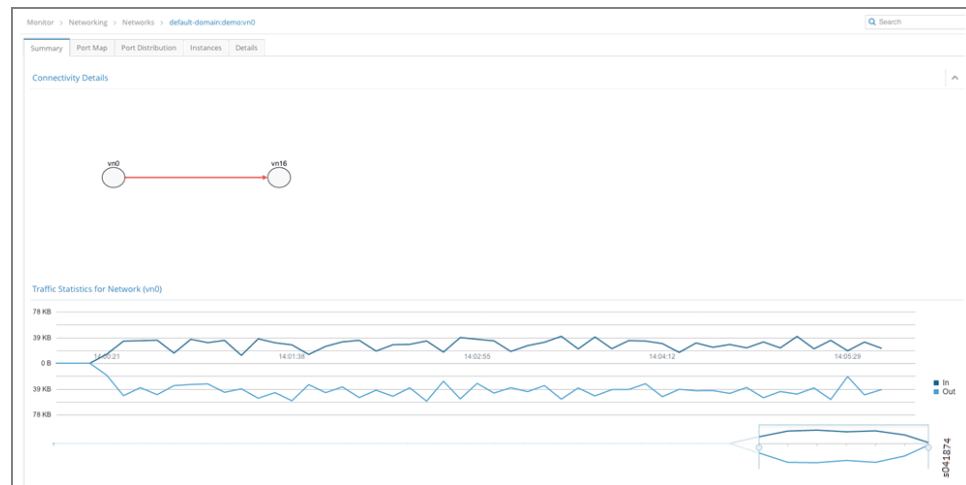


Figure 131 on page 310 shows the **Port Map** tab for an individual network, which displays the relative distribution of traffic for this network by protocol, by port.

**Figure 131: Individual Network— Port Map Tab**

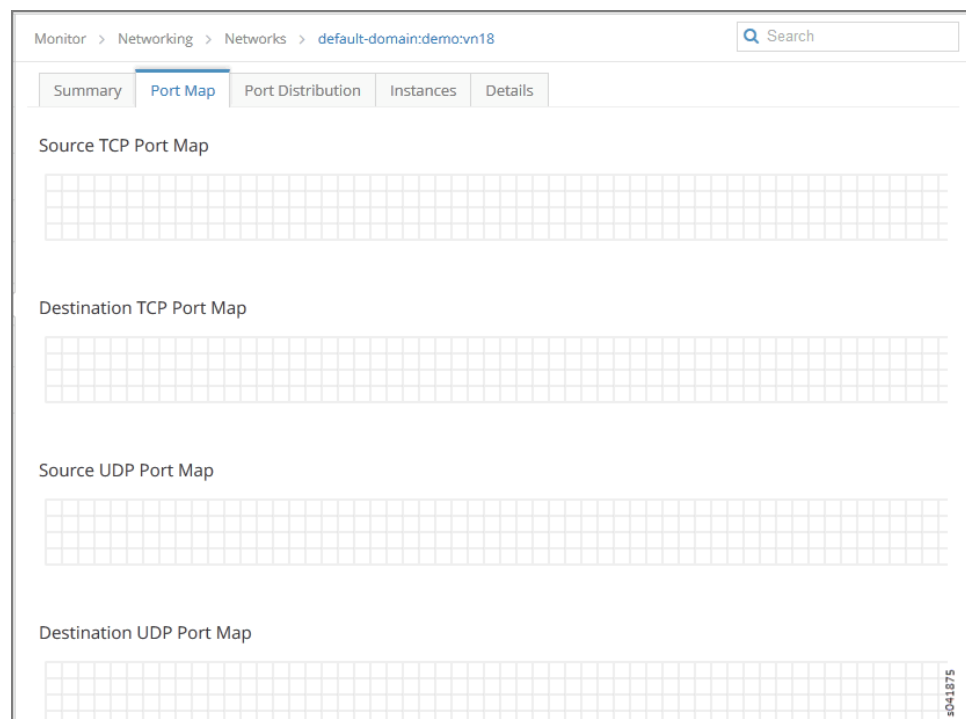


Figure 132 on page 311 shows the **Port Distribution** tab for an individual network, which displays the relative distribution of traffic in and out by source port and destination port.

Figure 132: Individual Network-- Port Distribution Tab

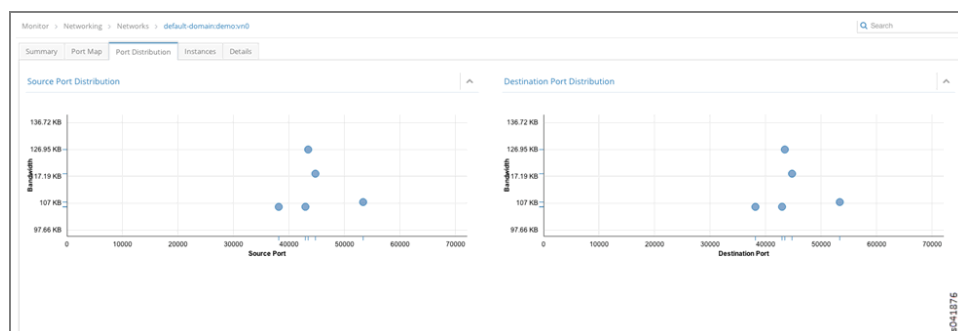


Figure 133 on page 311 shows the **Instances** tab for an individual network, which displays details for each instance associated with this network, including the number of interfaces, the associated vRouter, the instance IP address, and the volume of traffic in and out.

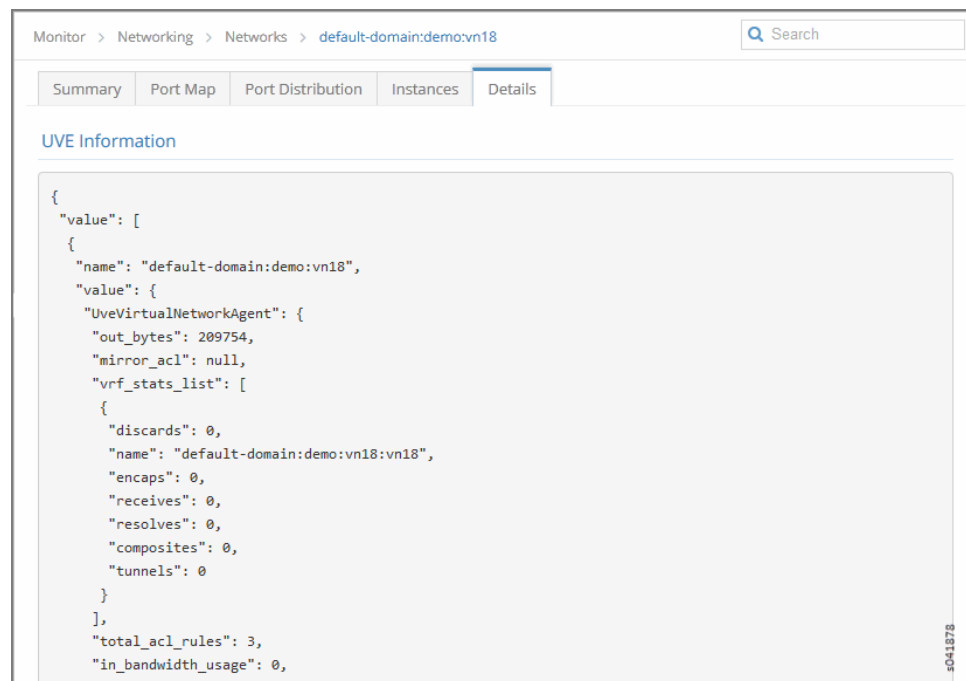
Additionally, you can click the arrow near the instance name to reveal even more details about the instance—the interfaces and their addresses, UUID, CPU (usage), and memory used of the total amount available.

Figure 133: Individual Network Instances Tab

Monitor > Networking > Networks > default-domain:demo:vn18						
Search						
Summary Port Map Port Distribution <b>Instances</b> 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 134 on page 312 shows the **Details** tab for an individual network, which displays the code used to define this network --the User Virtual Environment (UVE) code.

Figure 134: Individual Network Details Tab



## Query > Flows

Use the **Query > Flows** option 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 312](#)
- [Example: Query Flow Series on page 315](#)
- [Query > Flow Records on page 316](#)
- [Query > Flows > Query Queue on page 318](#)

### Query > Flows > Flow Series

Use **Query > Flows > Flow Series** to create queries on the flow series table, with results in the form of time series data for flow series. See [Figure 135 on page 313](#)

Figure 135: Query Flow Series

The query fields available on the screen for the **Flow Series** tab are described in [Table 61 on page 313](#). Enter query data into the fields to create a SQL-like query to display and analyze flows.

Table 61: Query Flow Series Fields

Field	Description
<b>Time Range</b>	<p>Select a range of time for which to see the flow series:</p> <ul style="list-style-type: none"> <li>• Last 10 Mins</li> <li>• Last 30 Mins</li> <li>• Last 1 Hr</li> <li>• Last 6 Hrs</li> <li>• Last 12 Hrs</li> <li>• Custom</li> </ul> <p>Click <b>Custom</b> to enter a specific custom time range in two new fields: <b>From Time</b> and <b>To Time</b>.</p>
<b>Select</b>	Click the edit button (pencil icon) to open a <b>Select</b> window ( <a href="#">Figure 136 on page 314</a> ), where you can click one or more boxes to select the fields to display from the flow series, such as <b>Source VN</b> , <b>Dest VN</b> , <b>Bytes</b> , <b>Packets</b> , and more.
<b>Where</b>	Click the edit button (pencil icon) to open a query-writing window, where you can specify query values for variables such as <b>sourcevn</b> , <b>sourceip</b> , <b>destvn</b> , <b>destip</b> , <b>protocol</b> , <b>sport</b> , <b>dport</b> .
<b>Direction</b>	Select the desired flow direction: <b>INGRESS</b> or <b>EGRESS</b> .
<b>Filter</b>	Click the edit button (pencil icon) to open a <b>Filter</b> window ( <a href="#">Figure 137 on page 315</a> ), where you can select filter items by which to sort, sort order, and limits to the number of results returned.
<b>Run Query</b>	Click this button to retrieve the flows that match the query you have created. The flows are listed on the lower portion of the screen in a box with columns identifying the selected fields for each flow.
(graph buttons)	When <b>Time Granularity</b> is selected, you have the option to view results in graph or flowchart form. Graph buttons appear on the screen above the <b>Export</b> button. Click a graph button to transform the tabular results into a graphical chart display.

Table 61: Query Flow Series Fields (*continued*)

Field	Description
<b>Export</b>	This button appears after you click <b>Run Query</b> , allowing 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 136 on page 314](#). The upper section of the **Select** window includes field names, and the lower portion lets you select units. Use the **Time Granularity** feature to aggregate bytes and packets in intervals by selecting **SUM(Bytes)** or **SUM(Packets)**.

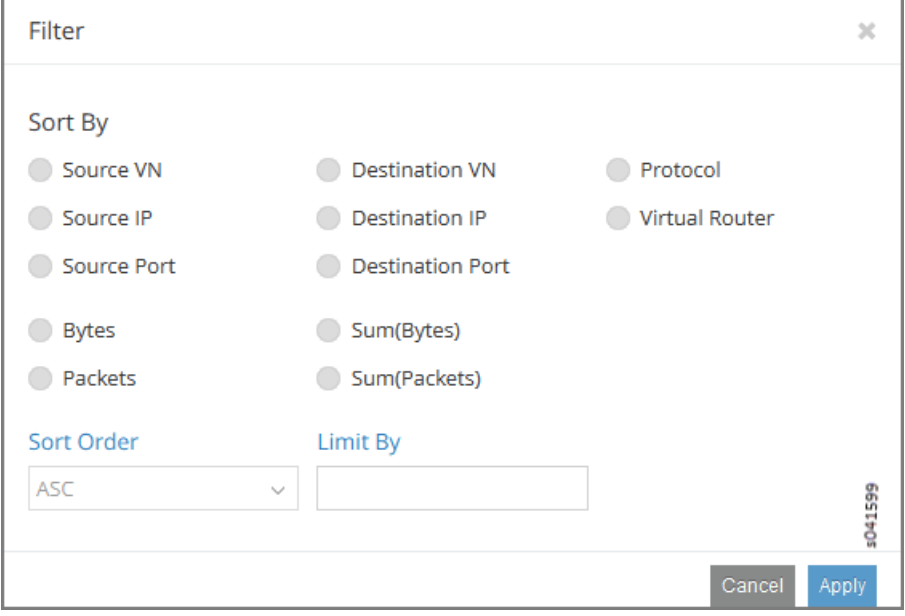
Figure 136: Flow Series Select

The screenshot shows a 'Select' dialog box with a close button (X) in the top right corner. The dialog contains a list of attributes with checkboxes: Source VN, Destination VN, Time Granularity, Source IP, Destination IP, Protocol, Source Port, Destination Port, Virtual Router, Bytes, SUM(Bytes), Packets, and SUM(Packets). The 'Bytes' and 'Packets' options are highlighted in blue. At the bottom right, there are 'Cancel' and 'Apply' buttons. A vertical label 's041600' is visible on the right side of the dialog.

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 137 on page 315](#).



Figure 137: Flow Series Filter



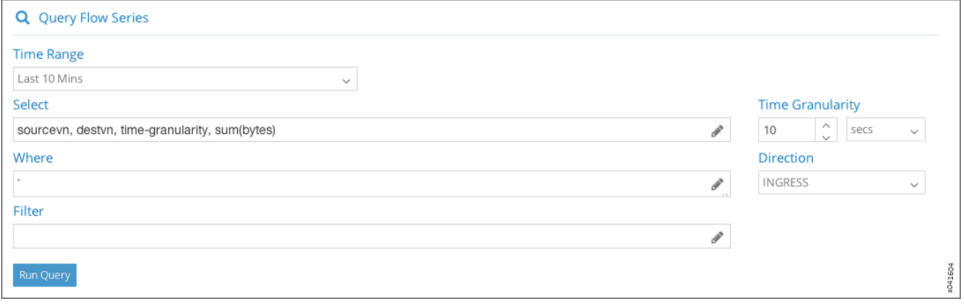
The 'Filter' dialog box contains the following elements:

- 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.
- Watermark:** A vertical watermark '© 2016 Juniper Networks, Inc.' is visible on the right side.

### Example: Query Flow Series

The following is an example flow series query that returns the time series of the summation of 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 138 on page 315](#).

Figure 138: Example: QueryFlow Series



The 'Query Flow Series' interface includes the following fields and controls:

- Time Range:** A dropdown menu set to 'Last 10 Mins'.
- Select:** A text input field containing the query: `sourcevn, destvn, time-granularity, sum(bytes)`.
- Where:** An empty text input field.
- Filter:** An empty text input field.
- Time Granularity:** A dropdown menu set to '10' with a unit of 'secs'.
- Direction:** A dropdown menu set to 'INGRESS'.
- Buttons:** A 'Run Query' button at the bottom left.
- Watermark:** A vertical watermark '© 2016 Juniper Networks, Inc.' is visible on the right side.

The query returns tabular time series data, see [Figure 139 on page 316](#), for the following combinations of Source VN and Dest VN:

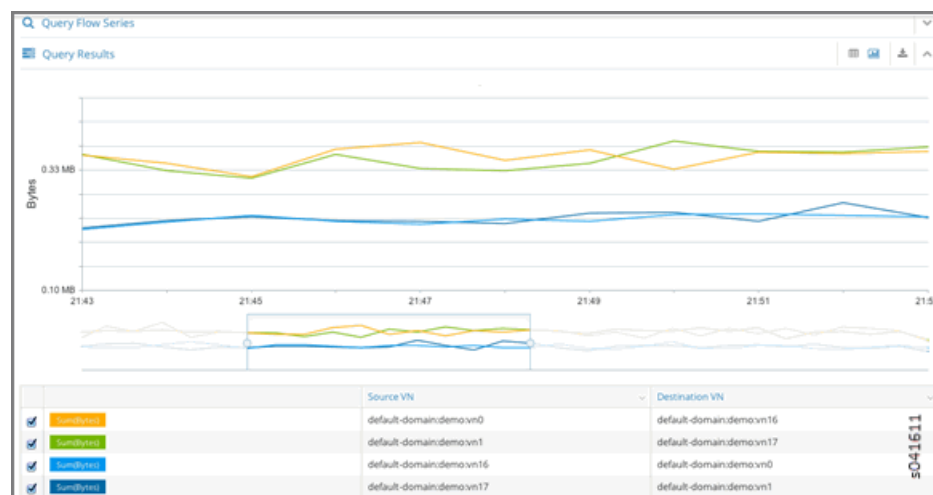
1. Flow Class 1: Source VN = default-domain:demo:front-end, Dest VN = \_\_UNKNOWN\_\_
2. Flow Class 2: Source VN = default-domain:demo:front-end, Dest VN = default-domain:demo:back-end

Figure 139: 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 the **Time Granularity** option was 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 now displayed in a graphical flow chart. See [Figure 140 on page 316](#).

Figure 140: Query Flow Series Graphical Results



## Query > Flow Records

Click **Query > Flow Records** to create queries on 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 141: Flow Records

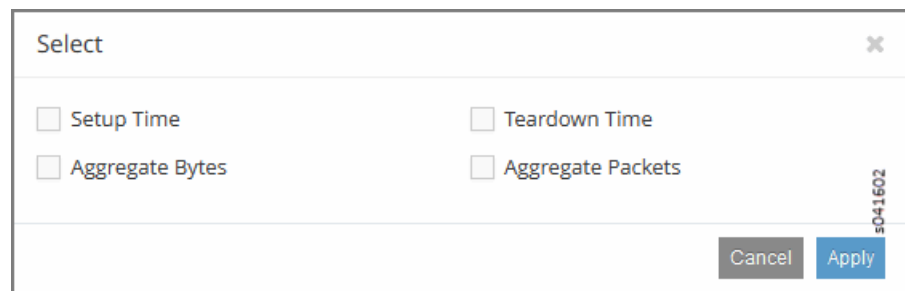
The query fields available on the screen for the **Flow Records** tab are described in [Table 62 on page 317](#). Enter query data into the fields to create a SQL-like query to display and analyze flows.

Table 62: Query Flow Records Fields

Field	Description
<b>Time Range</b>	<p>Select a range of time for which to see the flow records:</p> <ul style="list-style-type: none"> <li>• Last 10 Mins</li> <li>• Last 30 Mins</li> <li>• Last 1 Hr</li> <li>• Last 6 Hrs</li> <li>• Last 12 Hrs</li> <li>• Custom</li> </ul> <p>Click Custom to enter a specified custom time range in two new fields: <b>From Time</b> and <b>To Time</b>.</p>
<b>Select</b>	Click the edit button (pencil icon) to open a <b>Select</b> window ( <a href="#">Figure 142 on page 318</a> ), where you can click one or more boxes to select attributes to display for the flow records, including <b>Setup Time</b> , <b>Teardown Time</b> , <b>Aggregate Bytes</b> , and <b>Aggregate Packets</b> .
<b>Where</b>	Click the edit button (pencil icon) to open a query-writing window where you can specify query values for <b>sourcevn</b> , <b>sourceip</b> , <b>destvn</b> , <b>destip</b> , <b>protocol</b> , <b>sport</b> , <b>dport</b> . .
<b>Direction</b>	Select the desired flow direction: <b>INGRESS</b> or <b>EGRESS</b> .
<b>Run Query</b>	Click this button to retrieve the flow records that match the query you have created. The records are listed on the lower portion of the screen in a box with columns identifying the fields for each flow.
<b>Export</b>	This button appears after you click <b>Run Query</b> , 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 142 on page 318](#).

Figure 142: Flow Records Select Window



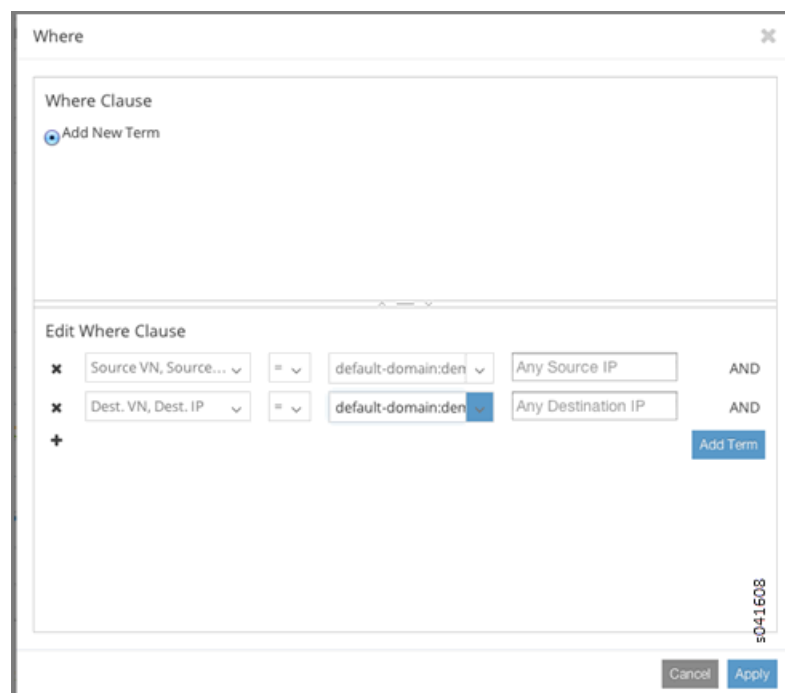
The 'Select' window contains four checkboxes: 'Setup Time', 'Teardown Time', 'Aggregate Bytes', and 'Aggregate Packets'. At the bottom right, there are 'Cancel' and 'Apply' buttons. A vertical label 's041602' is positioned on the right side of the window.

The user 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 logical OR of multiple AND terms. ( (term1 AND term2 AND term3..) OR (term4 AND term5) OR...).

Each term is a single variable expression such as **Source VN = VN1**.

Figure 143: 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 '+' icon 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

Click **Query > Flows > Query Queue** to view queries that are in queue, waiting to be performed on the data. See [Figure 144 on page 319](#).

Figure 144: 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

The query fields available on the screen for the **Flow Records** tab are described in [Table 63 on page 319](#). Enter query data into the fields to create a SQL-like query to display and analyze flows.

Table 63: Query Flow Records Fields

Field	Description
<b>Date</b>	The date and time the query was started.
<b>Query</b>	A display of the parameters set for the query.
<b>Progress</b>	The percentage completion of the query to date.
<b>Records</b>	The number of records matching the query to date.
<b>Status</b>	The status of the query, such as <b>completed</b> .
<b>Time Taken</b>	The amount of time in seconds it has taken the query to return the matching records.
(Action icon)	Click the <b>Action</b> icon and select <b>View Results</b> to view a list of the records that match the query, or click <b>Delete</b> to remove the query from the queue.

## Query > Logs

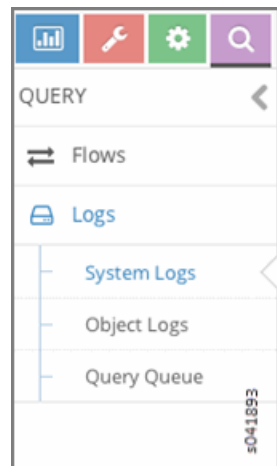
The **Query > Logs** option allows you to access the system log and object log activity of any Contrail Controller component from one central location.

- [Query > Logs Menu Options on page 319](#)
- [Query > Logs > System Logs on page 320](#)
- [Sample Query for System Logs on page 321](#)
- [Query > Logs > Object Logs on page 322](#)

## 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 145 on page 320](#).

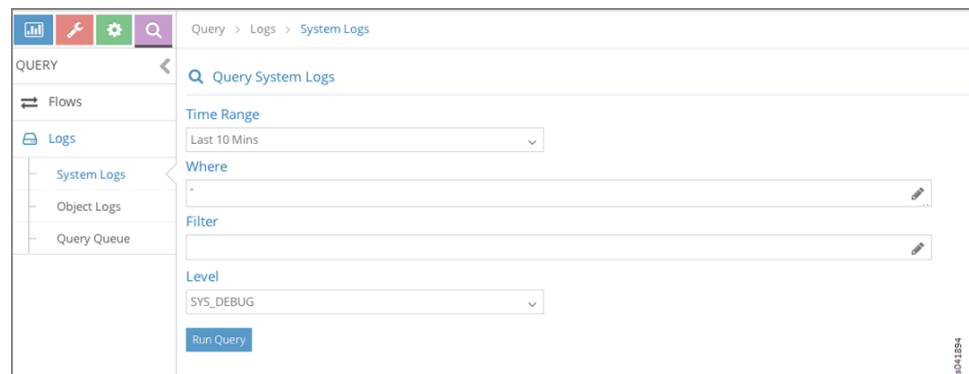
Figure 145: Query &gt; Logs



### 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 146 on page 320](#).

Figure 146: Query &gt; Logs &gt; System Logs



The query fields available on the **Query System Logs** screen are described in [Table 64 on page 320](#).

Table 64: Query System Logs Fields

Field	Description
<b>Time Range</b>	<p>Select a range of time for which to see the system logs:</p> <ul style="list-style-type: none"> <li>• Last 10 Mins</li> <li>• Last 30 Mins</li> <li>• Last 1 Hr</li> <li>• Last 6 Hrs</li> <li>• Last 12 Hrs</li> <li>• Custom</li> </ul> <p>If you click Custom, enter a desired time range in two new fields: <b>From Time</b> and <b>To Time</b>.</p>

Table 64: Query System Logs Fields (*continued*)

Field	Description
<b>Where</b>	Click the edit button (pencil icon) to open a query-writing window, where you can specify query values for variables such as Source, Module, MessageType, and the like, in order to retrieve specific information.
<b>Level</b>	<p>Select the message severity level to view:</p> <ul style="list-style-type: none"> <li>• SYS_NOTICE</li> <li>• SYS_EMERG</li> <li>• SYS_ALERT</li> <li>• SYS_CRIT</li> <li>• SYS_ERR</li> <li>• SYS_WARN</li> <li>• SYS_INFO</li> <li>• SYS_DEBUG</li> </ul>
<b>Run Query</b>	Click this button to retrieve the system logs that match the query. The logs are listed in a box with columns showing the <b>Time</b> , <b>Source</b> , <b>Module Id</b> , <b>Category</b> , <b>Log Type</b> , and <b>Log</b> message.
<b>Export</b>	This button appears after you click <b>Run Query</b> , allowing you to export the list of system messages to a text/csv file.

### Sample Query for System Logs

This section shows a sample system logs query designed to show all **System Logs** from **ModuleId = VRouterAgent** on **Source = b1s16** and filtered by **Level = SYS\_DEBUG**.

1. At the **Query System Logs** screen, click in the **Where** field to access the **Where** query screen and enter information defining the location to query in the **Edit Where Clause** section and click **OK**; see [Figure 147 on page 322](#).

Figure 147: Edit Where Clause

- The information you defined at the Where screen displays on the **Query System Logs**. Enter any more defining information needed; see [Figure 148 on page 322](#). When finished, click **Run Query** to display the results.

Figure 148: Sample Query System Logs

## 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 149 on page 323](#).



Figure 149: Query &gt; Logs &gt; Object Logs

The query fields available on the **Object Logs** screen are described in [Table 65 on page 323](#).

Table 65: Object Logs Query Fields

Field	Description
<b>Time Range</b>	<p>Select a range of time for which to see the logs:</p> <ul style="list-style-type: none"> <li>• Last 10 Mins</li> <li>• Last 30 Mins</li> <li>• Last 1 Hr</li> <li>• Last 6 Hrs</li> <li>• Last 12 Hrs</li> <li>• Custom</li> </ul> <p>If you click Custom, enter a desired time range in two new fields: <b>From Time</b> and <b>To Time</b>.</p>
<b>Object Type</b>	<p>Select the object type for which to show logs:</p> <ul style="list-style-type: none"> <li>• Virtual Network</li> <li>• Virtual Machine</li> <li>• Virtual Router</li> <li>• BGP Peer</li> <li>• Routing Instance</li> <li>• XMPP Connection</li> </ul>
<b>Object Id</b>	<p>Select from a list of available identifiers the name of the object you wish to use.</p>
<b>Select</b>	<p>Click the edit button (pencil icon) to open a window where you can select searchable types by clicking a checkbox:</p> <ul style="list-style-type: none"> <li>• ObjectLog</li> <li>• SystemLog</li> </ul>

Table 65: 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 <b>Source</b> , <b>ModuleId</b> , and <b>MessageType</b> , 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 <b>Time</b> , <b>Source</b> , <b>Module Id</b> , <b>Category</b> , <b>Log Type</b> , and <b>Log</b> message.
Export	This button appears after you click <b>Run Query</b> , allowing you to export the list of system messages to a text/csv file.

## System Log Receiver in Contrail Analytics

- [Overview on page 324](#)
- [Redirecting System Logs to Contrail Collector on page 324](#)
- [Exporting Logs from Contrail Analytics on page 324](#)

### 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.

## Example: Debugging Connectivity Using Monitoring for Troubleshooting


- Using Monitoring to Debug Connectivity on page 325

### 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 150 on page 325](#)

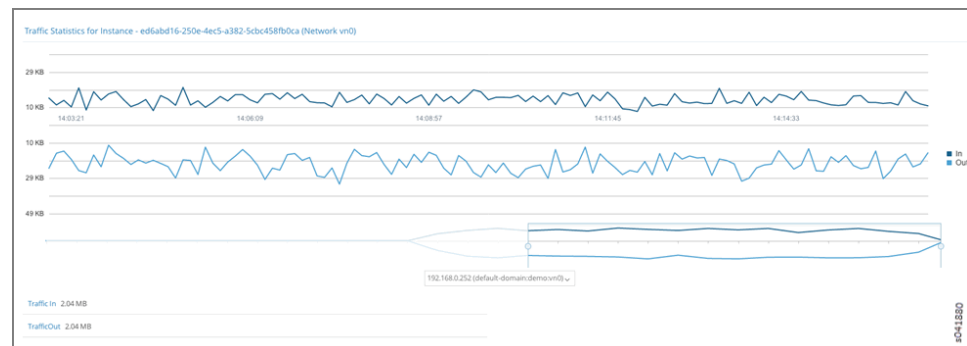
Figure 150: Navigate to Instance



Instance	Traffic In	Traffic Out
ed6abd16-250e-4ec5-a382-5cbc458fb0ca	1.73 MB	1.74 MB
682b7414-e8ba-45ee-91bc-9c22cd86c69d	1.72 MB	1.72 MB

2. Click the instance to view **Traffic Statistics for Instance**. see [Figure 151 on page 325](#).

Figure 151: Traffic Statistics for Instance



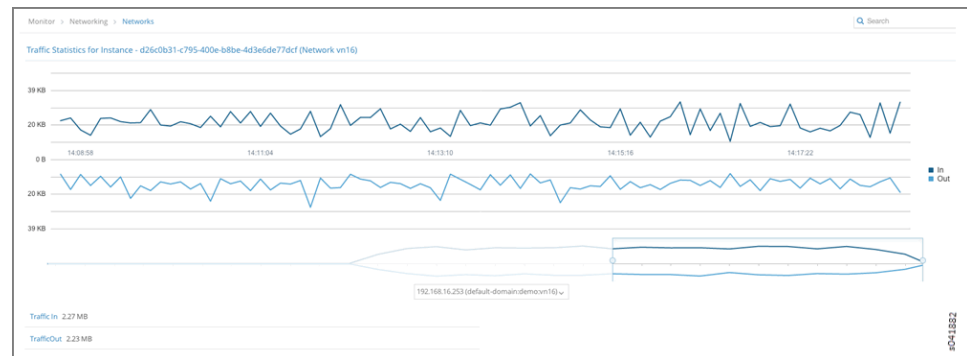
3. Instance **d26c0b31-c795-400e-b8be-4d3e6de77dcf** with IP address **192.168.0.253** in the virtual network **vn16**. see [Figure 152 on page 325](#) and [Figure 153 on page 326](#).

Figure 152: Navigate to Instance



Instance	Traffic In	Traffic Out
d26c0b31-c795-400e-b8be-4d3e6de77dcf	2.18 MB	2.13 MB
230d5415-b679-420e-8f5d-96c162d628be	2.11 MB	2.16 MB

Figure 153: Traffic Statistics for Instance



4. From **Monitor->Infrastructure->Virtual Routers->a3s18->Interfaces**, we can see that Instance **ed6abdl6-250e-4ec5-a382-5cbc458fb0ca** is hosted on Virtual Router **a3s18**; see [Figure 154 on page 326](#).

Figure 154: Navigate to a3s18 Interfaces

Monitor > Infrastructure > Virtual Routers > a3s18						
Details Console Interfaces Networks ACL Flows Routes						
Name	Label	Status	Network	IP Address	Floating IP	Instance
tap1d4e0721-4c	16	Up	default-domain:demo:vn0	192.168.0.252	None	ed6abdl6-250e-4ec5-a382-5cbc458fb0ca
tap249da2e1-97	18	Up	default-domain:demo:vn16	192.168.16.252	None	23045415-b679-4d8a-8f6d-96c162a28be
tap5b3b3b63-74	19	Up	default-domain:demo:vn17	192.168.17.252	None	99311eda-261e-47d8-b4a7-8d126d7d98f
tapc740843c-6b	17	Up	default-domain:demo:vn1	192.168.1.252	None	20244ef9-4a4d-4a32-803f-15c5323572e

5. From **Monitor->Infrastructure->Virtual Routers->a3s19->Interfaces**, we can see that Instance **d26c0b31-c795-400e-b8be-4d3e6de77dcf** is hosted on Virtual Router **a3s19**; see [Figure 155 on page 326](#).

Figure 155: Navigate to a3s19 Interfaces

Monitor > Infrastructure > Virtual Routers > a3s19						
Details Console Interfaces Networks ACL Flows Routes						
Name	Label	Status	Network	IP Address	Floating IP	Instance
tap295892f-c2	19	Up	default-domain:demo:vn16	192.168.16.253	None	d26c0b31-c795-400e-b8be-4d3e6de77dcf
tapb257d21d-43	18	Up	default-domain:demo:vn1	192.168.1.253	None	eebce321-7536-46e7-a454-cd1f13ac095
tapc13e3d87-66	17	Up	default-domain:demo:vn17	192.168.17.253	None	b2425f95-67fe-4060-9478-81a4a829541
tapc5ea07b3-55	16	Up	default-domain:demo:vn0	192.168.0.253	None	682b7414-c8ba-45ee-918c-9c22b86c69d

6. Virtual Routers **a3s18** and **a3s19** have the ACL entries to allow connectivity between **default-domain:demo:vn0** and **default-domain:demo:vn16** networks; see [Figure 156 on page 326](#) and [Figure 157 on page 327](#).

Figure 156: 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	
a72d32fe-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	
b32143a3-0e80-4aa2-9c...	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	
b8c9b10-ef9c-418b-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	
d7b47291-7a21-4f5e-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 157: ACL Connectivity a3s19

Monitor > Infrastructure > Virtual Routers > a3s19										
Details Console Interfaces Networks ACL Flows Routers										
UUID	Flows	Action	Protocol	Source Network or Prefix	Source Port	Destination Network or Prefix	Destination Port	Source Policy Rule	ACE Id	
a7249326e-3f50-477a-ad...	16	pass	any	default-domain:demo:vn0	any	default-domain:demo:vn16	any		1	a3s1886
		pass	any	default-domain:demo:vn16	any	default-domain:demo:vn0	any		2	
		pass	any	default-domain:demo:vn0	any	default-domain:demo:vn0	any		3	
b32143a3-0ed0-4ae2-9c...	16	pass	any	default-domain:demo:vn1	any	default-domain:demo:vn17	any		1	
		pass	any	default-domain:demo:vn17	any	default-domain:demo:vn1	any		2	
		pass	any	default-domain:demo:vn1	any	default-domain:demo:vn1	any		3	
b3c79810-e9fc-418b-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	
d7b47291-7a21-405e-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 158 on page 327](#) and [Figure 159 on page 327](#).

Figure 158: Routes default-domain:demo:vn0:vn0

Monitor > Infrastructure > Control Nodes > a3s15							
Details Console Peers Routes							
Routing Instance	default-domain:demo:vn0: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 159: Routes default-domain:demo:vn16:vn16

Monitor > Infrastructure > Control Nodes > a3s15							
Details Console Peers Routes							
Routing Instance	default-domain:demo:vn16:vn16	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
10.84.17.4:1:192.168.16.255,0.0.0.0	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	-
10.84.17.5:1:192.168.16.255,0.0.0.0	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 160 on page 328](#).

Figure 160: Verify Route and Next Hop a3s18

Monitor > Infrastructure > Virtual Routers > a3s18		
Details	Console	Interfaces
Networks	ACL	Flows
Routes		
VRF	default-domain:demo:vn0:vn0	Show Routes <input checked="" type="radio"/> Unicast <input type="radio"/> Multicast
Prefix	Next ho...	Next hop details
169.254.169.254 / 32	receive	Source: MData Dest VN: default-domain:default-project:__link_local__
192.168.0.252 / 32	interface	Interface: tap1dae0121-4c Dest VN: default-domain:demo:vn0
	interface	Interface: tap1dae0121-4c Dest VN: default-domain:demo:vn0
	interface	Interface: tap1dae0121-4c Dest VN: default-domain:demo:vn0
192.168.0.253 / 32	tunnel	Dest IP: 10.84.17.5 Dest VN: default-domain:demo:vn0 Label: 16
	tunnel	Dest IP: 10.84.17.5 Dest VN: default-domain:demo:vn0 Label: 16
192.168.0.254 / 32	interface	Interface: pkt0 Dest VN: default-domain:demo:vn0
192.168.16.252 / 32	interface	Interface: tap249de2e1-97 Dest VN: default-domain:demo:vn16
	interface	Interface: tap249de2e1-97 Dest VN: default-domain:demo:vn16
192.168.16.253 / 32	tunnel	Dest IP: 10.84.17.5 Dest VN: default-domain:demo:vn16 Label: 19

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 161 on page 328](#).

Figure 161: 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 <input checked="" type="radio"/> Unicast <input type="radio"/> 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 162 on page 329](#) and [Figure 163 on page 329](#).

Figure 162: Flows for a3s18

Protocol	Source Network	Source IP	Source Port	Destination Network	Destination IP	Destination Port	Bytes/Pkts	Setup Time
TCP	vn0	192.168.0.252	43434	vn16	192.168.16.253	9100	188458/5417	21:00:22.131180 2013-Aug-06
TCP	vn16	192.168.16.253	9100	vn0	192.168.0.252	43434	190968/5891	21:00:22.131193 2013-Aug-06
TCP	vn16	192.168.16.253	9101	vn0	192.168.0.252	53369	190300/5805	21:00:22.206222 2013-Aug-06
TCP	vn0	192.168.0.252	53369	vn16	192.168.16.253	9101	189008/5302	21:00:22.206207 2013-Aug-06
UDP	vn0	192.168.0.252	39522	vn16	192.168.16.252	9200	0/0	21:00:22.382861 2013-Aug-06
UDP	vn0	192.168.0.252	44794	vn16	192.168.16.253	9201	1707392/3144	21:00:24.104277 2013-Aug-06
UDP	vn16	192.168.16.253	9201	vn0	192.168.0.252	44794	1785789/3107	21:00:24.104293 2013-Aug-06
UDP	vn0	192.168.0.252	40561	vn16	192.168.16.253	9200	1693476/3067	21:00:22.037377 2013-Aug-06
UDP	vn16	192.168.16.253	9200	vn0	192.168.0.252	40561	1643324/3061	21:00:22.037387 2013-Aug-06
UDP	vn0	192.168.0.252	39522	vn16	192.168.16.252	9200	1676616/3074	21:00:22.306703 2013-Aug-06
TCP	vn0	192.168.0.252	34236	vn16	192.168.16.252	9100	1891368/5486	21:00:22.395695 2013-Aug-06
TCP	vn0	192.168.0.252	34236	vn16	192.168.16.252	9100	0/0	21:00:22.400371 2013-Aug-06

Figure 163: Flows for a3s19

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	1089880/1975	21:00:24.111324 2013-Aug-06
UDP	vn16	192.168.16.253	9201	vn0	192.168.0.252	44794	110904/1963	21:00:24.111380 2013-Aug-06
UDP	vn0	192.168.0.252	40561	vn16	192.168.16.253	9200	1046756/1877	21:00:22.047747 2013-Aug-06
UDP	vn16	192.168.16.253	9200	vn0	192.168.0.252	47270	1061900/1921	21:00:25.373941 2013-Aug-06
UDP	vn16	192.168.16.253	9200	vn0	192.168.0.252	40561	1010568/1914	21:00:22.047756 2013-Aug-06
TCP	vn16	192.168.16.253	9100	vn0	192.168.0.253	53314	1217772/3649	21:00:23.440564 2013-Aug-06
TCP	vn0	192.168.0.252	43434	vn16	192.168.16.253	9100	1196536/3400	21:00:22.137665 2013-Aug-06
TCP	vn16	192.168.16.253	9100	vn0	192.168.0.252	43434	1239616/3724	21:00:22.137679 2013-Aug-06
UDP	vn16	192.168.16.253	9200	vn0	192.168.0.253	47270	0/0	21:00:25.347868 2013-Aug-06
TCP	vn16	192.168.16.253	9100	vn0	192.168.0.253	53314	0/0	21:00:23.440990 2013-Aug-06
UDP	vn16	192.168.16.253	9201	vn0	192.168.0.253	53930	1088692/1953	21:00:25.443166 2013-Aug-06
UDP	vn16	192.168.16.253	9101	vn0	192.168.0.253	34551	0/0	21:00:23.514246 2013-Aug-06
TCP	vn16	192.168.16.253	9101	vn0	192.168.0.253	34551	1394273/3004	21:00:23.513463 2013-Aug-06

## 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.

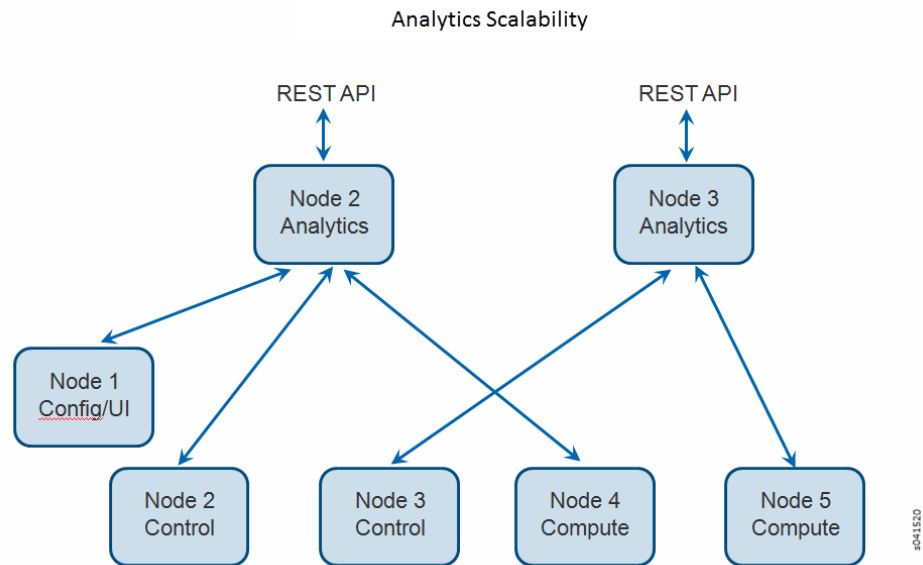
The following is the recommended locations for the various component roles of the Contrail system for a 5-node configuration.

- 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 164 on page 330 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 164: Analytics Scalability



The analytics nodes collect and store data and provide this data through various REST API queries. Scalability is provided for the control nodes, the compute nodes, and the REST API users, with the API output displayed in the Contrail user interface. As the number of control and compute nodes increase in the system, the analytics nodes can also be increased.

## High Availability for Analytics

Contrail supports multiple instances of analytics for high availability and load balancing.

Contrail analytics provides two broad areas of functionality:

- **contrail-collector** —Receives status, logs, and flow information from all Contrail processing elements (for example, generators) and records them.

Every generator is connected to one of the **contrail-collector** instances at any given time. If an instance fails (or is shut down), all the generators that are connected to it are automatically moved to another functioning instance, typically in a few seconds or less. Some messages may be lost during this movement. UVEs are resilient to message loss, so the state shown in a UVE is kept consistent to the state in the generator.

- **contrail-opserver** —Provides an external API to report UVEs and to query logs and flows.

Each analytics component exposes a northbound REST API represented by the **contrail-opserver** service (port 8081) so that the failure of one analytics component or one **contrail-opserver** service should not impact the operation of other instances.



These are the ways to manage connectivity to the **contrail-opserver** endpoints:

- Periodically poll the **contrail-opserver** service on a set of analytics nodes to determine the list of functioning endpoints, then make API requests from one or more of the functioning endpoints.
- Subscribe to the Contrail Discovery Service to get a list of functioning endpoints. If there are any failures, it can take 5-30 minutes for the Contrail Discovery Service to send an update.

The Contrail user interface makes use of the same northbound REST API to present dashboards, and reacts to any **contrail-opserver** high availability event automatically, using the Contrail Discovery Service.



## CHAPTER 13

# Application Programming Interfaces (APIs)

- [Contrail Analytics Application Programming Interfaces \(APIs\) and User-Visible Entities \(UVEs\) on page 333](#)
- [Contrail Node Status on page 344](#)
- [Log and Flow Information APIs on page 353](#)
- [Working with Neutron on page 359](#)
- [Support for Amazon VPC APIs on Contrail OpenStack on page 362](#)

## Contrail Analytics Application Programming Interfaces (APIs) and User-Visible Entities (UVEs)

---

The Contrail **analytics-api** server provides a REST API interface to extract the operational state of the Contrail system.

APIs are used by the Contrail Web user interface to present the operational state to users. Other applications might also use the server's REST APIs for analytics or other uses.

This section describes some of the more common APIs and their uses. To see all of the available APIs, navigate the URL tree at the REST interface, starting at the root

**`http://<ip>:<analytics-api-port>`**

- [User-Visible Entities on page 333](#)
- [Common UVEs in Contrail on page 335](#)
- [Virtual Network UVE on page 335](#)
- [Virtual Machine UVE on page 335](#)
- [vRouter UVE on page 335](#)
- [UVEs for Contrail Nodes on page 336](#)
- [Wild Card Query of UVEs on page 336](#)
- [Filtering UVE Information on page 337](#)

### User-Visible Entities

In Contrail, a User-Visible Entity (UVE) is an object entity that might span multiple components in Contrail and might require aggregation before the complete information

of the UVE is presented. Examples of UVEs in Contrail are virtual network, virtual machine, vRouter, and similar objects. Complete operational information for a virtual network might span multiple vRouters, config nodes, control nodes, and the like. The analytics-api server aggregates all of this information through REST APIs.

To get information about a UVE, you must have the UVE type and the UVE key. In Contrail, UVEs are identified by type, such as virtual network, virtual machine, vRouter, and so on. A system-wide unique key is associated with each UVE. The key type could be different, based on the UVE type. For example, perhaps a virtual network uses its name as its UVE key, and in the same system, a virtual machine uses its UUID as its key.

The URL `/analytics/uves` shows the list of all UVE types available in the system.

The following is sample output from `/analytics/uves`:

```
[
{
  href: "http://<ip address>:8081/analytics/uves/xmpp-peers",
  name: "xmpp-peers"
},
{
  href: "http://<ip address>:8081/analytics/uves/service-instances",
  name: "service-instances"
},
{
  href: "http://<ip address>:8081/analytics/uves/config-nodes",
  name: "config-nodes"
},
{
  href: "http://<ip address>:8081/analytics/uves/virtual-machines",
  name: "virtual-machines"
},
{
  href: "http://<ip address>:8081/analytics/uves/bgp-routers",
  name: "bgp-routers"
},
{
  href: "http://<ip address>:8081/analytics/uves/collectors",
  name: "collectors"
},
{
  href: "http://<ip address>:8081/analytics/uves/service-chains",
  name: "service-chains"
},
{
  href: "http://<ip address>:8081/analytics/uves/generators",
  name: "generators"
},
{
  href: "http://<ip address>:8081/analytics/uves/bgp-peers",
  name: "bgp-peers"
},
{
  href: "http://<ip address>:8081/analytics/uves/virtual-networks",
  name: "virtual-networks"
},
{
  href: "http://<ip address>:8081/analytics/uves/vrouters",
  name: "vrouters"
}
```

```

    },
    {
      href: "http://<ip address>:8081/analytics/uves/dns-nodes",
      name: "dns-nodes"
    }
  ]

```

## Common UVEs in Contrail

This section presents descriptions of some common UVEs in Contrail.

### Virtual Network UVE

This UVE provides information associated with a virtual network, such as:

- list of networks connected to this network
- list of virtual machines spawned in this network
- list of access control lists (ACLs) associated with this virtual network
- global input and output statistics
- input and output statistics per virtual network pair

The REST API to get a UVE for a specific virtual network is through HTTP GET, using the URL:

**/analytics/uves/virtual-network/<key>**

The REST API to get UVEs for all virtual machines is through HTTP GET, using the URL:

**/analytics/uves/virtual-networks**

### Virtual Machine UVE

This UVE provides information associated with a virtual machine, such as:

- list of interfaces in this virtual machine
- list of floating IPs associated with each interface
- input and output statistics

The REST API to get a UVE for a specific virtual machine is through HTTP GET, using the URL:

**/analytics/uves/virtual-machine/<key>**

The REST API to get UVEs for all virtual machines is through HTTP GET, using the URL:

**/analytics/uves/virtual-machines**

### vRouter UVE

This UVE provides information associated with a vRouter, such as:

- virtual networks present on this vRouter

- virtual machines spawned on the server of this vRouter
- statistics of the traffic flowing through this vRouter

The REST API to get a UVE for a specific vRouter is through HTTP GET, using the URL:

`/analytics/uves/vrouter/<key>`

The REST API to get UVEs for all virtual machines is through HTTP GET, using the URL:

`/analytics/uves/routers`

## UVEs for Contrail Nodes

There are multiple node types in Contrail (including the node type vRouter previously described). Other node types include control node, config node, analytics node, and compute node.

There is a UVE for each node type. The common information associated with each node UVE includes:

- the IP address of the node
- a list of processes running on the node
- the CPU and memory utilization of the running processes

Each UVE also has node-specific information, such as:

- the control node UVE has information about its connectivity to the vRouter and other control nodes
- the analytics node UVE has information about the number of generators connected

The REST API to get a UVE for a specific config node is through HTTP GET, using the URL:

`/analytics/uves/config-node/<key>`

The REST API to get UVEs for all config nodes is through HTTP GET, using the URL:

`/analytics/uves/config-nodes`



**NOTE:** Use similar syntax to get UVEs for each of the different types of nodes, substituting the node type that you want in place of config-node.

---

## Wild Card Query of UVEs

You can use wildcard queries when you want to get multiple UVEs at the same time. Example queries are the following:

The following HTTP GET with wildcard retrieves all virtual network UVEs:

`/analytics/uves/virtual-network/*`

The following HTTP GET with wildcard retrieves all virtual network UVEs with name starting with **project1**:

```
/analytics/uves/virtual-network/project1*
```

## Filtering UVE Information

It is possible to retrieve filtered UVE information. The following flags enable you to retrieve partial, filtered information about UVEs.

Supported filter flags include:

**sfilt** : filter by source (usually the hostname of the generator)

**mfilt** : filter by module (the module name of the generator)

**cfilt** : filter by content, useful when only part of a UVE needs to be retrieved

**kfilt** : filter by UVE keys, useful to get multiple, but not all, UVEs of a particular type

### Examples

The following HTTP GET with filter retrieves information about virtual network **vn1** as provided by the source **src1**:

```
/analytics/uves/virtual-network/vn1?sfilt=src1
```

The following HTTP GET with filter retrieves information about virtual network **vn1** as provided by all **ApiServer** modules:

```
/analytics/uves/virtual-network/vn1?mfilt=ApiServer
```

### Example Output: Virtual Network UVE

Example output for a virtual network UVE:

```
[user@host ~]# curl <ip
address>:8081/analytics/virtual-network/default-domain:demo:front-end | python
-mjson.tool
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 2576 100 2576 0 0 152k 0 --:--:-- --:--:-- --:--:-- 157k
{
  "UveVirtualNetworkAgent": {
    "acl": [
      [
        {
          "@type": "string"
        },
        "a3s18:VRouterAgent"
      ]
    ],
    "in_bytes": {
      "#text": "2232972057",
      "@aggtype": "counter",
      "@type": "i64"
    },
    "in_stats": {
      "@aggtype": "append",
```

```

"@type": "list",
"list": {
  "@size": "3",
  "@type": "struct",
  "UveInterVnStats": [
    {
      "bytes": {
        "#text": "2114516371",
        "@type": "i64"
      },
      "other_vn": {
        "#text": "default-domain:demo:back-end",
        "@aggtype": "listkey",
        "@type": "string"
      },
      "tpkts": {
        "#text": "5122001",
        "@type": "i64"
      }
    },
    {
      "bytes": {
        "#text": "1152123",
        "@type": "i64"
      },
      "other_vn": {
        "#text": "__FABRIC__",
        "@aggtype": "listkey",
        "@type": "string"
      },
      "tpkts": {
        "#text": "11323",
        "@type": "i64"
      }
    },
    {
      "bytes": {
        "#text": "8192",
        "@type": "i64"
      },
      "other_vn": {
        "#text": "default-domain:demo:front-end",
        "@aggtype": "listkey",
        "@type": "string"
      },
      "tpkts": {
        "#text": "50",
        "@type": "i64"
      }
    }
  ]
},
"in_tpkts": {
  "#text": "5156342",
  "@aggtype": "counter",

```



```

    "@type": "i64"
  },
  "interface_list": {
    "@aggtype": "union",
    "@type": "list",
    "list": {
      "@size": "1",
      "@type": "string",
      "element": [
        "tap2158f77c-ec"
      ]
    }
  },
  "out_bytes": {
    "#text": "2187615961",
    "@aggtype": "counter",
    "@type": "i64"
  },
  "out_stats": {
    "@aggtype": "append",
    "@type": "list",
    "list": {
      "@size": "4",
      "@type": "struct",
      "UvelInterVnStats": [
        {
          "bytes": {
            "#text": "2159083215",
            "@type": "i64"
          },
          "other_vn": {
            "#text": "default-domain:demo:back-end",
            "@aggtype": "listkey",
            "@type": "string"
          },
          "tpkts": {
            "#text": "5143693",
            "@type": "i64"
          }
        }
      ],
      {
        "bytes": {
          "#text": "1603041",
          "@type": "i64"
        },
        "other_vn": {
          "#text": "__FABRIC__",
          "@aggtype": "listkey",
          "@type": "string"
        },
        "tpkts": {
          "#text": "9595",
          "@type": "i64"
        }
      }
    ]
  },
  {

```

```

        "bytes": {
            "#text": "24608",
            "@type": "i64"
        },
        "other_vn": {
            "#text": "__UNKNOWN__",
            "@aggtype": "listkey",
            "@type": "string"
        },
        "tpkts": {
            "#text": "408",
            "@type": "i64"
        }
    },
    {
        "bytes": {
            "#text": "8192",
            "@type": "i64"
        },
        "other_vn": {
            "#text": "default-domain:demo:front-end",
            "@aggtype": "listkey",
            "@type": "string"
        },
        "tpkts": {
            "#text": "50",
            "@type": "i64"
        }
    }
]
}
},
"out_tpkts": {
    "#text": "5134830",
    "@aggtype": "counter",
    "@type": "i64"
},
"virtualmachine_list": {
    "@aggtype": "union",
    "@type": "list",
    "list": {
        "@size": "1",
        "@type": "string",
        "element": [
            "dd09f8c3-32a8-456f-b8cc-fab15189f50f"
        ]
    }
}
},
"UveVirtualNetworkConfig": {
    "connected_networks": {
        "@aggtype": "union",
        "@type": "list",
        "list": {
            "@size": "1",
            "@type": "string",
            "element": [

```

```

        "default-domain:demo:back-end"
    ]
}
},
"routing_instance_list": {
    "@aggtype": "union",
    "@type": "list",
    "list": {
        "@size": "1",
        "@type": "string",
        "element": [
            "front-end"
        ]
    }
},
"total_acl_rules": [
    [
        {
            "#text": "3",
            "@type": "i32"
        },
        ":",
        "a3s14:Schema"
    ]
]
}
}

```

**Example Output:** Example output for a virtual machine UVE:  
**Virtual Machine UVE**

```

[user@host ~]# curl <ip
address>:8081/analytics/virtual-machine/f38eb47e-63d2-4b39-80de-8fe68e6af1e4
| python -mjson.tool
% Total    % Received % Xferd Average Speed   Time    Time     Current
                                 Dload Upload Total   Spent  Left  Speed
100 736 100 736  0  0 160k  0 --:--:-- --:--:-- --:--:-- 179k
{
  "UveVirtualMachineAgent": {
    "interface_list": [
      [
        {
          "@type": "list",
          "list": {
            "@size": "1",
            "@type": "struct",
            "VmInterfaceAgent": [
              {
                "in_bytes": {
                  "#text": "2188895907",
                  "@aggtype": "counter",
                  "@type": "i64"
                },
                "in_pkts": {
                  "#text": "5130901",
                  "@aggtype": "counter",
                  "@type": "i64"
                }
              }
            ]
          }
        }
      ]
    ]
  }
}

```

```

    },
    "ip_address": {
      "#text": "192.168.2.253",
      "@type": "string"
    },
    "name": {
      "#text":
"f38eb47e-63d2-4b39-80de-8fe68e6af1e4:ccb085a0-c994-4034-be0f-6fd5ad08ce83",

      "@type": "string"
    },
    "out_bytes": {
      "#text": "2201821626",
      "@aggtype": "counter",
      "@type": "i64"
    },
    "out_pkts": {
      "#text": "5153526",
      "@aggtype": "counter",
      "@type": "i64"
    },
    "virtual_network": {
      "#text": "default-domain:demo:back-end",
      "@aggtype": "listkey",
      "@type": "string"
    }
  }
}
]
}
},
"a3s19:VRouterAgent"
]
}
}
}

```

**Example Output:** Example output for a vRouter UVE:

**vRouter UVE**

```

[user@host ~]# curl <ip address>:8081/analytics/vrouter/a3sxx | python -mjson.tool
% Total    % Received % Xferd Average Speed   Time    Time     Time Current
           Dload  Upload  Total   Spent    Left  Speed
100 706 100 706  0  0 142k    0 --:--:-- --:--:-- --:--:-- 172k
{
  "VrouterAgent": {
    "collector": [
      [
        {
          "#text": "<ip address>",
          "@type": "string"
        },
        "a3s18:VRouterAgent"
      ]
    ],
    "connected_networks": [
      [
        {

```

```

        "@type": "list",
        "list": {
            "@size": "1",
            "@type": "string",
            "element": [
                "default-domain:demo:front-end"
            ]
        }
    },
    "a3xxx:VRouterAgent"
],
"interface_list": [
    [
        {
            "@type": "list",
            "list": {
                "@size": "1",
                "@type": "string",
                "element": [
                    "tap2158f77c-ec"
                ]
            }
        },
        "a3s18:VRouterAgent"
    ]
],
"virtual_machine_list": [
    [
        {
            "@type": "list",
            "list": {
                "@size": "1",
                "@type": "string",
                "element": [
                    "dd09f8c3-32a8-456f-b8cc-fab15189f50f"
                ]
            }
        },
        "a3s18:VRouterAgent"
    ]
],
"xmpp_peer_list": [
    [
        {
            "@type": "list",
            "list": {
                "@size": "2",
                "@type": "string",
                "element": [
                    "<ip address>",
                    "<ip address>"
                ]
            }
        },
        "a3s18:VRouterAgent"
    ]
]

```

```
    ]  
  ]  
}  
}
```

## Contrail Node Status

---

- [Overview on page 344](#)
- [UVE for NodeStatus on page 344](#)
- [Node Status Features on page 345](#)
- [Using Introspect to Get Process Status on page 350](#)
- [contrail-status script on page 351](#)

### Overview

This topic describes how to view the status of a Contrail node on a physical server. Contrail nodes include config, control, analytics, compute, and so on.

### UVE for NodeStatus

The User-Visible Entity (UVE) mechanism is used to aggregate and send the status information. All node types send a NodeStatus structure in their respective node UVEs. The following is a control node UVE of NodeStatus:

```
struct NodeStatus {  
  
    1: string name (key="ObjectBgpRouter")  
  
    2: optional bool deleted  
  
    3: optional string status  
  
    // Sent by process  
  
    4: optional list<process_info.ProcessStatus> process_status (aggtype="union")  
  
    // Sent by node manager  
  
    5: optional list<process_info.ProcessInfo> process_info (aggtype="union")  
  
    6: optional string description  
  
}  
  
uve sandesh NodeStatusUVE {  
  
    1: NodeStatus data  
  
}
```

## Node Status Features

The most important features of NodeStatus include:

ProcessStatus

ProcessInfo

**ProcessStatus** Also process\_status, is sent by the processes corresponding to the virtual node, and displays the status of the process and an aggregate state indicating if the process is functional or non-functional. The process\_status includes the state of the process connections (ConnectionInfo) to important services and other information necessary for the process to be functional. Each process sends its NodeStatus information, which is aggregated as union (aggtype="union") at the analytics node. The following is the ProcessStatus structure:

```

1. struct ProcessStatus {
2.     1: string module_id
3.     2: string instance_id
4.     3: string state
5.     4: optional list<ConnectionInfo> connection_infos
6.     5: optional string description
7. }
8.
9. struct ConnectionInfo {
10.    1: string type
11.    2: string name
12.    3: optional list<string> server_addrs
13.    4: string status
14.    5: optional string description
15. }
```

**ProcessInfo** Sent by the node manager, /usr/bin/contrail-nodemgr. Node manager is a monitor process per contrail virtual node that tracks the running state of the processes. The following is the ProcessInfo structure:

```

16. struct ProcessInfo {
17.    1: string          process_name
18.    2: string          process_state
```

```
19.  3: u32                start_count
20.  4: u32                stop_count
21.  5: u32                exit_count
22.  // time when the process last entered running stage
23.  6: optional string     last_start_time
24.  7: optional string     last_stop_time
25.  8: optional string     last_exit_time
26.  9: optional list<string> core_file_list
27. }
```

**Example: NodeStatus** The following is an example output of NodeStatus obtained from the Rest API:

`http://:8081/analytics/uves/control-...ilt=NodeStatus .`

```
{
  NodeStatus:
  {
    process_info:
    [
      {
        process_name: "contrail-control",
        process_state: "PROCESS_STATE_RUNNING",
        last_stop_time: null,
        start_count: 1,
        core_file_list: [ ],
        last_start_time: "1409002143776558",
        stop_count: 0,
        last_exit_time: null,
        exit_count: 0
      },
    ]
  }
}
```



```
    process_name: "contrail-control-nodemgr",
    process_state: "PROCESS_STATE_RUNNING",
    last_stop_time: null,
    start_count: 1,
    core_file_list: [ ],
    last_start_time: "1409002141773481",
    stop_count: 0,
    last_exit_time: null,
    exit_count: 0
  },
{
  process_name: "contrail-dns",
  process_state: "PROCESS_STATE_RUNNING",
  last_stop_time: null,
  start_count: 1,
  core_file_list: [ ],
  last_start_time: "1409002145778383",
  stop_count: 0,
  last_exit_time: null,
  exit_count: 0
},
{
  process_name: "contrail-named",
  process_state: "PROCESS_STATE_RUNNING",
  last_stop_time: null,
  start_count: 1,
  core_file_list: [ ],
```

```
    last_start_time: "1409002147780118",
    stop_count: 0,
    last_exit_time: null,
    exit_count: 0
  }
],
process_status:
[

{
  instance_id: "0",
  module_id: "ControlNode",
  state: "Functional",
  description: null,
  connection_infos:
  [

    {
      server_addrs:
      [
        "10.84.13.45:8443"
      ],
      status: "Up",
      type: "IFMap",
      name: "IFMapServer",
      description: "Connection with IFMap Server (irond)"
    },

    {
      server_addrs:
      [
```

```
"10.84.13.45:8086"

],

status: "Up",

type: "Collector",

name: null,

description: "Established"

},

{

  server_addrs:

  [

    "10.84.13.45:5998"

  ],

  status: "Up",

  type: "Discovery",

  name: "Collector",

  description: "SubscribeResponse"

},

{

  server_addrs:

  [

    "10.84.13.45:5998"

  ],

  status: "Up",

  type: "Discovery",

  name: "IfmapServer",

  description: "SubscribeResponse"

},

{
```

```

    server_addrs:
    [
        "10.84.13.45:5998"
    ],
    status: "Up",
    type: "Discovery",
    name: "xmpp-server",
    description: "Publish Response - HeartBeat"
}
]
}
]
}
}

```

## Using Introspect to Get Process Status

The user can also view the state of a specific process by using the introspect mechanism.

### Example: Introspect of NodeStatus

The following is an example of the process state of contrail-control that is obtained by using

[http://server-ip:8083/Snh\\_SandeshUVECacheReq?x=NodeStatus](http://server-ip:8083/Snh_SandeshUVECacheReq?x=NodeStatus)



**NOTE:** The example output is the ProcessStatus of only one process of contrail-control. It does not show the full aggregated status of the control node through its UVE (as in the previous example).

```
root@a6s45:~# curl http://10.84.13.45:8083/Snh_SandeshU...q?x=NodeStatus
```

```

<?xml-stylesheet type="text/xsl" href="/universal_parse.xml"?><__NodeStatusUVE_list
type="slist"><NodeStatusUVE type="sandesh"><data type="struct"
identifier="1"><NodeStatus><name type="string" identifier="1"
key="ObjectBgpRouter">a6s45</name><process_status type="list" identifier="4"
aggtype="union"><list type="struct" size="1"><ProcessStatus><module_id type="string"
identifier="1">ControlNode</module_id><instance_id type="string"
identifier="2">0</instance_id><state type="string"
identifier="3">Functional</state><connection_infos type="list" identifier="4"><list
type="struct" size="5"><ConnectionInfo><type type="string"
identifier="1">IFMap</type><name type="string"

```

```

identifier="5"/> </description> </ProcessStatus> </list> </processStatus> </NodeStatus> </tbody> </NodeStatusUVE> </SandeshUVECacheResp>
type="sandesh"><returned type="u32" identifier="1">1</returned><more type="bool"
identifier="0">false</more></SandeshUVECacheResp></ NodeStatusUVE list>

```

## contrail-status script

The contrail-status script is used to give the status of the Contrail processes on a server.

The contrail-status script first checks if a process is running, and if it is, performs introspect into the process to get its functionality status, then outputs the aggregate status.

The possible states to display include:

- active - the process is running and functional; the internal state is good
- inactive - stopped by user
- failed – the process exited too quickly and has not restarted
- initializing - the process is running, but the internal state is not yet functional.

Example Output:  
Contrail-Status Script

The following is an example output from the `contrail-status` script.

```
root@a6s45:~# contrail-status
```

**== Contrail vRouter ==**

supervisor-vrouter:     active  
contrail-vrouter-agent   active  
contrail-vrouter-nodemgr  active

**== Contrail Control ==**

supervisor-control:     active  
contrail-control         active  
contrail-control-nodemgr  active  
contrail-dns             active  
contrail-named            active

**== Contrail Analytics ==**

supervisor-analytics:    active  
contrail-analytics-api    active  
contrail-analytics-nodemgr  active  
contrail-collector        active  
contrail-query-engine     active

**== Contrail Config ==**

supervisor-config:       active  
contrail-api:0            active  
contrail-config-nodemgr   active  
contrail-discovery:0      active  
contrail-schema           active  
contrail-svc-monitor      active  
ifmap                     active  
rabbitmq-server           active

== Contrail Web UI ==

supervisor-webui: active

contrail-webui active

contrail-webui-middleware active

redis-webui active

== Contrail Database ==

supervisord-contrail-database:active

contrail-database active

contrail-database-nodemgr active

## Log and Flow Information APIs

In Contrail, log and flow analytics information is collected and stored using a horizontally scalable Contrail collector and NoSQL database. The **analytics-api** server provides REST APIs to extract this information using queries. The queries use well-known SQL syntax, hiding the underlying complexity of the NoSQL tables.

- [HTTP GET APIs on page 353](#)
- [HTTP POST API on page 354](#)
- [POST Data Format Example on page 354](#)
- [Query Types on page 356](#)
- [Examining Query Status on page 356](#)
- [Examining Query Chunks on page 356](#)
- [Example Queries for Log and Flow Data on page 356](#)

### HTTP GET APIs

Use the following GET APIs to identify tables and APIs available for querying.

**/analytics/tables** -- lists the SQL-type tables available for querying, including the hrefs for each of the tables

**/analytics/table/<table>** -- lists the APIs available to get information for a given table

**/analytics/table/<table>/schema** -- lists the schema for a given table

## HTTP POST API

Use the following POST API information to extract data from a table.

**/analytics/query** -- format your query using the following SQL syntax:

```
SELECT field1, field2 ...  
FROM table1  
WHERE field1 = value1 AND field2 = value2 ...  
FILTER BY ...  
SORT BY ...  
LIMIT n
```

Additionally, it is mandatory to include the start time and the end time for the data range to define the time period for the query data. The parameters of the query are passed through POST data, using the following fields:

**start\_time** — the start of the time period  
**end\_time** — the end of the time period  
**table** — the table from which to extract data  
**select\_fields** — the columns to display in the extracted data  
**where** — the list of match conditions

## POST Data Format Example

The POST data is in **JSON** format, stored in an **idl** file. A sample file is displayed in the following.



**NOTE:** The result of the query API is also in **JSON** format.

```
/*  
 * Copyright (c) 2013 Juniper Networks, Inc. All rights reserved.  
 */  
  
/*  
 * query_rest.idl  
 *  
 * IDL definitions for query engine REST API  
 *  
 * PLEASE NOTE: After updating this file, do update json_parse.h  
 *  
 */  
  
enum match_op {  
    EQUAL = 1,  
    NOT_EQUAL = 2,  
    IN_RANGE = 3,  
};
```



```

    NOT_IN_RANGE = 4, // not supported currently
    // following are only for numerical column fields
    LEQ = 5, // column value is less than or equal to filter value
    GEQ = 6, // column value is greater than or equal to filter value
    PREFIX = 7, // column value has the "value" field as prefix
    REGEX_MATCH = 8 // for filters only
}

enum sort_op {
    ASCENDING = 1,
    DESCENDING = 2,
}

struct match {
    1: string name;
    2: string value;
    3: match_op op;
    4: optional string value2; // this is for only RANGE match
}

typedef list<match> term; (AND of match)

enum flow_dir_t {
    EGRESS = 0,
    INGRESS = 1
}

struct query {
    1: string table; // Table to query (FlowSeriesTable, MessageTable,
ObjectVNTTable, ObjectVMTable, FlowRecordTable)
    2: i64 start_time; // Microseconds in UTC since Epoch
    3: i64 end_time; // Microseconds in UTC since Epoch
    4: list<string> select_fields; // List of SELECT fields
    5: list<term> where; // WHERE (OR of terms)
    6: optional sort_op sort;
    7: optional list<string> sort_fields;
    8: optional i32 limit;
    9: optional flow_dir_t dir; // direction of flows being queried
    10: optional list<match> filter; // filter the processed result by value
}

struct flow_series_result_entry {
    1: optional i64 T; // Timestamp of the flow record
    2: optional string sourcevn;
    3: optional string sourceip;
    4: optional string destvn;
    5: optional string destip;
    6: optional i32 protocol;
    7: optional i32 sport;
    8: optional i32 dport;
    9: optional flow_dir_t direction_ing;
    10: optional i64 packets; // mutually exclusive to 12,13
    11: optional i64 bytes; // mutually exclusive to 12,13
    12: optional i64 sum_packets; // represented as "sum(packets)" in JSON
    13: optional i64 sum_bytes; // represented as "sum(bytes)" in JSON
};

typedef list<flow_series_result_entry> flow_series_result;

```

## Query Types

The **analytics-api** supports two types of queries. Both types use the same POST parameters as described in POST API.

- **sync** — Default query mode. The results are sent inline with the query processing.
- **async** — To execute a query in async mode, attach the following header to the POST request: **Expect: 202-accepted**.

## Examining Query Status

For an asynchronous query, the **analytics-api** responds with the code: **202 Accepted**. The response contents are a status entity href URL of the form: **/analytics/query/<QueryID>**. The QueryID is assigned by the **analytics-api**. To view the response contents, poll the status entity by performing a GET action on the URL. The status entity has a variable named **progress**, with a number between 0 and 100, representing the approximate percentage completion of the query. When progress is 100, the query processing is complete.

## Examining Query Chunks

The status entity has an element named **chunks** that lists portions (chunks) of query results. Each element of this list has three fields: **start\_time**, **end\_time**, **href**. The **analytics-api** determines how many chunks to list to represent the query data. A chunk can include an empty string ("" ) to indicate that the data query is not yet available. If a partial result is available, the chunk href is of the form: **/analytics/query/<QueryID>/chunk-partial/<chunk number>**. When the final result of a chunk is available, the href is of the form: **/analytics/query/<QueryID>/chunk-final/<chunk number>**.

## Example Queries for Log and Flow Data

The following example query lists the tables available for query.

```
[root@host ~]# curl 127.0.0.1:8081/analytics/tables | python -mjson.tool
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   Dload  Upload   Total   Spent    Left     Speed
100    846    100    846     0     0    509k      0  --:--:-- --:--:-- --:--:--   826k
[
  {
    "href": "http://127.0.0.1:8081/analytics/table/MessageTable",
    "name": "MessageTable"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectVNTTable",
    "name": "ObjectVNTTable"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectVMTable",
    "name": "ObjectVMTable"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectVRouter",
    "name": "ObjectVRouter"
  }
]
```

```

    },
    {
      "href": "http://127.0.0.1:8081/analytics/table/ObjectBgpPeer",
      "name": "ObjectBgpPeer"
    },
    {
      "href": "http://127.0.0.1:8081/analytics/table/ObjectRoutingInstance",
      "name": "ObjectRoutingInstance"
    },
    {
      "href": "http://127.0.0.1:8081/analytics/table/ObjectXmppConnection",
      "name": "ObjectXmppConnection"
    },
    {
      "href": "http://127.0.0.1:8081/analytics/table/FlowRecordTable",
      "name": "FlowRecordTable"
    },
    {
      "href": "http://127.0.0.1:8081/analytics/table/FlowSeriesTable",
      "name": "FlowSeriesTable"
    }
  ]
}

```

The following example query lists details for the table named **MessageTable**.

```

[root@host ~]# curl 127.0.0.1:8081/analytics/table/MessageTable | python
-mjson.tool
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total    Spent    Left  Speed
100   192   100   192    0    0   102k      0  --:--:-- --:--:-- --:--:--  187k
[
  {
    "href": "http://127.0.0.1:8081/analytics/table/MessageTable/schema",
    "name": "schema"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/MessageTable/column-values",
    "name": "column-values"
  }
]

```

The following example query lists the schema for the table named MessageTable.

```

[root@host ~]# curl 127.0.0.1:8081/analytics/table/MessageTable/schema | python
-mjson.tool
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total    Spent    Left  Speed
100    630   100    630    0    0   275k      0  --:--:-- --:--:-- --:--:--  307k
{
  "columns": [
    {
      "datatype": "int",
      "index": "False",
      "name": "MessageTS"
    },
    {
      "datatype": "string",
      "index": "True",
      "name": "Source"
    }
  ]
}

```

```

        "datatype": "string",
        "index": "True",
        "name": "ModuleId"
    },
    {
        "datatype": "string",
        "index": "True",
        "name": "Category"
    },
    {
        "datatype": "int",
        "index": "True",
        "name": "Level"
    },
    {
        "datatype": "int",
        "index": "False",
        "name": "Type"
    },
    {
        "datatype": "string",
        "index": "True",
        "name": "Messagetype"
    },
    {
        "datatype": "int",
        "index": "False",
        "name": "SequenceNum"
    },
    {
        "datatype": "string",
        "index": "False",
        "name": "Context"
    },
    {
        "datatype": "string",
        "index": "False",
        "name": "Xmlmessage"
    }
    ],
    "type": "LOG"
}

```

The following set of example queries explore a message table.

```

root@a6s45:~# cat filename
{ "end_time": "now" , "select_fields": ["MessageTS", "Source", "ModuleId",
"Category", "Messagetype", "SequenceNum", "Xmlmessage", "Type", "Level",
"NodeType", "InstanceId"] , "sort": 1 , "sort_fields": ["MessageTS"] ,
"start_time": "now-10m" , "table": "MessageTable" , "where": {"name": "ModuleId",
"value": "contrail-control", "op": 1, "suffix": null, "value2": null}, {"name":
"Messagetype", "value": "BGPRouterInfo", "op": 1, "suffix": null, "value2": null}
}

root@a6s45:~#
root@a6s45:~# curl -X POST --data @filename 127.0.0.1:8081/analytics/query --header
"Content-Type:application/json" | python -mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  9765    0  9297  100   468    9168    461   0:00:01  0:00:01 --:--:--  9177
{

```

```

"value": [
  {
    "Category": null,
    "InstanceId": "0",
    "Level": 2147483647,
    "MessageTS": 1428442589947392,
    "Messagetype": "BGPRouterInfo",
    "ModuleId": "contrail-control",
    "NodeType": "Control",
    "SequenceNum": 1302,
    "Source": "a6s45",
    "Type": 6,
    "Xmlmessage": "<BGPRouterInfo type=''><data
type=''><BgpRouterState><name type=''
>a6s45</name><cpu_info type=''><CpuLoadInfo><num_cpu type=''>4</num_cpu
><meminfo type=''><MemInfo><virt type=''>438436</virt><peakvirt type=''
>561048</peakvirt><res type=''>12016</res></MemInfo></meminfo><cpu_share
type=''>0.0416667</cpu_share></CpuLoadInfo></cpu_info><cpu_share type=''
>0.0416667</cpu_share></BgpRouterState></data></BGPRouterInfo>"
  },
  {
    "Category": null,
    "InstanceId": "0",
    "Level": 2147483647,
    ...

```

## Working with Neutron

OpenStack's networking solution, Neutron, has representative elements for Contrail elements for Network (VirtualNetwork), Port (VirtualMachineInterface), Subnet (IpamSubnets), and Security-Group. The Neutron plugin translates the elements from one representation to another.

- [Data Structure on page 359](#)
- [Network Sharing in Neutron on page 360](#)
- [Commands for Neutron Network Sharing on page 360](#)
- [Support for Neutron APIs on page 361](#)
- [Contrail Neutron Plugin on page 361](#)
- [DHCP Options on page 362](#)
- [Incompatibilities on page 362](#)

## Data Structure

Although the actual data between Neutron and Contrail is similar, the listings of the elements differs significantly. In the Contrail API, the networking elements list is a summary, containing only the UUID, FQ name, and an href, however, in Neutron, all details of each resource are included in the list.

The Neutron plugin has an inefficient list retrieval operation, especially at scale, because it:

- reads a list of resources (for example. `GET /virtual-networks`), then

- iterates and reads in the details of the resource (**GET /virtual-network/<uuid>** ).

As a result, the API server spends most of the time in this type of GET operation just waiting for results from the Cassandra database.

The following features in Contrail improve performance with Neutron:

- An optional detail query parameter is added in the GET of collections so that the API server returns details of all the resources in the list, instead of just a summary. This is accompanied by changes in the Contrail API library so that a caller gets returned a list of the objects.
- The existing Contrail list API takes in an optional **parent\_id** query parameter to return information about the resource anchored by the parent.
- The Contrail API server reads objects from Cassandra in a multiget format into **obj\_uuid\_cf**, where object contents are stored, instead of reading in an xget/get format. This reduces the number of round-trips to and from the Cassandra database.

## Network Sharing in Neutron

Using Neutron, a deployer can make a network accessible to other tenants or projects by using one of two attributes on a network:

- set the **shared** attribute to allow sharing
- set the **router:external** attribute, when the plugin supports an **external\_net** extension

### *Using the Shared Attribute*

When a network has the **shared** attribute set, users in other tenants or projects, including non-admin users, can access that network, using:

```
neutron net-list --shared
```

Users can also launch a virtual machine directly on that network, using:

```
nova boot <other-parameters> --nic net-id=<shared-net-id>
```

### *Using the Router:External Attribute*

When a network has the **router:external** attribute set, users in other tenants or projects, including non-admin users, can use that network for allocating floating IPs, using:

```
neutron floatingip-create <router-external-net-id>
```

then associating the IP address pool with their instances.

## Commands for Neutron Network Sharing

The following table summarizes the most common Neutron commands used with Contrail.

Action	Command
List all shared networks.	<b>neutron net-list --shared</b>

Action	Command
Create a network that has the shared attribute.	<code>neutron net-create &lt;net-name&gt; --shared</code>
Set the shared attribute on an existing network.	<code>neutron net-update &lt;net-name&gt; -shared</code>
List all <code>router:external</code> networks.	<code>neutron net-list --router:external</code>
Create a network that has the <code>router:external</code> attribute.	<code>neutron net-create &lt;net-name&gt; -router:external</code>
Set the <code>router:external</code> attribute on an existing network.	<code>neutron net-update &lt;net-name&gt; -router:external</code>

## Support for Neutron APIs

The OpenStack Neutron project provides virtual networking services among devices that are managed by the OpenStack compute service. Software developers create applications by using the OpenStack Networking API v2.0 (Neutron).

Contrail provides the following features to increase support for OpenStack Neutron:

- Create a port independently of a virtual machine.
- Support for more than one subnet on a virtual network.
- Support for allocation pools on a subnet.
- Per tenant quotas.
- Enabling DHCP on a subnet.
- External router can be used for floating IPs.

For more information about using OpenStack Networking API v2.0 (Neutron), refer to: <http://docs.openstack.org/api/openstack-network/2.0/content/> and the OpenStack Neutron Wiki at: <http://wiki.openstack.org/wiki/Neutron>.

## Contrail Neutron Plugin

The Contrail Neutron plugin provides an implementation for the following core resources:

- Network
- Subnet
- Port

It also implements the following standard and upstreamed Neutron extensions:

- Security group
- Router IP and floating IP
- Per-tenant quota
- Allowed address pair

The following Contrail-specific extensions are implemented:

- Network IPAM
- Network policy
- VPC table and route table
- Floating IP pools

The plugin does not implement native bulk, pagination, or sort operations and relies on emulation provided by the Neutron common code.

## DHCP Options

In Neutron commands, DHCP options can be configured using `extra-dhcp-options` in `port-create`.

**Example**      `neutron port-create net1 --extra-dhcp-opt  
opt_name=<dhcp_option_name>,opt_value=<value>`

The `opt_name` and `opt_value` pairs that can be used are maintained in GitHub:

<https://github.com/Juniper/contrail-controller/wiki/Extra-DHCP-Options> .

## Incompatibilities

In the Contrail architecture, the following are known incompatibilities with the Neutron API.

- Filtering based on any arbitrary key in the resource is not supported. The only supported filtering is by **id**, **name**, and **tenant\_id**.
- To use a floating IP, it is not necessary to connect the public subnet and the private subnet to a Neutron router. Marking a public network with **router:external** is sufficient for a floating IP to be created and associated, and packet forwarding to it will work.
- The default values for quotas are sourced from `/etc/contrail/contrail-api.conf` and not from `/etc/neutron/neutron.conf`.

---

## Support for Amazon VPC APIs on Contrail OpenStack

- [Overview of Amazon Virtual Private Cloud on page 363](#)
- [Mapping Amazon VPC Features to OpenStack Contrail Features on page 363](#)
- [VPC and Subnets Example on page 364](#)
- [Euca2ools CLI for VPC and Subnets on page 365](#)
- [Security in VPC: Network ACLs Example on page 365](#)
- [Euca2ools CLI for Network ACLs on page 366](#)
- [Security in VPC: Security Groups Example on page 366](#)
- [Euca2ools CLI for Security Groups on page 367](#)
- [Elastic IPs in VPC on page 368](#)
- [Euca2ools CLI for Elastic IPs on page 368](#)



- [Euca2ools CLI for Route Tables on page 368](#)
- [Supported Next Hops on page 369](#)
- [Internet Gateway Next Hop Euca2ools CLI on page 369](#)
- [NAT Instance Next Hop Euca2ools CLI on page 369](#)
- [Example: Creating a NAT Instance with Euca2ools CLI on page 370](#)

## Overview of Amazon Virtual Private Cloud

The current Grizzly release of OpenStack supports Elastic Compute Cloud (EC2) API translation to OpenStack Nova, Quantum, and Keystone calls. EC2 APIs are used in Amazon Web Services (AWS) and virtual private clouds (VPCs) to launch virtual machines, assign IP addresses to virtual machines, and so on. A VPC provides a container where applications can be launched and resources can be accessed over the networking services provided by the VPC.

Contrail enhances its use of EC2 APIs to support the Amazon VPC APIs.

The Amazon VPC supports networking constructs such as: subnets, DHCP options, elastic IP addresses, network ACLs, security groups, and route tables. The Amazon VPC APIs are now supported on the Openstack Contrail distribution, so users of the Amazon EC2 APIs for their VPC can use the same scripts to move to an Openstack Contrail solution.

**Euca2ools** are command-line tools for interacting with Amazon Web Services (AWS) and other AWS-compatible web services, such as OpenStack. **Euca2ools** have been extended in OpenStack Contrail to add support for the Amazon VPC, similar to the support that already exists for the Amazon EC2 CLI.

For more information about Amazon VPC and AWS EC2, see:

- Amazon VPC documentation:  
[http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC\\_Introduction.html](http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Introduction.html)
- Amazon VPC API list:  
<http://docs.aws.amazon.com/AWSEC2/latest/APIReference/query-apis.html>

## Mapping Amazon VPC Features to OpenStack Contrail Features

The following table compares Amazon VPC features to their equivalent features in OpenStack Contrail.

**Table 66: Amazon VPC and OpenStack Contrail Feature Comparison**

Amazon VPC Feature	OpenStack Contrail Feature
VPC	Project
Subnets	Networks (Virtual Networks)
DHCP options	IPAM
Elastic IP	Floating IP

Table 66: Amazon VPC and OpenStack Contrail Feature Comparison (*continued*)

Amazon VPC Feature	OpenStack Contrail Feature
Network ACLs	Network ACLs
Security Groups	Security Groups
Route Table	Route Table

## VPC and Subnets Example

When creating a new VPC, the user must provide a classless inter-domain routing (CIDR) block of which all subnets in this VPC will be part.

In the following example, a VPC is created with a CIDR block of 10.1.0.0/16. A subnet is created within the VPC CIDR block, with a CIDR block of 10.1.1.0/24. The VPC has a default network ACL named **acl-default**.

All subnets created in the VPC are automatically associated to the default network ACL. This association can be changed when a new network ACL is created. The last command in the list below creates a virtual machine using the image **ami-000000003** and launches with an interface in **subnet-5eb34ed2**.

```
# euca-create-vpc 10.1.0.0/16
VPC VPC:vpc-8352aa59 created

# euca-describe-vpcs
VpcId      CidrBlock      DhcpOptions
-----
vpc-8352aa59  10.1.0.0/16    None

# euca-create-subnet -c 10.1.1.0/24 vpc-8352aa59
Subnet: subnet-5eb34ed2 created

# euca-describe-subnets
Subnet-id   Vpc-id         CidrBlock
-----
subnet-5eb34ed2  vpc-8352aa59  10.1.1.0/24

# euca-describe-network-acls
AclId
-----
acl-default(def)
vpc-8352aa59

      Rule   Dir   Action  Proto  Port  Range  Cidr
      ----   --   -
      100    ingress allow   -1     0    65535  0.0.0.0/0
      100    egress  allow   -1     0    65535  0.0.0.0/0
      32767  ingress deny    -1     0    65535  0.0.0.0/0
      32767  egress  deny    -1     0    65535  0.0.0.0/0

      Association      SubnetId      AclId
      -----
      aclassoc-0c549d66  subnet-5eb34ed2  acl-default
```

```
# euca-run-instances -s subnet-5eb34ed2 ami-00000003
```

## Euca2ools CLI for VPC and Subnets

The following **euca2ools** CLI commands are used to create, define, and delete VPCs and subnets:

- **euca-create-vpc**
- **euca-delete-vpc**
- **euca-describe-vpcs**
- **euca-create-subnet**
- **euca-delete-subnet**
- **euca-describe-subnets**

## Security in VPC: Network ACLs Example

Network ACLs support ingress and egress rules for traffic classification and filtering. The network ACLs are applied at a subnet level.

In the following example, a new ACL, **acl-ba7158**, is created and an existing subnet is associated to the new ACL.

```
# euca-create-network-acl vpc-8352aa59
acl-ba7158c
```

```
# euca-describe-network-acls
```

```
AclId
```

```
-----
```

```
acl-default(def)
vpc-8352aa59
```

Rule	Dir	Action	Proto	Port	Range	Cidr
----	---	-----	-----	----	-----	----
100	ingress	allow	-1	0	65535	0.0.0.0/0
100	egress	allow	-1	0	65535	0.0.0.0/0
32767	ingress	deny	-1	0	65535	0.0.0.0/0
32767	egress	deny	-1	0	65535	0.0.0.0/0

Association	SubnetId	AclId
-----	-----	-----
aclassoc-0c549d66	subnet-5eb34ed2	acl-default

```
AclId
```

```
-----
```

```
acl-ba7158c
vpc-8352aa59
```

Rule	Dir	Action	Proto	Port	Range	Cidr
----	---	-----	-----	----	-----	----
32767	ingress	deny	-1	0	65535	0.0.0.0/0
32767	egress	deny	-1	0	65535	0.0.0.0/0

```
# euca-replace-network-acl-association -a aclassoc-0c549d66 acl-ba7158c
aclassoc-0c549d66

# euca-describe-network-acls
AcId
-----
acl-default(def)
vpc-8352aa59
      Rule   Dir   Action  Proto  Port  Range  Cidr
      ----   --   -
      100    ingress allow   -1     0    65535  0.0.0.0/0
      100    egress  allow   -1     0    65535  0.0.0.0/0
      32767  ingress deny    -1     0    65535  0.0.0.0/0
      32767  egress  deny    -1     0    65535  0.0.0.0/0

      Association      SubnetId      AcId
      -----
      acl-ba7158c
      vpc-8352aa59
      Rule   Dir   Action  Proto  Port  Range  Cidr
      ----   --   -
      32767  ingress deny    -1     0    65535  0.0.0.0/0
      32767  egress  deny    -1     0    65535  0.0.0.0/0

      Association      SubnetId      AcId
      -----
      aclassoc-0c549d66  subnet-5eb34ed2  acl-ba7158c
```

## Euca2ools CLI for Network ACLs

The following **euca2ools** CLI commands are used to create, define, and delete VPCs and subnets:

- **euca-create-network-acl**
- **euca-delete-network-acl**
- **euca-replace-network-acl-association**
- **euca-describe-network-acls**
- **euca-create-network-acl-entry**
- **euca-delete-network-acl-entry**
- **euca-replace-network-acl-entry**

## Security in VPC: Security Groups Example

Security groups provide virtual machine level ingress/egress controls. Security groups are applied to virtual machine interfaces.

In the following example, a new security group is created. The rules can be added or removed for the security group based on the commands listed for **euca2ools**. The last line launches a virtual machine using the newly created security group.

```
# euca-describe-security-groups
```

GroupId	VpcId	Name	Description
-----	-----	----	-----
sg-6d89d7e2	vpc-8352aa59	default	

Direction	Proto	Start	End	Remote
-----	-----	-----	---	-----
Ingress	any	0	65535	[0.0.0.0/0]
Egress	any	0	65535	[0.0.0.0/0]

```
# euca-create-security-group -d "TestGroup" -v vpc-8352aa59 testgroup
GROUP sg-c5b9d22a testgroup TestGroup
```

```
# euca-describe-security-groups
```

GroupId	VpcId	Name	Description
-----	-----	----	-----
sg-6d89d7e2	vpc-8352aa59	default	

Direction	Proto	Start	End	Remote
-----	-----	-----	---	-----
Ingress	any	0	65535	[0.0.0.0/0]
Egress	any	0	65535	[0.0.0.0/0]

GroupId	VpcId	Name	Description
-----	-----	----	-----
sg-c5b9d22a	vpc-8352aa59	testgroup	TestGroup

Direction	Proto	Start	End	Remote
-----	-----	-----	---	-----
Egress	any	0	65535	[0.0.0.0/0]

```
# euca-run-instances -s subnet-5eb34ed2 -g testgroup ami-00000003
```

## Euca2ools CLI for Security Groups

The following **euca2ools** CLI commands are used to create, define, and delete security groups:

- **euca-create-security-group**
- **euca-delete-security-group**
- **euca-describe-security-groups**
- **euca-authorize-security-group-egress**
- **euca-authorize-security-group-ingress**

- **euca-revoke-security-group-egress**
- **euca-revoke-security-group-ingress**

## Elastic IPs in VPC

Elastic IPs in VPCs are equivalent to the floating IPs in the Contrail Openstack solution.

In the following example, a floating IP is requested from the system and assigned to a particular virtual machine. The prerequisite is that the provider or Contrail administrator has provisioned a network named “public” and allocated a floating IP pool to it. This “public” floating IP pool is then internally used by the tenants to request public IP addresses that they can use and attach to virtual machines.

```
# euca-allocate-address --domain vpc
ADDRESS 10.84.14.253      eipalloc-78d9a8c9

# euca-describe-addresses --filter domain=vpc
Address      Domain    AllocationId      InstanceId(AssociationId)
-----
10.84.14.253  vpc      eipalloc-78d9a8c9

# euca-associate-address -a eipalloc-78d9a8c9 i-00000008
ADDRESS eipassoc-78d9a8c9

# euca-describe-addresses --filter domain=vpc
Address      Domain    AllocationId      InstanceId(AssociationId)
-----
10.84.14.253  vpc      eipalloc-78d9a8c9  i-00000008(eipassoc-78d9a8c9)
```

## Euca2ools CLI for Elastic IPs

The following **euca2ools** CLI commands are used to create, define, and delete elastic IPs:

- **euca-allocate-address**
- **euca-release-address**
- **euca-describe-addresses**
- **euca-associate-address**
- **euca-disassociate-address**

## Euca2ools CLI for Route Tables

Route tables can be created in an Amazon VPC and associated with subnets. Traffic exiting a subnet is then looked up in the route table and, based on the route lookup result, the next hop is chosen.

The following **euca2ools** CLI commands are used to create, define, and delete route tables:

- **euca-create-route-table**
- **euca-delete-route-table**

- **euca-describe-route-tables**
- **euca-associate-route-table**
- **euca-disassociate-route-table**
- **euca-replace-route-table-association**
- **euca-create-route**
- **euca-delete-route**
- **euca-replace-route**

## Supported Next Hops

The supported next hops for the current release are:

- Local Next Hop

Designating local next hop indicates that all subnets in the VPC are reachable for the destination prefix.

- Internet Gateway Next Hop

This next hop is used for traffic destined to the Internet. All virtual machines using the Internet gateway next hop are required to use an Elastic IP to reach the Internet, because the subnet IPs are private IPs.

- NAT instance

To create this next hop, the user needs to launch a virtual machine that provides network address translation (NAT) service. The virtual machine has two interfaces: one internal and one external, both of which are automatically created. The only requirement here is that a “public” network should have been provisioned by the admin, because the second interface of the virtual machine is created in the “public” network.

## Internet Gateway Next Hop Euc2ools CLI

The following **euc2ools** CLI commands are used to create, define, and delete Internet gateway next hop:

- **euca-attach-internet-gateway**
- **euca-create-internet-gateway**
- **euca-delete-internet-gateway**
- **euca-describe-internet-gateways**
- **euca-detach-internet-gateway**

## NAT Instance Next Hop Euc2ools CLI

The following **euc2ools** CLI commands are used to create, define, and delete NAT instance next hops:

- **euca-run-instances**

- `euca-terminate-instances`

## Example: Creating a NAT Instance with Euca2ools CLI

The following example creates a NAT instance and creates a default route pointing to the NAT instance.

```
# euca-describe-route-tables
RouteTableId    Main    VpcId          AssociationId    SubnetId
-----
rtb-default     yes     vpc-8352aa59   rtbassoc-0c549d66  subnet-5eb34ed2

                Prefix          NextHop
                -----
                10.1.0.0/16      local

# euca-describe-images
IMAGE  ami-00000003  None (ubuntu)      2c88a895fdea4461a81e9b2c35542130
IMAGE  ami-00000005  None (nat-service) 2c88a895fdea4461a81e9b2c35542130

# euca-run-instances ami-00000005

# euca-create-route --cidr 0.0.0.0/0 -i i-00000006 rtb-default

# euca-describe-route-tables
RouteTableId    Main    VpcId          AssociationId    SubnetId
-----
rtb-default     yes     vpc-8352aa59   rtbassoc-0c549d66  subnet-5eb34ed2

                Prefix          NextHop
                -----
                10.1.0.0/16      local
                0.0.0.0/0        i-00000006
```



## CHAPTER 14

# Optimizing

- [vRouter Command Line Utilities on page 371](#)
- [Route Target Filtering on page 387](#)
- [Source Network Address Translation \(SNAT\) on page 389](#)

### vRouter Command Line Utilities

---

- [Overview on page 371](#)
- [vif Command on page 372](#)
- [flow Command on page 374](#)
- [vrfstats Command on page 376](#)
- [rt Command on page 376](#)
- [dropstats Command on page 377](#)
- [mpls Command on page 381](#)
- [mirror Command on page 383](#)
- [vxlan Command on page 384](#)
- [nh Command on page 385](#)

### Overview

This section describes the shell prompt utilities available for examining the state of the vrouter kernel module in Contrail.

The most useful commands for inspecting the Contrail vrouter module are summarized in the following table.

Command	Description
<b>vif</b>	Inspect vrouter interfaces associated with the vrouter module.
<b>flow</b>	Display active flows in a system.
<b>vrfstats</b>	Display next hop statistics for a particular VRF.
<b>rt</b>	Display routes in a VRF.

Command	Description
<b>dropstats</b>	Inspect packet drop counters in the router.
<b>mpls</b>	Display the input label map programmed into the router.
<b>mirror</b>	Display the mirror table entries.
<b>vxlan</b>	Display the vxlan table entries.
<b>nh</b>	Display the next hops that the router knows.
<b>--help</b>	Display all command options available for the current command.

The following sections describe each of the router utilities in detail.

## vif Command

The router requires router interfaces (**vif**) to forward traffic. Use the **vif** command to see the interfaces that are known by the router.



**NOTE:** Having interfaces only in the OS (Linux) is not sufficient for forwarding. The relevant interfaces must be added to router. Typically, the set up of interfaces is handled by components like nova-compute or router agent.

### Example: vif --list

```
# vif --list
vif0/0 OS: pkt0
    Type:Agent HWaddr:00:00:5e:00:01:00 IPAddr:0
    Vrf:65535 Flags:L3 MTU:1514 Ref:2
    RX packets:6591 bytes:648577 errors:0
    TX packets:12150 bytes:1974451 errors:0
vif0/1 OS: vhost0
    Type:Host HWaddr:00:25:90:c3:08:68 IPAddr:0
    Vrf:0 Flags:L3 MTU:1514 Ref:3
    RX packets:3446598 bytes:4478599344 errors:0
    TX packets:851770 bytes:1337017154 errors:0
vif0/2 OS: p1p0p0 (Speed 1000, Duplex 1)
    Type:Physical HWaddr:00:25:90:c3:08:68 IPAddr:0
    Vrf:0 Flags:L3 MTU:1514 Ref:22
    RX packets:1643238 bytes:1391655366 errors:2812
    TX packets:3523278 bytes:6806058059 errors:0
vif0/18 OS: tap3214fc7e-88
    Type:Virtual HWaddr:00:00:5e:00:01:00 IPAddr:0
    Vrf:13 Flags:PL3L2 MTU:9160 Ref:6
    RX packets:60 bytes:4873 errors:0
    TX packets:21 bytes:2158 errors:0
```

Table 67: vif Fields

vif Output Field	Description
<b>vif0/X</b>	The vrouter assigned name, where 0 is the router id and X is the index allocated to the interface within the vrouter.
<b>OS: pkt0</b>	The <b>pkt0</b> (in this case) is the name of the actual OS (Linux) visible interface name. For physical interfaces, the speed and the duplex settings are also displayed.
<b>Type:xxxxx</b>	<p><b>Type:Virtual HWaddr:00:00:5e:00:01:00 IPAddr:0</b></p> <p>The type of interface and its IP address, as defined by vrouter. The values can be different from what is seen in the OS. Types defined by vrouter include:</p> <ul style="list-style-type: none"> <li>• Virtual – Interface of a virtual machine (VM).</li> <li>• Physical – Physical interface (NIC) in the system.</li> <li>• Host – An interface toward the host.</li> <li>• Agent – An interface used to trap packets to the vrouter agent when decisions need to be made for the forwarding path.</li> </ul>
<b>Vrf:xxxxx</b>	<p><b>Vrf:65535 Flags:L3 MTU:1514 Ref:2</b></p> <p>The identifier of the <b>vrf</b> to which the interface is assigned, the flags set on the interface, the MTU as understood by vrouter, and a reference count of how many individual entities actually hold reference to the interface (mainly of debugging value).</p> <p>Flag options identify that the following are enabled for the interface:</p> <ul style="list-style-type: none"> <li>• P - Policy</li> <li>• L3 - Layer 3 forwarding</li> <li>• L2 - Layer 2 bridging</li> <li>• X - Cross connect mode, only set on physical and host interfaces, indicating that packets are moved between physical and host directly, with minimal intervention by vrouter. Typically set when the agent is not alive or not in good shape.</li> <li>• Mt - Mirroring transmit direction</li> <li>• Mr - Mirroring receive direction</li> <li>• Tc - Checksum offload on the transmit side. Valid only on the physical interface.</li> </ul>
<b>Rx</b>	<p><b>RX packets:60 bytes:4873 errors:0</b></p> <p>Packets received by vrouter from this interface.</p>
<b>Tx</b>	<p><b>TX packets:21 bytes:2158 errors:0</b></p> <p>Packets transmitted out by vrouter on this interface.</p>

**vif Options** Use **vif --help** to display all options available for the vif command. Following is a brief description of each option.



**NOTE:** It is not recommended to use the following options unless you are very experienced with the system utilities.

```
# vif --help
Usage: vif [--create <intf_name> --mac <mac>]
      [--add <intf_name> --mac <mac> --vrf <vrf>
      --type [vhost|agent|physical|virtual][--policy, --mode <mode:x>]]
      [--delete <intf_id>]
      [--get <intf_id>][--kernel]
      [--set <intf_id> --vlan <vlan_id> --vrf <vrf_id>]
      [--list]
      [--help]
```

Option	Description
<b>--create</b>	Creates a 'Host' interface with name <b>&lt;intf_name&gt;</b> and mac <b>&lt;mac&gt;</b> on the host kernel. The 'vhost0' interface that you see on Linux is a typical example of invocation of this command.
<b>--add</b>	Adds the existing interfaces in the host OS to vrouter, with type and flag options.
<b>--delete</b>	Deletes the interface from vrouter. The <b>&lt;intf_id&gt;</b> is the vrouter interface id as given by <b>vif0/X</b> , where X is the IID
<b>--get</b>	Displays a specific interface. The <b>&lt;intf_id&gt;</b> is the vrouter interface id, unless the command is appended by the '--kernel' option, in which case the ID can be the kernel ID.
<b>--set</b>	Set working parameters of an interface. The only ones supported are the <b>vlan id</b> and the <b>vrf</b> . The <b>vlan id</b> as understood by vrouter differs from what one typically expects, and is relevant as of now only for interfaces of service instances.
<b>--list</b>	Display all of the interfaces of which the vrouter is aware.
<b>--help</b>	Display all options available for the current command.

## flow Command

Use the **flow** command to display all active flows in a system.

**Example: flow -l** Use **-l** to list everything in the flow table. The **-l** is the only relevant debugging option.

```
# flow -l
Flow table
  Index    Source:Port      Destination:Port  Proto(V)
-----
263484    1.1.1.252:1203   1.1.1.253:0      1 (3)
```

```

(Action:F, S(nh):91, Statistics:22/1848)
379480    1.1.1.253:1203    1.1.1.252:0    1 (3)
(Action:F, S(nh):75, Statistics:22/1848)

```

Each record in the flow table listing displays the index of the record, the source ip: source port, the destination ip: destination port, the inet protocol, and the source vrf to which the flow belongs.

Each new flow has to be approved by the vrouter agent. The agent does this by setting actions for each flow. There are three main actions associated with a flow table entry: Forward ('F'), Drop ('D'), and Nat ('N').

For NAT, there are additional flags indicating the type of NAT to which the flow is subject, including: SNAT (S), DNAT (D), source port translation (Ps), and destination port translation (Pd).

S(nh) indicates the source nexthop index used for the RPF check to validate that the traffic is from a known source. If the packet must go to an ECMP destination, E:X is also displayed, where 'X' indicates the destination to be used through the index within the ECMP next hop.

The Statistics field indicates the Packets/Bytes that hit this flow entry.

There is a Mirror Index field if the traffic is mirrored, listing the indices into the mirror table (which can be dumped by using **mirror --dump**).

If there is an explicit association between the forward and the reverse flows, as is the case with NAT, you will see a double arrow in each of the records with either side of the arrow displaying the flow index for that direction.

**Example: flow -r** Use **-r** to view all of the flow setup rates.

```

# flow -r
New =  2, Flow setup rate =  3 flows/sec, Flow rate =  3 flows/sec, for last  548 ms
New =  2, Flow setup rate =  3 flows/sec, Flow rate =  3 flows/sec, for last  543 ms
New = -2, Flow setup rate = -3 flows/sec, Flow rate = -3 flows/sec, for last  541 ms
New =  2, Flow setup rate =  3 flows/sec, Flow rate =  3 flows/sec, for last  544 ms
New = -2, Flow setup rate = -3 flows/sec, Flow rate = -3 flows/sec, for last  542 ms

```

**Example: flow --help** Use **--help** to display all options available for the flow command.

```

# flow --help
Usage:flow [-f flow_index][-d flow_index][-i flow_index]
        [--mirror=mirror table index]
        [-l]
    -f <flow_index> Set forward action for flow at flow_index <flow_index>
    -d <flow_index> Set drop action for flow at flow_index <flow_index>
    -i <flow_index> Invalidate flow at flow_index <flow_index>
    --mirror        mirror index to mirror to
    -l              List all flows
    -r              Start dumping flow setup rate
    --help          Print this help

```

## vrfstats Command

Use **vrfstats** to display statistics per next hop for a **vrf**. It is typically used to determine if packets are hitting the expected next hop.

### Example: vrfstats --dump

The **--dump** option displays the statistics for all vrfs that have seen traffic. In the following example, there was traffic only in **Vrf 0** (the public vrf). **Receives** shows the number of packets that came in the fabric destined to this location. **Encaps** shows the number of packets destined to the fabric.

If there is VM traffic going out on the fabric, the respective tunnel counters will increment.

```
# vrfstats --dump
Vrf: 0
Discards 414, Resolves 3, Receives 165334
Ecmp Composites 0, L3 Mcast Composites 0, L2 Mcast Composites 0, Fabric Composites
0, Multi Proto Composites 0
Udp Tunnels 0, Udp Mpls Tunnels 0, Gre Mpls Tunnels 0
L2 Encaps 0, Encaps 130955
```

### Example: vrfstats --get 0

Use **--get 0** to retrieve statistics for a particular **vrf**.

```
# vrfstats --get 0
Vrf: 0
Discards 418, Resolves 3, Receives 166929
Ecmp Composites 0, L3 Mcast Composites 0, L2 Mcast Composites 0, Fabric Composites
0, Multi Proto Composites 0
Udp Tunnels 0, Udp Mpls Tunnels 0, Gre Mpls Tunnels 0
L2 Encaps 0, Encaps 132179
```

### Example: vrfstats --help

```
Usage: vrfstats --get <vrf>
      --dump
      --help

--get <vrf>  Displays packet statistics for the vrf <vrf>

--dump      Displays packet statistics for all vrfs

--help      Displays this help message
```

## rt Command

Use the **rt** command to display all routes in a **vrf**.

### Example: rt --dump

The following example displays **inet** family routes for **vrf 0**.

```
# rt --dump 0

Kernel IP routing table 0/0/unicast

Destination    PPL    Flags    Label    Nexthop
0.0.0.0/8      0      -        5
1.0.0.0/8      0      -        5
```

2.0.0.0/8	0	-	5
3.0.0.0/8	0	-	5
4.0.0.0/8	0	-	5
5.0.0.0/8	0	-	5

In this example output, the first line displays the routing table that is being dumped. In **0/0/unicast**, the first 0 is for the router id, the next 0 is for the vrf id, and unicast identifies the unicast table. The router maintains separate tables for unicast and multicast routes. By default, if the **—table** option is not specified, only the unicast table is dumped.

Each record in the table output specifies the destination prefix length, the parent route prefix length from which this route has been expanded, the flags for the route, the MPLS label if the destination is a VM in another location, and the next hop id. To understand the second field “PPL”, it is good to keep in mind that the unicast routing table is internally implemented as an ‘mtree’.

The **Flags** field can have two values. **L** indicates that the label field is valid, and **H** indicates that **vroute** should proxy arp for this IP.

The **Nexthop** field indicates the next hop ID to which the route points.

**Example: rt --dump  
--table mcst**

To dump the multicast table, use the **—table** option with **mcst** as the argument.

```
# rt --dump 0 --table mcst

Kernel IP routing table 0/0/multicast

(Src,Group)                Nexthop

0.0.0.0,255.255.255.255
```

## dropstats Command

Use the dropstats command to see packet drop counters in router.

**Example: dropstats**

```
# dropstats

GARP                0

ARP notme           12904

Invalid ARPs        0

Invalid IF           0

Trap No IF           0

IF TX Discard        0

IF Drop              49
```

IF RX Discard	0
Flow Unusable	0
Flow No Memory	0
Flow Table Full	0
Flow NAT no rflow	0
Flow Action Drop	0
Flow Action Invalid	0
Flow Invalid Protocol	0
Flow Queue Limit Exceeded	0
Discards	34
TTL Exceeded	0
Mcast Clone Fail	0
Cloned Original	0
Invalid NH	2
Invalid Label	0
Invalid Protocol	0
Rewrite Fail	0
Invalid Mcast Source	0
Push Fails	0
Pull Fails	0
Duplicated	0
Head Alloc Fails	0
Head Space Reserve Fails	0
PCOW fails	0
Invalid Packet	0



Misc	0
Nowhere to go	0
Checksum errors	0
No Fmd	0
Invalid VNID	0
Fragment errors	0
Invalid Source	0

<b>dropstats ARP Block</b>	<p>GARP packets from VMs are dropped by vrouter, an expected behavior. In the example output, the first counter GARP indicates how many packets were dropped.</p> <p>ARP requests that are not handled by vrouter are dropped, for example, requests for a system that is not a host. These drops are counted by <b>ARP notme</b> counters.</p> <p>The <b>Invalid ARPs</b> counter is incremented when the Ethernet protocol is ARP, but the ARP operation was neither a request nor a response.</p>
<b>dropstats Interface Block</b>	<p><b>Invalid IF</b> counters are incremented normally during transient conditions, and should not be a concern.</p> <p><b>Trap No IF</b> counters are incremented when vrouter is not able to find the interface to trap the packets to vrouter agent, and should not happen in a working system.</p> <p><b>IF TX Discard</b> and <b>IF RX Discard</b> counters are incremented when vrouter is not in a state to transmit and receive packets, and typically happens when vrouter goes through a reset state or when the module is unloaded.</p> <p><b>IF Drop</b> counters indicate packets that are dropped in the interface layer. The increase can typically happen when interface settings are wrong.</p>
<b>dropstats Flow Block</b>	<p>When packets go through flow processing, the first packet in a flow is cached and the vrouter agent is notified so it can take actions on the packet according to the policies configured. If more packets arrive after the first packet but before the agent makes a decision on the first packet, then those new packets are dropped. The dropped packets are tracked by the Flow unusable counter.</p> <p>The <b>Flow No Memory</b> counter increments when the flow block doesn't have enough memory to perform internal operations.</p> <p>The <b>Flow Table Full</b> counter increments when the vrouter cannot install a new flow due to lack of available slots. A particular flow can only go in certain slots, and if all those slots are occupied, packets are dropped. It is possible that the flow table is not full, but the counter might increment.</p> <p>The <b>Flow NAT no rflow</b> counter tracks packets that are dropped when there is no reverse flow associated with a forward flow that had action set as NAT. For NAT, the vrouter needs both forward and reverse flows to be set properly. If they are not set, packets are dropped.</p> <p>The <b>Flow Action Drop</b> counter tracks packets that are dropped due to policies that prohibit a flow.</p> <p>The <b>Flow Action Invalid</b> counter usually does not increment in the normal course of time, and can be ignored.</p> <p>The <b>Flow Invalid Protocol</b> usually does not increment in the normal course of time, and can be ignored.</p> <p>The <b>Flow Queue Limit Exceeded</b> usually does not increment in the normal course of time, and can be ignored.</p>
<b>dropstats Miscellaneous Operational Block</b>	

The **Discard** counter tracks packets that hit a discard next hop. For various reasons interpreted by the agent and during some transient conditions, a route can point to a discard next hop. When packets hit that route, they are dropped.

The **TTL Exceeded** counter increments when the MPLS time-to-live goes to zero.

The **Mcast Clone Fail** happens when the vrouter is not able to replicate a packet for flooding.

The **Cloned Original** is an internal tracking counter. It is harmless and can be ignored.

The **Invalid NH** counter tracks the number of packets that hit a next hop that was not in a state to be used (usually in transient conditions) or a next hop that was not expected, or no next hops when there was a next hop expected. Such increments happen rarely, and should not continuously increment.

The **Invalid Label** counter tracks packets with an MPLS label unusable by vrouter because the value is not in the expected range.

The **Invalid Protocol** typically increments when the IP header is corrupt.

The **Rewrite Fail** counter tracks the number of times vrouter was not able to write next hop rewrite data to the packet.

The **Invalid Mcast Source** tracks the multicast packets that came from an unknown or unexpected source and thus were dropped.

The **Invalid Source** counter tracks the number of packets that came from an invalid or unexpected source and thus were dropped.

The remaining counters are of value only to developers.

## mpls Command

The **mpls** utility command displays the input label map that has been programmed in the vrouter.

### Example: mpls --dump

The **--dump** command dumps the complete label map. The output is divided into two columns. The first field is the label and the second is the next hop corresponding to the label. When an MPLS packet with the specified label arrives in the vrouter, it uses the next hop corresponding to the label to forward the packet.

```
# mpls --dump
```

```
MPLS Input Label Map
```

```
Label  NextHop
```

```
-----
```

```
16      9
```

```
17     11
```

You can inspect the operation on **nh 9** as follows:

```
# nh --get 9
```

```
Id:009 Type:Encap Fmly: AF_INET Flags:Valid, Policy, Rid:0 Ref_cnt:4
```

```
EncapFmly:0806 Oif:3 Len:14 Data:02 d0 60 aa 50 57 00 25 90 c3 08 69 08 00
```

The **nh** output shows that the next hop directs the packet to go out on the interface with index 3 (**Oif:3**) with the given rewrite data.

To check the index of 3, use the following:

```
# vif --get 3
```

```
vif0/3 OS: tapd060aa50-57
```

```
Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:0
```

```
Vrf:1 Flags:PL3L2 MTU:9160 Ref:6
```

```
RX packets:1056 bytes:103471 errors:0
```

```
TX packets:1041 bytes:102372 errors:0
```

The **--get 3** output shows that the index of 3 corresponds to a tap interface that goes to a VM.

You can also dump individual entries in the map using the **--get** option, as follows:

```
# mpls --get 16
```

```
MPLS Input Label Map
```

```
Label  NextHop
```

```
-----
```

```
16    9
```

**Example: mpls -help**

```
# mpls --help
```

```
Usage: mpls --dump
```

```
mpls --get <label>
```

```
mpls --help
```

```
--dump Dumps the mpls incoming label map
```

```
--get Dumps the entry corresponding to label <label>  
in the label map
```

```
--help Prints this help message
```

## mirror Command

Use the **mirror** command to dump the mirror table entries.

### Example: Inspect Mirroring

The following example inspects a mirror configuration where traffic is mirrored from network **vn1 (1.1.1.0/24)** to network **vn2 (2.2.2.0/24)**. A ping is run from 1.1.1.253 to 2.2.2.253, where both IPs are valid VM IPs, then the flow table is listed:

```
# flow -l

Flow table

Index      Source:Port  Destination:Port  Proto(V)
-----
135024     2.2.2.253:1208  1.1.1.253:0      1 (1)
           (Action:F, S(nh):17, Statistics:208/17472 Mirror Index : 0)

387324     1.1.1.253:1208  2.2.2.253:0      1 (1)
           (Action:F, S(nh):8, Statistics:208/17472 Mirror Index : 0)
```

In the example output, **Mirror Index:0** is listed, it is the index to the mirror table. The mirror table can be dumped with the **--dump** option, as follows:

```
# mirror --dump

Mirror Table

Index NextHop  Flags  References
-----
0     18        3
```

The mirror table entries point to next hops. In the example, the index 0 points to next hop 18. The **References** indicate the number of flow entries that point to this entry.

A next hop get operation on ID 18 is performed as follows:

```
# nh --get 18

Id:018 Type:Tunnel  Fmly: AF_INET  Flags:Valid, Udp,  Rid:0 Ref_cnt:2

Oif:0 Len:14 Flags Valid, Udp,  Data:00 00 00 00 00 00 00 25 90 c3 08 69 08 00

Vrf:-1 Sip:192.168.1.10 Dip:250.250.2.253

Sport:58818 Dport:8099
```

The **nh --get** output shows that mirrored packets go to a system with IP 250.250.2.253. The packets are tunneled as a UDP datagram and sent to the destination. **Vrf:-1** indicates that a lookup has to be done in the source **Vrf** for the destination.

You can also get an individual mirror table entry using the **--get** option, as follows:

```
# mirror --get 10

Mirror Table

Index  NextHop  Flags  References
-----
10     1         1
```

#### Example: mirror --help

```
# mirror --help

Usage: mirror --dump

      mirror --get <index>

      mirror --help

--dump  Dumps the mirror table

--get   Dumps the mirror entry corresponding to index <index>

--help  Prints this help message
```

## vxlan Command

The vxlan command can be used to dump the vxlan table. The vxlan table maps a network ID to a next hop, similar to an MPLS table.

If a packet comes with a vxlan header and if the VNID is one of those in the table, the vrouter will use the next hop identified to forward the packet.

#### Example: vxlan --dump

```
# vxlan --dump

VXLAN Table

VNID  NextHop
-----
4     16
5     16
```

#### Example: vxlan --get

You can use the **--get** option to dump a specific entry, as follows:

```
# vxlan --get 4

VXLAN Table
```

```

VNID  NextHop
-----

```

```

4      16

```

#### Example: vxlan --help

```

# vxlan --help

Usage: vxlan --dump

       vxlan --get <vnid>

       vxlan --help

--dump Dumps the vxlan table

--get  Dumps the entry corresponding to <vnid>

--help Prints this help message

```

## nh Command

The **nh** command enables you to inspect the next hops that are known by the router. Next hops tell the router the next location to send a packet in the path to its final destination. The processing of the packet differs based on the type of the next hop. The next hop types are described in the following table.

Next Hop Type	Description
Receive	Indicates that the packet is destined for itself and the router should perform Layer 4 protocol processing. As an example, all packets destined to the host IP will hit the receive next hop in the default VRF. Similarly, all traffic destined to the VMs hosted by the server and tunneled inside a GRE will hit the receive next hop in the default VRF first, because the outer packet that carries the traffic to the VM is that of the server.
Encap (Interface)	Used only to determine the outgoing interface and the Layer 2 information. As an example, when two VMs on the same server communicate with each other, the routes for each of them point to an encap next hop, because the only information needed is the Layer 2 information to send the packet to the tap interface of the destination VM. A packet destined to a VM hosted on one server from a VM on a different server will also hit an encap next hop, after tunnel processing.
Tunnel	Encapsulates VM traffic in a tunnel and sends it to the server that hosts the destination VM. There are different types of tunnel next hops, based on the type of tunnels used. Router supports two main tunnel types for Layer 3 traffic: MPLSoGRE and MPLSoUDP. For Layer 2 traffic, a VXLAN tunnel is used. A typical tunnel next hop indicates the kind of tunnel, the rewrite information, the outgoing interface, and the source and destination server IPs.
Discard	A catch-all next hop. If there is no route for a destination, the packet hits the discard next hop, which drops the packet.

Next Hop Type	Description
<b>Resolve</b>	Used by the agent to lazy install Layer 2 rewrite information.
<b>Composite</b>	Groups a set of next hops, called component next hops or sub next hops. Typically used when multi-destination distribution is needed, for example for multicast, ECMP, and so on.
<b>Vxlan</b>	A VXLAN tunnel is used for Layer 2 traffic. A typical tunnel next hop indicates the kind of tunnel, the rewrite information, the outgoing interface, and the source and destination server IPs.

**Example: nh --list**

```

Id:000 Type:Drop  Fmly: AF_INET  Flags:Valid,  Rid:0  Ref_cnt:1781

Id:001 Type:Resolve  Fmly: AF_INET  Flags:Valid,  Rid:0  Ref_cnt:244

Id:004 Type:Receive  Fmly: AF_INET  Flags:Valid, Policy,  Rid:0
      Ref_cnt:2 Oif:1

Id:007 Type:Encap  Fmly: AF_INET  Flags:Valid, Multicast,  Rid:0  Ref_cnt:3
      EncapFmly:0806 Oif:3 Len:14 Data:ff ff ff ff ff 00 25 90 c4 82 2c 08 00

Id:010 Type:Encap  Fmly:AF_BRIDGE  Flags:Valid, L2,  Rid:0  Ref_cnt:3
      EncapFmly:0000 Oif:3 Len:0 Data:

Id:012 Type:Vxlan Vrf  Fmly: AF_INET  Flags:Valid,  Rid:0  Ref_cnt:2
      Vrf:1

Id:013 Type:Composite  Fmly: AF_INET  Flags:Valid, Fabric,  Rid:0  Ref_cnt:3
      Sub NH(label): 19(1027)

Id:014 Type:Composite  Fmly: AF_INET  Flags:Valid, Multicast, L3,  Rid:0  Ref_cnt:3
      Sub NH(label): 13(0) 7(0)

Id:015 Type:Composite  Fmly:AF_BRIDGE  Flags:Valid, Multicast, L2,  Rid:0  Ref_cnt:3
      Sub NH(label): 13(0) 10(0)

Id:016 Type:Tunnel  Fmly: AF_INET  Flags:Valid, MPLSoGRE,  Rid:0  Ref_cnt:1
      Oif:2 Len:14 Flags Valid, MPLSoGRE,  Data:00 25 90 aa 09 a6 00 25 90 c4 82 2c 08
      00
      Vrf:0 Sip:10.204.216.72 Dip:10.204.216.21

Id:019 Type:Tunnel  Fmly: AF_INET  Flags:Valid, MPLSoUDP,  Rid:0  Ref_cnt:7
      Oif:2 Len:14 Flags Valid, MPLSoUDP,  Data:00 25 90 aa 09 a6 00 25 90 c4 82 2c 08

```



```
00
```

```
Vrf:0 Sip:10.204.216.72 Dip:10.204.216.21
```

```
Id:020 Type:Composite Fmly:AF_UNSPEC Flags:Valid, Multi Proto, Rid:0 Ref_cnt:2
```

```
Sub NH(label): 14(0) 15(0)
```

**Example: nh --get** Use the `--get` option to display information for a single next hop.

```
# nh --get 9
```

```
Id:009 Type:Encap Fmly:AF_BRIDGE Flags:Valid, L2, Rid:0 Ref_cnt:4
```

```
EncapFmly:0000 Oif:3 Len:0 Data:
```

**Example: nh --help**

```
# nh --help
```

```
Usage: nh --list
```

```
nh --get <nh_id>
```

```
nh --help
```

```
--list Lists All Nexthops
```

```
--get <nh_id> Displays nexthop corresponding to <nh_id>
```

```
--help Displays this help message
```

## Route Target Filtering

- [Introduction on page 387](#)
- [Debugging and Troubleshooting Route Target Filtering on page 388](#)
- [RTF Limitations in Contrail 1.10 on page 389](#)

### Introduction

BGP route target filtering (RTF) is a method for limiting the distribution of VPN routes to only those systems in the network for which the routes are necessary. If RTF is not active, the Contrail control node advertises all VPN routes to all of its VPN peers, which are either other control nodes or gateway routers such as an MX Series router. On the receiving side, the control node stores all VPN routes it receives from peers in the VPN table (for example, `bgp.l3vpn.0`). Any routes that do not include a route target extended community that is referenced by the local `vrf-import` policies are discarded by Junos.

The control node must send all route updates to its peers, even for unnecessary routes that are discarded. Continuous route updates are both CPU- and memory-intensive. The only routes that are necessary to advertise to gateway routers are those that belong to the virtual networks that are configured for public access. It is not necessary to advertise VM routes belonging to other virtual networks to gateway routers.

If a datacenter has more than two control nodes, the **vrouter-agent** only subscribes to two of the control nodes, indicated by the discovery service. When a VM is initially launched in a virtual network, it sends an XMPP subscribe request for the virtual network VRF and publishes the VM route to the connected control node. It is not necessary to advertise routes belonging to this type of VRF to control nodes that don't have the **vrouter-agent** subscribed in that VRF.

RTF is used to optimize the route distribution among control nodes and to the gateway routers to avoid unwanted route updates. If the BGP peer has not advertised or configured with RTF address family, then all routes belonging to the VPN table will be advertised.

RTF implementation in the control node does not support advertising and receiving of default route targets.

Constrained route distribution using route target reachability information is defined in RFC 4684, “*Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)*”.

## Debugging and Troubleshooting Route Target Filtering

Use the tips in this section to troubleshoot issues with RTF. Use various http introspect commands to reveal details about BGP neighbors for RTF. The following is a sample portion of an http introspect page.

When you access an introspect page, only the first panel of detail columns appears. Use a scroll bar or arrow keys to reveal more columns to the right, and vice versa.

BgpNeighborListResp						
neighbors						
peer	peer_address	deleted	peer_asn	local_address	local_asn	encoding
nodec13	10.204.216.70	false	0	10.204.216.70	0	XMPP
more						

- Use the following http introspect URL to display the details of each peer:

**http://(your\_node\_name):8083/Snh\_BgpNeighborReq**

For BGP peers, verify the configured and negotiated capability and the BGP table registration.

For XMPP peers, look at the **routing\_instances** column to get details about the VRF to which the displayed **vrouter-agent** has subscribed and to see the import **rtargets** of the VRFs.

- Use the following http introspect URL to dump the **bgp.rtarget.0** table to display the **RTargetRoutes**:

**http://(your\_node\_name):8083/Snh\_ShowRouteReq?x=bgp.rtarget.0**

- Use the following http introspect URL to dump the details for each of the route targets configured on the control node:

**http://(your\_node\_name):8083/Snh\_ShowRtGroupReq?**

For any given route target, this introspect displays the BGP table that imports and exports the route, the BGP peers that have shown interest in this route, and all dependent routes (when this route target has the extended community BGP attribute).

## RTF Limitations in Contrail 1.10

The following are RTF limitations in Contrail 1.10.

- The control node does not support advertising a default route target, which is an **rtarget** route with **target:0:0** or **0/0** as the prefix. This type of **rtarget** route enables a BGP peer to receive all VPN routes without **rtarget** filtering.
- The control node does not support receiving a default route target. If **rtarget** routes with a default **rtarget** prefix are received, they are silently ignored.
- A **keep all** configuration, typical for BGP peering for a control node on an MX Series router, does not have impact, because all VPN routes with an extended community route target, for which the MX has advertised the **rtarget** route, are sent to the MX. An example of this type of typical configuration is the following:

```
set protocols bgp group contrail-control-nodes type internal
set protocols bgp group contrail-control-nodes local-address 10.204.216.253
set protocols bgp group contrail-control-nodes keep all
set protocols bgp group contrail-control-nodes family inet-vpn unicast
set protocols bgp group contrail-control-nodes family route-target
set protocols bgp group contrail-control-nodes neighbor 10.204.216.16
```

## Source Network Address Translation (SNAT)

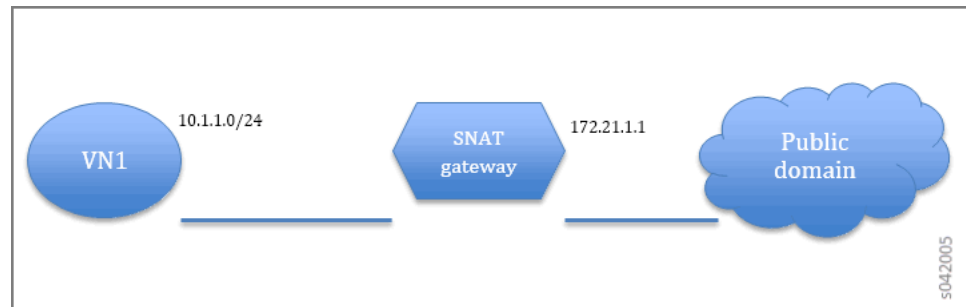
- [Overview on page 389](#)
- [Neutron APIs for Routers on page 390](#)
- [Network Namespace on page 391](#)
- [Using Web UI to Configure Routers with SNAT on page 391](#)

### Overview

Source Network Address Translation (source-nat or SNAT) allows traffic from a private network to go out to the internet. Virtual machines launched on a private network can get to the internet by going through a gateway capable of performing SNAT. The gateway has one arm on the public network and as part of SNAT, it replaces the source IP of the originating packet with its own public side IP. As part of SNAT, the source port is also updated so that multiple VMs can reach the public network through a single gateway public IP.

The following diagram shows a virtual network with the private subnet of 10.1.1.0/24. The default route for the virtual network points to the SNAT gateway. The gateway replaces the source-ip from 10.1.1.0/24 and uses its public address 172.21.1.1 for outgoing packets. To maintain unique NAT sessions the source port of the traffic also needs to be replaced.

**Figure 165: Virtual Network With a Private Subnet**



## Neutron APIs for Routers

OpenStack supports SNAT gateway implementation through its Neutron APIs for routers. The SNAT flag can be enabled or disabled on the external gateway of the router. The default is True (enabled).

The OpenContrail plugin supports the Neutron APIs for routers and creates the relevant service-template and service-instance objects in the API server. The service scheduler in OpenContrail instantiates the gateway on a randomly-selected virtual router. OpenContrail uses network namespace to support this feature.

### Example Configuration: SNAT for Contrail

The SNAT feature is enabled on OpenContrail through Neutron API calls.

The following configuration example shows how to create a test network and a public network, allowing the test network to reach the public domain through the SNAT gateway.

1. Create the public network and set the router external flag.

```

neutron net-create public
neutron subnet-create public 172.21.1.0/24
neutron net-update public -- --router:external=True
  
```

2. Create the test network.

```

neutron net-create test
neutron subnet-create --name test-subnet test 10.1.1.0/24
  
```

3. Create the router with one interface in test.

```

neutron router-create r1
  
```

```
neutron router-interface-add r1 test-subnet
```

4. Set the external gateway for the router.

```
neutron router-gateway-set r1 public
```

## Network Namespace

Setting the external gateway is the trigger for OpenContrail to set up the Linux network namespace for SNAT.

The network namespace can be cleared by issuing the following Neutron command:

```
neutron router-gateway-clear r1
```

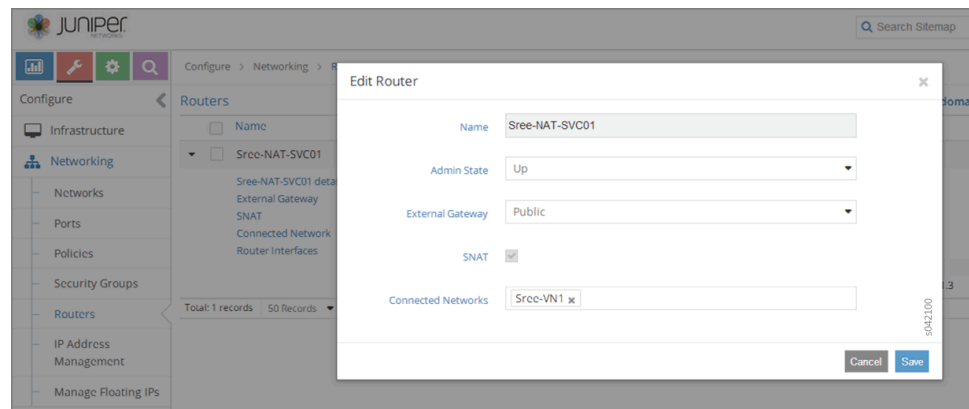
## Using Web UI to Configure Routers with SNAT

You can use the Contrail user interface to configure routers for SNAT and to check the SNAT status of routers.

To enable SNAT for a router, go to **Configure > Networking > Routers**. In the list of routers, select the router for which SNAT should be enabled. Click the Edit cog to reveal the **Edit Routers** window. Click the check box for SNAT to enable SNAT on the router.

The following shows a router for which SNAT has been **Enabled**.

**Figure 166: Edit Router Window to Enable SNAT**



When a router has been **Enabled** for SNAT, the configuration can be seen by selecting **Configure > Networking > Routers**. In the list of routers, click open the router of interest. In the list of features for that router, the status of SNAT is listed. The following shows a router that has been opened in the list. The status of the router shows that SNAT is **Enabled**.

Figure 167: Router Status for SNAT

Name	External Gateway	Connected Network	Admin State
Sree-NAT-SVC01	Public	Sree-VN1	Up

Sree-NAT-SVC01 details	SNAT	External Gateway	Connected Network	Router Interfaces	UUID	Network	IP
Enabled	Enabled	Sree-VN1			62a171f5-d323-4c70-942f-9cc02e973d53	Sree-VN1	100.1.1.3

You can view the real time status of a router with SNAT by viewing the instance console, as in the following.

Figure 168: Instance Details Window

**Instance Details: Sree-VM1**

Overview Log Console

Instance Console

If console is not responding to keyboard input, click the grey status bar below. [Click here to show only console](#)  
To exit the fullscreen mode, click the browser's back button.

```

Connected (unencrypted) to: OEMU (instance-80888953)
Send CtrlAltDel

inet addr:100.1.1.2 Bcast:100.1.1.255 Mask:255.255.255.0
inets addr: fe80::19c:9aff:fe20:64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:5611 errors:0 dropped:0 overruns:0 frame:0
TX packets:5790 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:551725 (551.7 KB) TX bytes:564998 (564.9 KB)

10
Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inets addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:1636 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

ubuntu@sree-vm1:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_req=1 ttl=44 time=19.1 ms
16
--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 19.107/19.107/19.107/0.000 ms
ubuntu@sree-vm1:~$
  
```

## Common Support Answers

- [Debugging Ping Failures for Policy-Connected Networks on page 393](#)
- [Debugging BGP Peering and Route Exchange in Contrail on page 399](#)
- [Troubleshooting the Floating IP Address Pool in Contrail on page 414](#)
- [Removing Stale Virtual Machines and Virtual Machine Interfaces on page 442](#)
- [Troubleshooting Link-Local Services in Contrail on page 446](#)

### Debugging Ping Failures for Policy-Connected Networks

---

This topic presents troubleshooting scenarios and steps for resolving reachability issues (ping failures) when working with policy-connected virtual networks.

These are the methods used to configure reachability for a virtual network or virtual machine:

- Use network policy to exchange virtual network routes.
- Use a floating IP address pool to associate an IP address from a destination virtual network to virtual machine(s) in the source virtual network.
- Use an ASN/RT configuration to exchange virtual network routes with an MX Series router gateway.
- Use a service instance static route configuration to route between service instances in two virtual networks.

This topic focuses on troubleshooting reachability for the first method --- using network policy to exchange routes between virtual networks.

#### *Troubleshooting Procedure for Policy-Connected Network*

1. Check the state of the virtual machine and interface.

Before doing anything else, check the status of the source and destination virtual machines.

- Is the **Status** of each virtual machine **Up**?
- Are the corresponding tap interfaces **Active**?

Check the virtual machine status in the Contrail UI:

Figure 169: Virtual Machine Status Window

Name	Label	Status	Network	IP Address	Floating IP	Instance
tapb8d9c6-67	16	Up	vn1 (admin)	31.1.1.253	10.204.219.108	549533ef-403e-40b0-b047-6c748e605c8 / vn1

Check the tap interface status in the http agent introspect, for example:

[http://nodef1.englab.juniper.net:8085/Snh\\_ItfReq?name=](http://nodef1.englab.juniper.net:8085/Snh_ItfReq?name=)

Figure 170: Tap Interface Status Window

Index	name	uuid	vrf_name	active
4	tapb8d9c6-67	b8d9c6-672e-4c1e-9b83-e053d6c911ab	default-domain:admin:vn1:vn1	Active

When the virtual machine status is verified **Up**, and the tap interface is **Active**, you can focus on other factors that affect traffic, including routing, network policy, security policy, and service instances with static routes.

## 2. Check reachability and routing.

Use the following troubleshooting guidelines whenever you are experiencing ping failures on virtual network routes that are connected by means of network policy.

Check the network policy configuration:

- Verify that the policy is attached to each of the virtual networks.
- Each attached policy should have either an explicit rule allowing traffic from one virtual network to the other, or an allow all traffic rule.
- Verify that the order of the actions in the policy rules is correct, because the actions are applied in the order in which they are listed.
- If there are multiple policies attached to a virtual network, verify that the policies are attached in a logical order. The first policy listed is applied first, and its rules are applied first, then the next policy is applied.
- Finally, if either of the virtual networks does not have an explicit rule to allow traffic from the other virtual network, the traffic flow will be treated as an **UNRESOLVED** or **SHORT** flow and all packets will be dropped.

Use the following sequence in the Contrail UI to check policies, attachments, and traffic rules:

Check VNI-VN2 ACL information from the compute node:



Figure 171: Policies, Attachments, and Traffic Rule Status Window

UUID	Flows	Action	Protocol	Source Network or Prefix	Source Port	Destination Network or Prefix	Destination Port	ACE Id
8b0329d7-ad9e-41ac-a72e-30f4dbc2b5ee	100000	pass	1-1	default-domain:admin:vn1	any	default-domain:admin:vn2	any	1
		pass	1-1	default-domain:admin:vn2	any	default-domain:admin:vn1	any	2
		pass	any	default-domain:admin:vn1	any	default-domain:admin:vn1	any	3
		deny	any	default-domain:admin:vn1	any	default-domain:admin:vn2	any	4
		deny	any	default-domain:admin:vn2	any	default-domain:admin:vn1	any	5

Check the virtual network policy configuration with route information:

Figure 172: Virtual Network Policy Configuration Window

Network	Attached Policies	IP Blocks
vn1	default-analyzer-analyzer-policy vn1-vn2	31.1.1.0/24
vn2	allow_all	32.1.1.0/24

Check the VN1 route information for VN2 routes:

Figure 173: Virtual Network Route Information Window

Prefix	Next hop Type	Next hop details
32.1.1.251 / 32	tunnel	Source IP: 192.168.40.11 Destination IP: 192.168.40.11 disabled Valid: true
32.1.1.252 / 32	interface	Interface: tap2e0a5c4d-1f Destination VN: default-domain:admin:vn2
32.1.1.253 / 32	tunnel	Source IP: 192.168.40.11 Destination IP: 192.168.40.11 disabled Valid: true

If a route is missing, ping fails. Flow inspection in the compute node displays **Action: D(rop)**.

Repeated dropstats commands confirms the drop by incrementing the **Flow Action Drop** counter with each iteration of dropstats.

Flow and dropstats commands issued at the compute node:

Figure 174: Flow and Dropstats Command List

```

root@nodefl1:~# flow -l | grep 32.1.1 -Al
root@nodefl1:~# flow -l | grep 32.1.1 -Al
root@nodefl1:~# flow -l | grep 32.1.1 -Al
73348          32.1.1.252:1911          31.1.1.253:0          1 (1)
                (Action:D, S(nh):0, Statistics:0/0)
--
423404          31.1.1.253:1911          32.1.1.252:0          1 (1)
                (Action:D, S(nh):8, Statistics:1/84)
root@nodefl1:~# dropstats | grep "Flow Action Drop "
Flow Action Drop          1588
root@nodefl1:~# dropstats | grep "Flow Action Drop "
Flow Action Drop          1589
root@nodefl1:~# dropstats | grep "Flow Action Drop "
Flow Action Drop          1590
root@nodefl1:~#

```

To help in debugging flows, you can use the detailed flow query from the agent introspect page for the compute node.

Fields of interest include:

- Inputs [from **flow -l** output]: **src/dest ip**, **src/dest ports**, **protocol**, and **vrf**
- Output from detailed flow query: **short\_flow**, **src\_vn**, **action\_str->action**

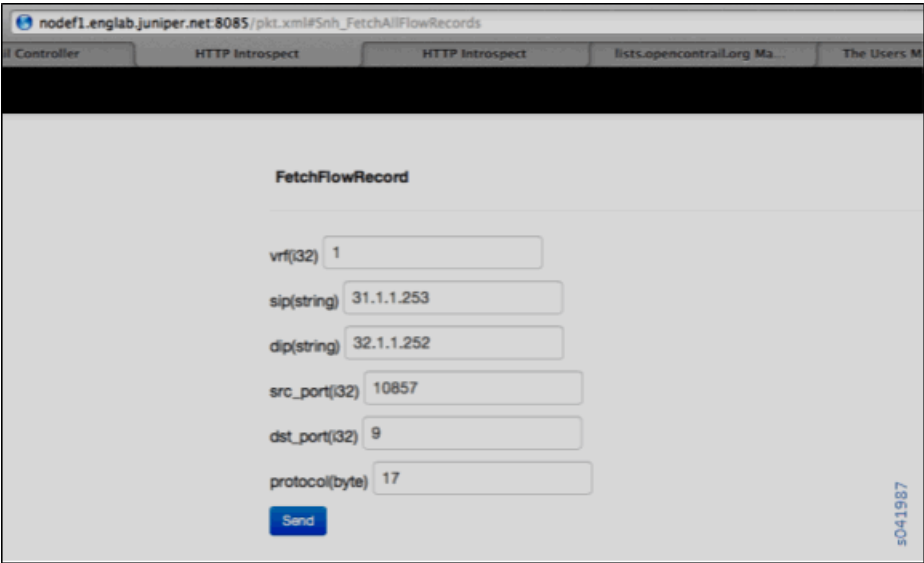
Flow command output:

Figure 175: Flow Command Output Window

Flow table			
Index	Source:Port	Destination:Port	Proto(V)
0	31.1.1.253:18572 (Action:D, S(nh):8, Statistics:4447/204562 Mirror Index : 0)	32.1.1.252:9	17 (1)
1	32.1.1.252:9 (Action:D, S(nh):15, Statistics:0/0 Mirror Index : 0)	31.1.1.253:34318	17 (1)
4	31.1.1.253:4391 (Action:D, S(nh):8, Statistics:4447/204562 Mirror Index : 0)	32.1.1.252:9	17 (1)
5	31.1.1.253:34163 (Action:D, S(nh):8, Statistics:4446/204516 Mirror Index : 0)	32.1.1.252:9	17 (1)

Fetching details of a single flow:

Figure 176: Fetch Flow Record Window



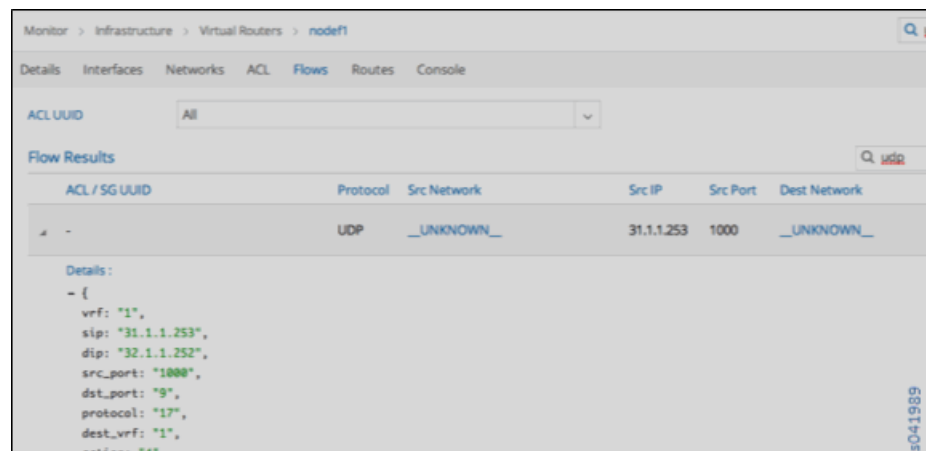
Output from **FetchFlowRecord** shows unresolved IP addresses:

Figure 177: Unresolved IP Address Window



You can also retrieve information about unresolved flows from the Contrail UI, as shown in the following:

Figure 178: Unresolved Flow Details Window

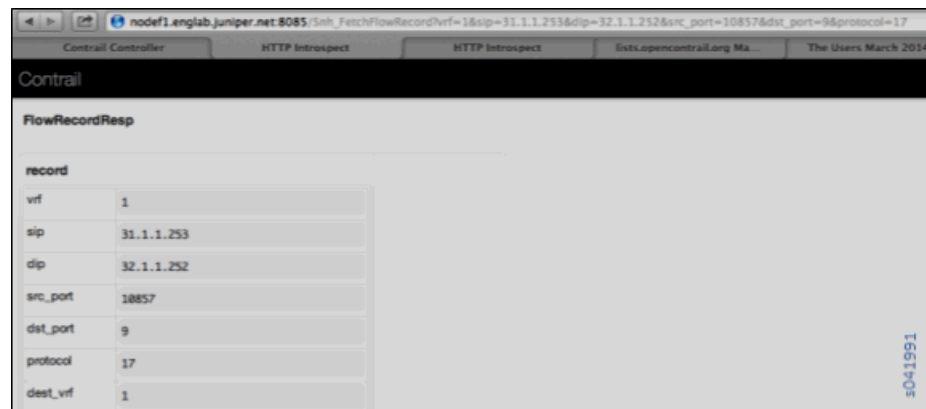


3. Check for protocol-specific network policy action.

If you are still experiencing reachability issues, troubleshoot any protocol-specific action, where routes are exchanged, but only specific protocols are allowed.

The following shows a sample query on a protocol-specific flow in the agent introspect:

Figure 179: Protocol-Specific Flow Sample



The following shows that although the virtual networks are resolved (not \_\_UNKNOWN\_\_), and not a short flow (the flow entry exists for a defined aging time), the policy action clearly displays **deny** as the action.

Figure 180: Protocol-Specific Flow Sample With Deny Action

implicit_deny	no																
short_flow	no																
setup_time_utc	1394972834710415																
local_flow	no																
src_vn	default-domain:admin:vn1																
dst_vn	default-domain:admin:vn2																
reverse_flow	no																
policy	<table> <tr> <td>policy</td><td></td></tr> <tr> <td>action</td><td>g</td></tr> <tr> <td>acl</td><td> <table> <tr> <td>acl</td><td></td></tr> <tr> <td>uuid</td><td>8b0329d7-ad9e-41ac-af2e-30f4dbc2b50e</td></tr> </table> </td></tr> <tr> <td>action_str</td><td> <table> <tr> <td>action_str</td><td></td></tr> <tr> <td>action</td><td>deny</td></tr> </table> </td></tr> </table>	policy		action	g	acl	<table> <tr> <td>acl</td><td></td></tr> <tr> <td>uuid</td><td>8b0329d7-ad9e-41ac-af2e-30f4dbc2b50e</td></tr> </table>	acl		uuid	8b0329d7-ad9e-41ac-af2e-30f4dbc2b50e	action_str	<table> <tr> <td>action_str</td><td></td></tr> <tr> <td>action</td><td>deny</td></tr> </table>	action_str		action	deny
policy																	
action	g																
acl	<table> <tr> <td>acl</td><td></td></tr> <tr> <td>uuid</td><td>8b0329d7-ad9e-41ac-af2e-30f4dbc2b50e</td></tr> </table>	acl		uuid	8b0329d7-ad9e-41ac-af2e-30f4dbc2b50e												
acl																	
uuid	8b0329d7-ad9e-41ac-af2e-30f4dbc2b50e																
action_str	<table> <tr> <td>action_str</td><td></td></tr> <tr> <td>action</td><td>deny</td></tr> </table>	action_str		action	deny												
action_str																	
action	deny																

### Summary

This topic explores one area —debugging for policy-based routing. However, in a complex system, a virtual network might have one or more configuration methods combined that influence reachability and routing.

For example, an environment might have a virtual network VN-X configured with policy-based routing to another virtual network VN-Y. At the same time, there are a few virtual machines in VN-X that have a floating IP to another virtual network VN-Z, which is connected to VN-XX via a NAT service instance. This is a complex scenario, and you need to debug step-by-step, taking into account all of the features working together.

Additionally, there are other considerations beyond routing and reachability that can affect traffic flow. For example, the rules of network policies and security groups can affect traffic to the destination. Also, if multi-path is involved, then ECMP and RPF need to be taken into account while debugging.

## Debugging BGP Peering and Route Exchange in Contrail

Use the troubleshooting steps and guidelines in this topic when you have errors with Contrail BGP peering and route exchange.

- [Example Cluster on page 400](#)
- [Verifying the BGP Routers on page 400](#)
- [Verifying the Route Exchange on page 403](#)
- [Debugging Route Exchange with Policies on page 405](#)
- [Debugging Peering with an MX Series Router on page 407](#)
- [Debugging a BGP Peer Down Error with Incorrect Family on page 408](#)

- [Configuring MX Peering \(iBGP\) on page 410](#)
- [Checking Route Exchange with an MX Series Peer on page 411](#)
- [Checking the Route in the MX Series Router on page 413](#)

## Example Cluster

Examples in this document refer to a virtual cluster that is set up as follows:

Config Nodes : ['nodea22', 'nodea20']

Control Nodes : ['nodea22', 'nodea20']

Compute Nodes : ['nodea22', 'nodea20']

Collector : ['nodea22']

WebU : nodea22

Openstack : nodea22

## Verifying the BGP Routers

Use this procedure to launch various introspects to verify the setup of the BGP routers in your system.

Use this procedure to launch various introspects to verify the setup of the BGP routers in your system.

1. Verify the BGP routers.

All of the configured control nodes and external BGP routers are visible from the following location, shown using the sample node setup.

<http://nodea22.testlab.juniper.net:8082/bgp-routers>



**NOTE:** Throughout this procedure, replace `nodea22.testlab.juniper.net` with the correct location for your system to see the setup in your system.

Figure 181: Sample Output, BGP Routers:

```
{
  - bgp-routers: [
    - {
      href: "http://nodea22.testlab.juniper.net:8082/bgp-router/1da579c5-0907-4c98-a7ad-37671f00cf60",
      - fq_name: [
        "default-domain",
        "default-project",
        "ip-fabric",
        "__default__",
        "nodea20"
      ],
      uuid: "1da579c5-0907-4c98-a7ad-37671f00cf60"
    },
    - {
      href: "http://nodea22.testlab.juniper.net:8082/bgp-router/9702853f-5e48-417f-bd72-c00a12cc0200",
      - fq_name: [
        "default-domain",
        "default-project",
        "ip-fabric",
        "__default__",
        "nodea22"
      ],
      uuid: "9702853f-5e48-417f-bd72-c00a12cc0200"
    }
  ]
}
```

5041945

2. Verify the BGP peering.

The following statement is entered to check the `bgp_router_refs` object on the API server to validate the peering on the sample setup.

<http://<ip address>:8082/bgp-router/1da579c5-0907-4c98-a7ad-37671f00cf60>

Figure 182: Sample Output, BGP Router References:

```

- bgp_router_parameters: {
  vendor: "contrail",
  autonomous_system: 64512,
  vnc_managed: null,
  address: "10.204.216.16",
  identifier: "10.204.216.16",
  port: 179,
  - address_families: {
    - family: {
      "inet-vpn",
      "e-vpn"
    }
  }
},
- bgp_router_refs: {
  - {
    - to: {
      "default-domain",
      "default-project",
      "ip-fabric",
      "__default__",
      "nodea22"
    },
    href: "http://nodea22.englab.juniper.net:8082/bgp-router/9702853f-5e48-417f-bd72-c00a12cc0200",
    - attr: {
      - session: {
        - attributes: {
          - {
            bgp_router: null,
            - address_families: {
              - family: {
                "inet-vpn",
                "e-vpn"
              }
            }
          },
          uid: null
        }
      },
      uid: "9702853f-5e48-417f-bd72-c00a12cc0200"
    }
  }
},
}

```

3. Verify the command line arguments that are passed to the control-node.

On the control-node, use **ps aux | grep control-node** to see the arguments that are passed to the control-node.

#### Example

```

/usr/bin/control-node --map-user <ip address> --map-password <ip address>
--hostname nodea22 --host-ip <ip address> --bgp-port 179 --discovery-server <ip
address>

```

The hostname is the **bgp-router** name. Ensure that the bgp-router config can be found for the hostname, using the procedure in Step 1.

4. Validate the BGP neighbor config and the BGP peering config object.

**http://<ip address>:8083/Snh\_ShowBgpNeighborConfigReq?**

Figure 183: Sample Output, BGP Neighbor Config:

neighbors						
instance_name	name	vendor	autonomous_system	identifier	address	address_families
default-domain:default-project:ip-fabric:__default__	default-domain:default-project:ip-fabric:__default__:nodea22	contrail	64512	10.204.216.16	10.204.216.16	address_families
						inet-vpn
						e-vpn

**http://<ip address>:8083/Snh\_ShowBgpPeeringConfigReq?**



Figure 184: Sample Output, BGP Peering Config:

ShowBgpPeeringConfigResp

peerings		neighbor_count	sessions
instance_name	name		
default-domain:default-project:ip-fabric:default...	addr(default-domain:default-project:ip-fabric:default...mode20,default-domain:default-project:ip-fabric:default...mode21)	1	<div>sessions</div> <div>valid attributes</div> <div>attributes</div> <div>bgp.router.address.families</div> <div>address_families</div> <div>inet-vpn</div> <div>e-vpn</div>

5. Check the BGP neighbor states on the sample setup.

http://<ip address>:8083/Snh\_BgpNeighborReq?ip\_address=&domain=

Figure 185: Sample Output, BGP Neighbor States:

BgpNeighborListResp

neighbors											
peer	peer_address	peer_asn	local_address	local_asn	encoding	peer_type	state	send_state	last_event	last_state	last_state_at
10.204.216.16	10.204.216.16	64512	10.204.216.18	64512	BGP	Internal	Established	In sync	fan::EvBgpKeepalive	OpenConfirm	2014-Feb-10 07:08:21.177632
10.204.216.253	10.204.216.253	64512	10.204.216.18	64512	BGP	Internal	Established	In sync	fan::EvBgpUpdate	OpenConfirm	2014-Feb-10 11:24:45.851632

If the peer is not in an established state, check the **last\_error** and the **flap\_count**. Debug the BGP state machine by using information displayed in the output, such as **last\_state** and **last\_event**.



**NOTE:** The image displayed is truncated to fit this page. On the console screen you can scroll horizontally to see more columns and data.

Verifying the Route Exchange

The following two virtual networks are used in the sample debugging session for route exchange.

vn1 -> 1.1.1.0/24

vn2 -> 2.2.2.0/24

Example Procedure for Verifying Route Exchange

- 1. Validate the presence of the routing instance for each virtual network in the sample system.

http://<ip address>:8083/Snh\_ShowRoutingInstanceReq?name=



**NOTE:** Throughout this example, replace <ip address> with the correct location for the control node on your system.

Figure 186: Sample Output, Show Routing Instance:

default-domain:demo:vn1:vn1	default-domain:demo:vn1	4	import_target target:64512:1	export_target target:64512:1	values	name	peers
						default-domain:demo:vn1:vn1.inet.0	peers nodes28
						default-domain:demo:vn1:vn1.inet.0	peers nodes28
						default-domain:demo:vn1:vn1.l2vpncast.0	peers nodes28
default-domain:demo:vn2:vn2	default-domain:demo:vn2	5	import_target target:64512:2	export_target target:64512:2	values	name	peers
						default-domain:demo:vn2:vn2.inet.0	peers nodes22
						default-domain:demo:vn2:vn2.inet.0	peers nodes22
						default-domain:demo:vn2:vn2.l2vpncast.0	peers nodes22

In the sample output, you can see the **import\_target** and the **export\_target** configured on the routing instance. Also shown are the **xmpp peers (vroutes)** registered to the table.

The user can click on the **inet** table of the required routing instance to display the routes that belong to the instance.

Use the information in Step 2 to validate a route.

2. Validate a route in a given routing instance in the sample setup:

**[http://<ip address>:8083/Snh\\_ShowRouteReq?x=default-domain:demo:vn1:vn1.inet.0](http://<ip address>:8083/Snh_ShowRouteReq?x=default-domain:demo:vn1:vn1.inet.0)**

In the following sample output (truncated), the user can validate the BGP paths for the protocol and for the source of the route to verify which XMPP agent or vRouter has pushed the route. If the path source is BGP, the route is imported to the VRF table from a BGP peer, either another control-node or an external bgp router such as an MX Series router. BGP paths are displayed in the order of path selection.

Figure 187: Sample Output, Validate Route:

ShowRouteResp									
tables									
routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	infeasible_paths	routes		
default-domain:demo:vn1:vn1	default-domain:demo:vn1:vn1.inet.0	1	2	1	1	0	prefix	last_modified	paths
							1.1.1.253/32	2014-Feb-10 11:34:12.227288	paths
							protocol		
							XMPP		
							BGP		

3. Validate the **l3vpn** table.

**[http://<ip address>:8083/Snh\\_ShowRouteReq?x=bgp.l3vpn.0](http://<ip address>:8083/Snh_ShowRouteReq?x=bgp.l3vpn.0)**

The extended community (communities column), determines the VRF table to which this VPN route is imported. The **origin\_vn** shows the virtual network where this route was created, information useful for applying ACL

Figure 190: Create Policy Window

2. Validate that the routing instances have the correct import\_target configuration.

[http://<ip address>:8083/Snh\\_ShowRoutingInstanceReq?name=](http://<ip address>:8083/Snh_ShowRoutingInstanceReq?name=)

Figure 191: Sample Output, Validate Import Target:

default-domain:demo:vn1:vn1	default-domain:demo:vn1	4	import_target target:64512:1 target:64512:2	export_target target:64512:1
default-domain:demo:vn2:vn2	default-domain:demo:vn2	5	import_target target:64512:1 target:64512:2	export_target target:64512:2

3. Validate that the routes are imported from VRF.

Use the BGP path attribute to check the replication status of the path. The route from the destination VRF should be replicated and validate the origin-vn.

Figure 192: Sample Output, Route Import:

routing_instance	routing_table_name	prefix	paths	primary_paths	secondary_paths	feasible_paths	routes	last_modified	path	protocol
default-domain:demo:vn2:vn2	default-domain:demo:vn2:vn2:inet.0	2	4	1	3	8	prefix 1.1.1.253/32	2014-Feb-18 12:02:47.261344	path	XMP
							2.2.2.253/32	2014-Feb-18 11:34:35.469899	path	XMP

## Debugging Peering with an MX Series Router

This section sets up an example BGP MX Series peer and provides some troubleshooting scenarios.

1. Set the Global AS number of the control-node for an MX Series BGP peer, using the Contrail WebUI (eBGP).

Figure 193: Edit Global ASN Window

2. Configure the eBGP peer for the MX Series router. Use the Contrail Web UI or Python provisioning.

Figure 194: Create BGP Peer Window

Configuring the MX Series BGP peer with the Python provision utility:

```
python ./provision_mx.py --router_name mx --router_ip <ip address> --router_asn
<asn> --api_server_ip <ip address> --api_server_port 8082 --oper add --admin_user
admin --admin_password <password> --admin_tenant_name admin
```

3. Configure a control-node peer on the MX Series router, using Junos CLI:

```
set protocols bgp group contrail-control-nodes type external
```

```
set protocols bgp group contrail-control-nodes local-address <ip address>
```

```
set protocols bgp group contrail-control-nodes keep all
```

```
set protocols bgp group contrail-control-nodes peer-as <as>
```

```
set protocols bgp group contrail-control-nodes local-as 12345
```

```
set protocols bgp group contrail-control-nodes neighbor <ip address>
```

## Debugging a BGP Peer Down Error with Incorrect Family

Use this procedure to identify and resolve errors that arise from *families* mismatched configurations.



**NOTE:** This example uses locations at `http://<ip address>:`. Be sure to replace `<ip address>` with the correct address for your environment.

1. Check the BGP peer UVE.

```
<ip address>:8081/analytics/uves/bgp-peers
```

2. Search for the MX Series BGP peer by name in the list.

In the sample output, **families** is the family advertised by the peer and **configured\_families** is what is provisioned. In the sample output, the families configured on the peer has a mismatch, thus the peer doesn't move to an established state. You can verify it in the peer UVE.

Figure 195: Sample BGP Peer UVE

```
{
  - BgpPeerInfoData: {
    - state_info: {
      last_state: "Idle",
      state: "Idle",
      last_state_at: 1394778927107639
    },
    - families: [
      "IPv4:Unicast"
    ],
    peer_type: "external",
    local_asn: 54321,
    - configured_families: [
      "inet-vpn"
    ],
    - event_info: {
      last_event_at: 1394778927107880,
      last_event: "fsm::EvStart"
    },
    local_id: 181196816,
    send_state: "not advertising",
    peer_address: "10.204.216.253",
    peer_id: 181197053,
    hold_time: 90,
    peer_asn: 12345
  }
}
```

3. Fix the **families** mismatch in the sample by updating the configuration on the MX Series router, using Junos CLI:

```
set protocols bgp group contrail-control-nodes family inet-vpn unicast
```

4. After committing the CLI configuration, the peer comes up. Verify this with UVE.

```
http://<ip address>:8081/analytics/uves/bgp-peers
```

Figure 196: Sample Established BGP Peer UVE

```

{
  - BgpPeerInfoData: {
    - state_info: {
      last_state: "OpenConfirm",
      state: "Established",
      last_state_at: 1394779652932460
    },
    - families: [
      "IPv4:Vpn"
    ],
    peer_type: "external",
    local_asn: 54321,
    - configured_families: [
      "inet-vpn"
    ],
    - event_info: {
      last_event_at: 1394779652992071,
      last_event: "fsm::EvBgpUpdate"
    },
    local_id: 181196816,
    send_state: "in sync",
    peer_address: "10.204.216.253",
    peer_id: 181197053,
    peer_asn: 12345
  }
}

```

5. Verify the peer status on the MX Series router, using Junos CLI:

```

run show bgp neighbor 10.204.216.16
Peer: <ip address>+46924 AS <as> Local: <ip address>3+179 AS 12345

Type: External    State: Established    Flags: <ImportEval Sync>

Last State: OpenConfirm    Last Event: RecvKeepAlive

Last Error: None

Options: <Preference LocalAddress KeepAll AddressFamily PeerAS LocalAS
Rib-group Refresh>

Address families configured: inet-vpn-unicast

Local Address: <ip address> Holdtime: 90 Preference: 170 Local AS: 12345
Local System AS: <AS>

Number of flaps: 0

Error: 'Cease' Sent: 0 Recv: 2

Peer ID: <ip address>    Local ID: <ip address>    Active Holdtime: 90

Keepalive Interval: 30          Group index: 1    Peer index: 0

BFD: disabled, down

Local Interface: ge-1/0/2.0

NLRI for restart configured on peer: inet-vpn-unicast

NLRI advertised by peer: inet-vpn-unicast

NLRI for this session: inet-vpn-unicast

```

Peer does not support Refresh capability

Stale routes from peer are kept for: 300

Peer does not support Restarter functionality

Peer does not support Receiver functionality

Peer does not support 4 byte AS extension

Peer does not support Addpath

## Configuring MX Peering (iBGP)

1. Edit the Global ASN.

Figure 197: Edit Global ASN Window

2. Configure the MX Series iBGP peer, using Contrail WebUI or Python provisioning.

Figure 198: Create BGP Peer Window

Configuring the MX Series BGP peer with the Python provision utility:

```
python ./provision_mx.py --router_name mx--router_ip <ip address> --router_asn <ASN>
--api_server_ip <ip address> --api_server_port 8082 --oper add --admin_user admin
--admin_password <password> --admin_tenant_name admin
```

3. Verify the peer from UVE.



<http://<ip address>:8081/analytics/uves/bgp-peers>

Figure 199: Sample Established IBGP Peer UVE

```
{
  - BgpPeerInfoData: {
    - state_info: {
      last_state: "OpenConfirm",
      state: "Established",
      last_state_at: 1394788178225128
    },
    - families: {
      "IPv4:Vpn"
    },
    peer_type: "internal",
    local_asn: 64512,
    - configured_families: {
      "inet-vpn"
    },
    - event_info: {
      last_event_at: 1394788178267208,
      last_event: "fsm::EvBgpUpdate"
    },
    local_id: 181196816,
    send_state: "in_sync",
    peer_address: "10.204.216.253",
    peer_id: 181197053,
    peer_asn: 64512
  }
}
```

- 4. You can verify the same information at the HTTP introspect page of the control node (8443 in this example).

[http://<ip address>:8083/Snh\\_BgpNeighborReq?ip\\_address=&domain=](http://<ip address>:8083/Snh_BgpNeighborReq?ip_address=&domain=)

Figure 200: Sample Established IBGP Peer Introspect Window

neighbors											
peer	peer_address	peer_asn	local_address	local_asn	encoding	peer_type	state	send_state	last_event	last_state	
10.204.216.253	10.204.216.253	64512	10.204.216.16	64512	BGP	internal	Established	in_sync	fsm::EvBgpKeepalive	OpenConfirm	

Checking Route Exchange with an MX Series Peer

- 1. Check the route table in the bgp.l3vpn.0 table.

Figure 201: Routing Instance Route Table

routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	infeasible_paths	routes
default-domain:default-project:ip-fabric...default...	bgp.l3vpn.0	2	2	2	0	0	routes
							prefix
							10.204.216.253:5:0.0.0.0/0
							10.204.216.253:5:10.204.218.0/24

- 2. Configure a public virtual network.

### Figure 202: Routing Instance Route Table

routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	infeasible_paths	routes
default-domain:default-project:ip-fabric::default...	bgp-13vpn-0	2	2	2	0	0	<div> <div>routes</div> <div>prefix</div> <div>10.204.216.253:5:0.0.0.0/0</div> <div>10.204.216.253:5:10.204.218.0/24</div> </div>

3. Verify the routes in the public.inet.0 table.

http: //<ip

```
address>:8083/Snh_ShowRouteReq?x=default-domain:admin:public:public.inet.0
```

### Figure 203: Routing Instance Public IPv4 Route Table

tables							
routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	infeasible_paths	routes
default-domain:admin:public:public	default-domain:admin:public:public:.inet.0	2	2	0	2	0	<div>routes</div> <div>prefix</div> <div>0.0.0.0/0</div> <div>10.204.218.0/24</div>

4. Launch a virtual machine (11.2.3.253 in the sample case) in the public network and verify the route in the public.inet.0 table.

http://<ip address>:8083/

Snh\_ShowRouteReq?x=default-domain:admin:public:public.inet.0

Figure 204: Virtual Machine Routing Instance Public IPv4 Route Table

table							routes		
routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	infeasible_paths	prefix	last_modified	paths
default-domain:admin:public:public	default-domain:admin:public:public:inet_8	8	8	1	2	0	0.0.0.0/9	2014-Mar-14 18:05:05.719926	paths protocol BGP
							10.204.718.0/24	2014-Mar-14 18:05:05.719817	paths protocol BGP
							11.2.3.253/32	2014-Mar-14 18:18:48.797958	paths protocol BGP

5. Verify the route in the bgp.l3vpn.0 table.

http://<ip address>:8083/Snh\_ShowRouteReq?x=bgp.l3vpn.0

Figure 205: BGP Routing Instance Route Table

routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	infeasible_paths	routes
default-domain:default-project:ip-fabric:default...	bgp.13vpn.0	3	3	2	1	0	<pre> prefix 10.204.216.253:5:0.0.0.0/0  10.204.216.253:5:10.204.216.0/24  10.204.216.70:1:11.2.3.253/32 </pre>

## Checking the Route in the MX Series Router

Use Junos CLI show commands from the router to check the route.

```
run show route table public.inet.0
```

```
public.inet.0: 5 destinations, 6 routes (5 active, 0 holddown, 0 hidden)
```

```
+ = Active Route, - = Last Active, * = Both
```

```
0.0.0.0/0    *[Static/5] 15w6d 08:50:34
```

```
> to <ip address> via ge-1/0/1.0
```

```
<ip address>/24  *[Direct/0] 15w6d 08:50:35
```

```
> via ge-1/0/1.0
```

```
<ip address>/32  *[Local/0] 15w6d 08:50:51
```

```
Local via ge-1/0/1.0
```

```
<ip address>/32  *[BGP/170] 01:13:34, localpref 100, from <ip address>
```

```
AS path: ?, validation-state: unverified
```

```
> via gr-1/0/0.32771, Push 16
```

```
[BGP/170] 01:13:34, localpref 100, from <ip address>
```

```
AS path: ?, validation-state: unverified
```

```
> via gr-1/0/0.32771, Push 16
```

```
<ip address>/32  *[BGP/170] 00:03:20, localpref 100, from <ip address>
```

```
AS path: ?, validation-state: unverified
```

```
> via gr-1/0/0.32769, Push 16
```

```
run show route table bgp.l3vpn.0 receive-protocol bgp <ip address> detail

bgp.l3vpn.0: 92 destinations, 130 routes (92 active, 0 holddown, 0 hidden)

* 10.xxx.xxx.70:1:11.2.3.253/32 (1 entry, 0 announced)

Import Accepted

Route Distinguisher: 10.xxx.xxx.70:1

VPN Label: 16

Nexthop: 10.xxx.xxx.70

Localpref: 100

AS path: ?

Communities: target:64512:1 target:64512:10003 unknown iana 30c unknown iana
30c unknown type 8004 value fc00:1 unknown type 8071 value fc00:4
```

---

## Troubleshooting the Floating IP Address Pool in Contrail

This document provides troubleshooting methods to use when you have errors with the floating IP address pool when using Contrail.

- [Example Cluster on page 415](#)
- [Example on page 415](#)
- [Example: MX80 Configuration for the Gateway on page 416](#)
- [Ping the Floating IP from the Public Network on page 419](#)
- [Troubleshooting Details on page 419](#)
- [Get the UUID of the Virtual Network on page 419](#)
- [View the Floating IP Object in the API Server on page 420](#)
- [View floating-ips in floating-ip-pools in the API Server on page 423](#)
- [Check Floating IP Objects in the Virtual Machine Interface on page 426](#)
- [View Floating IP Objects in the IFMAP Server View on page 429](#)
- [View the BGP Peer Status on the Control Node on page 433](#)
- [Querying Routes in the Public Virtual Network on page 434](#)
- [Verification from the MX80 Gateway on page 435](#)
- [Viewing the Compute Node Vnsw Agent on page 437](#)
- [Advanced Troubleshooting on page 440](#)

Example Cluster

Examples in this document refer to a virtual cluster that is set up as follows:

- Config Nodes : ['nodec6', 'nodec7', 'nodec8']
- Control Nodes : ['nodec7', 'nodec8']
- Compute Nodes : ['nodec9', 'nodec10']
- Collector : ['nodec6', 'nodec8']
- WebUI : nodec7
- Openstack : nodec6

The following virtual networks are used in the examples in this document:

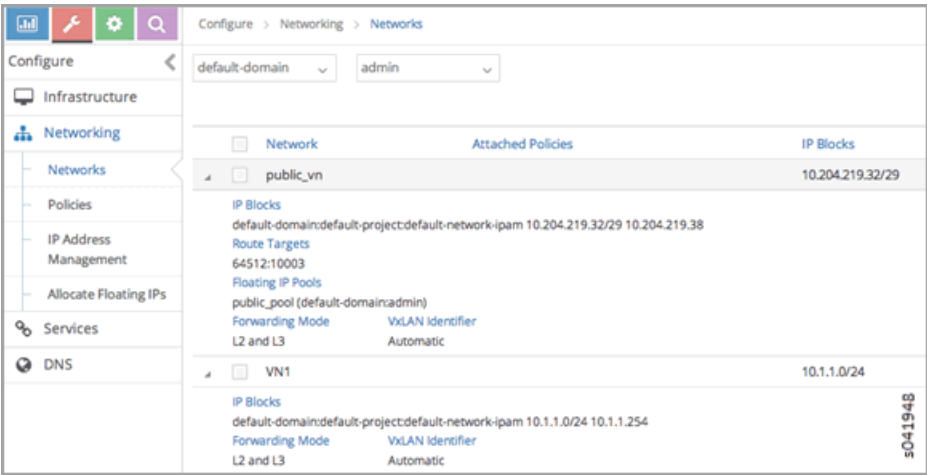
Public virtual network:

- Virtual network name: **public\_vn**
- Public addresses range: **10.204.219.32 to 10.204.219.37**
- Route Target: **64512:10003**
- Floating IP pool name: **public\_pool**

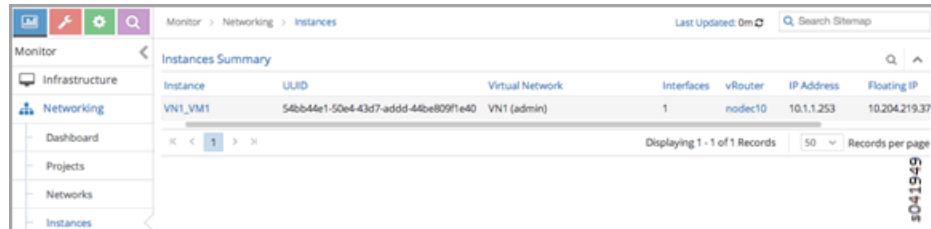
Private virtual network:

- Virtual network name: **vn1**
- Subnet: **10.1.1.0/24**

Example



A virtual machine is created in the virtual network VN1 with the name VN1\_VM1 and with the IP address 10.1.1.253. A floating IP address of 10.204.219.37 is associated to the VN1\_VM1 instance.



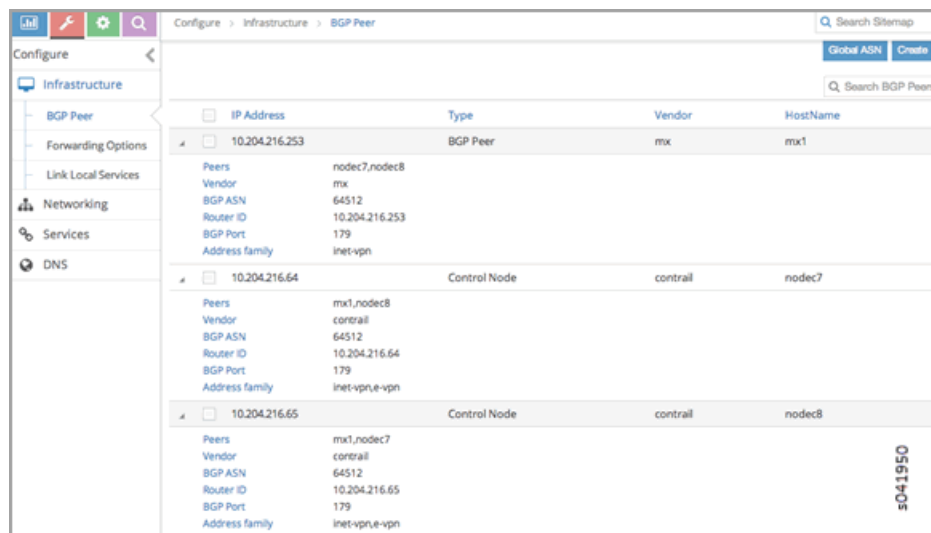
Monitor > Networking > Instances

Instances Summary

Instance	UUID	Virtual Network	Interfaces	vRouter	IP Address	Floating IP
VN1_VM1	54bb44e1-50e4-43d7-addr-44be809f1e40	VN1 (admin)	1	nodec10	10.1.1.253	10.204.219.37

Displaying 1 - 1 of 1 Records | 50 Records per page

An MX80 router is configured as a gateway to peer with control nodes nodec7 and nodec8.



Configure > Infrastructure > BGP Peer

IP Address	Type	Vendor	HostName
10.204.216.253	BGP Peer	mx	mx1
<b>Peers</b> nodec7,nodec8 <b>Vendor</b> mx <b>BGP ASN</b> 64512 <b>Router ID</b> 10.204.216.253 <b>BGP Port</b> 179 <b>Address family</b> inet-vpn			
10.204.216.64	Control Node	contrail	nodec7
<b>Peers</b> mx1,nodec8 <b>Vendor</b> contrail <b>BGP ASN</b> 64512 <b>Router ID</b> 10.204.216.64 <b>BGP Port</b> 179 <b>Address family</b> inet-vprive-vpn			
10.204.216.65	Control Node	contrail	nodec8
<b>Peers</b> mx1,nodec7 <b>Vendor</b> contrail <b>BGP ASN</b> 64512 <b>Router ID</b> 10.204.216.65 <b>BGP Port</b> 179 <b>Address family</b> inet-vprive-vpn			

### Example: MX80 Configuration for the Gateway

The following is the Junos OS configuration for the MX80 gateway. The route 10.204.218.254 is the route to the external world.

```
chassis {

  fpc 1 {

    pic 0 {

      tunnel-services;

    }

  }

}

interfaces {

  ge-1/0/1 {
```

```
unit 0 {  
    family inet {  
        address 10.204.218.1/24;  
    }  
}  
  
ge-1/0/2 {  
    unit 0 {  
        family inet {  
            address 10.204.216.253/24;  
        }  
    }  
}  
  
routing-options {  
    static {  
        route 0.0.0.0/0 next-hop 10.204.216.254;  
    }  
    router-id 10.204.216.253;  
    route-distinguisher-id 10.204.216.253;  
    autonomous-system 64512;  
    dynamic-tunnels {  
        tun1 {  
            source-address 10.204.216.253;  
            gre;  
            destination-networks {  
                10.204.216.0/24;  
                10.204.217.0/24;  
            }  
        }  
    }  
}
```

```
    }
  }
}
}
protocols {
  bgp {
    group control-nodes {
      type internal;
      local-address 10.204.216.253;
      keep all;
      family inet-vpn {
        unicast;
      }
      neighbor 10.204.216.64;
      neighbor 10.204.216.65;
    }
  }
}
routing-instances {
  public {
    instance-type vrf;
    interface ge-1/0/1.0;
    vrf-target target:64512:10003;
    vrf-table-label;
    routing-options {
      static {
        route 0.0.0.0/0 next-hop 10.204.218.254;
      }
    }
  }
}
```



```

    }
  }
}

```

## Ping the Floating IP from the Public Network

From the public network, ping the floating IP 10.204.219.37.

```

user1-test:~ user1$ ping 10.204.219.37

PING 10.204.219.37 (10.204.219.37): 56 data bytes

64 bytes from 10.204.219.37: icmp_seq=0 ttl=54 time=62.439 ms
64 bytes from 10.204.219.37: icmp_seq=1 ttl=54 time=56.018 ms
64 bytes from 10.204.219.37: icmp_seq=2 ttl=54 time=55.915 ms
64 bytes from 10.204.219.37: icmp_seq=3 ttl=54 time=57.755 ms

^C

--- 10.204.219.37 ping statistics ---

5 packets transmitted, 4 packets received, 20.0% packet loss

round-trip min/avg/max/stddev = 55.915/58.032/62.439/2.647 ms

```

## Troubleshooting Details

The following sections show details of ways to get related information, view, troubleshoot, and validate floating IP addresses in a Contrail system.

## Get the UUID of the Virtual Network

Use the following to get the universal unique identifier (UUID) of the virtual network.

```

[root@nodec6 ~]# (source /etc/contrail/openstackrc; quantum net-list -F id -F name)
2>/dev/null

+-----+-----+
| id          | name          |
+-----+-----+
| 43707766-75f3-4d48-80d9-1b7240fb161d | public_vn      |
| 2ab7ea04-8f5f-4b8d-acbf-a7c29c9b4112 | VN1            |
| 1c59ded0-38e8-4168-b91f-4c51aba10d30 | default-virtual-network |
| 5b0a1040-91e4-47ff-bd4c-0a81e1901a1f | ip-fabric      |

```

```
| 7efddf64-ff3c-44d2-aeb2-45d7472b7a64 | __link_local__ |  
+-----+-----+
```

## View the Floating IP Object in the API Server

Use the following to view the floating IP pool information in the API server. API server requests can be made on http port 8082.

The Contrail API servers have the virtual-network public\_vn object that contains floating IP pool information. Use the following to view the floating-ip-pools object information.

**curl http://<API-Server\_IP>:8082/virtual-network/<UUID\_of\_VN>**

*Example*

```
root@nodec6 ~]# curl  
http://nodec6:8082/virtual-network/43707766-75f3-4d48-80d9-1b7240fb161d | python  
-m json.tool
```

```
{  
  "virtual-network": {  
    "floating_ip_pools": [  
      {  
        "href":  
        "http://127.0.0.1:8095/floating-ip-pool/663737c1-f3ab-40ff-9442-bdb6c225e3c3",  
        "to": [  
          "default-domain",  
          "admin",  
          "public_vn",  
          "public_pool"  
        ],  
        "uuid": "663737c1-f3ab-40ff-9442-bdb6c225e3c3"  
      }  
    ],  
    "fq_name": [  
      "default-domain",  
      "admin",
```

```
"public_vn"

],

"href":
"http://127.0.0.1:8095/virtual-network/43707766-75f3-4d48-80d9-1b7240fb161d",

"id_perms": {

    "created": "2014-02-07T08:58:40.892803",

    "description": null,

    "enable": true,

    "last_modified": "2014-02-07T10:06:42.234423",

    "permissions": {

        "group": "admin",

        "group_access": 7,

        "other_access": 7,

        "owner": "admin",

        "owner_access": 7

    },

    "uuid": {

        "uuid_lslong": 9284482284331406877,

        "uuid_mslong": 4859515279882014024

    }

},

"name": "public_vn",

"network_ipam_refs": [

    {

        "attr": {

            "ipam_subnets": [

                {

                    "default_gateway": "10.204.219.38",

                    "subnet": {
```

```
        "ip_prefix": "10.204.219.32",
        "ip_prefix_len": 29
      }
    }
  ]
},
  "href":
"http://127.0.0.1:8095/network-ipam/39b0e8da-fcd4-4b35-856c-8d18570b1483",
  "to": [
    "default-domain",
    "default-project",
    "default-network-ipam"
  ],
  "uuid": "39b0e8da-fcd4-4b35-856c-8d18570b1483"
}
],
  "parent_href":
"http://127.0.0.1:8095/project/deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f",
  "parent_type": "project",
  "parent_uuid": "deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f",
  "route_target_list": {
    "route_target": [
      "target:64512:10003"
    ]
  },
  "routing_instances": [
    {
      "href":
"http://127.0.0.1:8095/routing-instance/3c6254ac-cfde-417e-916d-e7a1c0efad92",
```

```

      "to": [
        "default-domain",
        "admin",
        "public_vn",
        "public_vn"
      ],
      "uuid": "3c6254ac-cfde-417e-916d-e7a1c0efad92"
    }
  ],
  "uuid": "43707766-75f3-4d48-80d9-1b7240fb161d",
  "virtual_network_properties": {
    "extend_to_external_routers": null,
    "forwarding_mode": "l2_l3",
    "network_id": 4,
    "vxlan_network_identifier": null
  }
}

```

## View floating-ips in floating-ip-pools in the API Server

Once you have located the floating-ip-pools object, use the following to review its floating-ips object.

The floating-ips object should display the floating IP that is shown in the Contrail UI. The floating IP should have a reference to the virtual machine interface (VMI) object that is bound to the floating IP.

*Example*

```

[root@nodec6 ~]#
curlhttp://nodec6:8082/floating-ip-pool/663737c1-f3ab-40ff-9442-bdb6c225e3c3 |
python -m json.tool

```

```

{

```

```
"floating-ip-pool": {
  "floating_ips": [
    {
      "href":
"http://127.0.0.1:8095/floating-ip/f3eec4d6-889e-46a3-a8f0-879dfaff6ca0",
      "to": [
        "default-domain",
        "admin",
        "public_vn",
        "public_pool",
        "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0"
      ],
      "uuid": "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0"
    }
  ],
  "fq_name": [
    "default-domain",
    "admin",
    "public_vn",
    "public_pool"
  ],
  "href":
"http://127.0.0.1:8095/floating-ip-pool/663737c1-f3ab-40ff-9442-bdb6c225e3c3",
  "id_perms": {
    "created": "2014-02-07T08:58:41.136572",
    "description": null,
    "enable": true,
    "last_modified": "2014-02-07T08:58:41.136572",
    "permissions": {
```

```

        "group": "admin",
        "group_access": 7,
        "other_access": 7,
        "owner": "admin",
        "owner_access": 7
    },
    "uuid": {
        "uuid_lslong": 10683309858715198403,
        "uuid_mslong": 7365417021744038143
    }
},
"name": "public_pool",
"parent_href":
"http://127.0.0.1:8095/virtual-network/43707766-75f3-4d48-80d9-1b7240fb161d",
"parent_type": "virtual-network",
"parent_uuid": "43707766-75f3-4d48-80d9-1b7240fb161d",
"project_back_refs": [
    {
        "attr": {},
        "href": "http://127.0.0.1:8095/project/deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f",

        "to": [
            "default-domain",
            "admin"
        ],
        "uuid": "deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f"
    }
],
"uuid": "663737c1-f3ab-40ff-9442-bdb6c225e3c3"

```

```
}
}
```

## Check Floating IP Objects in the Virtual Machine Interface

Use the following to retrieve the virtual machine interface of the virtual machine from either the quantum port-list command or from the Contrail UI. Then get the virtual machine interface identifier and check its floating IP object associations.

- Using **quantum port-list** to get the virtual machine interface:

### Example

```
[root@nodec6 ~]# quantum port-list -F id -F fixed_ips
```

```
+-----+-----+
| id          | fixed_ips          |
+-----+-----+
| cdca35ce-84ad-45da-9331-7bc67b7fcca6 | {"subnet_id":
"e80f480b-98d4-43cc-847c-711e637295db", "ip_address": "10.1.1.253"} |
+-----+-----+
```

- Using Contrail UI to get the virtual machine interface:



### Checking Floating IP Objects on the Virtual Machine Interface

Once you have obtained the virtual machine interface identifier, check the floating-ip objects that are associated with the virtual machine interface.

```
[root@nodec6 ~]# curl
```



```
http://127.0.0.1:8095/floating-ip/f3eec4d6-889e-46a3-a8f0-879dfaff6ca0 | python
-m json.tool
```

```
{
  "floating-ip": {
    "floating_ip_address": "10.204.219.37",
    "fq_name": [
      "default-domain",
      "admin",
      "public_vn",
      "public_pool",
      "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0"
    ],
    "href": "http://127.0.0.1:8095/floating-ip/f3eec4d6-889e-46a3-a8f0-879dfaff6ca0",

    "id_perms": {
      "created": "2014-02-07T10:07:05.869899",
      "description": null,
      "enable": true,
      "last_modified": "2014-02-07T10:36:36.820926",
      "permissions": {
        "group": "admin",
        "group_access": 7,
        "other_access": 7,
        "owner": "admin",
        "owner_access": 7
      },
      "uuid": {
        "uuid_lslong": 12173378905373109408,
```

```
        "uuid_mslong": 17577202821367744163
      }
    },
    "name": "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0",
    "parent_href":
    "http://127.0.0.1:8095/floating-ip-pool/663737c1-f3ab-40ff-9442-bdb6c225e3c3",
    "parent_type": "floating-ip-pool",
    "parent_uuid": "663737c1-f3ab-40ff-9442-bdb6c225e3c3",
    "project_refs": [
      {
        "attr": null,
        "href": "http://127.0.0.1:8095/project/deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f",

        "to": [
          "default-domain",
          "admin"
        ],
        "uuid": "deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f"
      }
    ],
    "uuid": "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0",
    "virtual_machine_interface_refs": [
      {
        "attr": null,
        "href":
        "http://127.0.0.1:8095/virtual-machine-interface/cdca35ce-84ad-45da-9331-7bc67b7fcca6",

        "to": [
          "54bb44e1-50e4-43d7-addd-44be809f1e40",
          "cdca35ce-84ad-45da-9331-7bc67b7fcca6"
        ]
      }
    ]
  }
}
```

```

    ],
    "uuid": "cdca35ce-84ad-45da-9331-7bc67b7fcca6"
  }
]
}
}

```

### View Floating IP Objects in the IFMAP Server View

Use the following to view the output of `/usr/bin/ifmap_view.py` on the config-nodes. The IFMAP server example output shows the BGP peering configurations and the configurations of the virtual networks VN1 and public\_vn.

```
[root@nodec6 ~]# (source /opt/contrail/api-venv/bin/activate ; python
/usr/bin/ifmap_view.py nodec6 8443 test3 test3 -v 2 )
```

```
....
```

```
....
```

```
....
```

```
project = admin
```

```
floating-ip = f3eec4d6-889e-46a3-a8f0-879dfaff6ca0
```

```
project = admin
```

```
floating-ip-pool = public_pool
```

```
security-group = default
```

```
access-control-list = default-access-control-list
```

```
virtual-network = VN1
```

```
network-ipam = default-network-ipam
```

```
{
```

```
"ipam_subnets": [
```

```
{
```

```
"subnet": {
```

```
"ip_prefix": "10.1.1.0",
```

```
"ip_prefix_len": 24
```

```
    },
    "default_gateway": "10.1.1.254"
  }
],
"host_routes": null
}

routing-instance = VN1

route-target = 2

{
  "import_export": null
}

virtual-network = public_vn

floating-ip-pool = public_pool

floating-ip = f3eec4d6-889e-46a3-a8f0-879dfaff6ca0

virtual-machine-interface = cdca35ce-84ad-45da-9331-7bc67b7fcca6

network-ipam = default-network-ipam

{
  "ipam_subnets": [
    {
      "subnet": {
        "ip_prefix": "10.204.219.32",
        "ip_prefix_len": 29
      },
      "default_gateway": "10.204.219.38"
    }
  ],
  "host_routes": null
}
```

```
routing-instance = public_vn

route-target = 10003

{
  "import_export": null
}

route-target = 1

{
  "import_export": null
}

....

....

project = default-project

virtual-network = ip-fabric

routing-instance = __default__

bgp-router = nodec8

bgp-router = nodec7

{
  "session": [
    {
      "attributes": [
        {
          "bgp_router": null,
          "address_families": {
            "family": [
              "inet-vpn",
              "e-vpn"
            ]
          }
        }
      ]
    }
  ]
}
```

```
    }
  ],
  "uuid": null
}
]
}
bgp-router = nodec7
bgp-router = mx1
bgp-router = nodec7
{
  "session": [
    {
      "attributes": [
        {
          "bgp_router": null,
          "address_families": {
            "family": [
              "inet-vpn"
            ]
          }
        }
      ],
      "uuid": null
    }
  ],
  "bgp-router = nodec8"
}
```

```

    "session": [
      {
        "attributes": [
          {
            "bgp_router": null,
            "address_families": {
              "family": [
                "inet-vpn"
              ]
            }
          }
        ],
        "uuid": null
      }
    ]
  }
....
....
....

```

### View the BGP Peer Status on the Control Node

Use the Contrail UI or the control node http introspect on port 8083 to view the BGP peer status. In the following example, the control nodes are **nodec7** and **nodec8**.

Ensure that the BGP peering state is displayed as **Established** for the control nodes and the gateway MX.

*Example*

- Using the Contrail UI:

The screenshot shows the Contrail UI interface. The left sidebar contains navigation links: Dashboard, Control Nodes, Virtual Routers, Analytics Nodes, Config Nodes, and Networking. The main content area is titled 'Monitor > Infrastructure > Control Nodes > nodec8'. Below this, there are tabs for Details, Peers, Routes, and Console. The 'Peers' tab is active, displaying a table with the following data:

Peer	Peer Type	Peer ASN	Status	Last Rap	Messages (Recv/Sent)
10.204.216.253	BGP	64512	Established, in sync	-	1707/ 1590
10.204.216.64	BGP	64512	Established, in sync	2/7/2014 11:46:32 AM	1595/ 1597

At the bottom of the table, it says 'Displaying 1 - 2 of 2 Records' and '50 Records per page'. The bottom right corner shows 's041952'.

- Using the control-node Introspect:

`http://nodec7:8083/Snh_BgpNeighborReq?ip_address=&domain=`

`http://nodec8:8083/Snh_BgpNeighborReq?ip_address=&domain=`

## Querying Routes in the Public Virtual Network

On each control-node, a query on the routes in the **public\_vn** lists the routes that are pushed by the MX gateway, which in the following example are 0.0.0.0/0 and 10.204.218.0/24.

In the following results, the floating IP route of 10.204.217.32 is installed by the compute node (nodec10) that hosts that virtual machine.

*Example*

- Using the Contrail UI:

The screenshot shows the Contrail UI interface. The left sidebar contains navigation links: Dashboard, Control Nodes, Virtual Routers, Analytics Nodes, Config Nodes, and Networking. The main content area is titled 'Monitor > Infrastructure > Control Nodes > nodec8'. Below this, there are tabs for Details, Peers, Routes, and Console. The 'Routes' tab is active, displaying a table with the following data:

Routing Table	Prefix	Protocol	Source	Next hop	Label	Secur...	Origin VN
default-domainadminpublic_vnpublic_vn	0.0.0.0/0	BGP	10.204.216.253	10.204.216.253	16	-	default-domainadminpublic_vn
	10.204.218.0/24	BGP	10.204.216.253	10.204.216.253	16	-	default-domainadminpublic_vn
	10.204.217.32/32	XMPP	nodec10	10.204.216.67	16	1	default-domainadminpublic_vn

- Using the http Introspect:

Following is the format for using an introspect query.

`http://<nodename/ip>:8083/Snh_ShowRouteReq?x=<RoutingInstance of public VN>.inet.0`

*Example*



`http://nodec8:8083/Snh_BgpNeighborReq?ip_address=&domain=`

Contrail							Collapse	Expand	Wrap	Fullscreen
Show route table										
tables										
routing_instance	routing_table_name	prefixes	paths	primary_path	secondary_path	inheritable_paths	routes			
default-domain-admin-public-vm-public-vm	default-domain-admin-public-vm-public-vm	8.8.8.8/8	4	1	3	0	routes			
							prefix	last_modified	paths	
							8.8.8.8/8	2014-Feb-07 00:10:41.288405	paths	
									protocol	
									BGP	

**View Corresponding BGP LL3VPN Routes**

Use the Contrail UI or the http introspect to view the public route's corresponding BGP L3VPN routes, as in the following.

*Example*

- Using the Contrail UI:

Routing Table	Prefix	Protocol	Source	Next Hop	Label	Security	Origin/VN
bgp.l3vpn.0	10.204.216.253/0.0.0.0	BGP	10.204.216.253	10.204.216.253	16	-	-
	10.204.216.253/10.204.216.0/24	BGP	10.204.216.253	10.204.216.253	16	-	-
	10.204.216.67/10.1.1.253/32	XMPP	nodec10	10.204.216.67	16	1	default-domain-admin/vn.1
		BGP	10.204.216.64	10.204.216.67	16	1	default-domain-admin/vn.1
	10.204.216.67/2:10.204.216.37/32	XMPP	nodec10	10.204.216.67	16	1	default-domain-admin/vn.1
		BGP	10.204.216.64	10.204.216.67	16	1	default-domain-admin/vn.1

- Using the control-node Introspect:

`http://nodec7:8083/Snh_ShowRouteReq?x=bgp.l3vpn.0`

`http://nodec8:8083/Snh_ShowRouteReq?x=bgp.l3vpn.0`

**Verification from the MX80 Gateway**

This section provides options for verifying floating IP pools from the MX80 gateway.

**Verify BGP Sessions are Established**

Use the following commands from the gateway to verify that BGP sessions are established with the control nodes nodec7 and nodec8:

`root@mx-host> show bgp neighbor 10.204.216.64`

Peer: 10.204.216.64+59287 AS 64512 Local: 10.204.216.253+179 AS 64512

Type: Internal State: Established Flags: <Sync>  
Last State: OpenConfirm Last Event: RecvKeepAlive  
Last Error: Hold Timer Expired Error  
Options: <Preference LocalAddress KeepAll AddressFamily Rib-group Refresh>  
Address families configured: inet-vpn-unicast  
Local Address: 10.204.216.253 Holdtime: 90 Preference: 170  
Number of flaps: 216  
Last flap event: HoldTime  
Error: 'Hold Timer Expired Error' Sent: 68 Recv: 0  
Error: 'Cease' Sent: 0 Recv: 43  
Peer ID: 10.204.216.64 Local ID: 10.204.216.253 Active Holdtime: 90  
Keepalive Interval: 30 Group index: 0 Peer index: 3  
BFD: disabled, down  
NLRI for restart configured on peer: inet-vpn-unicast  
NLRI advertised by peer: inet-vpn-unicast  
NLRI for this session: inet-vpn-unicast  
Peer does not support Refresh capability  
Stale routes from peer are kept for: 300  
Peer does not support Restarter functionality  
Peer does not support Receiver functionality  
Peer does not support 4 byte AS extension  
Peer does not support Addpath

**Show Routes Learned  
from Control Nodes**

From the MX80, use show route to display the routes for the virtual machine 10.204.219.37 that are learned from both control-nodes.

In the following example, the routes learned are 10.204.216.64 and 10.204.216.65, pointing to a dynamic GRE tunnel next hop with a label of 16 (of the virtual machine).

public.inet.0: 4 destinations, 5 routes (4 active, 0 holddown, 0 hidden)

+ = Active Route, - = Last Active, \* = Both

```
0.0.0.0/0    *[Static/5] 10w6d 18:47:50
              > to 10.204.218.254 via ge-1/0/1.0

10.204.218.0/24  *[Direct/0] 10w6d 18:47:51
              > via ge-1/0/1.0

10.204.218.1/32  *[Local/0] 10w6d 18:48:07
              Local via ge-1/0/1.0

10.204.219.37/32 *[BGP/170] 09:42:43, localpref 100, from 10.204.216.64
              AS path: ?, validation-state: unverified
              > via gr-1/0/0.32779, Push 16

[BGP/170] 09:42:43, localpref 100, from 10.204.216.65
              AS path: ?, validation-state: unverified
              > via gr-1/0/0.32779, Push 16
```

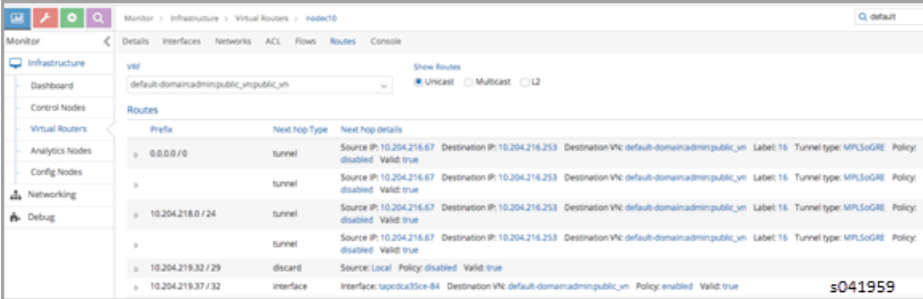
## Viewing the Compute Node VnsW Agent

The compute node introspect can be accessed from port 8085. In the following examples, the compute nodes are nodec9 and nodec10.

**View Routing Instance Next Hops** On the routing instance of VN1, the routes 0.0.0.0/0 and 10.204.218.0/24 should have the next hop pointing to the MX gateway (10.204.216.253).

*Example*

Using the Contrail UI:



Prefix	Next hop Type	Next hop details
0.0.0.0/0	tunnel	Source IP: 10.204.216.67 Destination IP: 10.204.216.253 Destination VN: default-domainadminpublic_vn Label: 16 Tunnel type: MPLSoGRE Policy: disabled Valid: true
10.204.218.0/24	tunnel	Source IP: 10.204.216.67 Destination IP: 10.204.216.253 Destination VN: default-domainadminpublic_vn Label: 16 Tunnel type: MPLSoGRE Policy: disabled Valid: true
10.204.219.32/29	discard	Source: Local Policy: disabled Valid: true
10.204.219.37/32	interface	Interface: tap0ca35ce-84 Destination VN: default-domainadminpublic_vn Policy: enabled Valid: true

Using the Unicast  
Route Table Index to  
View Next Hops

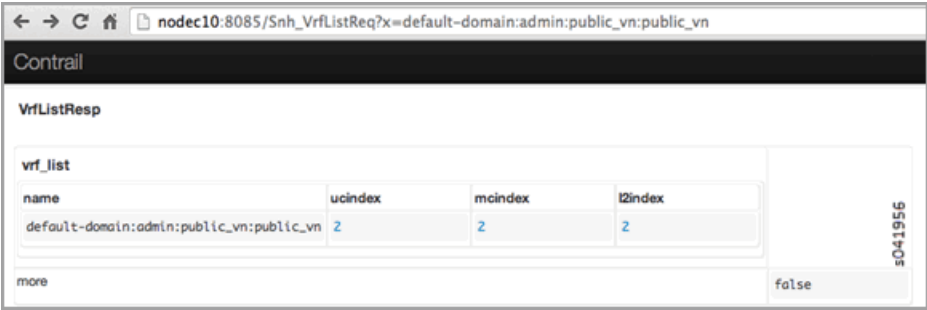
Alternatively, from the agent introspect, you can view the next hops at the unicast route table.

First, use the following to get the unicast route table index (ucindex ) for the routing instance **default-domain:admin:public\_vn:public\_vn**.

**http://nodec10:8085/Snh\_VrfListReq?x=default-domain:admin:public\_vn:public\_vn**

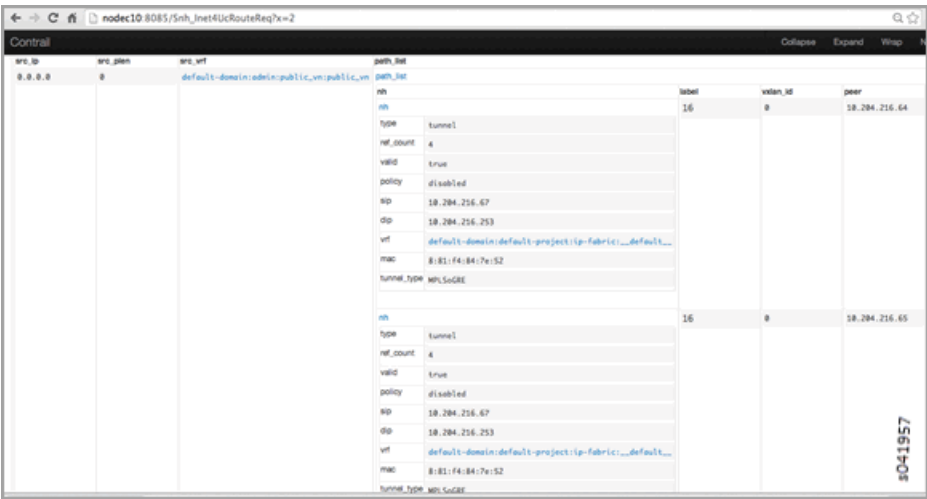
Example

In the following example, the unicast route table index is 2.



Next, perform a route request query on ucindex 2, as shown in the following. The tunnel detail indicates the source and destination endpoints of the tunnel and the MPLS label 16 (the label of the virtual machine).

The query should also show a route for 10.204.219.37 with an interface next hop of tap-interface. **http://nodec10:8085/Snh\_Inet4UcRouteReq?x=2**



10.204.219.37	32	default-domain:admin:public_vn:public_vn	path_list			
			nh	label	vxlan_id	peer
			nh	16	0	10.204.216.64
			type	interface		
			ref_count	g		
			valid	true		
			policy	enabled		
			if	tapcdca35ce-84		
			mac	2:cd:ca:35:ce:84		
			mcast	disabled		
			nh	16	0	10.204.216.65
			type	interface		
			ref_count	g		
			valid	true		
			policy	enabled		
			if	tapcdca35ce-84		
			mac	2:cd:ca:35:ce:84		
			mcast	disabled		

A ping from the MX gateway to the virtual machine's floating IP in the public routing-instance should work.

## Advanced Troubleshooting

If you still have reachability problems after performing all of the tests in this article, for example, a ping between the virtual machine and the MX IP or to public addresses is failing, try the following:

- Validate that all the required Contrail processes are running by using the **contrail-status** command on all of the nodes.
- On the compute node where the virtual machine is present (nodec10 in this example), perform a tcpdump on the tap interface (**tcpdump -ni tapcdca35ce-84**). The output should show the incoming packets from the virtual machine.

- Check to see if any packet drops occur in the kernel vrouter module:

**http://nodec10:8085/Snh\_KDropStatsReq?**

In the output, scroll down to find any drops. Note: You can ignore any ds\_invalid\_arp increments.

- On the physical interface where packets transmit onto the compute-node, perform a tcpdump matching the host IP of the MX to show the GRE encapsulated packets, as in the following.

```
[root@nodec10 ~]# cat /etc/contrail/agent.conf |grep -A 1 eth-port
```

```
<eth-port>
```

```
<name>p1p0p0</name>
```

```
</eth-port>
```

```
<metadata-proxy>
```

```
[root@nodec10 ~]# tcpdump -ni p1p0p0 host 10.204.216.253 -vv

tcpdump: WARNING: p1p0p0: no IPv4 address assigned

tcpdump: listening on p1p0p0, link-type EN10MB (Ethernet), capture size 65535 bytes

02:06:51.729941 IP (tos 0x0, ttl 64, id 57430, offset 0, flags [DF], proto GRE (47),
length 112)

    10.204.216.253 > 10.204.216.67: GREv0, Flags [none], length 92

        MPLS (label 16, exp 0, [S], ttl 54)

        IP (tos 0x0, ttl 54, id 35986, offset 0, flags [none], proto ICMP (1), length 84)

    172.29.227.6 > 10.204.219.37: ICMP echo request, id 53240, seq 242, length 64

02:06:51.730052 IP (tos 0x0, ttl 64, id 324, offset 0, flags [none], proto GRE (47),
length 112)

    10.204.216.67 > 10.204.216.253: GREv0, Flags [none], length 92

        MPLS (label 16, exp 0, [S], ttl 64)

        IP (tos 0x0, ttl 64, id 33909, offset 0, flags [none], proto ICMP (1), length 84)

    10.204.219.37 > 172.29.227.6: ICMP echo reply, id 53240, seq 242, length 64

02:06:52.732283 IP (tos 0x0, ttl 64, id 12675, offset 0, flags [DF], proto GRE (47),
length 112)

    10.204.216.253 > 10.204.216.67: GREv0, Flags [none], length 92

        MPLS (label 16, exp 0, [S], ttl 54)

        IP (tos 0x0, ttl 54, id 54155, offset 0, flags [none], proto ICMP (1), length 84)

    172.29.227.6 > 10.204.219.37: ICMP echo request, id 53240, seq 243, length 64

02:06:52.732355 IP (tos 0x0, ttl 64, id 325, offset 0, flags [none], proto GRE (47),
length 112)

    10.204.216.67 > 10.204.216.253: GREv0, Flags [none], length 92

        MPLS (label 16, exp 0, [S], ttl 64)

        IP (tos 0x0, ttl 64, id 33910, offset 0, flags [none], proto ICMP (1), length 84)

    10.204.219.37 > 172.29.227.6: ICMP echo reply, id 53240, seq 243, length 64

^C

4 packets captured

5 packets received by filter
```

0 packets dropped by kernel

[root@nodec10 ~]#

- On the MX gateway, use the following to inspect the GRE tunnel rx/tx (received/transmitted) packet count:

```
root@mx-host> show interfaces gr-1/0/0.32779 |grep packets
```

Input packets : 542

Output packets: 559

```
root@blr-mx1> show interfaces gr-1/0/0.32779 |grep packets
```

Input packets : 544

Output packets: 561

- Look for any packet drops in the FPC, as in the following:

```
show pfe statistics traffic fpc <id>
```

- Also inspect the dynamic tunnels, using the following:

```
show dynamic-tunnels database
```

---

## Removing Stale Virtual Machines and Virtual Machine Interfaces

This topic gives examples for removing stale VMs (virtual machines) and VMIs (virtual machine interfaces). Before you can remove a stale VM or VMI, you must first remove any back references associated to the VM or VMI.

- [Problem Example on page 442](#)
- [Show Virtual Machines on page 443](#)
- [Show Virtual Machines Using Python API on page 445](#)
- [Delete Methods on page 446](#)

### Problem Example

The troubleshooting examples in this topic are based on the following problem example. A **net-delete** of the virtual machine 2a8120ec-bd18-49f4-aca0-acfc6e8fe74f returned the following messages that there are two VMIs that still have back-references to the stale VM.

The two VMIs must be deleted first, then the Neutron **net-delete <vm\_ID>** command will complete without errors.

From neutron.log:

2014-03-10 14:18:05.208



```

DEBUG [urllib3.connectionpool]
"DELETE/virtual-network/2a8120ec-bd18-49f4-aca0-acfc6e8fe74f HTTP/1.1" 409 203
2014-03-10 14:18:05.278
ERROR [neutron.api.v2.resource] delete failed
Traceback (most recent call last):
  File "/usr/lib/python2.7/dist-packages/neutron/api/v2/resource.py", line
84, in resource
    result = method(request=request, **args)
  File "/usr/lib/python2.7/dist-packages/neutron/api/v2/base.py", line
432, in delete
    obj_deleter(request.context, id, **kwargs)
  File
"/usr/lib/python2.7/dist-packages/neutron/plugins/juniper/contrail/contrail
plugin.py", line 294, in delete_network
    raise e
RefsExistError: Back-References from
http://127.0.0.1:8082/virtual-machine-interface/51daf6f4-7366-4463-a819-bd1
17fe3a8c8,
http://127.0.0.1:8082/virtual-machine-interface/30882e66-e175-4fbb-862e-354
bb700b579 still exist

```

## Show Virtual Machines

Use the following command to show all of the virtual machines known to the Contrail API server. Replace the variable **<config-node-IP>** shown in the example with the IP address of the **config-node** in your setup.

```
http://<config-node-IP>:8082/virtual-machines
```

### Example

In the following example, 03443891-99cc-4784-89bb-9d1e045f8aa6 is a stale VM that needs to be removed.

```
virtual-machines:
```

```
[
```

```
{
  href:"http:
//example-node:8082/virtual-machine/03443891-99cc-4784-89bb-9d1e045f8aa6",
  fq_name:
  [
    "03443891-99cc-4784-89bb-9d1e045f8aa6"
  ],
  uuid:"03443891-99cc-4784-89bb-9d1e045f8aa6"
},
```

When the user attempts to delete the stale VM, a message displays that children to the VM still exist:

```
root@example-node:~# curl -X DELETE -H "Content-Type: application/json;
charset=UTF-8" http:
//127.0.0.1:8082/virtual-machine/03443891-99cc-4784-89bb-9d1e045f8aa6
Children http:
//127.0.0.1:8082/virtual-machine-interface/0c32a82a-7bd3-46c7-b262-6d85b9911a0d
still exist
root@example-node:~#
```

The user opens `http://example-node:8082/virtual-machine/03443891-99cc-4784-89bb-9d1e045f8aa6`, and sees a **virtual-machine-interface** (VMI) attached to it. The VMI must be removed before the VM can be removed.

However, when the user attempts to delete the VMI from the stale VM, they get a message that there is still a back-reference:

```
root@example-node:~# curl -X DELETE -H "Content-Type: application/json;
charset=UTF-8" http:
//<example-IP>:8082/virtual-machine-interface/0c32a82a-7bd3-46c7-b262-6d85b9911a0d

Back-References from http:
//<example-IP>:8082/instance-ip/6ffa29a1-023f-462b-b205-353da8e3a2a4 still exist

root@example-node:~#
```

Because there is a back-reference from an **instance-ip** object still present, the **instance-ip** object must first be deleted, as follows:

```
root@example-node:~# curl -X DELETE -H "Content-Type: application/json;
charset=UTF-8" http:
//<example-IP>:8082/instance-ip/6ffa29a1-023f-462b-b205-353da8e3a2a4

root@example-node:~#
```

When the **instance-ip** is deleted, then the VMI and the VM can be deleted.



**NOTE:** To prevent inconsistency, be certain that the VM is not present in the Nova database before deleting the VM.

## Show Virtual Machines Using Python API

The following example shows how to view virtual machines using a Python API. This example shows virtual machines and back-references. Once you identify back-references and existing children, you can delete them first, then delete the stale VM.

```
root@example-node:~# source /opt/contrail/api-venv/bin/activate
```

```
File "<stdin>", line 1, in <module>
```

```
File
```

```
"/opt/contrail/api-venv/lib/python2.7/site-packages/vnc_api/gen/vnc_api_client_gen.py",  
line 3793, in virtual_machine_interface_delete
```

```
content = self._request_server(rest.OP_DELETE, uri)
```

```
File "/opt/contrail/api-venv/lib/python2.7/site-packages/vnc_api/vnc_api.py", line  
342, in _request_server
```

```
raise RefsExistError(content)
```

```
cfgm_common.exceptions.RefsExistError: Back-References from http: //  
<example-IP>:8082/instance-ip/6ffa29a1-023f-462b-b205-353da8e3a2a4 still exist
```

```
>>> (api-venv)root@example-node:~# python
```

```
Python 2.7.5 (default, Mar 10 2014, 03:55:35)
```

```
[GCC 4.6.3] on linux2
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> from vnc_api.vnc_api import VncApi
```

```
>>> vh=VncApi()
```

```
>>>
```

```
vh.virtual_machine_interface_delete(id='0c32a82a-7bd3-46c7-b262-6d85b9911a0d')
```

Traceback (most recent call last):

```
File "<stdin>", line 1, in <module>
```

```
File
```

```
"/opt/contrail/api-venv/lib/python2.7/site-packages/vnc_api/gen/vnc_api_client_gen.py",  
line 3793, in virtual_machine_interface_delete
```

```
content = self._request_server(rest.OP_DELETE, uri)
```

```
File "/opt/contrail/api-venv/lib/python2.7/site-packages/vnc_api/vnc_api.py", line
```

```
342, in _request_server

    raise RefsExistError(content)

cfgm_common.exceptions.RefsExistError: Back-References from http: //
<example-IP>:8082/instance-ip/6ffa29a1-023f-462b-b205-353da8e3a2a4 still exist

>>>
```

## Delete Methods

Use help (**vh**) to show all delete methods supported.

Typical commands for deleting VMs and VMIs include:

- **virtual\_machine\_delete()** to delete a virtual machine
- **instance\_ip\_delete()** to delete an **instance-ip**.

---

## Troubleshooting Link-Local Services in Contrail

Use the troubleshooting steps and guidelines in this topic when you have errors with Contrail link-local services.

- [Overview of Link-Local Services on page 446](#)
- [Troubleshooting Procedure for Link-Local Services on page 447](#)
- [Metadata Service on page 449](#)
- [Troubleshooting Procedure for Link-Local Metadata Service on page 449](#)

## Overview of Link-Local Services

Virtual machines might be set up to access specific services hosted on the fabric infrastructure. For example, a virtual machine might be a Nova client that requires access to the Nova API service running in the fabric network. Access to services hosted on the fabric network can be provided by configuring the services as link-local services.

A link-local address and a service port is chosen for the specific service running on a TCP / UDP port on a server in the fabric. With the link-local service configured, virtual machines can access the service using the link-local address. For link-local services, Contrail uses the address range 169.254.169.x.

Link-local service can be configured using the Contrail WebUI: **Configure > Infrastructure > Link Local Services**.

## Troubleshooting Procedure for Link-Local Services

Use the following steps when you are troubleshooting link-local services errors.

1. Verify the reachability of the fabric server that is hosting the link-local service from the compute node.
2. Check the state of the virtual machine and the interface:
  - Is the **Status** of virtual machine **Up**?
  - Is the corresponding tap interface **Active**?

Checking the virtual machine status in the Contrail UI:

Name	Label	Status	Network	IP Address	Floating IP	Instance
tap4b094dbe-f0	18	Up	vn1 (demo)	1.2.3.247	None	4f4b917a-a071-4517-961a-0e41067fec53 / vn1-v m2

Checking the tap interface status in the http agent introspect:

**http://<compute-node-ip>:8085/Snh\_ItfReq?name=**

index	name	uuid	vrf_name	active
3	tap722a7a11-6d	722a7a11-6d2e-47e9-a4cc-687a105a240f	default-domain:demo:vn1:vn1	Active

3. Check the link-local configuration in the vrouter agent. Make sure the configured link-local service is displayed.

**http://<compute-node-ip>:8085/Snh\_LinkLocalServiceInfo?**

linklocal_service_name	linklocal_service_ip	linklocal_service_port	ipfabric_dns_name	ipfabric_ip	ipfabric_port
ntp	169.254.169.100	123	-	172.17.28.5	123

4. Validate the BGP neighbor config and the BGP peering config object. When the virtual machine communicates with the configured link-local service, a forward and reverse

flow for the communication is set up. Check that the flow for this communication is created and the flow action is NAT.

**`http://<compute-node-ip>:8085/Snh_KFlowReq?flow_idx=`**

Check that all flow entries display NAT action programmed and display flags for the fields (source or destination IP and ports) that have NAT programmed. Also shown are the number of packets and bytes transmitted in the respective flows.

flow_list									
index	rflow	sip	sport	dip	dport	proto	vrf_id	action	flags
467472	234436	1.2.3.247	123	169.254.169.100	123	17	1	NAT	ACTIVE   VEFT   SNAT   SPAT   DNAT   SPAT
234436	467472	172.17.28.5	123	10.204.216.72	43226	17	0	NAT	ACTIVE   VEFT   SNAT   DNAT   S041957

The forward flow displays the source IP of the virtual machine and the destination IP of the link-local service. The reverse flow displays the source IP of the fabric host and the destination IP of the compute node's vhost interface. If the service is hosted on the same compute node, the destination address of the reverse flow displays the metadata address allocated to the virtual machine.

Note that the **index** and **rflow** index for the two flows are reversed.

You can also view similar information in the vrouter agent introspect page, where you can see the policy and security group for the flow. Check that the flow actions display as **pass**.

**`http://<compute-node-ip>:8085/Snh_FetchAllFlowRecords?`**

## Metadata Service

OpenStack allows virtual instances to access metadata by sending an HTTP request to the link-local address 169.254.169.254. The metadata request from the instance is proxied to Nova, with additional HTTP header fields added, which Nova uses to identify the source instance. Then Nova responds with appropriate metadata.

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

## Troubleshooting Procedure for Link-Local Metadata Service

Metadata service is also a link-local service, with a fixed service name (metadata), a fixed service address (169.254.169.254:80), and a fabric address pointing to the server where the OpenStack Nova API server is running. All of the configuration and troubleshooting procedures for Contrail link-local services also apply to the metadata service.

However, for metadata service, the flow is always set up to the compute node, so the router agent will update and proxy the HTTP request. The router agent listens on a local port to receive the metadata requests. Consequently, the reverse flow has the compute node as the source IP, the local port on which the agent is listening is the source port, and the instance's metadata IP is the destination IP address.

After performing all of the troubleshooting procedures for link-local services, the following additional steps can be used to further troubleshoot metadata service.

1. Check the metadata statistics for: the number of metadata requests received by the router agent, the number of proxy sessions set up with the Nova API server, and number of internal errors encountered.

`http://<compute-node-ip>:8085/Snh_MetadataInfo?`

The port on which the router agent listens for metadata requests is also displayed.

metadata_server_port	45094
metadata_requests	2
metadata_responses	0
metadata_proxy_sessions	2
metadata_internal_errors	0

2. Check the metadata trace messages, which show the trail of metadata requests and responses.

**`http://<compute-node-ip>:8085/Snh_SandeshTraceRequest?x=Metadata`**

3. Check the Nova configuration. On the server running the OpenStack service, inspect the **nova.conf** file.

- Ensure that the metadata proxy is enabled, as follows:

**`service_neutron_metadata_proxy = True`**

**`service_quantum_metadata_proxy = True`** (on older installations)

- Check to see if the metadata proxy shared secret is set:

**`neutron_metadata_proxy_shared_secret`**

**`quantum_metadata_proxy_shared_secret`** (on older installations)

If the shared secret is set in **nova.conf**, the same secret must be configured on each compute node in the file **/etc/contrail/contrail-vrouter-agent.conf**, and the same shared secret must be updated in the **METADATA** section as **`metadata_proxy_secret=<secret>`**.

4. Restart the vrouter agent after modifying the shared secret:

**`service contrail-vrouter restart`**



## CHAPTER 16

# Contrail Commands

- [contrail-logs \(Accessing Log File Messages\)](#) on page 451
- [contrail-status \(Viewing Node Status\)](#)
- [contrail-version \(Viewing Version Information\)](#)
- [service \(Managing Services\)](#)
- [Backing Up and Restoring Configurations](#) on page 458

### **contrail-logs (Accessing Log File Messages)**

---

A command-line utility, **contrail-logs**, uses REST APIs to retrieve system log messages, object log messages, and trace messages.

- [Command-Line Options for Contrail-Logs](#) on page 451
- [Option Descriptions](#) on page 452
- [Example Uses](#) on page 453

### **Command-Line Options for Contrail-Logs**

The command-line utility for accessing log file information is **contrail-logs** in the analytics node. The following are the options supported at the command line for **contrail-logs**, as viewed using the **--help** option.

```
[root@host]# contrail-logs --help
usage: contrail-logs [-h]
                    [--opserver-ip OPSERVER_IP]
                    [--opserver-port OPSERVER_PORT]
                    [--start-time START_TIME]
                    [--end-time END_TIME]
                    [--last LAST]
                    [--source SOURCE]
                    [--module {ControlNode, VRouterAgent, ApiServer, Schema, OpServer, Collector,
QueryEngine, ServiceMonitor, DnsAgent}]
                    [--category CATEGORY]
                    [--level LEVEL]
                    [--message-type MESSAGE_TYPE]
                    [--reverse]
                    [--verbose]
                    [--all]
                    [--object {ObjectVNTable, ObjectVMTable, ObjectSITable, ObjectVRouter,
```

```
ObjectBgpPeer, ObjectRoutingInstance, ObjectBgpRouter, ObjectXmppConnection,  
ObjectCollectorInfo, ObjectGeneratorInfo, ObjectConfigNode]]  
    [--object-id OBJECT_ID]  
    [--object-select-field {ObjectLog,SystemLog}]  
    [--trace TRACE]
```

## Option Descriptions

The following are the descriptions for each of the option arguments available for **contrail-logs**.

optional arguments:

```
-h, --help  
    show this help message and exit  
--opserver-ip OPSERVER_IP  
    IP address of OpServer (default: 127.0.0.1)  
--opserver-port OPSERVER_PORT  
    Port of OpServer (default: 8081)  
--start-time START_TIME  
    Logs start time (format now-10m, now-1h) (default: now-10m)  
--end-time END_TIME  
    Logs end time (default: now)  
--last LAST  
    Logs from last time period (format 10m, 1d) (default: None)  
--source SOURCE  
    Logs from source address (default: None)  
--module {ControlNode, VRouterAgent, ApiServer, Schema, OpServer, Collector,  
QueryEngine, ServiceMonitor, DnsAgent}  
    Logs from module (default: None)  
--category CATEGORY  
    Logs of category (default: None)  
--level LEVEL  
    Logs of level (default: None)  
--message-type MESSAGE_TYPE  
    Logs of message type (default: None)  
--reverse  
    Show logs in reverse chronological order (default: False)  
--verbose  
    Show internal information (default: True)  
--all  
    Show all logs (default: False)  
--object {ObjectVNTTable, ObjectVMTable, ObjectSITable, ObjectVRouter,  
ObjectBgpPeer, ObjectRoutingInstance, ObjectBgpRouter, ObjectXmppConnection,  
ObjectCollectorInfo, ObjectGeneratorInfo, ObjectConfigNode}  
    Logs of object type (default: None)  
--object-id OBJECT_ID  
    Logs of object name (default: None)  
--object-select-field {ObjectLog,SystemLog}  
    Select field to filter the log (default: None)  
--trace TRACE  
    Dump trace buffer (default: None)
```

## Example Uses

The following examples show how you can use the option arguments available for **contrail-logs** to retrieve the information you specify.

1. View only the system log messages from all boxes for the last 10 minutes.

**contrail-logs**

2. View all log messages (systemlog, objectlog, uve, ...) from all boxes for the last 10 minutes.

**contrail-logs --all**

3. View only the control node system log messages from all boxes for the last 10 minutes.

**contrail-logs --module ControlNode**

**--module** accepts the following values - **ControlNode, VRouterAgent, ApiServer, Schema, ServiceMonitor, Collector, OpServer, QueryEngine, DnsAgent**

4. View the control node system log messages from source **a6s23.contrail.juniper.net** for the last 10 minutes.

**contrail-logs --module ControlNode --source a6s23.contrail.juniper.net**

5. View the XMPP category system log messages from all modules on all boxes for the last 10 minutes.

**contrail-logs --category XMPP**

6. View the system log messages from all the boxes from the last hour.

**contrail-logs --last 1h**

7. View the system log messages from the VN object named **demo:admin:vn1** from all boxes for the last 10 minutes.

**contrail-logs --object ObjectVNTable --object-id demo:admin:vn1**

**--object** accepts the following values - **ObjectVNTable, ObjectVMTable, ObjectSITable, ObjectVRouter, ObjectBgpPeer, ObjectRoutingInstance, ObjectBgpRouter, ObjectXmppConnection, ObjectCollectorInfo**

8. View the system log messages from all boxes for the last 10 minutes in reverse chronological order:

**contrail-logs --reverse**

9. View the system log messages from a specific time interval and display them in a specified date format.

**contrail-logs --start-time "2013 May 12 18:30:27.0" --end-time "2013 May 12 18:31:27.0"**

## contrail-status (Viewing Node Status)

---

<b>Syntax</b>	<code>[root@host ~]# contrail-status</code>
<b>Release Information</b>	Command introduced in Contrail Release 1.0.
<b>Description</b>	Display a list of all components of a Contrail server node (such as control, configuration, database, Web-UI, analytics, or vrouter) and report their current status of active or inactive.
<b>Required Privilege Level</b>	admin

### Sample Output

The following example usage displays on a server that is configured for the roles of **analytics**, **configuration**, **web-ui**, and **database**. It is not configured with the roles **control** or **vrouter**.

### Sample Output

```
root@host> contrail-status
VRouter is NOT PRESENT
Agent is NOT PRESENT
== Control node ==
supervisor-control:      active
contrail-control         active
supervisor-dns:          active
contrail-dns              active
contrail-named            active
== Analytics ==
supervisor-analytics:    active
contrail-analytics-nodemgr active
contrail-collector        active
contrail-opserver        active
contrail-qe              active
redis-query              active
redis-sentinel           active
redis-uve                active
== Contrail API server ==
supervisor-config:       active
contrail-api             active
contrail-discovery        active
contrail-schema          active
contrail-svc-monitor     active
ifmap                    active
== Contrail quantum ==
== Contrail Web UI ==
supervisor-webui:        active
contrail-webui           active
contrail-webui-middleware active
== Contrail Database ==
supervisord-contrail-database:active
contrail-database        active
```

## contrail-version (Viewing Version Information)

<b>Syntax</b>	<code>[root@host]# contrail-version</code>
<b>Release Information</b>	Command introduced in Contrail Release 1.0.
<b>Description</b>	Display a list of all installed components with their version and build numbers.
<b>Required Privilege Level</b>	admin

### Sample Output

The following example shows version and build information for all installed components.

#### Sample Output

```

root@host> contrail-version
Package                               Version                               Build-ID | Repo |
RPM Name
-----
contrail-analytics                   1-1309090026.e16                    141
contrail-analytics-venv              0.1-1309062310.e16                  141
contrail-api                         0.1-1309090026.e16                  141
contrail-api-lib                     0.1-1309090026.e16                  141
contrail-api-venv                    0.1-1309080539.e16                  141
contrail-control                     2012.0-1309090026.e16               141
contrail-database                    0.1-1309050028                      141
contrail-dns                         1-1309090026.e16                    141
contrail-fabric-utils                1-1309090026                        141
contrail-libs                        1-1309090026.e16                    141
contrail-nodejs                      0.8.15-1309090026.e16              141
contrail-openstack-analytics         0.1-1309090026.e16                  141
contrail-openstack-cfgm              0.1-1309090026.e16                  141
contrail-openstack-control           0.1-1309090026.e16                  141

```

### Sample Output

The following example shows version and build information for only the installed contrail components.

#### Sample Output

```

root@host> contrail-version | grep contrail
Package                               Version                               Build-ID | Repo |
RPM Name
-----
contrail-analytics                   1-1309090026.e16                    141
contrail-analytics-venv              0.1-1309062310.e16                  141
contrail-api                         0.1-1309090026.e16                  141
contrail-api-lib                     0.1-1309090026.e16                  141

```

contrail-api-venv	0.1-1309080539.e16	141
contrail-control	2012.0-1309090026.e16	141
contrail-database	0.1-1309050028	141
contrail-dns	1-1309090026.e16	141
contrail-fabric-utils	1-1309090026	141
contrail-libs	1-1309090026.e16	141
contrail-nodejs	0.8.15-1309090026.e16	141
contrail-openstack-analytics	0.1-1309090026.e16	141
contrail-openstack-cfgm	0.1-1309090026.e16	141
contrail-openstack-control	0.1-1309090026.e16	141
contrail-openstack-database	0.1-1309090026.e16	141
contrail-openstack-webui	0.1-1309090026.e16	141
contrail-setup	1-1309090026.e16	141
contrail-webui	1-1309090026	141
openstack-quantum-contrail	2013.2-1309090026	141

## service (Managing Services)

<b>Syntax</b>	<code>service contrail-service ( start   stop   restart   status )</code>
<b>Release Information</b>	Standard Linux command used for managing and viewing services in Contrail Controller Release 1.0.
<b>Description</b>	<p>Start, stop, or restart a Contrail service. Display the status of a Contrail service.</p> <p>All contrail services are managed by the process <b>supervisord</b>, which is open source software written in Python. Each Contrail node type, such as compute, control, and so on, has an instance of <b>supervisord</b> that, when running, launches Contrail services as child processes. All <b>supervisord</b> instances display in <b>contrail-status</b> output with the prefix <b>supervisor</b>. If the <b>supervisord</b> instance of a particular node type is not up, none of the services for that node type are up. For more details about the open source <b>supervisord</b> process, see <a href="http://www.supervisord.org">http://www.supervisord.org</a>.</p>
<b>Options</b>	<ul style="list-style-type: none"> <li>• <b>start</b>—start a named service.</li> <li>• <b>stop</b>—stop a named service.</li> <li>• <b>restart</b>—stop and restart a named service.</li> <li>• <b>status</b>—display the status of a named service.</li> </ul>
<b>Required Privilege Level</b>	admin

### Sample Output

The following examples show usage for the **contrail-collector** service, which is only configured on nodes that have the roles of **analytics**, **configuration**, **web-ui**, or **database**.

### Sample Output

```
[root@host service supervisor-analytics status
supervisord (pid 32116) is running... [
[root@host]# service contrail-collector restart

contrail-collector: stopped
contrail-collector: started
[root@host]# service contrail-collector stop

contrail-collector: stopped
[root@host]# service contrail-collector start

contrail-collector: started
[root@host]# service contrail-collector status

contrail-collector                RUNNING    pid 20071, uptime 0:00:04
```

## Backing Up and Restoring Configurations

---

- [Back up Procedure on page 458](#)
- [Restore Procedure on page 459](#)
- [Restore Steps Continued on page 469](#)
- [Finishing on page 469](#)

### Back up Procedure

#### Configuration Backup and Restore

1. Take a snapshot of the Cassandra database on all database nodes. Copy it to a different host if you intend to reimage or reset the same servers.

```
root@a4s1:~# nodetool -h localhost -p 7199 snapshot
```

```
Requested creating snapshot for: all keyspaces
```

```
Snapshot directory: 1403160262349
```



**NOTE:** The snapshot could be in /home/cassandra/ or /var/lib/cassandra. Zip the cassandra directory and store it in a remote host.

2. Get a back up of the MySQL database in the OpenStack node.

```
root@a4s1:~# cat /etc/contrail/mysql.token
```

```
422beb8ab4e9a6bdc5e7
```

```
root@a4s1:~# mysqldump -u root --password=422beb8ab4e9a6bdc5e7 --all-databases  
> openstack.sql
```

3. For testing purposes only: Bring servers to a clean state

```
fab reset_config
```



**NOTE:** This step is for testing purposes ONLY. This is not necessary if you are bringing up another node.



## Restore Procedure

1. Stop Nova services.

```
root@a4s1:~# service nova-api stop
nova-api stop/waiting
root@a4s1:~# service nova-compute stop
nova-compute stop/waiting
root@a4s1:~# service nova-scheduler stop
nova-scheduler stop/waiting
root@a4s1:~# service nova-conductor stop
nova-conductor stop/waiting
```

2. Stop Glance service.

```
root@a4s1:~# service glance-api stop
glance-api stop/waiting
root@a4s1:~# service glance-registry stop
glance-registry stop/waiting
```

3. Stop Keystone service.

```
root@a4s1:~# service keystone stop

keystone stop/waiting
```

4. Stop Config service.

```
fab stop_cfgm
```

5. Stop Collector service.

```
fab stop_collector
```

6. Stop Database service.

```
fab stop_database
```

7. Restore All OpenStack databases.

```
root@a4s1:~# cat /etc/contrail/mysql.token

e5814139795b0c06e90a

root@a4s1:~# mysql -u root --password=e5814139795b0c06e90a < openstack.sql
```

8. Restore the Cassandra database using the script **cass-db-restore-v4.sh**.



**NOTE:** Please copy the backed up Cassandra database to this server.

```
root@a4s1:~# ./cass-db-restore-v4.sh
```

NAME

Script to restore Cassandra database from snapshot

SYNOPSIS

```
cass-db-restore-v4.sh [--help|-h] [--base_db_dir|-b] [--snapshot_dir|-s]
[--snapshot_name|-n]
```

MUST OPTIONS: base\_db\_dir, snapshot\_dir, snapshot\_name

DESCRIPTION

--base\_db\_dir, -b

Location of running Cassandra database

--snapshot\_dir, -s

Snapshot location of Cassandra database

--snapshot\_name, -n

Snapshot name

#### Restore Example

```
cass-db-restore-v4.sh -b /var/lib/cassandra/data -s /root/data.ss -n 1403068337967
```

```
root@a4s1:~# ./cass-db-restore-v4.sh -b /home/cassandra/data -s
/root/data.ss/cassandra/data -n 1403160262349
```

Snapshot available...continuing..

-----dirs to be restored-----

to\_bgp\_keyspace/route\_target\_table/

ContrailAnalytics/MessageTableSource/

ContrailAnalytics/MessageTableMessageType/

ContrailAnalytics/StatsTableByU64StrTag/

ContrailAnalytics/MessageTableModuleId/

ContrailAnalytics/ObjectValueTable/

ContrailAnalytics/MessageTable/

ContrailAnalytics/StatsTableByStrStrTag/

ContrailAnalytics/MessageTableTimestamp/

```

ContrailAnalytics/MessageTableCategory/

ContrailAnalytics/SystemObjectTable/

ContrailAnalytics/ObjectTable/

config_db_uuid/obj_fq_name_table/

config_db_uuid/obj_uuid_table/

system/schema_columns/

system/local/

system/schema_columnfamilies/

system/schema_keyspaces/

-----db files in snapshots-----

=====check /home/cassandra/data/to_bgp_keyspace/route_target_table//
=====

to_bgp_keyspace-route_target_table-ic-1-CompressionInfo.db
to_bgp_keyspace-route_target_table-ic-2-CompressionInfo.db

to_bgp_keyspace-route_target_table-ic-1-Data.db
to_bgp_keyspace-route_target_table-ic-2-Data.db

to_bgp_keyspace-route_target_table-ic-1-Filter.db
to_bgp_keyspace-route_target_table-ic-2-Filter.db

to_bgp_keyspace-route_target_table-ic-1-Index.db
to_bgp_keyspace-route_target_table-ic-2-Index.db

to_bgp_keyspace-route_target_table-ic-1-Statistics.db
to_bgp_keyspace-route_target_table-ic-2-Statistics.db

to_bgp_keyspace-route_target_table-ic-1-Summary.db
to_bgp_keyspace-route_target_table-ic-2-Summary.db

to_bgp_keyspace-route_target_table-ic-1-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/MessageTableSource//
=====

ContrailAnalytics-MessageTableSource-ic-1-CompressionInfo.db
ContrailAnalytics-MessageTableSource-ic-2-CompressionInfo.db

ContrailAnalytics-MessageTableSource-ic-1-Data.db
ContrailAnalytics-MessageTableSource-ic-2-Data.db

ContrailAnalytics-MessageTableSource-ic-1-Filter.db
ContrailAnalytics-MessageTableSource-ic-2-Filter.db

ContrailAnalytics-MessageTableSource-ic-1-Index.db

```

ContrailAnalytics-MessageTableSource-ic-2-Index.db

ContrailAnalytics-MessageTableSource-ic-1-Statistics.db  
ContrailAnalytics-MessageTableSource-ic-2-Statistics.db

ContrailAnalytics-MessageTableSource-ic-1-Summary.db  
ContrailAnalytics-MessageTableSource-ic-2-Summary.db

ContrailAnalytics-MessageTableSource-ic-1-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/MessageTableMessageType//  
=====

ContrailAnalytics-MessageTableMessageType-ic-1-CompressionInfo.db  
ContrailAnalytics-MessageTableMessageType-ic-2-CompressionInfo.db

ContrailAnalytics-MessageTableMessageType-ic-1-Data.db  
ContrailAnalytics-MessageTableMessageType-ic-2-Data.db

ContrailAnalytics-MessageTableMessageType-ic-1-Filter.db  
ContrailAnalytics-MessageTableMessageType-ic-2-Filter.db

ContrailAnalytics-MessageTableMessageType-ic-1-Index.db  
ContrailAnalytics-MessageTableMessageType-ic-2-Index.db

ContrailAnalytics-MessageTableMessageType-ic-1-Statistics.db  
ContrailAnalytics-MessageTableMessageType-ic-2-Statistics.db

ContrailAnalytics-MessageTableMessageType-ic-1-Summary.db  
ContrailAnalytics-MessageTableMessageType-ic-2-Summary.db

ContrailAnalytics-MessageTableMessageType-ic-1-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/StatsTableByU64StrTag//  
=====

ContrailAnalytics-StatsTableByU64StrTag-ic-1-CompressionInfo.db  
ContrailAnalytics-StatsTableByU64StrTag-ic-1-Index.db

ContrailAnalytics-StatsTableByU64StrTag-ic-1-Data.db  
ContrailAnalytics-StatsTableByU64StrTag-ic-1-Statistics.db

ContrailAnalytics-StatsTableByU64StrTag-ic-1-Filter.db  
ContrailAnalytics-StatsTableByU64StrTag-ic-1-Summary.db

=====check /home/cassandra/data/ContrailAnalytics/MessageTableModuleId//  
=====

ContrailAnalytics-MessageTableModuleId-ic-1-CompressionInfo.db  
ContrailAnalytics-MessageTableModuleId-ic-2-CompressionInfo.db

ContrailAnalytics-MessageTableModuleId-ic-1-Data.db  
ContrailAnalytics-MessageTableModuleId-ic-2-Data.db

ContrailAnalytics-MessageTableModuleId-ic-1-Filter.db  
ContrailAnalytics-MessageTableModuleId-ic-2-Filter.db

```

ContrailAnalytics-MessageTableModuleId-ic-1-Index.db
ContrailAnalytics-MessageTableModuleId-ic-2-Index.db

ContrailAnalytics-MessageTableModuleId-ic-1-Statistics.db
ContrailAnalytics-MessageTableModuleId-ic-2-Statistics.db

ContrailAnalytics-MessageTableModuleId-ic-1-Summary.db
ContrailAnalytics-MessageTableModuleId-ic-2-Summary.db

ContrailAnalytics-MessageTableModuleId-ic-1-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/ObjectValueTable//
=====

ContrailAnalytics-ObjectValueTable-ic-1-CompressionInfo.db
ContrailAnalytics-ObjectValueTable-ic-2-CompressionInfo.db

ContrailAnalytics-ObjectValueTable-ic-1-Data.db
ContrailAnalytics-ObjectValueTable-ic-2-Data.db

ContrailAnalytics-ObjectValueTable-ic-1-Filter.db
ContrailAnalytics-ObjectValueTable-ic-2-Filter.db

ContrailAnalytics-ObjectValueTable-ic-1-Index.db
ContrailAnalytics-ObjectValueTable-ic-2-Index.db

ContrailAnalytics-ObjectValueTable-ic-1-Statistics.db
ContrailAnalytics-ObjectValueTable-ic-2-Statistics.db

ContrailAnalytics-ObjectValueTable-ic-1-Summary.db
ContrailAnalytics-ObjectValueTable-ic-2-Summary.db

ContrailAnalytics-ObjectValueTable-ic-1-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/MessageTable//
=====

ContrailAnalytics-MessageTable-ic-1-CompressionInfo.db
ContrailAnalytics-MessageTable-ic-2-CompressionInfo.db

ContrailAnalytics-MessageTable-ic-1-Data.db
ContrailAnalytics-MessageTable-ic-2-Data.db

ContrailAnalytics-MessageTable-ic-1-Filter.db
ContrailAnalytics-MessageTable-ic-2-Filter.db

ContrailAnalytics-MessageTable-ic-1-Index.db
ContrailAnalytics-MessageTable-ic-2-Index.db

ContrailAnalytics-MessageTable-ic-1-Statistics.db
ContrailAnalytics-MessageTable-ic-2-Statistics.db

ContrailAnalytics-MessageTable-ic-1-Summary.db
ContrailAnalytics-MessageTable-ic-2-Summary.db

```

ContrailAnalytics-MessageTable-ic-1-TOC.txt  
ContrailAnalytics-MessageTable-ic-2-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/StatsTableByStrStrTag//  
=====

ContrailAnalytics-StatsTableByStrStrTag-ic-1-CompressionInfo.db  
ContrailAnalytics-StatsTableByStrStrTag-ic-2-CompressionInfo.db

ContrailAnalytics-StatsTableByStrStrTag-ic-1-Data.db  
ContrailAnalytics-StatsTableByStrStrTag-ic-2-Data.db

ContrailAnalytics-StatsTableByStrStrTag-ic-1-Filter.db  
ContrailAnalytics-StatsTableByStrStrTag-ic-2-Filter.db

ContrailAnalytics-StatsTableByStrStrTag-ic-1-Index.db  
ContrailAnalytics-StatsTableByStrStrTag-ic-2-Index.db

ContrailAnalytics-StatsTableByStrStrTag-ic-1-Statistics.db  
ContrailAnalytics-StatsTableByStrStrTag-ic-2-Statistics.db

ContrailAnalytics-StatsTableByStrStrTag-ic-1-Summary.db  
ContrailAnalytics-StatsTableByStrStrTag-ic-2-Summary.db

ContrailAnalytics-StatsTableByStrStrTag-ic-1-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/MessageTableTimestamp//  
=====

ContrailAnalytics-MessageTableTimestamp-ic-1-CompressionInfo.db  
ContrailAnalytics-MessageTableTimestamp-ic-2-CompressionInfo.db

ContrailAnalytics-MessageTableTimestamp-ic-1-Data.db  
ContrailAnalytics-MessageTableTimestamp-ic-2-Data.db

ContrailAnalytics-MessageTableTimestamp-ic-1-Filter.db  
ContrailAnalytics-MessageTableTimestamp-ic-2-Filter.db

ContrailAnalytics-MessageTableTimestamp-ic-1-Index.db  
ContrailAnalytics-MessageTableTimestamp-ic-2-Index.db

ContrailAnalytics-MessageTableTimestamp-ic-1-Statistics.db  
ContrailAnalytics-MessageTableTimestamp-ic-2-Statistics.db

ContrailAnalytics-MessageTableTimestamp-ic-1-Summary.db  
ContrailAnalytics-MessageTableTimestamp-ic-2-Summary.db

ContrailAnalytics-MessageTableTimestamp-ic-1-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/MessageTableCategory//  
=====

ContrailAnalytics-MessageTableCategory-ic-1-CompressionInfo.db  
ContrailAnalytics-MessageTableCategory-ic-2-CompressionInfo.db

ContrailAnalytics-MessageTableCategory-ic-1-Data.db

```

ContrailAnalytics-MessageTableCategory-ic-2-Data.db

ContrailAnalytics-MessageTableCategory-ic-1-Filter.db
ContrailAnalytics-MessageTableCategory-ic-2-Filter.db

ContrailAnalytics-MessageTableCategory-ic-1-Index.db
ContrailAnalytics-MessageTableCategory-ic-2-Index.db

ContrailAnalytics-MessageTableCategory-ic-1-Statistics.db
ContrailAnalytics-MessageTableCategory-ic-2-Statistics.db

ContrailAnalytics-MessageTableCategory-ic-1-Summary.db
ContrailAnalytics-MessageTableCategory-ic-2-Summary.db

ContrailAnalytics-MessageTableCategory-ic-1-TOC.txt

=====check /home/cassandra/data/ContrailAnalytics/SystemObjectTable//
=====

ContrailAnalytics-SystemObjectTable-ic-1-CompressionInfo.db
ContrailAnalytics-SystemObjectTable-ic-1-Statistics.db

ContrailAnalytics-SystemObjectTable-ic-1-Data.db
ContrailAnalytics-SystemObjectTable-ic-1-Summary.db

ContrailAnalytics-SystemObjectTable-ic-1-Filter.db
ContrailAnalytics-SystemObjectTable-ic-1-TOC.txt

ContrailAnalytics-SystemObjectTable-ic-1-Index.db

=====check /home/cassandra/data/ContrailAnalytics/ObjectTable//
=====

ContrailAnalytics-ObjectTable-ic-1-CompressionInfo.db
ContrailAnalytics-ObjectTable-ic-2-CompressionInfo.db

ContrailAnalytics-ObjectTable-ic-1-Data.db
ContrailAnalytics-ObjectTable-ic-2-Data.db

ContrailAnalytics-ObjectTable-ic-1-Filter.db
ContrailAnalytics-ObjectTable-ic-2-Filter.db

ContrailAnalytics-ObjectTable-ic-1-Index.db
ContrailAnalytics-ObjectTable-ic-2-Index.db

ContrailAnalytics-ObjectTable-ic-1-Statistics.db
ContrailAnalytics-ObjectTable-ic-2-Statistics.db

ContrailAnalytics-ObjectTable-ic-1-Summary.db
ContrailAnalytics-ObjectTable-ic-2-Summary.db

ContrailAnalytics-ObjectTable-ic-1-TOC.txt

=====check /home/cassandra/data/config_db_uuid/obj_fq_name_table//
=====

```





```

system-schema_columns-ic-5-Statistics.db    system-schema_columns-ic-6-Filter.db

=====check /home/cassandra/data/system/local// =====

system-local-ic-6-CompressionInfo.db system-local-ic-6-TOC.txt
system-local-ic-7-Summary.db    system-local-ic-8-Statistics.db

system-local-ic-6-Data.db        system-local-ic-7-CompressionInfo.db
system-local-ic-7-TOC.txt        system-local-ic-8-Summary.db

system-local-ic-6-Filter.db    system-local-ic-7-Data.db
system-local-ic-8-CompressionInfo.db system-local-ic-8-TOC.txt

system-local-ic-6-Index.db    system-local-ic-7-Filter.db
system-local-ic-8-Data.db

system-local-ic-6-Statistics.db    system-local-ic-7-Index.db
system-local-ic-8-Filter.db

system-local-ic-6-Summary.db    system-local-ic-7-Statistics.db
system-local-ic-8-Index.db

=====check /home/cassandra/data/system/schema_columnfamilies//
=====

system-schema_columnfamilies-ic-45-CompressionInfo.db
system-schema_columnfamilies-ic-47-CompressionInfo.db

system-schema_columnfamilies-ic-45-Data.db
system-schema_columnfamilies-ic-47-Data.db

system-schema_columnfamilies-ic-45-Filter.db
system-schema_columnfamilies-ic-47-Filter.db

system-schema_columnfamilies-ic-45-Index.db
system-schema_columnfamilies-ic-47-Index.db

system-schema_columnfamilies-ic-45-Statistics.db
system-schema_columnfamilies-ic-47-Statistics.db

system-schema_columnfamilies-ic-45-Summary.db
system-schema_columnfamilies-ic-47-Summary.db

system-schema_columnfamilies-ic-45-TOC.txt
system-schema_columnfamilies-ic-47-TOC.txt

system-schema_columnfamilies-ic-46-CompressionInfo.db
system-schema_columnfamilies-ic-48-CompressionInfo.db

system-schema_columnfamilies-ic-46-Data.db
system-schema_columnfamilies-ic-48-Data.db

system-schema_columnfamilies-ic-46-Filter.db
system-schema_columnfamilies-ic-48-Filter.db

```

```
system-schema_columnnfamilies-ic-46-Index.db
system-schema_columnnfamilies-ic-48-Index.db

system-schema_columnnfamilies-ic-46-Statistics.db
system-schema_columnnfamilies-ic-48-Statistics.db

system-schema_columnnfamilies-ic-46-Summary.db
system-schema_columnnfamilies-ic-48-Summary.db

system-schema_columnnfamilies-ic-46-TOC.txt

=====check /home/cassandra/data/system/schema_keyspaces//
=====

system-schema_keyspaces-ic-5-CompressionInfo.db
system-schema_keyspaces-ic-6-CompressionInfo.db
system-schema_keyspaces-ic-7-CompressionInfo.db

system-schema_keyspaces-ic-5-Data.db      system-schema_keyspaces-ic-6-Data.db
system-schema_keyspaces-ic-7-Data.db

system-schema_keyspaces-ic-5-Filter.db    system-schema_keyspaces-ic-6-Filter.db
system-schema_keyspaces-ic-7-Filter.db

system-schema_keyspaces-ic-5-Index.db    system-schema_keyspaces-ic-6-Index.db
system-schema_keyspaces-ic-7-Index.db

system-schema_keyspaces-ic-5-Statistics.db
system-schema_keyspaces-ic-6-Statistics.db
system-schema_keyspaces-ic-7-Statistics.db

system-schema_keyspaces-ic-5-Summary.db
system-schema_keyspaces-ic-6-Summary.db
system-schema_keyspaces-ic-7-Summary.db

system-schema_keyspaces-ic-5-TOC.txt      system-schema_keyspaces-ic-6-TOC.txt

root@a4s1:~#
```

## Restore Steps Continued

9. Start Nova services.

```
root@a4s1:~# service nova-api start
nova-api start/running, process 25075

root@a4s1:~# service nova-compute start
nova-compute start/running, process 25527

root@a4s1:~# service nova-scheduler start
nova-scheduler start/running, process 25509

root@a4s1:~# service nova-conductor start
nova-conductor start/running, process 25545
```

10. Start Glance service.

```
root@a4s1:~# service glance-api start
glance-api start/running, process 25779

root@a4s1:~# service glance-registry start
glance-registry start/running, process 25793
```

11. Start Keystone service.

```
root@a4s1:~# service keystone start

keystone start/running, process 25806
```

12. Start Config service.

```
fab start_cfgm
```

13. Start Collector service.

```
fab start_collector
```

14. Start Database service.

```
fab start_database
```

## Finishing

**Purpose** Once all the services are started again, you should be able to restore all the VNs and policies in the new node setup.



## PART 6

# Index

- [Index on page 473](#)



# Index

## Symbols

#, comments in configuration statements.....	xxvii
( ), in syntax descriptions.....	xxvii
< >, in syntax descriptions.....	xxvii
[ ], in configuration statements.....	xxvii
{ }, in configuration statements.....	xxvii
(pipe), in syntax descriptions.....	xxvii

## A

ASN	
global.....	26

## B

BGP peers.....	50
braces, in configuration statements.....	xxvii
brackets	
angle, in syntax descriptions.....	xxvii
square, in configuration statements.....	xxvii

## C

comments, in configuration statements.....	xxvii
compute nodes	
vRouter.....	4
XMPP agent.....	4
configure custom	
hostname.....	13
IP address.....	13
LAN port.....	13
nameserver.....	13
Contrail ISO.....	24
Contrail packages.....	24
contrail-logs.....	29, 33, 451
control node	
configuring.....	50
control nodes.....	4
conventions	
text and syntax.....	xxvi
curly braces, in configuration statements.....	xxvii
customer support.....	xxviii
contacting JTAC.....	xxviii

## D

dashboard.....	273
DHCP.....	230
DKMS.....	134
DNS.....	229
configuring.....	232
DHCP.....	230
IPAM.....	230
record types.....	231
scripts.....	237
<i>See also</i> Domain Name System	
documentation	
comments on.....	xxvii
Domain Name System.....	229
<i>See also</i> DNS	

## E

EX 4200.....	187
existing OpenStack.....	36

## F

font conventions.....	xxvi
-----------------------	------

## H

hardware requirements.....	10
heat template.....	225
high availability.....	33, 193
hostname	
configure custom.....	13

## I

image	
creating.....	146
infrastructure.....	273
install.....	24
instance	
virtual machine.....	148
IP address	
configure custom.....	13
IP Address Management.....	229
<i>See also</i> IPAM	
IP address pool	
allocating.....	164
creating.....	162
floating.....	162, 164
IPAM.....	229, 230
<i>See also</i> IP Address Management	

## L

LAN port	
configure custom.....	13

## M

manuals	
comments on.....	xxvii
monitor.....	273
multi-tier example.....	181
multitenancy.....	251
MX 80.....	187

## N

nameserver	
configure custom.....	13
network	
create.....	143
delete.....	142, 144
Juniper.....	142
OpenStack.....	144
network policy	
associating to a network.....	153, 160
creating.....	151, 157
Juniper.....	151, 153
OpenStack.....	157, 160

## O

object log.....	29, 33, 451
OpenStack.....	3

## P

parentheses, in syntax descriptions.....	xxvii
policy	
associating to a network.....	153, 160
creating.....	151, 157
Juniper.....	151, 153
OpenStack.....	157, 160
projects	
creating.....	138

## R

roles	
cfgm.....	9
collector.....	9
compute.....	9
control.....	9
webui.....	9
rpm.....	24

## S

security groups	
associating to an instance.....	165
service chain	
creating.....	210, 217, 222
example.....	210, 217, 222
service instance	
commands.....	210, 217, 222
service policy	
commands.....	210, 217, 222
service template	
commands.....	210, 217, 222
support, technical See technical support	
syntax conventions.....	xxvi
syslog.....	29, 33, 451

## T

technical support	
contacting JTAC.....	xxviii
testbed definitions.....	26, 36
testbed.py.....	26, 36
trace messages.....	29, 33, 451

## V

virtual machine	
instance.....	148
launching.....	148
virtual network	
creating.....	139, 143
Juniper.....	139
OpenStack.....	143
VSRX.....	210, 217, 222