Engineering Simplicity

Contrail™

Contrail Getting Started Guide

Published 2023-11-02

RELEASE 4.1

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

Contrail™ Contrail Getting Started Guide
4.1

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide | xii

1 **Overview**

Understanding Contrail | 2

Contrail Overview | 2

Contrail Description | 3

Contrail Installation Overview | 4

2 **Installing and Upgrading Contrail**

Supported Platforms and Server Requirements | 8

Supported Platforms Contrail 4.1 | 8

Server Requirements | 12

Installing Contrail and Provisioning Roles | 14

Introduction to Containerized Contrail Modules | 14

Downloading Installation Software | 18

Installing the Operating System and Contrail Packages | 18

Installing Containerized Contrail Clusters Using Server Manager | 20

Installing Containerized Contrail Using Server Manager Lite (SM-Lite) | 24

Supporting Multiple Interfaces on Servers and Nodes | 27

Configuring the Control Node with BGP | 31

Adding a New Node to an Existing Containerized Contrail Cluster | 36

Using contrailctl to Configure Services Within Containers | 39

Supporting Multiple Interfaces on Servers and Nodes | 42

Contrail Global Controller | 45

Role and Resource-Based Access Control | 47

Installation and Configuration Scenarios | 58

Setting Up and Using a Simple Virtual Gateway with Contrail 4.0 | 58

Introduction to the Simple Gateway | 58

How the Simple Gateway Works | 59

Setup Without Simple Gateway | 59

Setup With a Simple Gateway | 60

Simple Gateway Configuration Features | 61

Packet Flows with the Simple Gateway | 62

Packet Flow Process From the Virtual Network to the Public Network | 63

Packet Flow Process From the Public Network to the Virtual Network | 63

Methods for Configuring the Simple Gateway | 64

Using the vRouter Configuration File to Configure the Simple Gateway | 64

Using Thrift Messages to Dynamically Configure the Simple Gateway | 64

How to Dynamically Create a Virtual Gateway | 65

How to Dynamically Delete a Virtual Gateway | 66

Using Devstack to Configure the Simple Gateway | 67

Common Issues with Simple Gateway Configuration | 67

Simple Underlay Connectivity without Gateway | 68

Using Server Manager to Automate Provisioning | 72

Installing Server Manager | 72

Using Server Manager to Automate Provisioning | 79

Overview of Server Manager | 79

Server Manager Requirements and Assumptions | 80

Server Manager Component Interactions | 81

Configuring Server Manager | 82

Configuring the Cobbler DHCP Template | 84

User-Defined Tags for Server Manager | 85

Server Manager Client Configuration File | 85

Restart Services | 86

Accessing Server Manager | 86

Communicating with the Server Manager Client | 87

Server Manager Commands for Configuring Servers | 88

Server Manager Commands Common Options | 88

Add New Servers or Update Existing Servers | 89

Delete Servers | 90

Display Server Configuration | 91
Server Manager Commands for Managing Clusters | 92
Server Manager Commands for Managing Tags | 94
Server Manager Commands for Managing Images | 96
Server Manager Operational Commands for Managing Servers | 100
Reimaging Server(s) | 100
Provisioning and Configuring Roles on Servers | 102
Restarting Server(s) | 103
Show Status of Server(s) | 104
Show Status of Provision | 105
Server Manager REST API Calls | 105
REST APIs for Server Manager Configuration Database Entries | 106
API: Add a Server | 106
API: Delete Servers | 106
API: Retrieve Server Configuration | 107
API: Add an Image | 107
API: Upload an Image | 108
API: Get Image Information | 108
API: Delete an Image | 108
API: Add or Modify a Cluster | 109
API: Delete a Cluster | 109
API: Get Cluster Configuration | 109
API: Get All Server Manager Configurations | 110
API: Reimage Servers | 110
API: Provision Servers | 110
API: Restart Servers | 111
Example: Reimaging and Provisioning a Server | 111
Using the Server Manager Web User Interface | 113
Log In to Server Manager | 113
Create a Cluster for Server Manager | 114
Edit a Cluster through Edit JSON | 125
Working with Servers in the Server Manager User Interface | 125
Add a Server | 126
Edit Tags for Servers | 129
Using the Edit Config Option for Multiple Servers | 129

Edit a Server through Server Manager, Edit JSON | 130
Filter Servers by Tag | 131
Viewing Server Details | 131
Configuring Images and Packages | 134
Add New Image or Package | 135
Selecting Server Manager Actions for Clusters | 135
Reimage a Cluster | 136
Provision a Cluster | 136

Installing and Using Server Manager Lite | 137

Installing and Using Contrail Storage | 140

Installing and Using Contrail Storage | 140
Overview of the Contrail Storage Solution | 140
Basic Storage Functionality with Contrail | 141
Ceph Block and Object Storage Functionality | 141
Using the Contrail Storage User Interface | 142
Hardware Specifications | 143
Contrail Storage Provisioning | 143

Upgrading Contrail Software | 146

Upgrading Contrail 4.0 to 4.1 | 146

Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3 | 148

Upgrade Procedure for Ubuntu-based Contrail 4.1.3 to Contrail 4.1.4 Using Juju with Netronome SmartNIC | 161

Upgrade Procedure for RHOSP-based Contrail 4.1.3 to Contrail 4.1.4 | 170

Dynamic Kernel Module Support (DKMS) for vRouter | 185

3 **Configuring Contrail**

Configuring Virtual Networks | 188

Creating Projects in OpenStack for Configuring Tenants in Contrail | 188

Creating a Virtual Network with Juniper Networks Contrail | 190

Creating a Virtual Network with OpenStack Contrail | 194

Creating an Image for a Project in OpenStack Contrail | 196

Creating a Floating IP Address Pool | 200

Using Security Groups with Virtual Machines (Instances) | 202

- Security Groups Overview | 202
- Creating Security Groups and Adding Rules | 202

Support for IPv6 Networks in Contrail | 206

Configuring EVPN and VXLAN | 210

- Configuring the VXLAN Identifier Mode | 212
- Configuring Forwarding | 214
- Configuring the VXLAN Identifier | 215
- Configuring Encapsulation Methods | 216

Example of Deploying a Multi-Tier Web Application Using Contrail | 220

Example: Deploying a Multi-Tier Web Application | 220

- Multi-Tier Web Application Overview | 220
- Example: Setting Up Virtual Networks for a Simple Tiered Web Application | 221
- Verifying the Multi-Tier Web Application | 224
- Sample Addressing Scheme for Simple Tiered Web Application | 224
- Sample Physical Topology for Simple Tiered Web Application | 225
- Sample Physical Topology Addressing | 226

Sample Network Configuration for Devices for Simple Tiered Web Application | 228

Configuring Services | 235

Configuring DNS Servers | 235

- DNS Overview | 235
- Defining Multiple Virtual Domain Name Servers | 236
- IPAM and Virtual DNS | 237
- DNS Record Types | 237
- Configuring DNS Using the Interface | 238
- Configuring DNS Using Scripts | 246

Support for Multicast | 247

- Subnet Broadcast | 248
- All-Broadcast/Limited-Broadcast and Link-Local Multicast | 249
- Host Broadcast | 249

Using Static Routes with Services | 250

Static Routes for Service Instances | 250
Configuring Static Routes on a Service Instance | 251
Configuring Static Routes on Service Instance Interfaces | 252
Configuring Static Routes as Host Routes | 253

Configuring Metadata Service | 254

Configuring Service Chaining | 256

Service Chaining | 256

Service Chaining Basics | 256
Service Chaining Configuration Elements | 258

Service Chaining MX Series Configuration | 260

ECMP Load Balancing in the Service Chain | 262

Customized Hash Field Selection for ECMP Load Balancing | 263

Using the Contrail Heat Template | 268

Service Chain Route Reorigination | 273

Service Instance Health Checks | 295

Health Check Object | 295
Bidirectional Forwarding and Detection Health Check over Virtual Machine Interfaces | 300
Bidirectional Forwarding and Detection Health Check for BGPaaS | 300
Health Check of Transparent Service Chain | 301
Service Instance Fate Sharing | 301

Examples: Configuring Service Chaining | 303

Example: Creating an In-Network or In-Network-NAT Service Chain | 303

Example: Creating a Transparent Service Chain | 313

Example: Creating a Service Chain With the CLI | 319

4 **Monitoring and Troubleshooting the Network Using Contrail Analytics**

Understanding Contrail Analytics | 325

Understanding Contrail Analytics | 325

Contrail Alerts | 326

Underlay Overlay Mapping in Contrail | 330

Overview: Underlay Overlay Mapping using Contrail Analytics | 331

Underlay Overlay Analytics Available in Contrail | 331

Architecture and Data Collection | 332

New Processes/Services for Underlay Overlay Mapping | 332

External Interfaces Configuration for Underlay Overlay Mapping | 333

Physical Topology | 333

SNMP Configuration | 334

Link Layer Discovery Protocol (LLDP) Configuration | 334

IPFIX and sFlow Configuration | 334

Sending pRouter Information to the SNMP Collector in Contrail | 337

pRouter UVEs | 337

Contrail User Interface for Underlay Overlay Analytics | 339

Enabling Physical Topology on the Web UI | 340

Viewing Topology to the Virtual Machine Level | 340

Viewing the Traffic of any Link | 340

Trace Flows | 341

Search Flows and Map Flows | 342

Overlay to Underlay Flow Map Schemas | 343

Module Operations for Overlay Underlay Mapping | 346

SNMP Collector Operation | 346

Topology Module Operation | 348

IPFIX and sFlow Collector Operation | 349

Troubleshooting Underlay Overlay Mapping | 350

Script to add pRouter Objects | 350

Configuring Contrail Analytics | 353

Analytics Scalability | 353

High Availability for Analytics | 354

Role-Based Access Control for Analytics | 355

System Log Receiver in Contrail Analytics | 356

Overview | 357

Redirecting System Logs to Contrail Collector | 357

Exporting Logs from Contrail Analytics | 357

Sending Flow Messages to the Contrail System Log | 357

More Efficient Flow Queries | 358

Ceilometer Support in a Contrail Cloud | 359

- Overview | 359
- Ceilometer Details | 360
- Verification of Ceilometer Operation | 360
- Contrail Ceilometer Plugin | 363
- Ceilometer Installation and Provisioning | 366

Using Contrail Analytics to Monitor and Troubleshoot the Network | 367

Monitoring the System | 367

Debugging Processes Using the Contrail Introspect Feature | 371

Monitor > Infrastructure > Dashboard | 376

- Monitor Dashboard | 377
- Monitor Individual Details from the Dashboard | 377
- Using Bubble Charts | 378
- Color-Coding of Bubble Charts | 379

Monitor > Infrastructure > Control Nodes | 380

- Monitor Control Nodes Summary | 380
- Monitor Individual Control Node Details | 381
- Monitor Individual Control Node Console | 383
- Monitor Individual Control Node Peers | 386
- Monitor Individual Control Node Routes | 388

Monitor > Infrastructure > Virtual Routers | 391

- Monitor vRouters Summary | 391
- Monitor Individual vRouters Tabs | 393
- Monitor Individual vRouter Details Tab | 393
- Monitor Individual vRouters Interfaces Tab | 395
- Monitor Individual vRouters Networks Tab | 397
- Monitor Individual vRouters ACL Tab | 398
- Monitor Individual vRouters Flows Tab | 400
- Monitor Individual vRouters Routes Tab | 401
- Monitor Individual vRouter Console Tab | 402

Monitor > Infrastructure > Analytics Nodes | 405

Monitor Analytics Nodes | 405
Monitor Analytics Individual Node Details Tab | 407
Monitor Analytics Individual Node Generators Tab | 408
Monitor Analytics Individual Node QE Queries Tab | 409
Monitor Analytics Individual Node Console Tab | 410

Monitor > Infrastructure > Config Nodes | 413

Monitor Config Nodes | 413
Monitor Individual Config Node Details | 414
Monitor Individual Config Node Console | 415

Monitor > Networking | 417

Monitor > Networking Menu Options | 417
Monitor -> Networking -> Dashboard | 418
Monitor > Networking > Projects | 420
Monitor Projects Detail | 421
Monitor > Networking > Networks | 424

Query > Flows | 429

Query > Flows > Flow Series | 430
Example: Query Flow Series | 433
Query > Flow Records | 435
Query > Flows > Query Queue | 438

Query > Logs | 439

Query > Logs Menu Options | 440
Query > Logs > System Logs | 440
Sample Query for System Logs | 442
Query > Logs > Object Logs | 444

Example: Debugging Connectivity Using Monitoring for Troubleshooting | 446

Using Monitoring to Debug Connectivity | 446

About This Guide

Contrail Release 4.0

RELATED DOCUMENTATION

- Contrail Release 4.1
- Release Notes 4.1
- Day One: Understanding OpenContrail Architecture
- Juniper Contrail Configuration API Reference
- Juniper Networks TechWiki: Contrail

1

PART

Overview

Understanding Contrail | 2

CHAPTER 1

Understanding Contrail

IN THIS CHAPTER

- Contrail Overview | 2
- Contrail Description | 3
- Contrail Installation Overview | 4

Contrail Overview

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

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

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

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

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

RELATED DOCUMENTATION

Contrail Description | 3

Contrail Description

IN THIS SECTION

Contrail Major Components | 3
Contrail Solution | 4

Contrail Major Components

The following are the major components of Contrail.

Contrail Control Nodes

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

Contrail Compute Nodes – XMPP Agent and vRouter

- Responsible for managing the data plane.

- Functionality can reside on a host OS.

Contrail Solution

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

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

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

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

RELATED DOCUMENTATION

Contrail Overview | 2
Contrail Roles Overview

Contrail Installation Overview

IN THIS SECTION

- Installing Contrail on Different Operating Systems | 5

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

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

Installing Contrail on Different Operating Systems

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

Supported Platforms Contrail 4.1

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

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

Juniper OpenStack Ubuntu Installation

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

- "Introduction to Containerized Contrail Modules" on page 14
- "Downloading Installation Software" on page 18
- "Installing the Operating System and Contrail Packages" on page 18
- "Installing Containerized Contrail Clusters Using Server Manager" on page 20
- "Installing Containerized Contrail Using Server Manager Lite (SM-Lite)" on page 24

Using VMware vCenter with Containerized Contrail, Release 4.0.1 and Greater

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

- Installing and Provisioning VMware vCenter with Containerized Contrail
- Underlay Network Configuration for Containerized ContrailVM
- Sample JSON Configuration Files for vCenter with Containerized Contrail 4.0.1 and Greater

Using VMware vCenter with Contrail, through Release 4.0

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

- Installing and Provisioning VMware vCenter with Contrail
- Underlay Network Configuration for ContrailVM
- Sample Testbed.py Files for Contrail vCenter

Using Red Hat with Contrail

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

- Installing Red Hat OpenShift Container Platform with Contrail Networking

Using Contrail with Kubernetes Automation Platform

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

- *Contrail Integration with Kubernetes*
- Installing and Provisioning Containerized Contrail Controller for Kubernetes
- *Verifying Configuration for CNI for Kubernetes*
- Using Kubernetes Helm to Provision Contrail

Using APIs with Contrail

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

Juniper Contrail Configuration API Reference

2
PART

Installing and Upgrading Contrail

Supported Platforms and Server Requirements | 8

Installing Contrail and Provisioning Roles | 14

Installation and Configuration Scenarios | 58

Using Server Manager to Automate Provisioning | 72

Installing and Using Contrail Storage | 140

Upgrading Contrail Software | 146

CHAPTER 2

Supported Platforms and Server Requirements

IN THIS CHAPTER

- Supported Platforms Contrail 4.1 | 8
- Server Requirements | 12

Supported Platforms Contrail 4.1

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

Table 1: Supported Platforms

Contrail Release	Orchestrator Release	Operating System and Kernel Versions
Contrail Release 4.1.5	OpenStack Newton	• RHEL7.5—Linux Kernel Version 3.10.0-862.14.4 (RHOSP 10.0) [Satellite content synced on Oct 29, 2018] • RHEL7.7—Linux Kernel Version 3.10.0-1062.12.1 (RHOSP 10.0.14) [Satellite content synced on May 20. 2020]
	OpenStack Ocata	• Ubuntu 16.04.6 - Linux Kernel Version 4.15.0-112-generic

Table 1: Supported Platforms *(Continued)*

Contrail Release	Orchestrator Release	Operating System and Kernel Versions
Contrail Release 4.1.4.1	OpenStack Newton	• RHEL7.5—Linux Kernel Version 3.10.0-862.14.4 (RHOSP 10.0) • RHEL7.7—Linux Kernel Version 3.10.0-1062.9.1 (RHOSP 10.0.14)
Contrail Release 4.1.4	OpenStack Ocata	• Ubuntu 16.04.2—Linux kernel version 4.4.0-165-generic • VMware vCenter 6.0, 6.5—Ubuntu 16.04.2 kernel version 4.4.0-62-generic
	OpenStack Newton	• RHEL7.7—Linux Kernel Version 3.10.0-1062.1.2 (RHOSP 10.0.12) • Ubuntu 16.04.2—Linux kernel version 4.4.0-165-generic
	OpenStack Mitaka	• Ubuntu 14.04.5—Linux kernel versions 3.13.0-171-generic
Contrail Release 4.1.3	OpenStack Ocata	• RHEL 7.5—Linux kernel version 3.10.0-862.11.6 and Linux kernel version 3.10.0-957 (RHOSP11) • Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic • VMware vCenter 6.0, 6.5—Ubuntu 16.04.2 kernel version 4.4.0-62-generic
	OpenStack Newton	• RHEL 7.5—Linux kernel version 3.10.0-862.11.6 • RHEL 7.6—Linux kernel version 3.10.0-957 (RHOSP10) • Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic

Table 1: Supported Platforms *(Continued)*

Contrail Release	Orchestrator Release	Operating System and Kernel Versions
	OpenStack Mitaka	• Ubuntu 14.04.5—Linux kernel versions 3.13.0-142-generic and 4.4.0-116-generic
Contrail Release 4.1.2	Kubernetes 1.7.5	• Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic
	OpenStack Ocata	• RHEL 7.5—Linux kernel version 3.10.0-862.11.6 and Linux kernel version 3.10.0-957 (RHOSP11) • Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic • VMware vCenter 6.0, 6.5—Ubuntu 16.04.2 kernel version 4.4.0-62-generic
	OpenStack Newton	• RHEL 7.5—Linux kernel version 3.10.0-862.11.6 • Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic
	OpenStack Mitaka	• Ubuntu 14.04.5—Linux kernel versions 3.13.0-142-generic and 4.4.0-116-generic
Contrail Release 4.1.1	Kubernetes 1.7.5	• Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic
	Openshift 3.6	• RHEL 7.5—Linux kernel version 3.10.0-862.3.2

Table 1: Supported Platforms *(Continued)*

Contrail Release	Orchestrator Release	Operating System and Kernel Versions
	OpenStack Ocata	• RHEL 7.5—Linux kernel version 3.10.0-862.3.2 (RHOSP11) • Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic • VMware vCenter 6.0, 6.5—Ubuntu 16.04.2 kernel version 4.4.0-62-generic
	OpenStack Newton	• RHEL 7.5—Linux kernel version 3.10.0-862.3.2 (RHOSP10) • Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic
	OpenStack Mitaka	• Ubuntu 14.04.5—Linux kernel versions 3.13.0-142-generic and 4.4.0-116-generic
Contrail Release 4.1	Kubernetes 1.7.5	• Ubuntu 16.04.2—Linux kernel version 4.4.0-62-generic
	Openshift 3.6	• RHEL 7.4—Linux kernel version 3.10.0-693
	OpenStack Ocata	• RHEL 7.4—Linux kernel version 3.10.0-693 (RHOSP11) • Ubuntu 16.04.2—Linux kernel version 4.4.0-62-generic • VMware vCenter 6.0, 6.5—Ubuntu 16.04.2 kernel version 4.4.0-62-generic

Table 1: Supported Platforms *(Continued)*

Contrail Release	Orchestrator Release	Operating System and Kernel Versions
	OpenStack Newton	• RHEL 7.4—Linux kernel version 3.10.0-693 (RHOSP10) • Ubuntu 16.04.2—Linux kernel version 4.4.0-62-generic
	OpenStack Mitaka	• Ubuntu 14.04.5—Linux kernel versions 3.13.0-110-generic and 4.4.0-34-generic

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

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

Server Requirements

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

Each server must have a minimum of:

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

RELATED DOCUMENTATION

Contrail Roles Overview

Downloading Installation Software | 18

CHAPTER 3

Installing Contrail and Provisioning Roles

IN THIS CHAPTER

- Introduction to Containerized Contrail Modules | 14
- Downloading Installation Software | 18
- Installing the Operating System and Contrail Packages | 18
- Installing Containerized Contrail Clusters Using Server Manager | 20
- Installing Containerized Contrail Using Server Manager Lite (SM-Lite) | 24
- Supporting Multiple Interfaces on Servers and Nodes | 27
- Configuring the Control Node with BGP | 31
- Adding a New Node to an Existing Containerized Contrail Cluster | 36
- Using contrailctl to Configure Services Within Containers | 39
- Supporting Multiple Interfaces on Servers and Nodes | 42
- Contrail Global Controller | 45
- Role and Resource-Based Access Control | 47

Introduction to Containerized Contrail Modules

IN THIS SECTION

- Why Use Containers? | 15
- Overview of Contrail Containers | 15
- Contrail 4.0 Containers | 16
- Summary of Container Design, Configuration Management, and Orchestration | 17

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

Why Use Containers?

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

These dependencies lead to complexities of deployment, including:

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

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

Overview of Contrail Containers

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

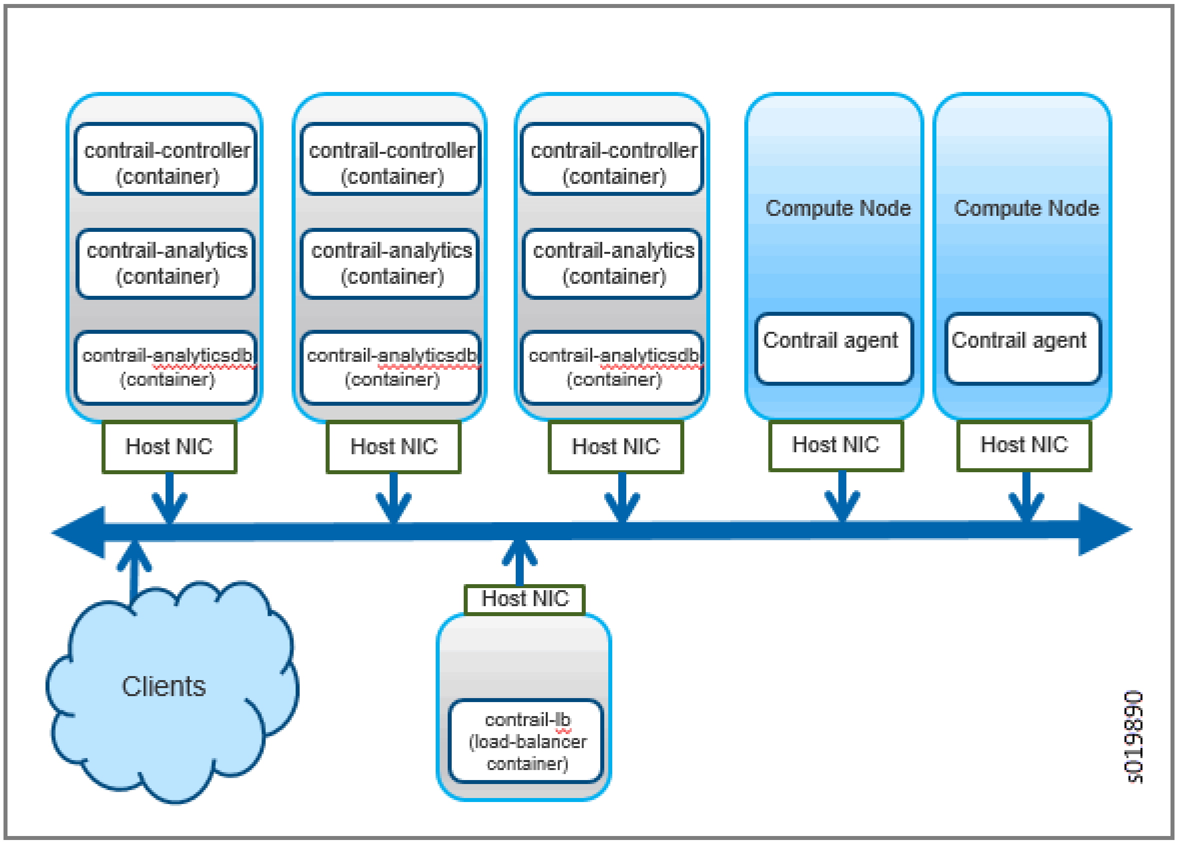
In Contrail 4.0, the containerized components include Contrail controller, analytics, and load-balancer applications. Contrail OpenStack components are not containerized at this time.

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

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

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

Figure 1: Sample Configuration Containerized Contrail



Contrail 4.0 Containers

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

contrail-controller

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

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

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

contrail-analytics

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

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

contrail-analyticsdb

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

contrail-lb

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

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

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

Summary of Container Design, Configuration Management, and Orchestration

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

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

RELATED DOCUMENTATION

Using contrailctl to Configure Services Within Containers | 39

Downloading Installation Software

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

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

The Contrail image includes the following software:

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

Installing the Operating System and Contrail Packages

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

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

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

   Sample JSON Snippet

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

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

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

For full installation information, see "Installing Containerized Contrail Clusters Using Server Manager" on page 20 and Installing Containerized Contrail for Single- and Multi-Node Systems Using Server Manager Lite

RELATED DOCUMENTATION

Introduction to Containerized Contrail Modules | 14

Contrail Roles Overview

Installing Containerized Contrail Clusters Using Server Manager | 20

Installing Containerized Contrail Using Server Manager Lite (SM-Lite) | 24
Upgrading Contrail 3.2 to 4.0
Download Software

Installing Containerized Contrail Clusters Using Server Manager

IN THIS SECTION

- Installing Server Manager | 20
- Creating Objects with Server Manager and JSONs | 21
- Preparing the Target System for Provisioning | 23
- Provisioning the System | 23

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

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

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

The installation is completed using the following major activities:

Installing Server Manager

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

1. Install the Server Manager wrapper package:

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

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

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

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

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

a. In the `setup.sh` script, comment out the line containing the `docker run` command.

b. `dpkg --purge contrail-server-manager`

c. `setup.sh --all --hostip=[IP address of SM]`

3. When the Server Manager install completes with no errors, modify the DHCP template at `/etc/cobbler/dhcp.template` to include the details of the subnet being reimaged or provisioned. Be sure to include DNS details.

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

4. Start the Server Manager process:

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

For more details about the Server Manager installation process, refer to "Installing Server Manager" on page 72.

Creating Objects with Server Manager and JSONs

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

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

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

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

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

1. Configure the images needed for reimaging and provisioning.

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

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

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

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

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

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

2. Configure the cluster(s).

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

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

   For a multi-node system:

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

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

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

3. Configure the server.

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

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

Preparing the Target System for Provisioning

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

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

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

- For a multi-node system:

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

Provisioning the System

Launch the system provisioning.

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

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

- For a multi-node system:

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

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

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

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

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

RELATED DOCUMENTATION

- Sample JSONs for an all-in-one, single node with roles
- Sample JSONs for a Multinode Cluster
- Sample JSONs for a Multinode Cluster with High Availability
- Introduction to Containerized Contrail Modules | 14
- Contrail Roles Overview
- Installing the Operating System and Contrail Packages | 18
- Installing Containerized Contrail Using Server Manager Lite (SM-Lite) | 24

Upgrading Contrail 3.2 to 4.0

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

IN THIS SECTION

- Preparing for SM-Lite Installation | 24
- Installing SM-Lite | 25
- Provisioning Contrail Using SM-Lite | 26
- Sample JSONs and Testbed.py | 26

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

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

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

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

Preparing for SM-Lite Installation

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

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

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

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

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

Prerequisites

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

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

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

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

    `<Host non mgmt IP> <server hostname>.<domain_name> <server hostname>`

  - The domain name is present in **/etc/resolv.conf** as follows:

    `search <domain_name>`

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

Installing SM-Lite

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

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

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

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

```
Help:
`/opt/contrail/contrail_server_manager/provision_containers.sh -h`
`-h --help`
`-cj <cluster json path>`
`-sj <server json path>`
`-ij <image json path>`
`-t|--testbed <testbed.py path>`
`-c <contrail cloud docker package path>`
```

```
`-cid|--cluster-id <cluster-id>`
`-ni|--no-install-sm-lite`
```

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

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

Provisioning Contrail Using SM-Lite

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

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

- Using JSONs

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

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

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

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

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

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

Sample JSONs and Testbed.py

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

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

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

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

RELATED DOCUMENTATION

Sample JSONs for an All-In-One-Node Cluster (for demo)
Sample JSONs for a Multinode Cluster with Two Nodes
Sample JSONs for a Multinode Cluster with 7 Nodes and High Availability
Sample testbed.py for Provisioning Containers with SM-Lite
Introduction to Containerized Contrail Modules | 14
Contrail Roles Overview
Installing the Operating System and Contrail Packages | 18
Installing Containerized Contrail Clusters Using Server Manager | 20
Upgrading Contrail 3.2 to 4.0

Supporting Multiple Interfaces on Servers and Nodes

IN THIS SECTION

- Support for Multiple Interfaces | 28
- Server Interface Examples | 30
- Interface Naming and Configuration Management | 30

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

Support for Multiple Interfaces

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

Examples of control traffic include the following:

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

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

Number of cfgm Nodes Supported

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

Uneven Number of Database Nodes Required

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

Support for VLAN Interfaces

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

```
“network”: {
     “interfaces”: [
         {
            “name”: “vlan2003”,
            “type”  : “vlan”,
            “vlan”: “2003”,
            “parent_interface”: “bond0”,
```

```
            “ip_address": “10.224.11.10/24”,
            “default_gateway”:  “10.224.12.1”
           }
       ]
   }
```

Support for Bonding Options

Contrail provides support for bond interface options.

The default bond interface options are:

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

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

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

Support for Static Route Options

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

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

```
          “network”: {
                       "routes": [
                           {
                              "gateway": "3.3.2.254",
                              "interface": "enp129s0f0",
                              "netmask": "255.255.255.0",
```

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

Server Interface Examples

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

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

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

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

Interface Naming and Configuration Management

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

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

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

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

Configuring the Control Node with BGP

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

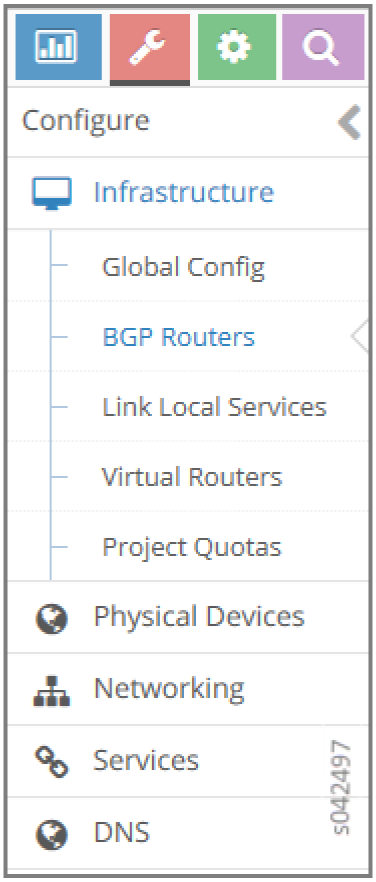
Before you begin, ensure that the following tasks are completed:

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

To configure BGP peering in the control node:

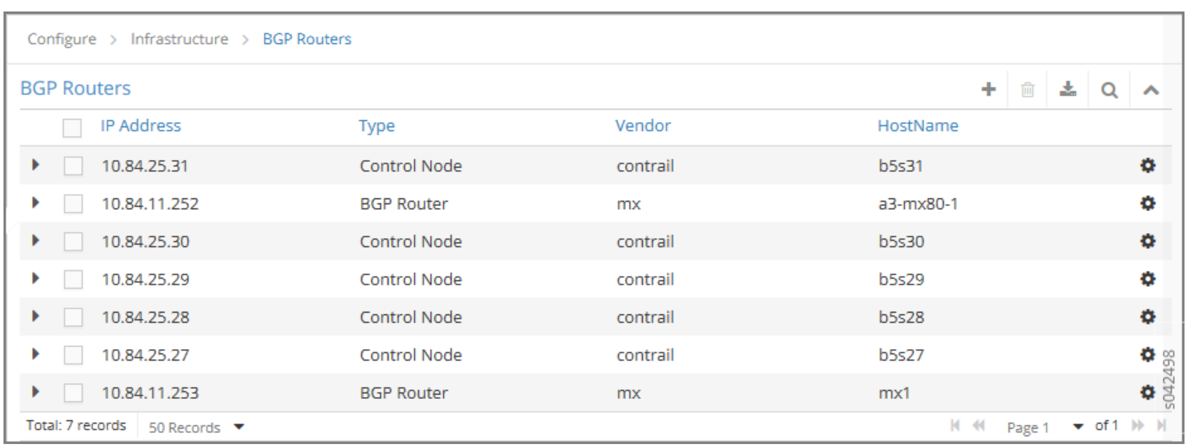
1. From the Contrail Controller module control node (**http://***nn.nn.nn.nn***:8080**), select **Configure > Infrastructure > BGP Routers**; see Figure 2 on page 32.

Figure 2: Configure> Infrastructure > BGP Routers



A summary screen of the control nodes and BGP routers is displayed; see Figure 3 on page 32.

Figure 3: BGP Routers Summary

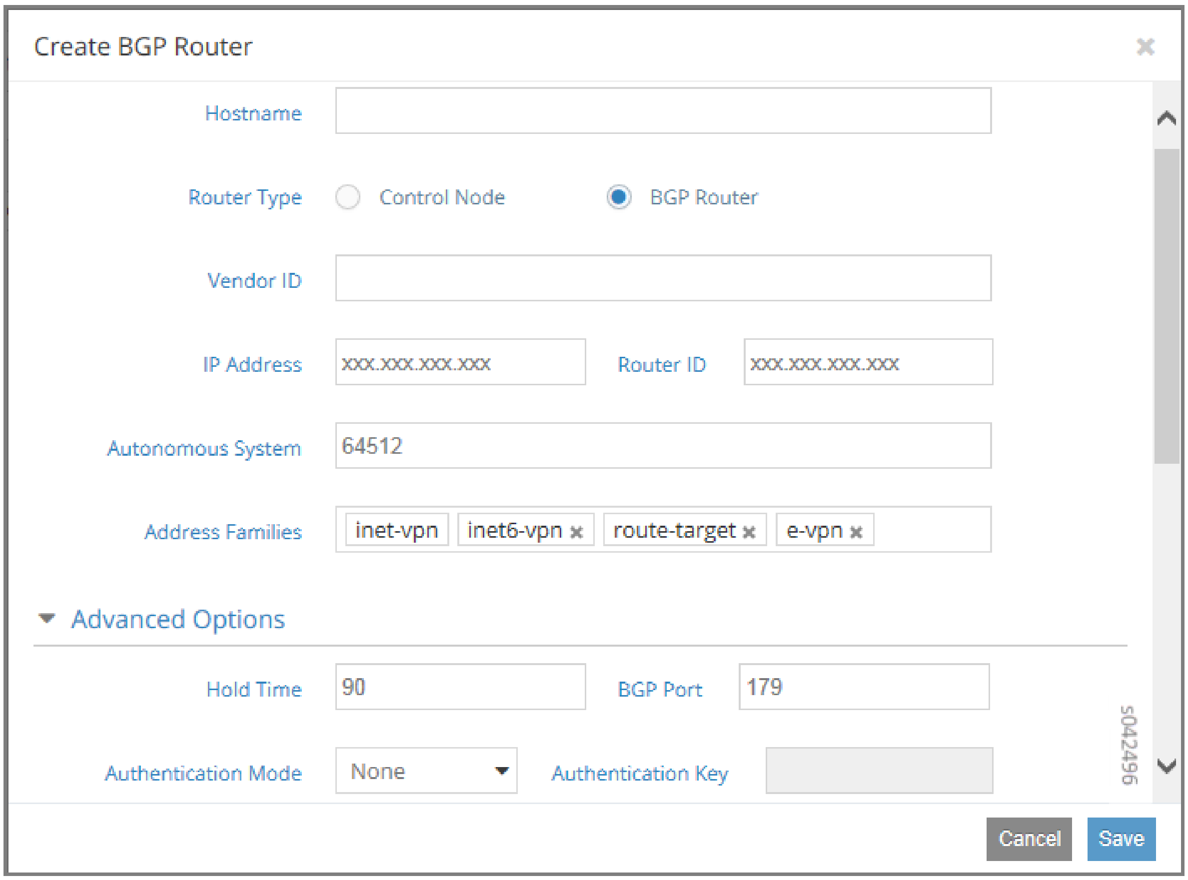


IP Address	Type	Vendor	HostName
10.84.25.31	Control Node	contrail	b5s31
10.84.11.252	BGP Router	mx	a3-mx80-1
10.84.25.30	Control Node	contrail	b5s30
10.84.25.29	Control Node	contrail	b5s29
10.84.25.28	Control Node	contrail	b5s28
10.84.25.27	Control Node	contrail	b5s27
10.84.11.253	BGP Router	mx	mx1

2. (Optional) The global AS number is 64512 by default. To change the AS number, on the **BGP Router** summary screen click the gear wheel and select **Edit**. In the Edit BGP Router window enter the new number.
3. To create control nodes and BGP routers, on the **BGP Routers** summary screen, click the

   icon. The **Create BGP Router** window is displayed; see Figure 4 on page 33.

Figure 4: Create BGP Router



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

   For each node you want to add, populate the fields with values for your system. See Table 2 on page 34.

Table 2: Create BGP Router Fields

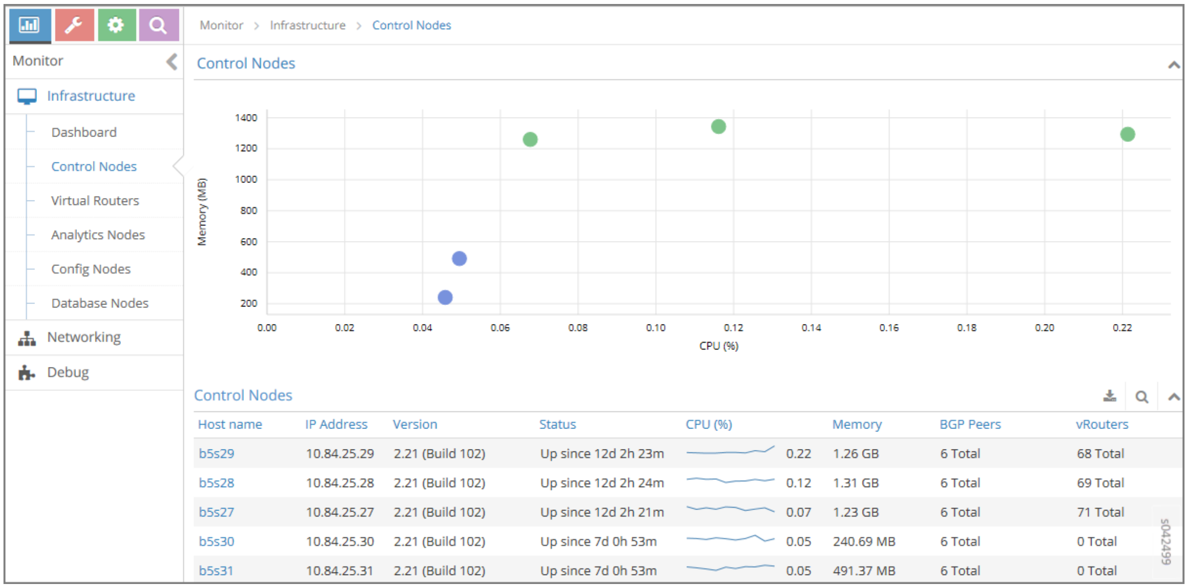
Field	Description
Hostname	Enter a name for the node being added.
Vendor ID	Required for external peers. Populate with a text identifier, for example, “MX-0”. (BGP peer only)
IP Address	The IP address of the node.
Router ID	Enter the router ID.
Autonomous System	Enter the AS number for the node. (BGP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

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

   Click **<<** to remove a node from the **Configured Peers** box.

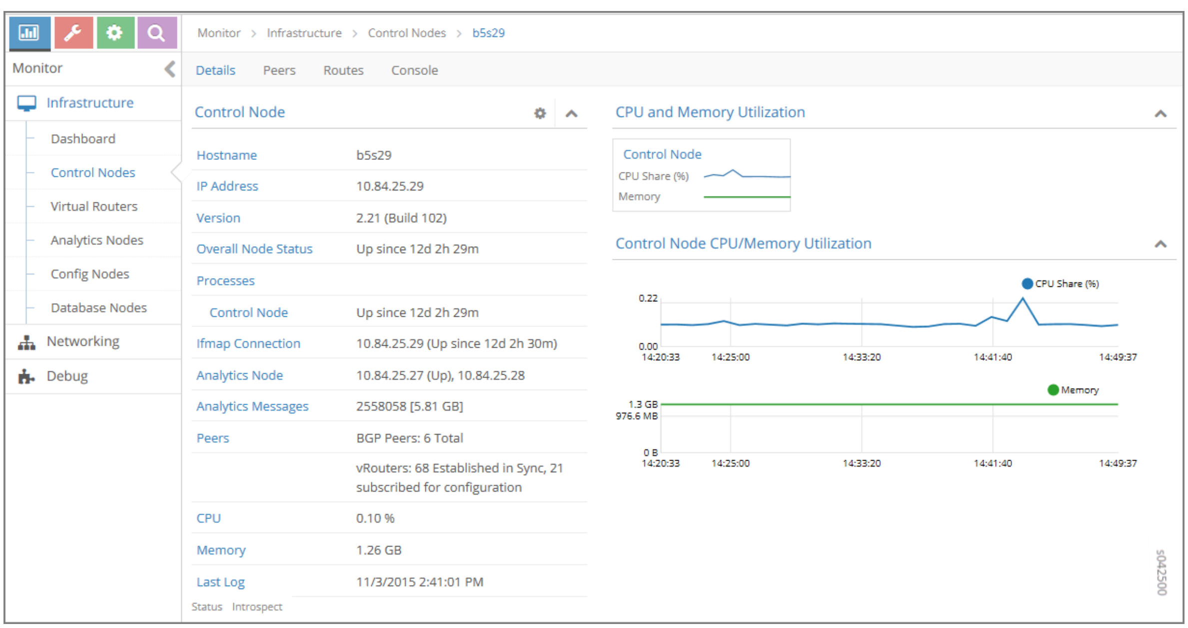
7. You can check for peers by selecting **Monitor > Infrastructure > Control Nodes**; see Figure 5 on page 35.

Figure 5: Control Nodes



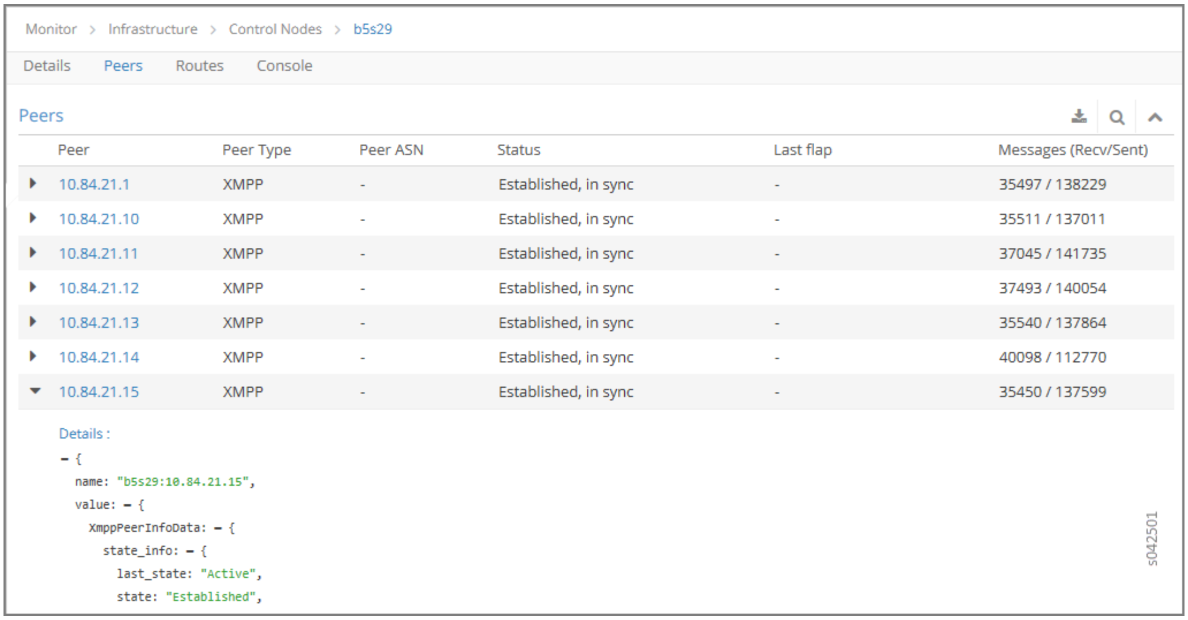
In the **Control Nodes** window, click any hostname in the memory map to view its details; see Figure 6 on page 35.

Figure 6: Control Node Details



8. Click the **Peers** tab to view the peers of a control node; see Figure 7 on page 36.

Figure 7: Control Node Peers Tab



RELATED DOCUMENTATION

Creating a Virtual Network with Juniper Networks Contrail | 190
Creating a Virtual Network with OpenStack Contrail | 194

Adding a New Node to an Existing Containerized Contrail Cluster

IN THIS SECTION

- Controller Configuration | 37

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

Controller Configuration

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

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

     See examples on github at:

     contrail-docker/tools/python-contrailctl/examples/configs/controller.conf

   - Start the controller container. For more information, see *How to run Contrail Docker containers*.

   - Wait for the new containers to come up completely.

2. Configure the existing cluster nodes with new nodes.

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

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

Using contrailctl to add node configuration on existing containers

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

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

1. Determine which node is the leader node.

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

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

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

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

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

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

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

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

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

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

Manually configure contrailctl on all containers and sync the configs

1. Determine which node is the leader node.

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

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

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

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

3. Run contrailctl within the containers.

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

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

Removing Nodes in an Existing Containerized Cluster

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

Using contrailctl to Configure Services Within Containers

IN THIS SECTION

- What is contrailctl? | 39
- Command Operations | 40

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

What is contrailctl?

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

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

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

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

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

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

Sample container configuration files can be seen at

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

Command Operations

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

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

Example:

docker exec controller contrailctl config sync -c controller -Fv

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

Command Syntax and Options

```
$ contrailctl config sync -h
usage: contrailctl config sync [-h] [-f CONFIG_FILE] -c

{controller,analyticsdb,analytics,agent,lb,kubemanager,mesosmanager}
                              [-F] [-t TAGS]

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

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

Updating and Syncing Service Configurations Within the Container

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

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

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

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

Syntax and Usage: config sync

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

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

Example: config sync

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

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

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

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

RELATED DOCUMENTATION

Introduction to Containerized Contrail Modules | 14

Supporting Multiple Interfaces on Servers and Nodes

IN THIS SECTION

- Support for Multiple Interfaces | 42
- Server Interface Examples | 44
- Interface Naming and Configuration Management | 45

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

Support for Multiple Interfaces

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

Examples of control traffic include the following:

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

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

Number of cfgm Nodes Supported

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

Uneven Number of Database Nodes Required

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

Support for VLAN Interfaces

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

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

Support for Bonding Options

Contrail provides support for bond interface options.

The default bond interface options are:

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

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

```
“network”: {
     “interfaces”: [
```

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

Support for Static Route Options

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

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

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

Server Interface Examples

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

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

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

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

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

Interface Naming and Configuration Management

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

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

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

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

Contrail Global Controller

IN THIS SECTION

- Resource Identifier Management | 46
- Multiple Location Resource Provisioning | 46

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

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

Resource Identifier Management

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

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

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

Multiple Location Resource Provisioning

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

Requirements, Assumptions, and Constraints

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

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

Platform Support

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

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

Installation

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

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

Role and Resource-Based Access Control

IN THIS SECTION

- Contrail Role and Resource-Based Access (RBAC) Overview | 47
- API-Level Access Control | 48
- Object Level Access Control | 49
- Configuration | 49
- Utilities | 52
- Upgrading from Previous Releases | 53
- Configuring RBAC Using the Contrail User Interface | 54
- RBAC Resources | 57

Contrail Role and Resource-Based Access (RBAC) Overview

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

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

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

API-Level Access Control

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

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

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

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

Where:

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

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

`role` The Keystone role name.

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

Example: ACL RBAC Object

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

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

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

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

Rule Sets and ACL Objects

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

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

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

Object Level Access Control

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

The `perms2` property has the following fields:

`owner` This field is populated at the time of creation with the tenant UUID value extracted from the token.

`share list` The share list gets built when the object is selected for sharing with other users. It is a list of tuples with which the object is shared.

The `permission` field has the following options:

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

Access is allowed as follows:

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

Configuration

This section describes the parameters used in Contrail RBAC.

Parameter: aaa-mode

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

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

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

> **NOTE**: The `multi_tenancy` parameter is deprecated, starting with Contrail 3.0. The parameter should be removed from the configuration. Instead, use the `aaa_mode` parameter for RBAC to take effect.
>
> If the `multi_tenancy` parameter is not removed, the `aaa-mode` setting is ignored.

Parameter: cloud_admin_role

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

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

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

Configuration Files with Cloud Admin Credentials

The following configuration files contain `cloud_admin_role` credentials:

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

Changing Cloud Admin Configuration Files

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

1. Change the configuration files with the new information.
2. Restart the following:
   - API server

     `service supervisor-config restart`
   - Neutron server

     `service neutron-server restart`
   - WebUI

     `service supervisor-webui restart`

Global Read-Only Role

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

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

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

Setting the Global Read-Only Role

To set the global read-only role:

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

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

   `global_read_only_role = <new-admin-read-role>`
2. Restart the `contrail-api` service:

   `service contrail-api restart`

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

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

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

Utilities

This section describes the utilities available for Contrail RBAC.

Utility: rbacutil.py

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

Read RBAC rule-set using UUID or FQN

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

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

Create RBAC rule-set using FQN domain/project

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

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

Delete RBAC group using FQN or UUID

To delete an RBAC group using FQN or UUID:

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

Add rule to existing RBAC group

To add a rule to an existing RBAC group:

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

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

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

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

Utility: chmod2.py

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

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

Upgrading from Previous Releases

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

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

Configuring RBAC Using the Contrail User Interface

To use the Contrail UI with RBAC:

1. Set the aaa_mode to no_auth.

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

   ```
   aaa_mode = no-auth
   ```

2. Restart the `analytics-api` service.

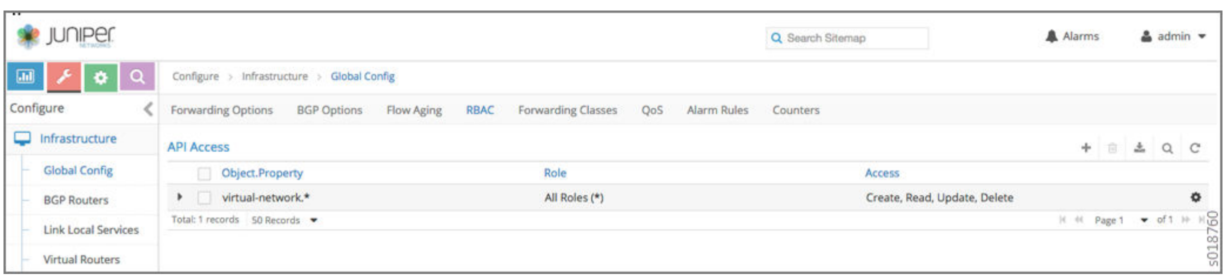
   ```
   service contrail-analytics-api restart
   ```

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

Configuring RBAC at the Global Level

To configure RBAC at the global level, navigate to **Configure** > **Infrastructure** > **Global Config** > **RBAC**, see Figure 8 on page 54.

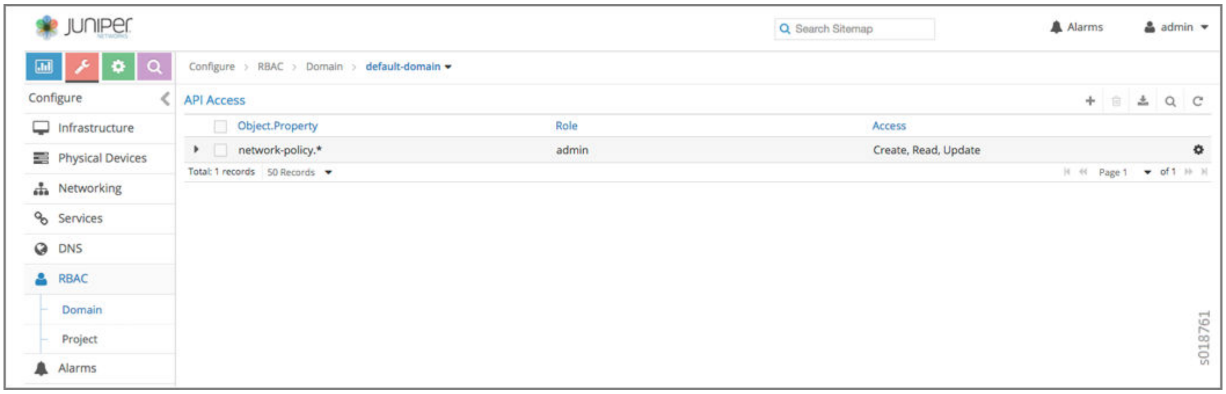
Figure 8: RBAC Global Level



Configuring RBAC at the Domain Level

To configure RBAC at the domain level, navigate to **Configure** > **RBAC** > **Domain**, see Figure 9 on page 55.

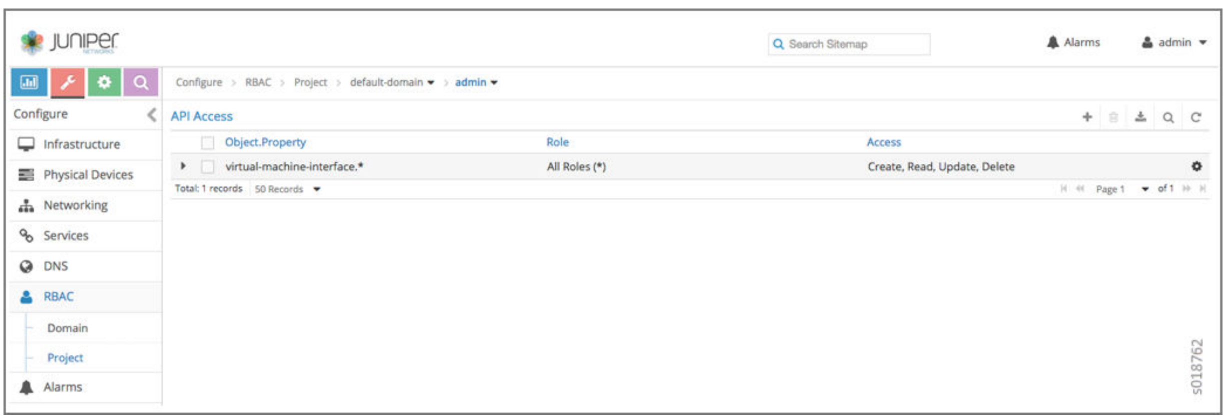
Figure 9: RBAC Domain Level



Configuring RBAC at the Project Level

To configure RBAC at the project level, navigate to **Configure > RBAC > Project**, see Figure 10 on page 55.

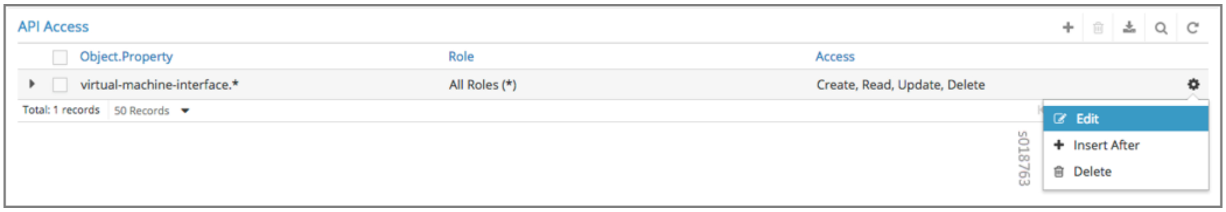
Figure 10: RBAC Project Level



Configuring RBAC Details

Configuring RBAC is similar at all of the levels. To add or edit an API access list, navigate to the global, domain, or project page, then click the plus (+) icon to add a list, or click the gear icon to select from Edit, Insert After, or Delete, see Figure 11 on page 56.

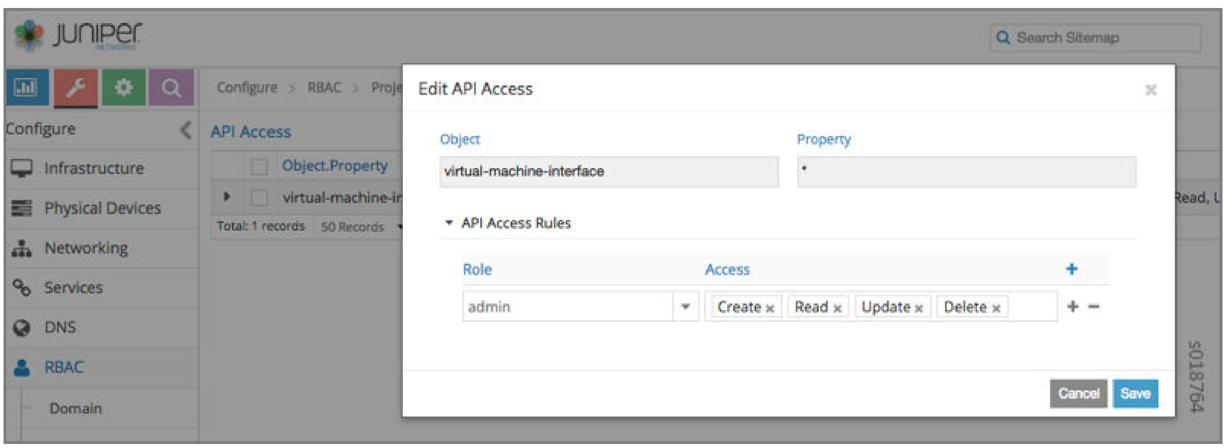
Figure 11: RBAC Details API Access



Creating or Editing API Level Access

Clicking create, edit, or insert after activates the Edit API Access popup window, where you enter the details for the API Access Rules. Enter the user type in the Role field, and use the + icon in the Access filed to enter the types of access allowed for the role, including, Create, Read, Update, Delete, and so on, see Figure 12 on page 56.

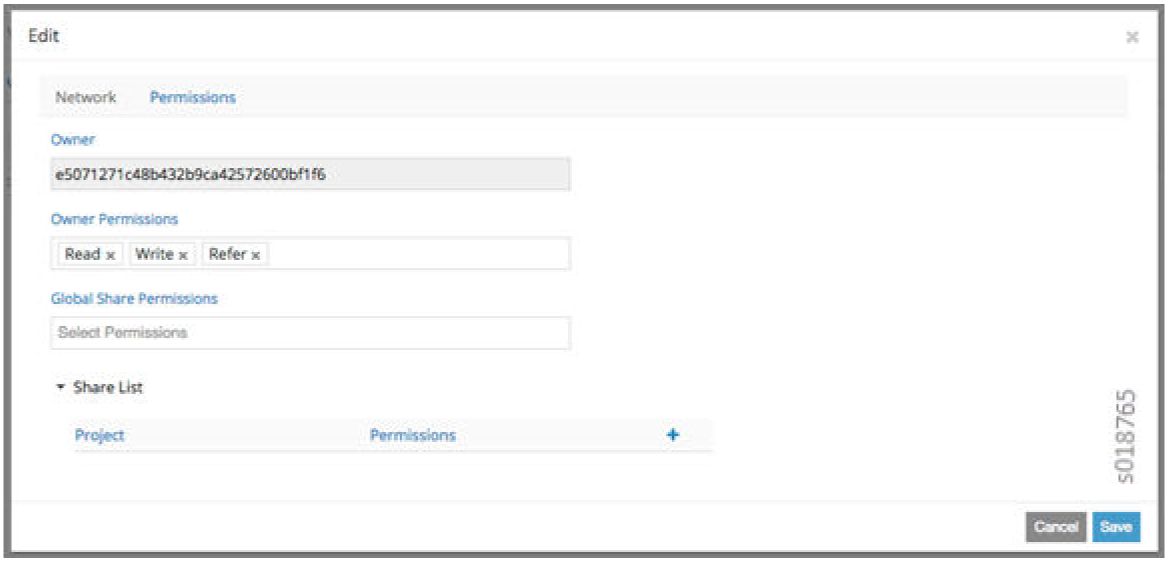
Figure 12: Edit API Access



Creating or Editing Object Level Access

You can configure fine-grained access control by resource. A **Permissions** tab is available on all create or edit popups for resources. Use the **Permissions** popup to configure owner permissions and global share permissions. You can also share the resource to other tenants by configuring it in the **Share List**, see Figure 13 on page 57.

Figure 13: Edit Object Level Access



RBAC Resources

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

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

CHAPTER 4

Installation and Configuration Scenarios

IN THIS CHAPTER

- Setting Up and Using a Simple Virtual Gateway with Contrail 4.0 | 58
- Simple Underlay Connectivity without Gateway | 68

Setting Up and Using a Simple Virtual Gateway with Contrail 4.0

IN THIS SECTION

- Introduction to the Simple Gateway | 58
- How the Simple Gateway Works | 59
- Setup Without Simple Gateway | 59
- Setup With a Simple Gateway | 60
- Simple Gateway Configuration Features | 61
- Packet Flows with the Simple Gateway | 62
- Packet Flow Process From the Virtual Network to the Public Network | 63
- Packet Flow Process From the Public Network to the Virtual Network | 63
- Methods for Configuring the Simple Gateway | 64
- Using the vRouter Configuration File to Configure the Simple Gateway | 64
- Using Thrift Messages to Dynamically Configure the Simple Gateway | 64
- Common Issues with Simple Gateway Configuration | 67

Introduction to the Simple Gateway

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

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

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

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

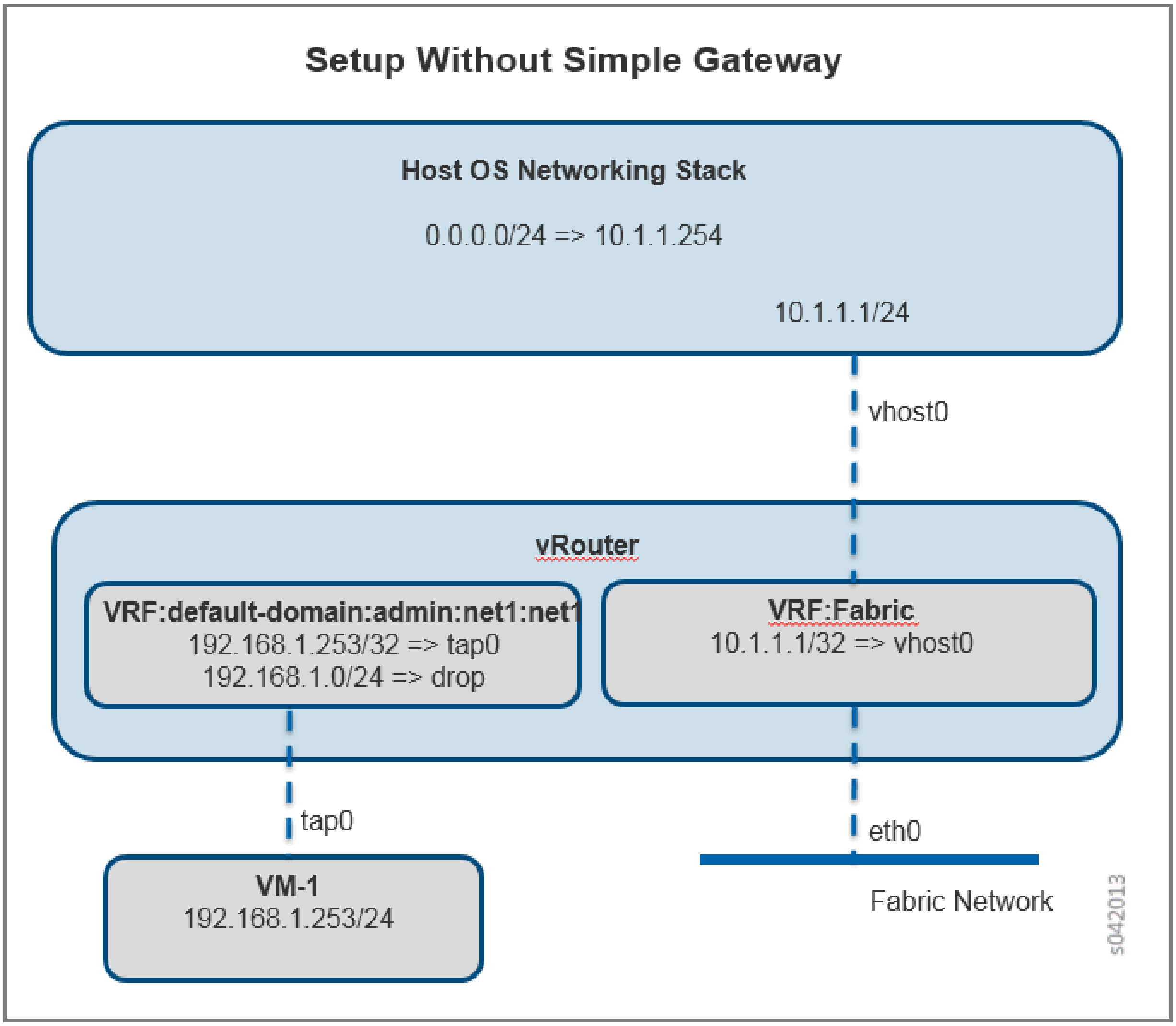
How the Simple Gateway Works

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

Setup Without Simple Gateway

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

- A virtual network, `default-domain:admin:net1`, is configured with the subnet 192.168.1.0/24.
- The routing instance `default-domain:admin:net1:net1` is associated with the `default-domain:admin:net1` virtual network .
- A virtual machine with the 192.168.1.253 IP address is spawned in net1.
- A virtual machine is spawned on compute server 1.
- An interface, `vhost0`, is in the host OS of server 1 and is assigned the 10.1.1.1/24 IP address.
- The `vhost0` interface is added to the vRouter in the routing instance fabric.
- The simple gateway is not configured.



Setup With a Simple Gateway

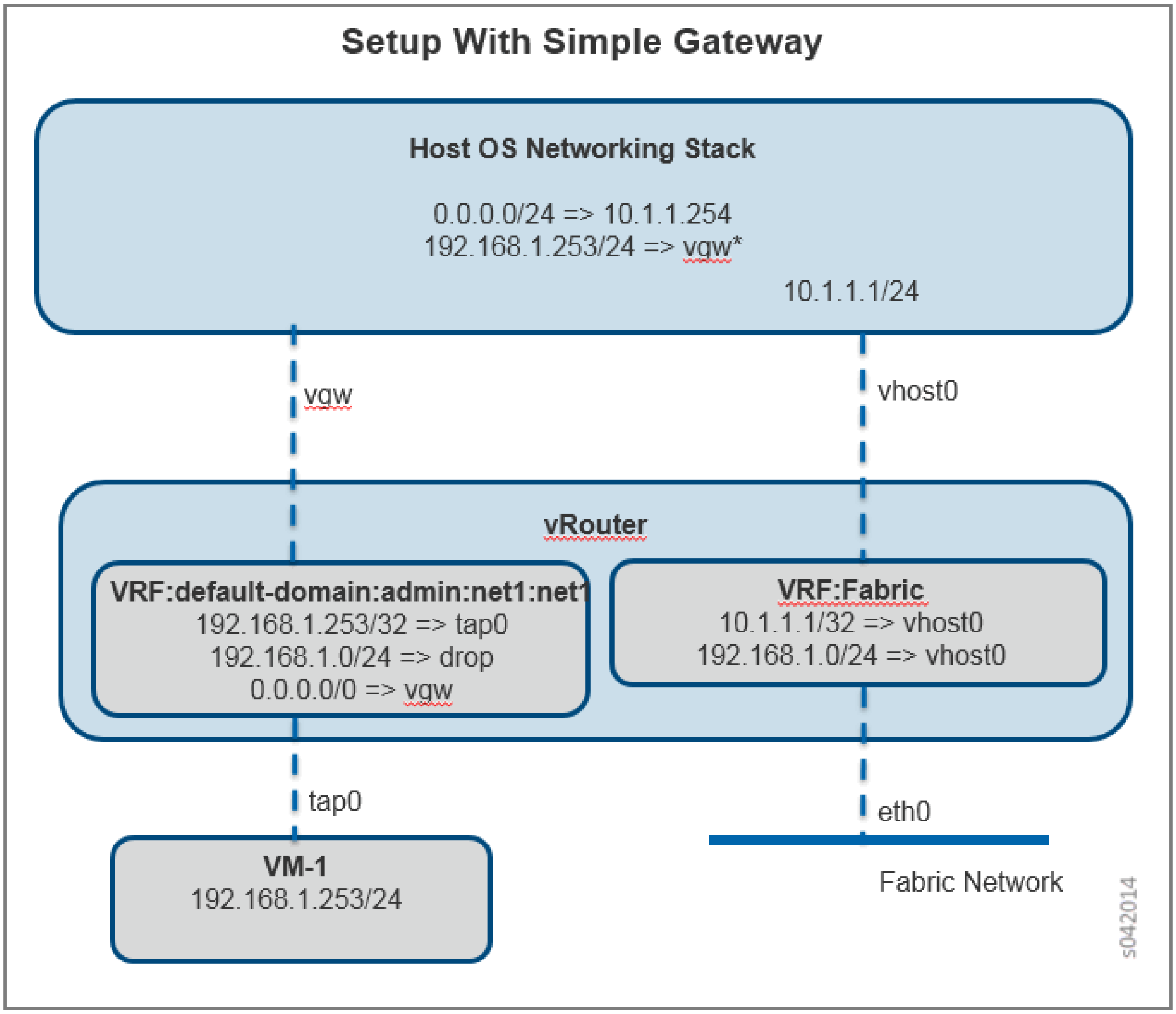
Figure 14 on page 61 shows a virtual network setup with the simple gateway configured for the `default-domain:admin:net1` virtual network.

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

Figure 14 on page 61 shows the packet flows between the fabric VRF and the `default-domain:admin:net1` virtual network.

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

Figure 14: Virtual Network Setup With a Simple Gateway



Simple Gateway Configuration Features

The simple gateway configuration has the following features.

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

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

Simple Gateway Restrictions

The following are restrictions of the simple gateway:

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

Packet Flows with the Simple Gateway

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

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

Packet Flow Process From the Virtual Network to the Public Network

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

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

Packet Flow Process From the Public Network to the Virtual Network

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

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

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

Methods for Configuring the Simple Gateway

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

Using the vRouter Configuration File to Configure the Simple Gateway

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

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

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

Using Thrift Messages to Dynamically Configure the Simple Gateway

IN THIS SECTION

- How to Dynamically Create a Virtual Gateway | 65
- How to Dynamically Delete a Virtual Gateway | 66
- Using Devstack to Configure the Simple Gateway | 67

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

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

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

How to Dynamically Create a Virtual Gateway

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

When run, the script does the following:

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

Example: Dynamically Create a Virtual Gateway

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

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

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

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

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

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

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

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

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

How to Dynamically Delete a Virtual Gateway

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

When run, the script does the following:

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

Example: Dynamically Create a Virtual Gateway

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

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

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

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

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

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

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

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

Using Devstack to Configure the Simple Gateway

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

The following parameters are available:

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

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

Example: Devstack Configuration for Simple Gateway

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

```
CONTRAIL_VGW_INTERFACE=vgw1

CONTRAIL_VGW_PUBLIC_SUBNET=192.168.1.0/24

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

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

Common Issues with Simple Gateway Configuration

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

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

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

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

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

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

Simple Underlay Connectivity without Gateway

IN THIS SECTION

- Simple Routing of Packets Without a Gateway | 68
- Supported Use Cases | 69
- Implementation: Routing Instances | 69
- Implementation | 71

Simple Routing of Packets Without a Gateway

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

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

The following features can be enabled when using this method:

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

Supported Use Cases

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

The following use cases for no SDN gateway are supported.

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

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

- Virtual networks with IP fabric VN as their provider network, communicating with other VNs that do not have any provider network based on policy configured, using overlay with tunneling.

- Vhost communication , with other compute vhosts and with VMs and containers in the IP fabric network or other VNs with IP fabric network as the provider network based on policy configured, using underlay and no tunneling.

- Vhost communication with VMs in any virtual network based on policy configuration, using overlay with tunneling.

Implementation: Routing Instances

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

- A default routing instance, `ip-fabric:default`, which is used for all forwarding decisions by the data path.

- A new routing instance, `ip-fabric:ip-fabric`, to carry L3VPN routes for endpoints in IP fabric. Network policy and security groups are applied based on these entries.

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

IPAM Addressing Schemes

Two IPAM addressing schemes are supported for IP fabric:

- Common subnet mode with a set of subnet prefixes.

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

Operation

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

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

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

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

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

ARP Handling

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

- VM-to-VM communication, on the same compute or on different compute nodes— Respective vrouters respond to ARP requests from the VMs with the vrouter's MAC. Agent resolves the ARP for other compute nodes to fill the next hop corresponding to remote VMs.

- Vhost connectivity to VM on the same compute node—Vrouter responds with vhost MAC (its own MAC) for ARP requests from vhost. ARP requests from the VM will be responded with vrouter's MAC.

Each subnet in the networks, IP fabric network or other VNs using IP fabric as the provider network, has a subnet route in the compute host pointing to the vhost interface. There is a Layer 3 route in the fabric default VRF for each VM, with the next hop pointing to its VMI. Traffic is forwarded to the VM based on this route. The next hop is a Layer 3 interface next hop with the source MAC being the vrouter's MAC.

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

- Vhost connectivity to VM on a different compute node—ARP requests for VMs on a different compute node are flooded on the fabric interface. The compute node hosting the VM has a Layer 3 route for the VM, with the next hop pointing to its VMI. The vrouter on that node responds to the ARP request with its vhost MAC address. The VM's ARP request is always responded to by with vrouter's MAC.

- Vhost connectivity to another compute node—As in the previous example, the ARP request is transmitted on the fabric interface. Other vrouters cross connect the ARP request to their vhost interface because there is not any Layer 3 route pointing to the VMI. The host responds to the ARP request.

Broadcast and Multicast Traffic

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

Implementation

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

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

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

CHAPTER 5

Using Server Manager to Automate Provisioning

IN THIS CHAPTER

- Installing Server Manager | 72
- Using Server Manager to Automate Provisioning | 79
- Using the Server Manager Web User Interface | 113
- Installing and Using Server Manager Lite | 137

Installing Server Manager

IN THIS SECTION

- Installation Requirements for Server Manager | 72
- Installing Server Manager | 74
- Upgrading Server Manager Software | 75
- Server Manager Installation Completion Checks | 76
- Sample Configurations for Server Manager Templates | 77

Installation Requirements for Server Manager

This document provides details for installing Server Manager.

Platform Support

As of Contrail 4.0, Server Manager can be installed on, and used to reimage and provision, the following platform operating systems:

- Ubuntu 16.04.01

- Ubuntu 16.04.02
- Ubuntu 14.04.5
- Ubuntu 14.04.4
- Ubuntu 14.04.2
- Ubuntu 14.04.1
- Ubuntu 14.04

As of Contrail 4.0, Server Manager installation supports Contrail provisioning for only the following OpenStack versions:

- Ocata, on Ubuntu 16.04 platform, only
- Newton, on Ubuntu 16.04 platform, only
- Mitaka
- Liberty
- Kilo, on Contrail networking only

Installation Prerequisites

Before installing Server Manager ensure the following prerequisites are met.

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

  > **NOTE**: Server Manager is tested with only the following versions of dependent packages: Ansible 2.2.0.0, Docker 1.13.0, Puppet 3.7.3-1, and Cobbler 2.6.3-1. These tested versions are installed during Server Manager installation.

- Puppet Master requires the fully-qualified domain name (FQDN) of the Server Manager for key generation. The domain name is taken from the `/etc/hosts` file. If the server is part of multiple domains, specify the domain name by using the `--domain` option during the installation.
- On multi-interface systems, specify the interface on which Server Manager needs to listen by using the `--hostip` option during installation. If the listening interface is not specified, the first available interface from the `ifconfig` list is used.
- The system administrator might need to configure the Linux kernel security module AppArmor to allow `server-manager` access.

Installing Server Manager

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

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

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

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

1. Install the Server Manager packages:

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

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

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

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

   - To set up all components:

     `./setup.sh --all`

   - To set up only the Server Manager server:

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

   - To set up only the Server Manager client:

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

   - To set up only the Server Manager user interface:

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

   - To set up only Server Manager monitoring:

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

   Other options include:

   - `--sm-cliff-client`
   - `--nowebui`

- `--nosm-mon`

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

Finishing the Installation

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

1. `/etc/cobbler/dhcp.template`
2. `/etc/cobbler/named.template`
3. `/etc/bind/named.conf.options`
4. `/etc/cobbler/settings`
5. `/etc/cobbler/modules.conf`
6. `/etc/mail/sendmail.cf`

Starting the Server Manager Service

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

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

Upgrading Server Manager Software

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

Steps for Upgrading

Use the following steps to upgrade your Server Manager installation.

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

1. `dpkg -i` *<contrail-server-manager-installer*deb>*

2. `cd /opt/contrail/contrail_server_manager`

3. `./setup.sh -all`

4. After the setup script has completed running, you can restart Server Manager by issuing:

   `service contrail-server-manager restart`

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

Server Manager Installation Completion Checks

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

Server Manager Checks

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

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

  `service contrail-server-manager status`

  `service cobblerd status`

  `cobbler sync`

  `service bind9 status`

  `service isc-dhcp-server status`

  `service apache2 status`

  `service docker status`

- Also verify processes using the following command:

  `ps auwx | grep Passenger`

Server Manager Client Checks

- Verify the items listed:

  `which server-manager`

- Check the client configuration at `/etc/contrail/sm-client-config.ini`

- Make sure that `listen_ip_addr` is configured with the correct Server Manager IP address.

Server Manager WebUI Checks

- Verify the status of the Server Manager WebUI:

  ```
  service supervisor-webui-sm status
  ```

- Check the webui access from the browser:
  - Contrail release 4.0 and greater—`http:`*<server manager ip>* `:9143`
  - Contrail releases 3.0, 3.1, and 3.2—`http:`*<server manager ip>* `:9080`
  - Contrail release 2.2 and lower—`http:`*<server manager ip>* `:8080`

Sample Configurations for Server Manager Templates

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

Sample Settings

```
bind_master: 10.XX.11.6

manage_forward_zones: ['contrail.juniper.net']

manage_reverse_zones: ['10.XX.11']

next_server: 10.XX.11.6

server: 10.XX.11.6
```

Sample dhcp.template File

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

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

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

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

Sample named.conf.options File

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

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

You can also configure the following parameter:

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

Sample named.template File

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

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

The sendmail.cf File

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

RELATED DOCUMENTATION

Using Server Manager to Automate Provisioning | 79
Using the Server Manager Web User Interface | 113
Installing and Using Server Manager Lite

Using Server Manager to Automate Provisioning

IN THIS SECTION

- Overview of Server Manager | 79
- Server Manager Requirements and Assumptions | 80
- Server Manager Component Interactions | 81
- Configuring Server Manager | 82
- Configuring the Cobbler DHCP Template | 84
- User-Defined Tags for Server Manager | 85
- Server Manager Client Configuration File | 85
- Restart Services | 86
- Accessing Server Manager | 86
- Communicating with the Server Manager Client | 87
- Server Manager Commands for Configuring Servers | 88
- Server Manager REST API Calls | 105
- Example: Reimaging and Provisioning a Server | 111

Overview of Server Manager

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

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

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

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

The Server Manager:

- Provides REST APIs to handle customer requests.
- Manages its own database to store information about the servers.

- Interacts with other open source products such as Cobbler, Puppet, and Ansible to configure servers based on user requests.

Server Manager Requirements and Assumptions

The following are requirements and assumptions for the Server Manager:

- The Server Manager runs on a Linux server (bare metal or virtual machine) and assumes availability of several software products with which it interacts to provide the functionality of managing servers.
- The Server Manager has network connectivity to the servers it is trying to manage.
- The Server Manager has access to a remote power management tool to power cycle the servers that it manages.
- The Server Manager uses Cobbler software for Linux provisioning to configure and download software to physical servers. Cobbler resides on the same server that is running the Server Manager daemon.
  - Server Manager assumes that DNS and DHCP servers embedded with Cobbler provide IP addresses and names to the servers being managed, although it is possible to use external DNS and DHCP servers.
- The Server Manager uses Puppet software, an open source configuration management tool, to accomplish the configuration management of target servers, including the installation and configuration of different software packages and the launching of various services.
- Starting with Contrail Release 4.0, Server Manager uses Ansible software, an open source configuration management tool primarily used to automate the configuration and provisioning of Contrail components inside containers.
- The Server Manager also uses Docker to load and move these Contrail containers to the target servers. The Server Manager maintains a local registry on the Server Manager machine and users also have an option to use an external registry from which they can copy their Contrail Docker images directly onto the target servers.
- SQLite3 database management software is used to maintain and manage server configurations and it runs on the same machine where the Server Manager daemon is running.
- Because the server-manager process listens on port 9001, and the server-manager webui listens on ports 9080 and 9143, the firewall must be enabled for those ports.
- Server Manager needs a minimum of 4GB of RAM, 2 CPU cores, and 80GB of disks (to support multiple Contrail installations).
- Server Manager assumes that SSH is enabled on target nodes.

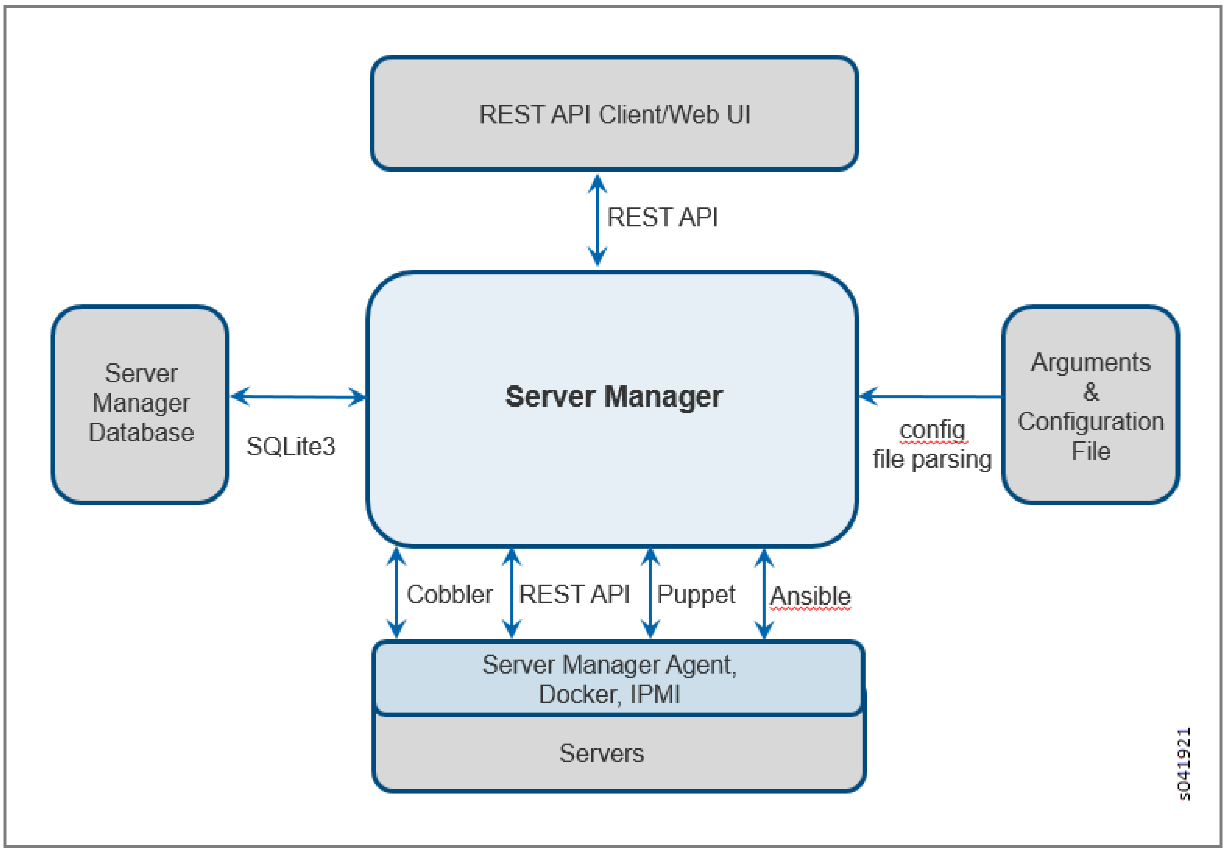
Server Manager Component Interactions

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

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

Figure 15 on page 81 illustrates several high-level components with which the Server Manager interacts.

Figure 15: Server Manager Component Interactions



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

While managing the servers, the Server Manager also communicates with other software components. It uses Cobbler for reimaging target servers, Docker to host Contrail containers, and Ansible and Puppet for provisioning, thereby ensuring necessary software packages are installed and configured, required services are running, and so on.

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

Configuring Server Manager

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

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

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

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

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

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

```
[SERVER-MANAGER]

listen_ip_addr = <SM-IP-address>

listen_port    = <port-number>


cobbler_ip_address = <cobbler-ip-address>

cobbler_port   = <cobbler-port-number>

cobbler_username = <cobbler-username>

cobbler_password = <cobbler-password>


ipmi_username = <IPMI username>

ipmi_password = <IPMI password>
```

```
ipmi_type = <IPMI type>
```

Table 3 on page 83 provides details for each of the parameters in the SERVER-MANAGER section.

Table 3: Server Manager Parameters

Parameter	Configuration
listen_ip_addr	Specify the IP address of the server on which the Server Manager is listening for REST API requests.
listen_port	The port number on which the Server Manager is listening for REST API requests. The default is 9001.
cobbler_ip_address	The IP address used to access Cobbler. This address MUST be the same address as the listen_ip_address. The Server Manager assumes that the Cobbler service is running on the same server as the Server Manager service.
cobbler_port	The port on which Cobbler listens for user requests. Leave this field blank.
cobbler_username	Specify the user name to access the Cobbler service. Specify testing unless your Cobbler settings have been modified to use a different user name.
cobbler_password	Specify the password to access the Cobbler service. Specify testing unless your Cobbler settings have been modified to use a different password.
ipmi_username	The IPMI username for power management.
ipmi_password	The IPMI password for power management.
ipmi_type	The IPMI type (ipmilan, lanplus, or other Cobbler-supported types).

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

4.0 Docker containers. The following example shows the format and parameters of the ANSIBLE-SERVER section:

```
[ANSIBLE-SERVER]

docker_insecure_registries = <IP address:Port>

docker_registry = <IP address:Port>

ansible_srvr_ip = <IP address>

ansible_srvr_port = <Port>

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

Table 4 on page 84 provides details for each of the parameters in the ANSIBLE-SERVER section.

Table 4: Ansible Server Parameters

Parameter	Configuration
docker_insecure_registries	Specify the IP address and port of the server on which the insecure Docker registry used by the Server Manager resides
docker_registry	Specify the IP address and port of the server on which the Docker registry used by the Server Manager resides
ansible_srvr_ip	Specify the IP address of the Server Manager machine on which the Ansible daemon will run
ansible_srvr_port	Specify the port on the Server Manager machine on which the Ansible daemon will run
ansible_log_path	Specify the log path where the Ansible daemon stores its log messages

Configuring the Cobbler DHCP Template

In addition to configuring the sm_config.ini file, you must manually change the settings in the/etc/cobbler/dhcp.template file to use the correct subnet address, mask, and DNS domain name for your environment.

Optionally, you can also restrict the use of the current instance of Server Manager and Cobbler to a subset of servers in the network.

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

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

Sample dhcp.template

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

User-Defined Tags for Server Manager

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

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

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

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

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

Server Manager Client Configuration File

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

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

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

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

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

Restart Services

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

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

Accessing Server Manager

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

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

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

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

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

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

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

Communicating with the Server Manager Client

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

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

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

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

Server Manager Commands for Configuring Servers

IN THIS SECTION

- Server Manager Commands Common Options | 88
- Add New Servers or Update Existing Servers | 89
- Delete Servers | 90
- Display Server Configuration | 91
- Server Manager Commands for Managing Clusters | 92
- Server Manager Commands for Managing Tags | 94
- Server Manager Commands for Managing Images | 96
- Server Manager Operational Commands for Managing Servers | 100
- Reimaging Server(s) | 100
- Provisioning and Configuring Roles on Servers | 102
- Restarting Server(s) | 103
- Show Status of Server(s) | 104
- Show Status of Provision | 105

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

Server Manager Commands Common Options

The common options in Table 5 on page 89 are available with every Server Manager command.

Table 5: Common Command Options

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

Add New Servers or Update Existing Servers

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

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

Table 6 on page 89 lists the optional arguments.

Table 6: Server Manager Add Server Command Options

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

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

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

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

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

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

Delete Servers

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

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

Table 7 on page 90 lists the optional arguments.

Table 7: Server Manager Delete Server Command Options

Option	Description
`--server_id SERVER_ID`	The server ID for the server or servers to be deleted.
`--mac MAC`	The MAC address for the server or servers to be deleted.
`--ip IP`	The IP address for the server or servers to be deleted.
`--cluster_id CLUSTER_ID`	The cluster ID for the server or servers to be deleted.
`--tag TagName=TagValue`	The TagName that is to be matched with the Tagvalue. Up to seven TagName and Tagvalue pairs separated by commas can be provided.

The criteria for identifying servers to be deleted can be specified by providing the **server_id** or the server: `mac address, ip, cluster_id`, or the `TagName = TagValue`.

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

Display Server Configuration

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

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

Table 8 on page 91 lists the optional arguments.

Table 8: Server Manager Display Server Command Options

Option	Description
`--server_id SERVER_ID`	The server ID for the server or servers to be deleted.
`--mac MAC`	The MAC address for the server or servers to be displayed.
`--ip IP`	The IP address for the server or servers to be displayed.
`--cluster_id CLUSTER_ID`	The cluster ID for the server or servers to be displayed.
`--tag TagName=TagValue`	The TagName that is to be matched with the Tagvalue. Up to seven TagName and Tagvalue pairs separated by commas can be provided.
`--detail, -d`	Flag to indicate if details are requested.

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

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

Server Manager Commands for Managing Clusters

IN THIS SECTION

- Create a New Cluster or Update an Existing Cluster | 92
- Delete a Cluster | 93
- Display Cluster Configuration | 93

A cluster is used to store parameter values that are common to all servers belonging to that cluster. The commands in this section facilitate managing clusters in the Server Manager database, enabling you to add, modify, delete, and view clusters.

NOTE: Whenever a server is created with a specific `cluster_id`, Server Manager checks to see if a cluster with that ID has already been created. If there is no matching `cluster_id` already in the database, an error is returned.

Create a New Cluster or Update an Existing Cluster

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

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

Table 9 on page 92 lists the optional arguments.

Table 9: Server Manager Add Cluster Command Options

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

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

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

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

Delete a Cluster

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

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

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

Table 10 on page 93 lists the optional arguments.

Table 10: Server Manager Delete Cluster Command Options

Option	Description
`--cluster_id CLUSTER_ID`	The cluster ID for the server or servers to be displayed.

Display Cluster Configuration

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

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

Table 11 on page 93 lists the optional arguments.

Table 11: Server Manager Display Cluster Command Options

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

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

Server Manager Commands for Managing Tags

IN THIS SECTION

- Create a New Tag or Update an Existing Tag | 94
- Display Tag Configuration | 95

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

Create a New Tag or Update an Existing Tag

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

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

Table 12 on page 94 lists the optional arguments.

Table 12: Server Manager Add New Tag

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

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

```
{
```

```
    "tag1" : "data-center",

    "tag2" : "floor",

    "tag3" : "",

    "tag4" : "pod",

    "tag5" : "rack",

}
```

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

Display Tag Configuration

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

```
server-manager display tag
```

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

```
{

    "tag1": "datacenter",

    "tag2": "floor",

    "tag3": "hall",

    "tag4": "rack",

    "tag5": "user_tag"

}
```

Server Manager Commands for Managing Images

IN THIS SECTION

- Creating New Images or Updating Existing Images | 97
- Add an Image | 97
- Upload an Image | 98
- Delete an Image | 99
- Display Image Configuration | 100

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

Table 13 on page 96 lists the image types supported.

Table 13: Server Manager Image Types

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

Creating New Images or Updating Existing Images

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

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

- Use `add` when the new image is present locally on the Server Manager machine. The path provided is the image path on the Server Manager machine.
- Use `upload_image` when the new image is present on the machine where the client program is being invoked. The path provided is the image path on the client machine.

Add an Image

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

Table 14 on page 97 lists the optional arguments.

Table 14: Server Manager Add Image

Option	Description
`--file_name FILE_NAME, -f FILE_NAME`	The name of the JSON file that contains the image parameter values.

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

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

```
{

    "image": [

        {
```

```
            "id": "ubuntu-<x.xx.x>",

            "type": "ubuntu",

            "version": "ubuntu-<x.xx.x>",

            "path": "/iso/ubuntu-<x.xx.x>-server-amd64.iso"

        },

        {

            "id": "centos-<x.xx>",

            "type": "centos",

            "version": "centos-<x.xx>",

            "path": "/iso/CentOS-<x.xx>-x86_64-minimal.iso"

        },

        {

            "id": "contrail-ubuntu-<x.xx>",

            "type": "contrail-ubuntu-package",

            "version": "contrail-ubuntu-<x.xx>",

            "path": "/iso/contrail-cloud-docker_<x.xx-xx>_all.deb"


        }

    ]

}
```

Upload an Image

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

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

Table 15 on page 99 lists the optional arguments.

Table 15: Server Manager Upload Image

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

Delete an Image

Use the `server-manager delete` command to delete an image from the Server Manager database. When an image is deleted from the Server Manager database, the corresponding distribution, profile, or repository for the image is also deleted from the Cobbler database.

```
server-manager delete  image --image_id <image_id>
```

Table 16 on page 99 lists the optional arguments.

Table 16: Server Manager Delete Image

Option	Description
`image_id`	The image ID for the image to be deleted.

Display Image Configuration

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

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

Table 17 on page 100 lists the optional arguments.

Table 17: Server Manager Display Image Configuration

Option	Description
image_id	The image ID for the image or images.
--detail, -d	Flag to indicate if details are requested.

Server Manager Operational Commands for Managing Servers

The Server Manager commands in the following sections are operational commands for performing a specific operation on a server or a group of servers. These commands assume that the base configuration of entities required to execute the operational commands is already completed using configuration CLI commands.

Reimaging Server(s)

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

Before issuing the reimage command, the images must be added to the Server Manager, which also adds the images to Cobbler. The set of servers to be reimaged can be specified by providing match criteria for servers already added to the Server Manager database, using server_id.

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

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

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

            [--no_reboot]

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

            [--no_confirm]
             base_image_id
```

Options include the following:

Table 18 on page 101 lists the optional arguments.

Table 18: Server Manager Reimage

Option	Description
`base_image_id`	The image ID of the base image to be used.
`--package_image_id PACKAGE_IMAGE_ID, -p PACKAGE_IMAGE_ID`	The optional Contrail package to be used to reimage the server or servers.
`--no_reboot, -n`	Optional parameter to indicate that the server should not be rebooted following the reimage setup.
`--server_id SERVER_ID`	The server ID for the server or servers to be reimaged.
`--cluster_id CLUSTER_ID`	The cluster ID for the server or servers to be reimaged.
`--tag TagName=TagValue`	TagName which is to be matched with Tagvalue
`--no_confirm, -F`	Flag to bypass confirmation message, default = do NOT bypass.

Provisioning and Configuring Roles on Servers

Use the `server-manager provision` command to provision identified server(s) with configured roles for the virtual network system. The servers can be selected from the database configuration (using standard server match criteria), identified in a JSON file, or provided interactively.

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

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

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

Options include the following:

Table 19 on page 102 lists the optional arguments.

Table 19: Server Manager Provision

Option	Description
`package_image_id`	The Contrail package image ID to be used for provisioning.
`--server_id SERVER_ID`	The server ID for the server or servers to be provisioned.
`--cluster_id CLUSTER_ID`	The cluster ID for the server or servers to be provisioned.
`--tag TagName=TagValue`	TagName to be matched with Tagvalue.
`--provision_params_file PROVISION_PARAMS_FILE, -f PROVISION_PARAMS_FILE`	Optional JSON file containing the parameters for provisioning the server(s).
--no_confirm, -F	Flag to bypass confirmation message, default = do NOT bypass.

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

Restarting Server(s)

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

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

             [--net_boot]

             [--no_confirm]
```

Table 20 on page 103 lists the optional arguments.

Table 20: Server Manager Restart

Option	Description
`--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.

Table 20: Server Manager Restart *(Continued)*

Option	Description
--no_confirm, -F	Flag to bypass confirmation message, default = do NOT bypass.

Show Status of Server(s)

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

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

Table 21 on page 104 lists the optional arguments.

Table 21: Server Manager Status Server

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

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

Status outputs include the following:

1. `restart_issued`

   `reimage_started`

   `provision_issued`

   `provision_completed`

   `openstack_started`

```
openstack_completed
```

Show Status of Provision

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

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

Table 22 on page 105 lists the optional arguments.

Table 22: Server Manager Status Provision

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

Server Manager REST API Calls

IN THIS SECTION

- REST APIs for Server Manager Configuration Database Entries | 106
- API: Add a Server | 106
- API: Delete Servers | 106
- API: Retrieve Server Configuration | 107
- API: Add an Image | 107
- API: Upload an Image | 108
- API: Get Image Information | 108
- API: Delete an Image | 108

- API: Add or Modify a Cluster | **109**
- API: Delete a Cluster | **109**
- API: Get Cluster Configuration | **109**
- API: Get All Server Manager Configurations | **110**
- API: Reimage Servers | **110**
- API: Provision Servers | **110**
- API: Restart Servers | **111**

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

REST APIs for Server Manager Configuration Database Entries

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

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

API: Add a Server

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

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

Method: `PUT`

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

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

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

API: Delete Servers

Use one of the following formats to delete a server.

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

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

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

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

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

Method : DELETE

Payload : None

API: Retrieve Server Configuration

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

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

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

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

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

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

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

Method : GET

Payload : None

API: Add an Image

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

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

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

Method: PUT

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

See sample payload in the following:

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

API: Upload an Image

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

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

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

Method: `PUT`

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

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

API: Get Image Information

Use the following to get image information.

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

Method: `GET`

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

API: Delete an Image

Use the following to delete an image.

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

Method: `DELETE`

Payload: Specifies criteria for the image being deleted.

API: Add or Modify a Cluster

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

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

Method: `PUT`

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

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

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

API: Delete a Cluster

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

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

Method: `DELETE`

Payload: None

API: Get Cluster Configuration

Use this API to get a cluster configuration.

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

Method: `GET`

Payload: None

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

API: Get All Server Manager Configurations

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

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

Method: GET

Payload: None

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

API: Reimage Servers

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

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

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

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

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

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

Method: POST

Payload: None

API: Provision Servers

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

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

Method: POST

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

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

API: Restart Servers

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

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

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

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

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

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

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

`http://`*<SM-IP-Address>*: *<SM-Port>*`/server/restart ? [netboot&]tag=<tag_name>=<tag_value>`

Method: `POST`

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

Example: Reimaging and Provisioning a Server

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

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

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

```
[SERVER-MANAGER]

listen_ip_addr = 192.168.1.10 (Server Manager IP address)

listen_port = 9001
```

Overview

The steps to be followed include:

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

Procedure

1. Configure a cluster.

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

2. Configure the server.

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

3. Configure images.

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

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

4. Reimage servers.

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

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

5. Provision servers.

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

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

SEE ALSO

Installing Server Manager | 72

Using the Server Manager Web User Interface | 113

Installing and Using Server Manager Lite

Using the Server Manager Web User Interface

IN THIS SECTION

- Log In to Server Manager | 113
- Create a Cluster for Server Manager | 114
- Edit a Cluster through Edit JSON | 125
- Working with Servers in the Server Manager User Interface | 125
- Add a Server | 126
- Edit Tags for Servers | 129
- Using the Edit Config Option for Multiple Servers | 129
- Edit a Server through Server Manager, Edit JSON | 130
- Filter Servers by Tag | 131
- Viewing Server Details | 131
- Configuring Images and Packages | 134
- Add New Image or Package | 135
- Selecting Server Manager Actions for Clusters | 135
- Reimage a Cluster | 136
- Provision a Cluster | 136

When the Server Manager is installed on your Contrail system, you can also install a Server Manager Web user interface that you can use to access the features of Server Manager.

Log In to Server Manager

The Server Manager user interface can be accessed using:

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

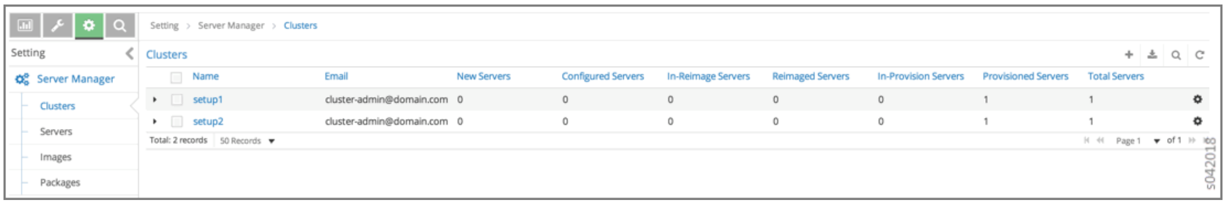
Where *<server-manager-user-interface-ip>* is the IP address of the server on which the Server Manager web user interface is installed.

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

Create a Cluster for Server Manager

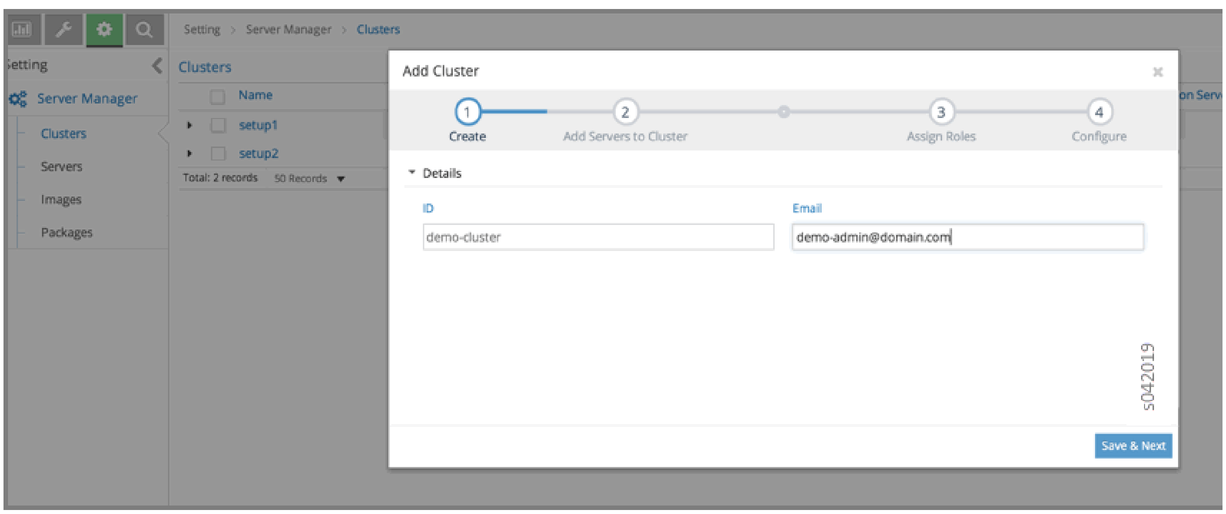
Select **Add Cluster** to identify a cluster to be managed by the Server Manager. Select **Setting > Server Manager > Clusters**, to access the **Clusters** page, see Figure 16 on page 114.

Figure 16: Server Manager > Clusters



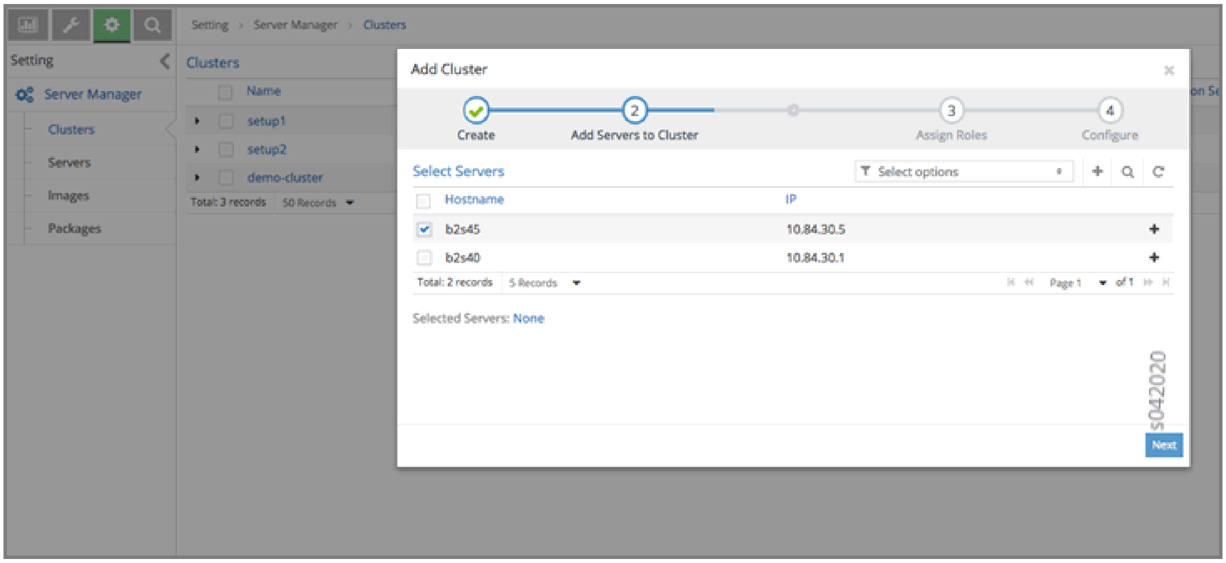
To create a new cluster, click the plus icon in the upper right of the **Clusters** page. The **Add Cluster** window is displayed. In the **Add Cluster** window, you can add a new cluster ID and the domain e-mail address of the cluster. See Figure 17 on page 114.

Figure 17: Add Cluster



When you are finished adding information about the new cluster in the **Add Clusters** window, click **Save & Next**. Now you can add servers to the cluster, see Figure 18 on page 115.

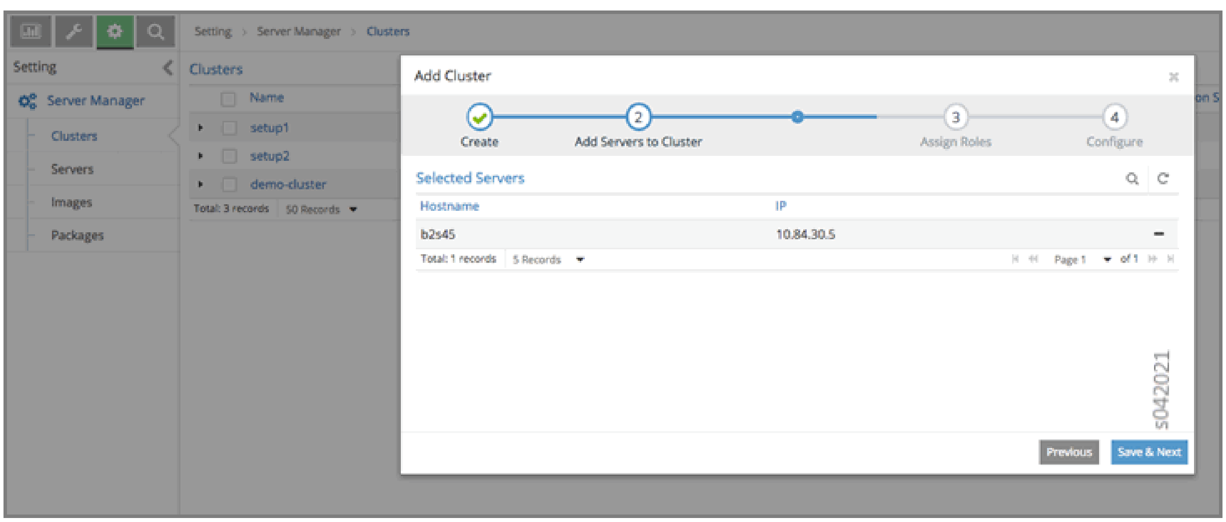
Figure 18: Add Servers to Cluster



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

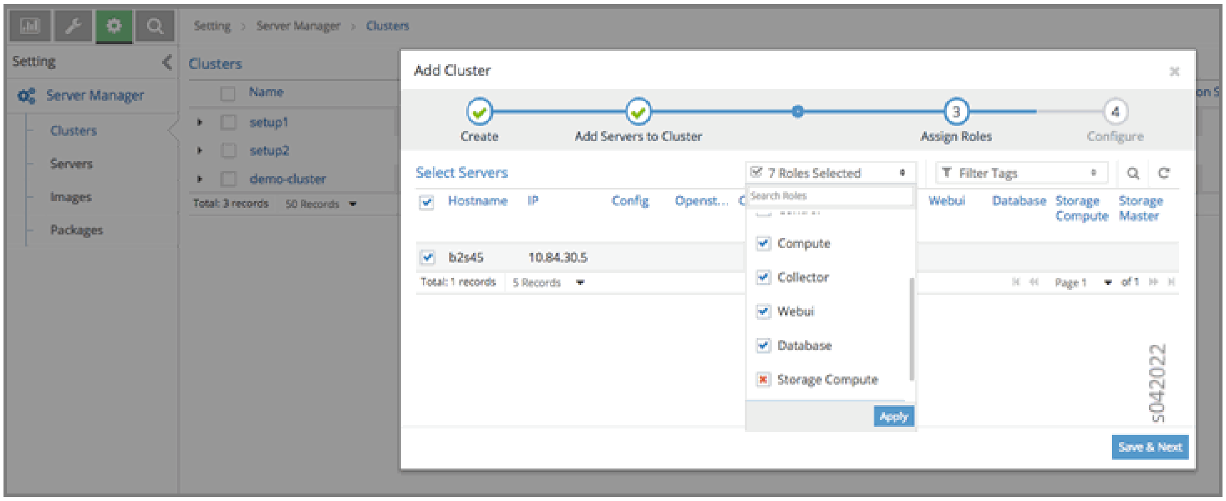
When you are finished, click **Next**. The selected servers are added to the cluster, see Figure 19 on page 115.

Figure 19: Add Servers to Cluster, Next



When you are finished adding servers, click **Save & Next**. Now you can assign Contrail roles to servers that you select in the cluster. Roles available are Config, OpenStack, Control, Compute, and Collector. Select each role assignment for the selected server. You can also unselect any assigned role. The assigned roles correspond to the role functions in operation on the server, see Figure 20 on page 116.

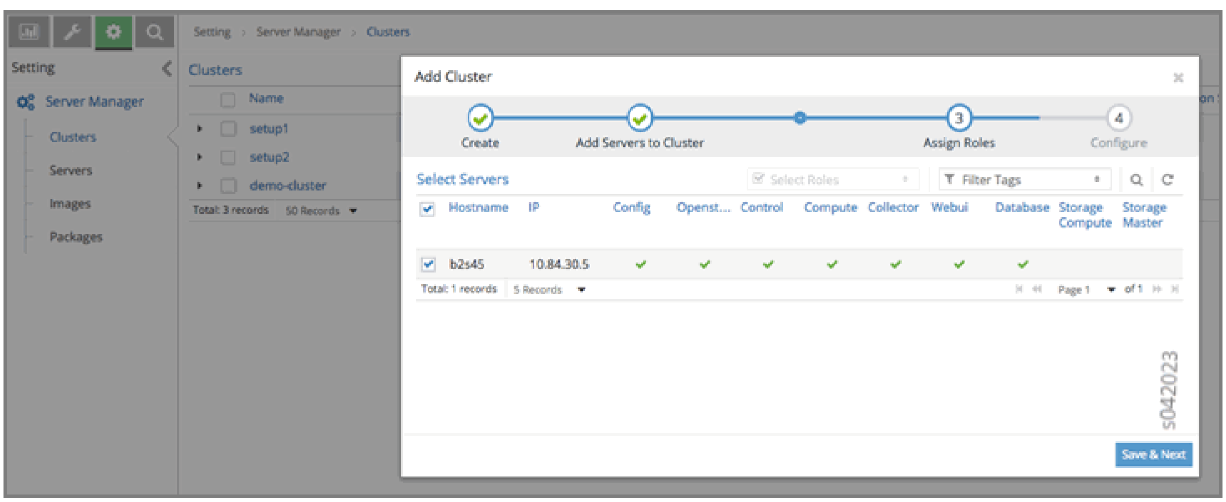
Figure 20: Assign Roles



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

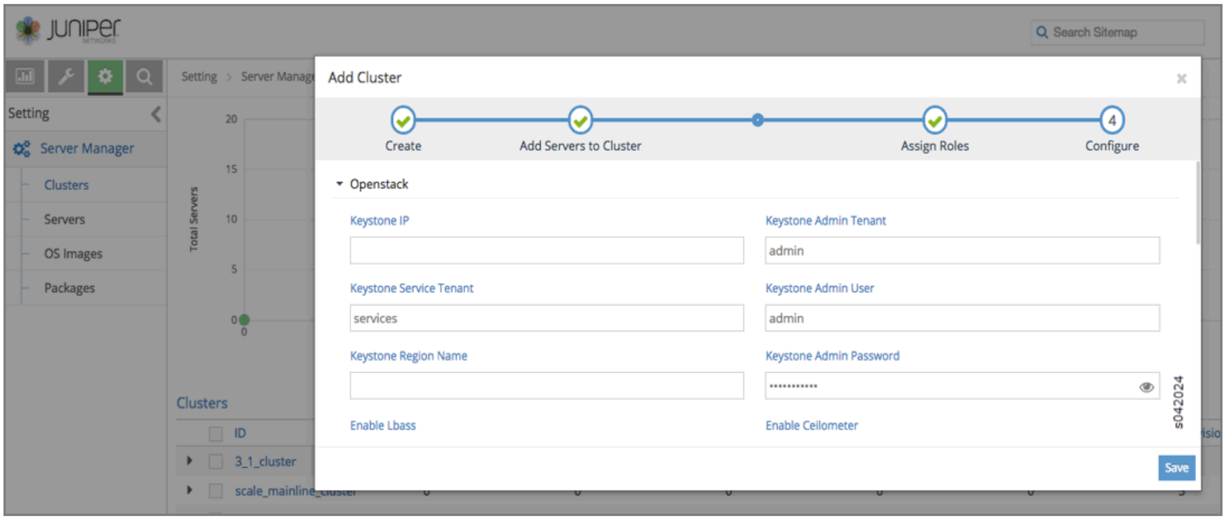
Click **Save & Next** to view your selections. Check marks are displayed in the columns of the **Add Cluster** window, see Figure 21 on page 116.

Figure 21: Roles Assigned



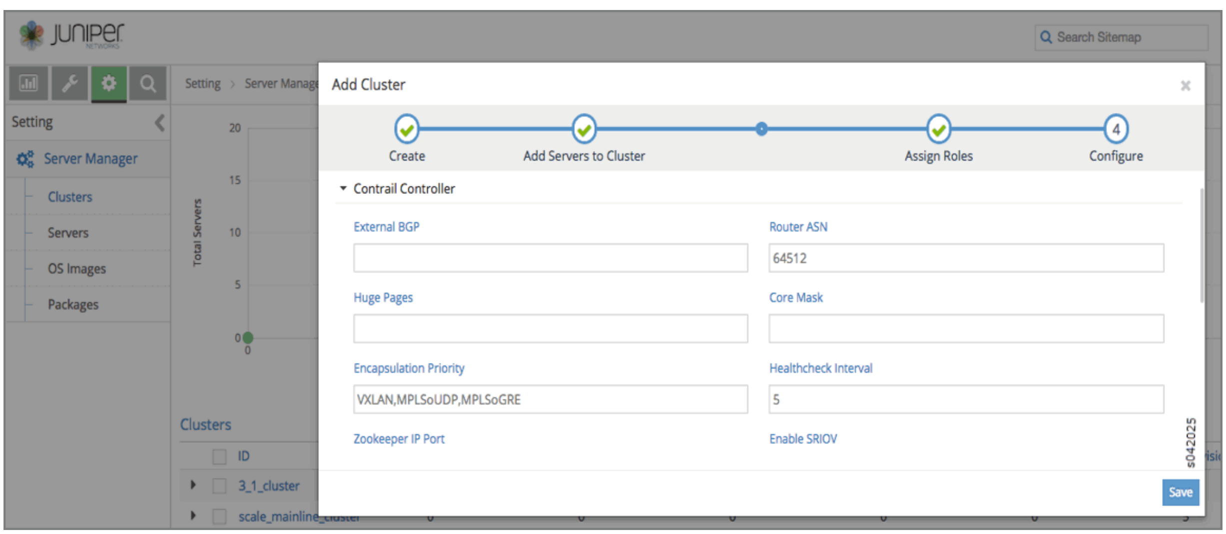
The next step after roles are assigned is to enter the cluster configuration information for OpenStack. After viewing the assigned roles, click **Save & Next**. The **Add Cluster** window is displayed. Click an icon that opens a set of fields where you can enter OpenStack or Contrail configuration information for the cluster. In the following image, the **Openstack** icon is selected. You can enter **Keystone** configuration information, such as IP, Admin tenant, user, and password, service tenant, and region name. You can also enable LBaaS and Ceilometer, see Figure 22 on page 117.

Figure 22: OpenStack Configuration



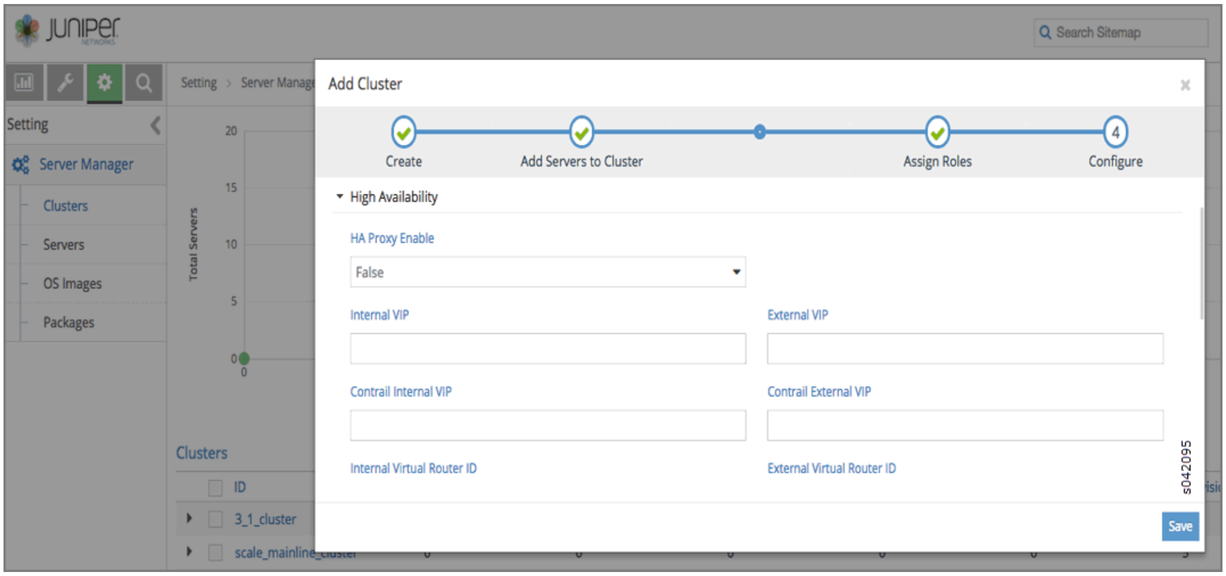
In the following image, the Contrail controller icon is selected. You can enter configuration information for Contrail, such as **External BGP, Router ASN, Huge Pages, Core Mask, Encapsulation Priority, Healthcheck Interval, Zookeeper IP Port, Enable SRIOV,** and so on, see Figure 23 on page 117.

Figure 23: Configure Contrail



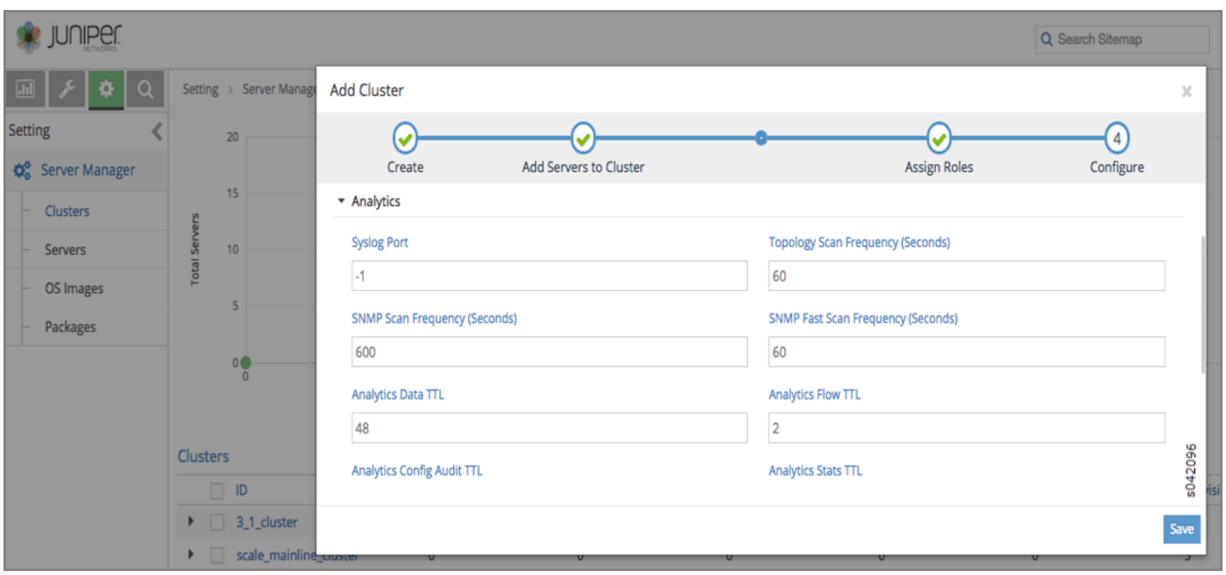
In the following image, the High Availability (HA) icon is selected. You can configure high availability parameters such as `HA Proxy Enable`, `Internal` and `External VIP`, and so on, see Figure 24 on page 118.

Figure 24: Configure High Availability



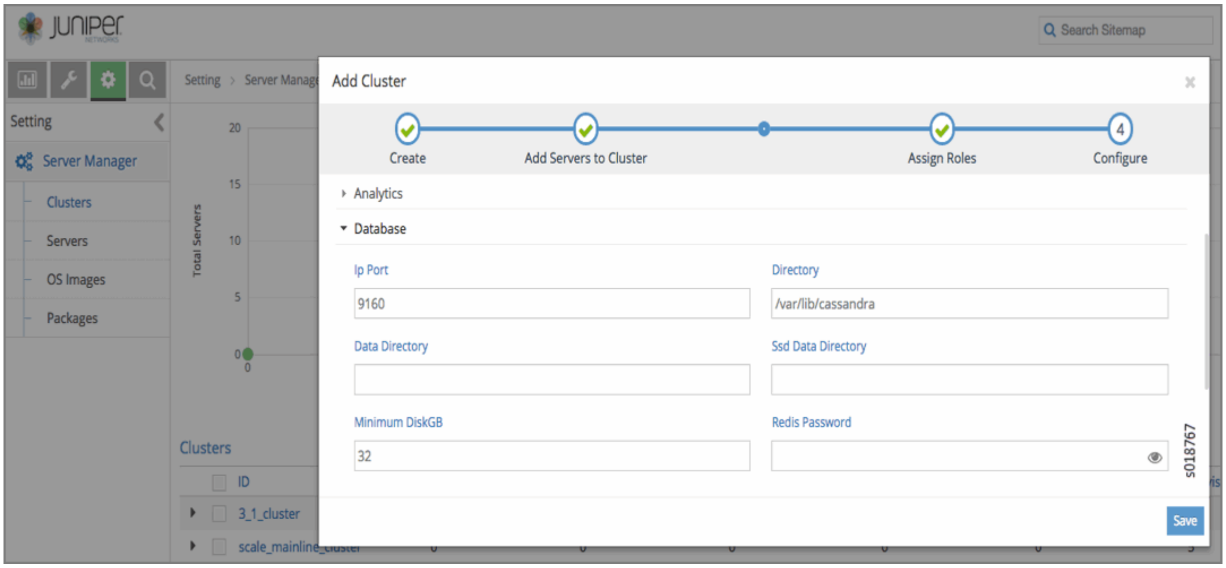
In the following image, the **Analytics** icon is selected. Here you can configure parameters for Contrail Analytics, including **Syslog Port**, various scan frequencies, and various TTL settings, see Figure 25 on page 118.

Figure 25: Configure Analytics



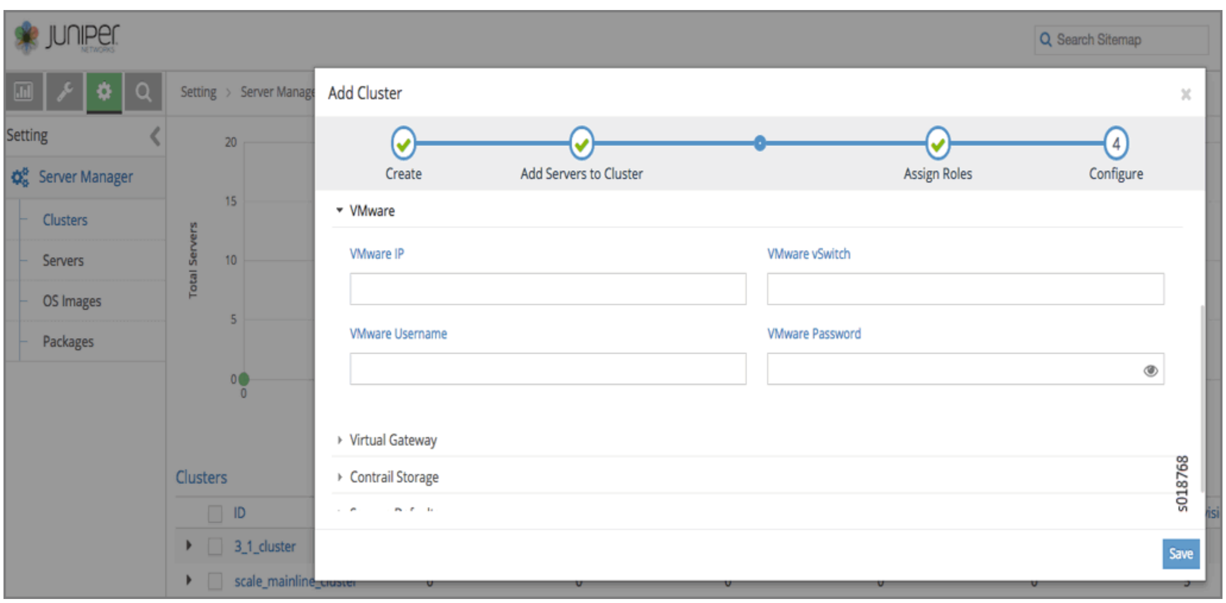
In following image, the **Database** icon is selected. You can configure parameters for the Contrail database, including **IP Port, Directory, Minimum Disk GB,** and so on, see Figure 26 on page 119.

Figure 26: Configure Database



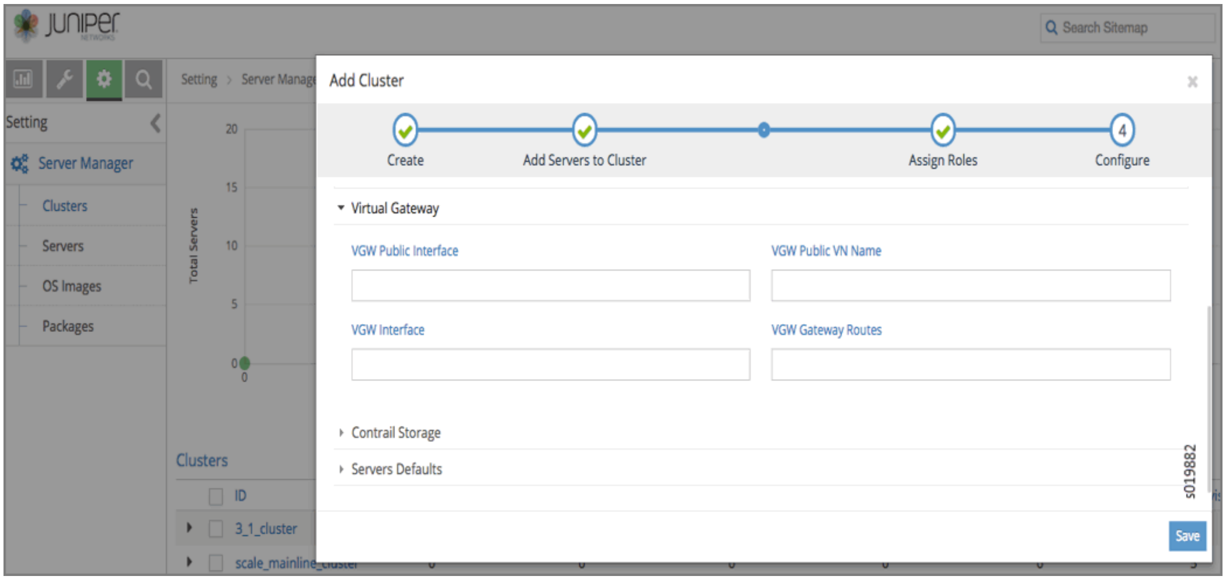
In following image, the **VMware** icon is selected. You can configure parameters for Contrail VMware , including **VMware IP, VMware vSwitch, Username, Password** , and so on, see Figure 27 on page 119.

Figure 27: Configure VMware



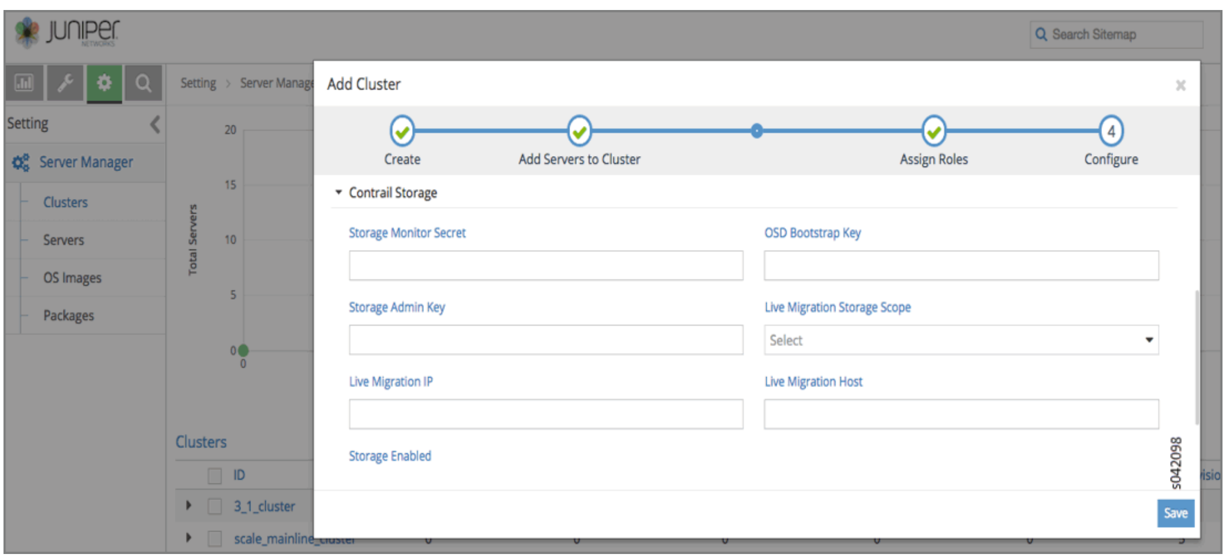
In following image, the **Virtual Gateway** icon is selected. You can configure parameters for the Contrail Virtual Gateway, including **VGW Public Interface, VGW Public VN Name, VGW Interface, Routes** , and so on, see Figure 28 on page 120.

Figure 28: Configure Virtual Gateway



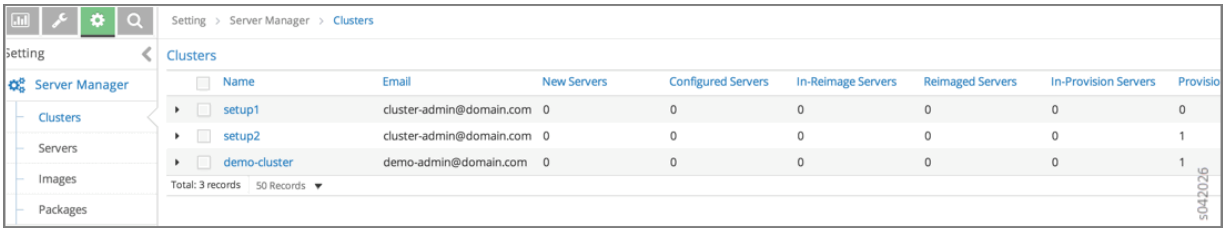
In following image, the **Contrail Storage** icon is selected. You can configure parameters for Contrail Storage, including **Storage Monitor Secret, OSD Bootstrap Key, Admin Key**, and so on, see Figure 29 on page 120.

Figure 29: Configure Contrail Storage



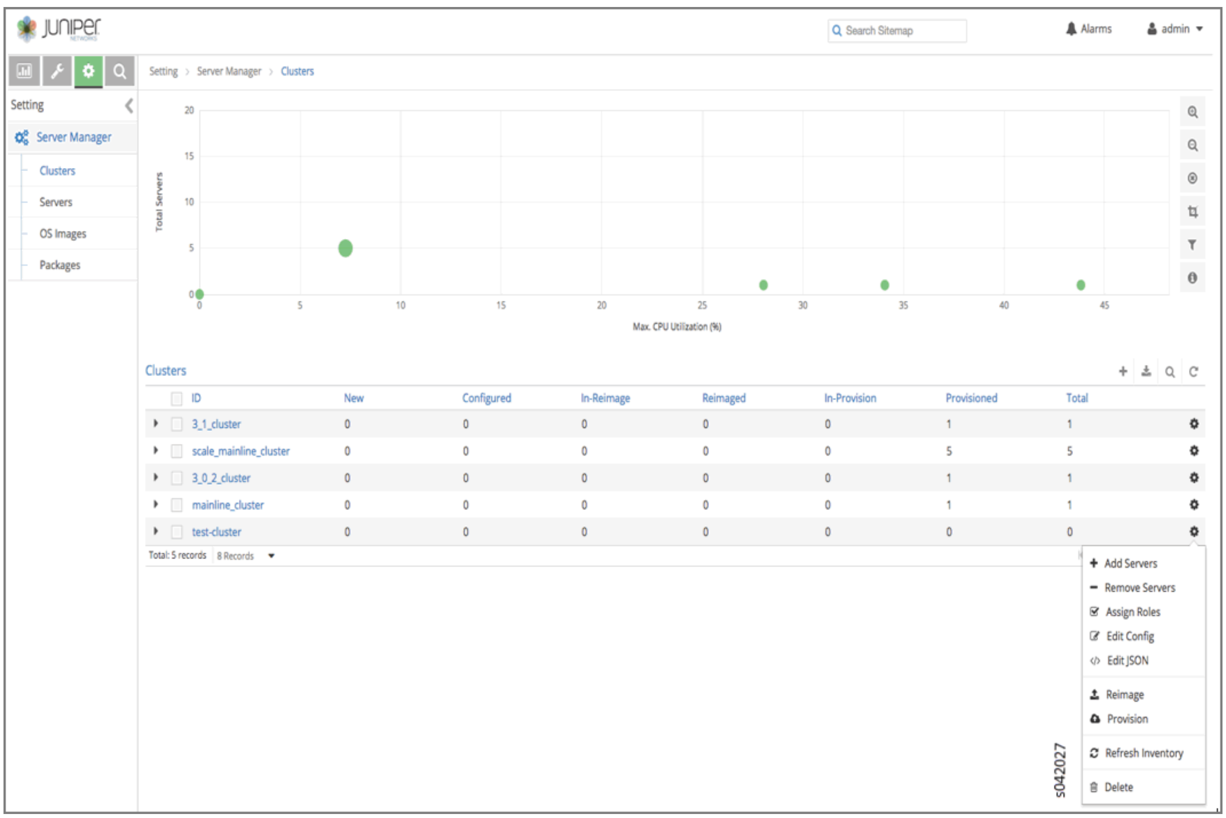
When you are finished entering all of the cluster configuration information, click **Save** to submit the configurations. You can view all configured clusters on the **Clusters** window by selecting **Setting > Server Manager > Clusters**, see Figure 30 on page 121.

Figure 30: View Configured Clusters



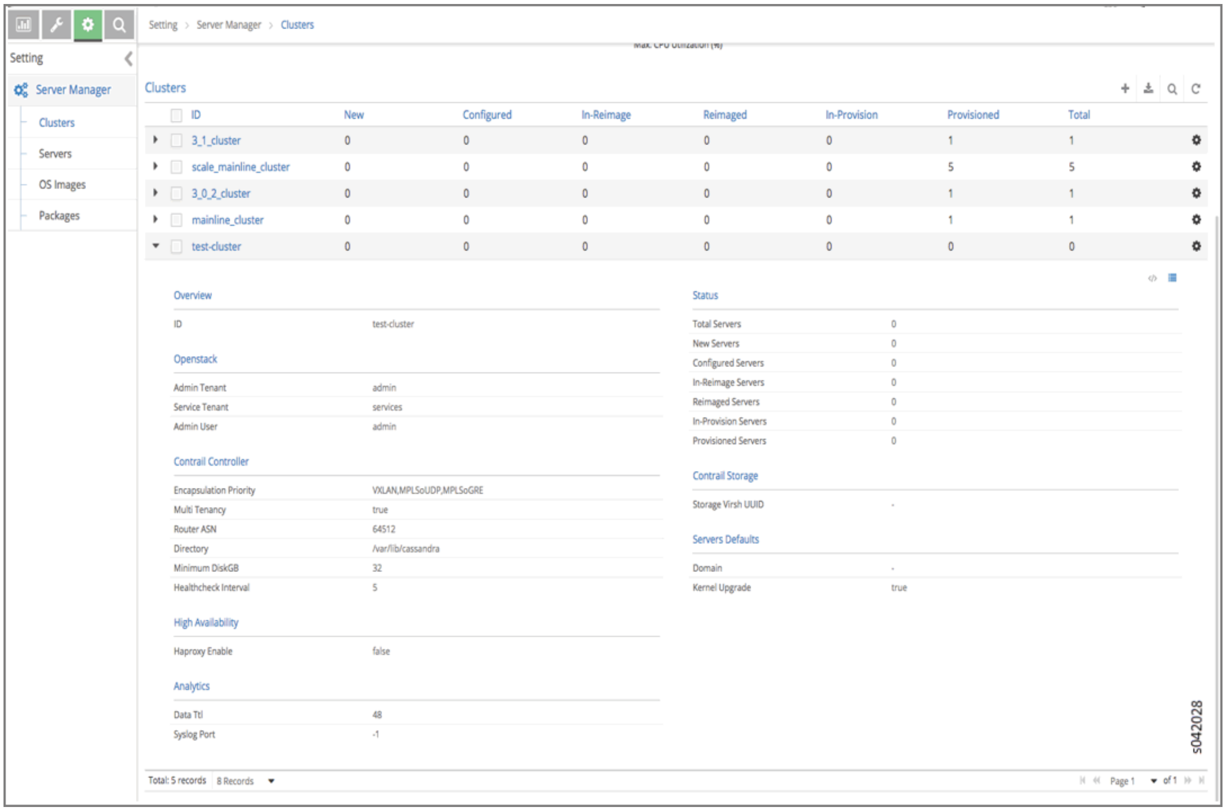
To perform an action on one of the configured clusters, click the gear wheel icon at the right to select from a menu of actions available for that cluster, including **Add Servers, Remove Servers, Assign Roles, Edit Config, Reimage, Provision**, and **Delete**, see Figure 31 on page 121.

Figure 31: Select Cluster Action



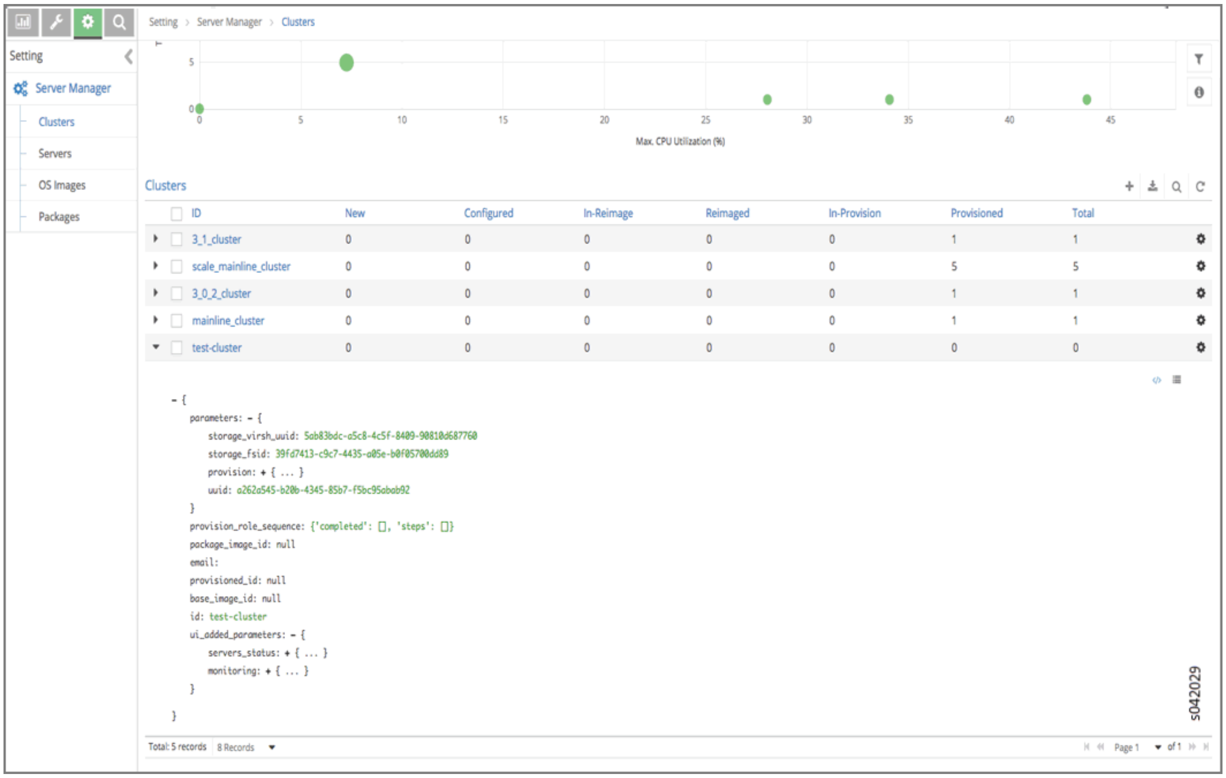
You can also click the expansion icon on the left side of the cluster name to display the details of that cluster in an area below the name line, see Figure 32 on page 122.

Figure 32: Display Cluster Details



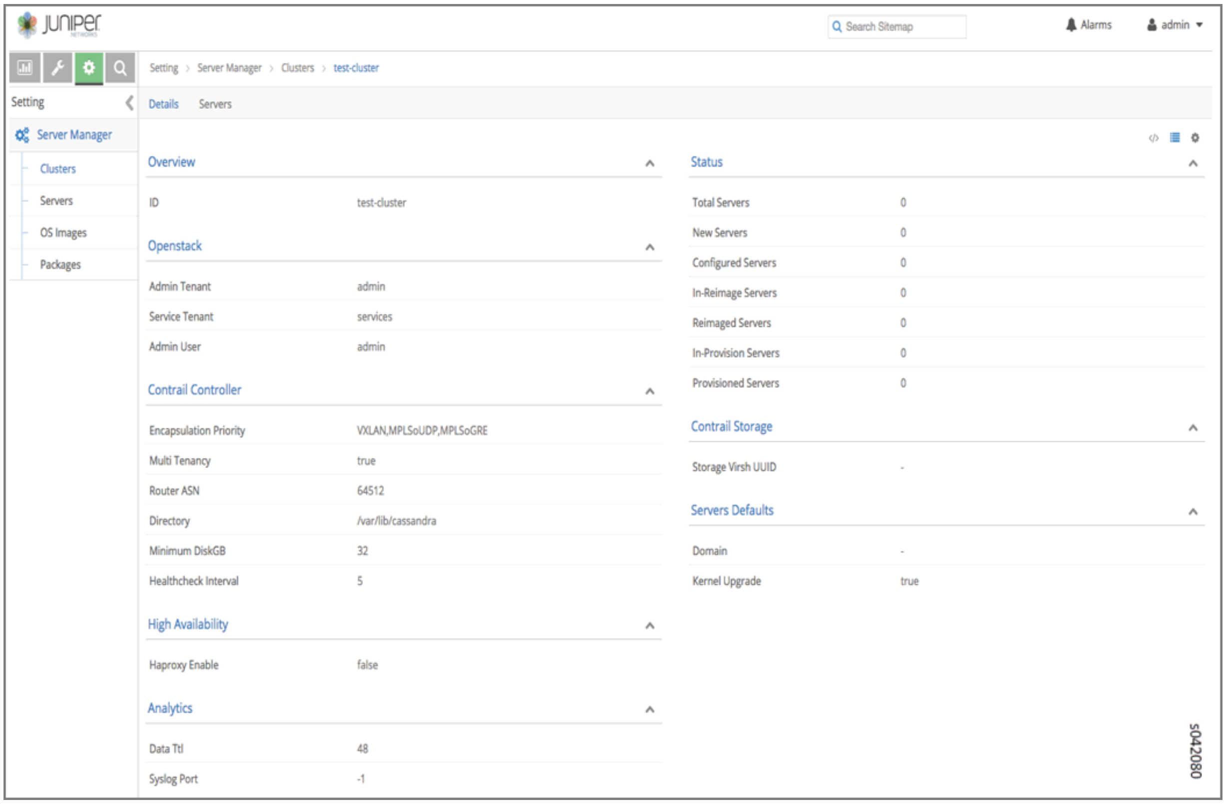
Click the upper right icon to switch to the JSON view to see the contents of the JSON file for the cluster, see Figure 33 on page 123.

Figure 33: View Cluster JSON



The cluster name is a link, click the cluster name to display the cluster **Details** page, see Figure 34 on page 124.

Figure 34: Link to View Cluster Details



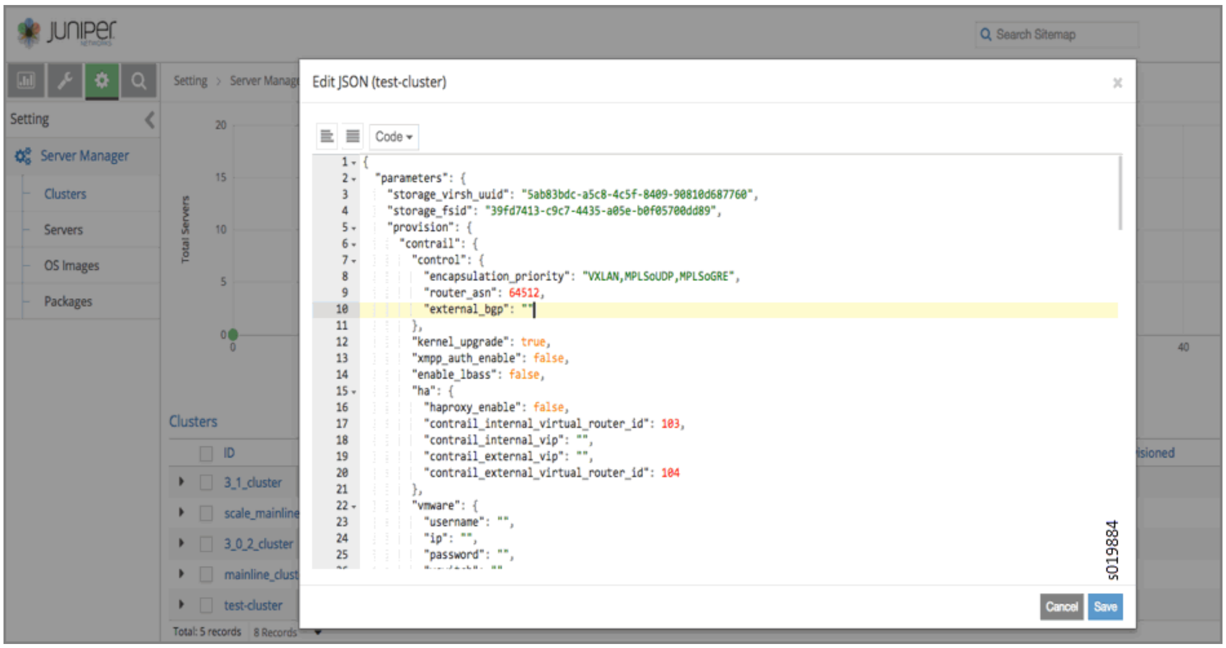
Click the **Servers** tab to display the servers under that cluster, see Figure 35 on page 124.

Figure 35: Display Servers for Cluster

Edit a Cluster through Edit JSON

Select **Edit JSON** to edit a cluster by editing the JSON file. Make changes to the JSON code and click **Save** to save the edited configuration for the cluster, see Figure 36 on page 125.

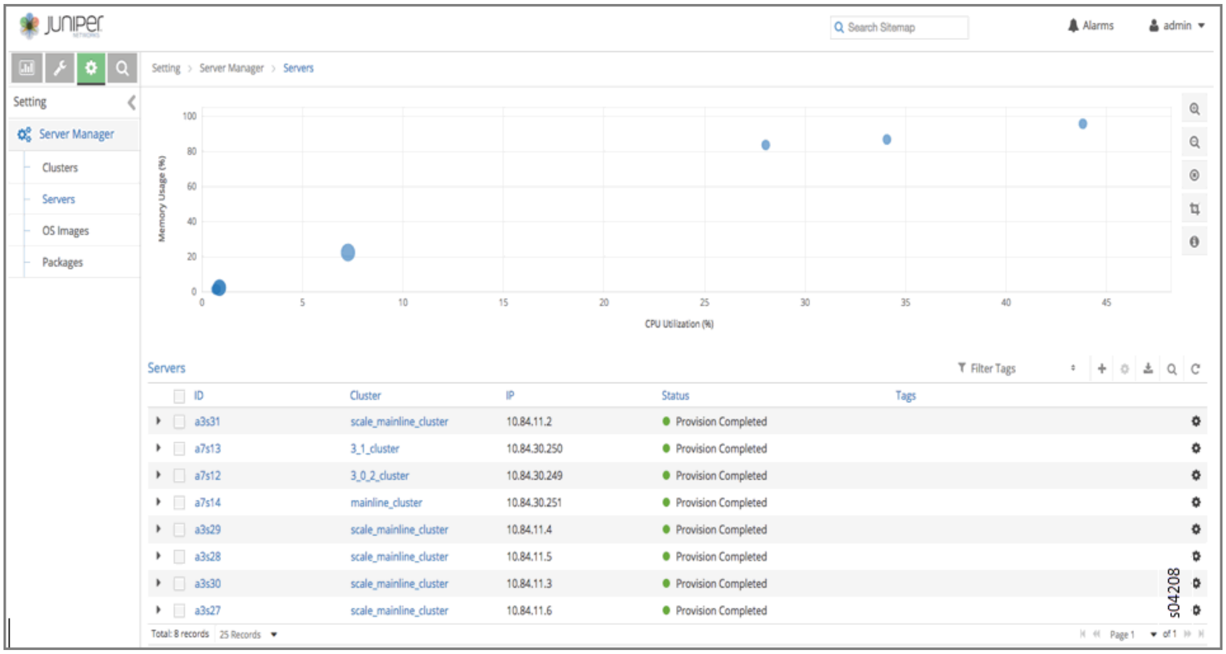
Figure 36: Edit Cluster JSON



Working with Servers in the Server Manager User Interface

Select **Setting > Server Manager** and click the **Servers** link in the left sidebar at to view a list of all servers, see Figure 37 on page 126.

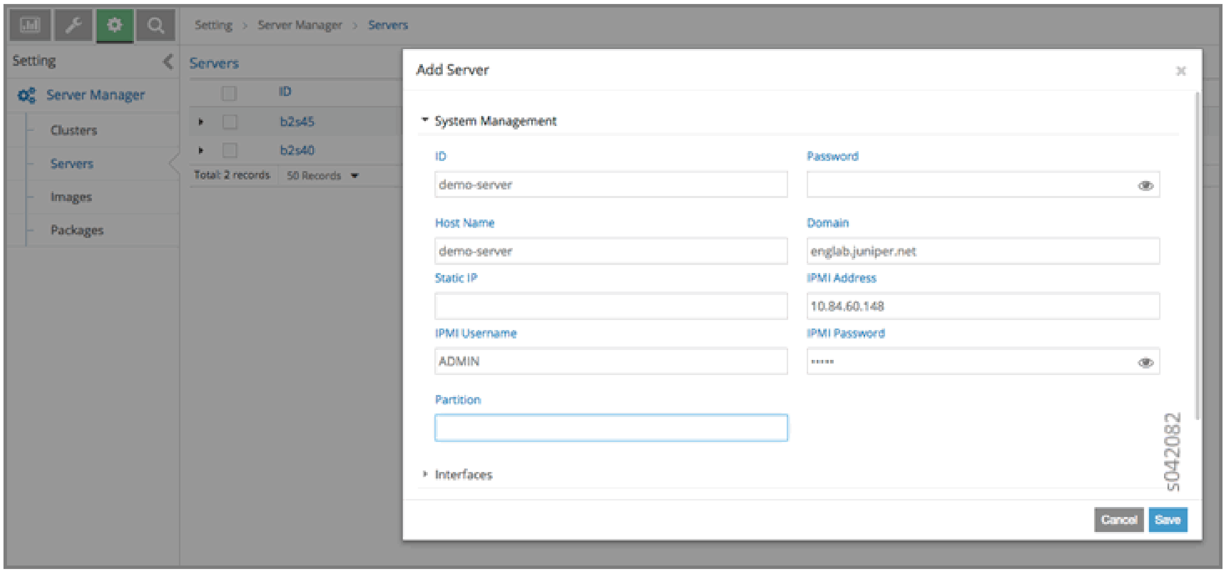
Figure 37: View Servers



Add a Server

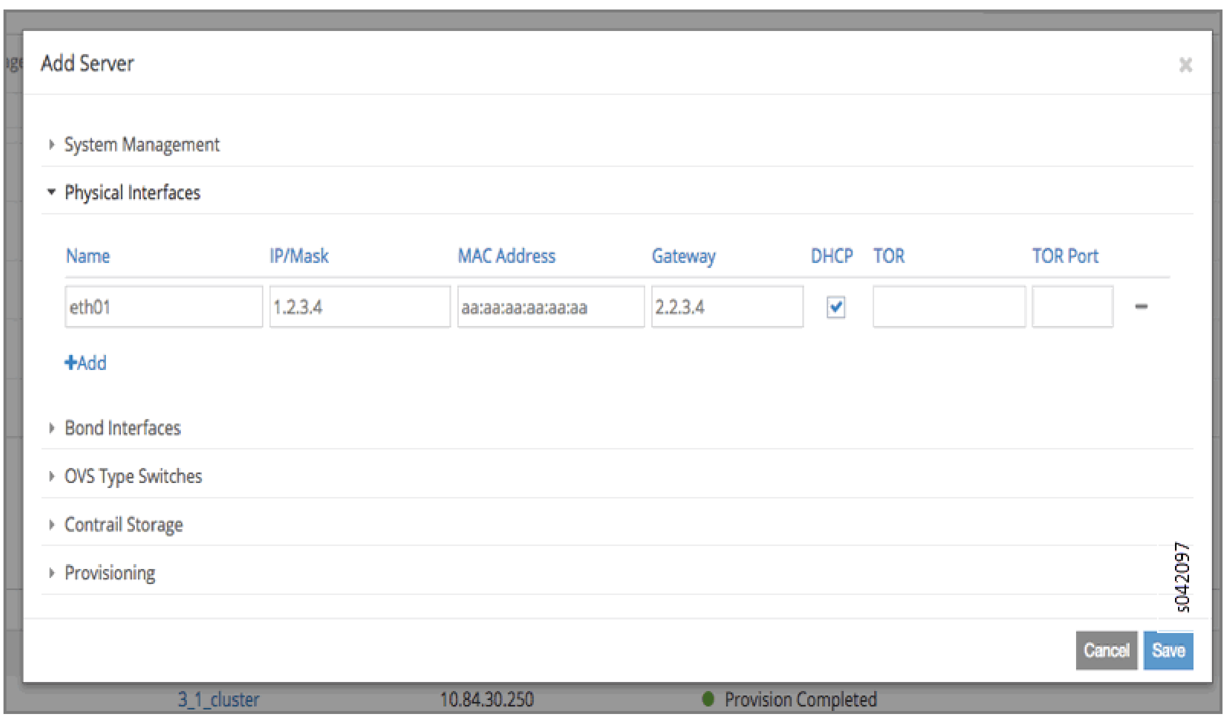
To add a new server, select **Setting** > **Server Manager** > **Servers** and click the plus (+) icon at the upper right side in the header line. The **Add Server** window is displayed, see Figure 38 on page 127, in which the **System Management** tab is expanded. Here you enter the details of `ID`, `Password`, `Domain`, `Partition`, and so on for the server.

Figure 38: Add Server, System Management



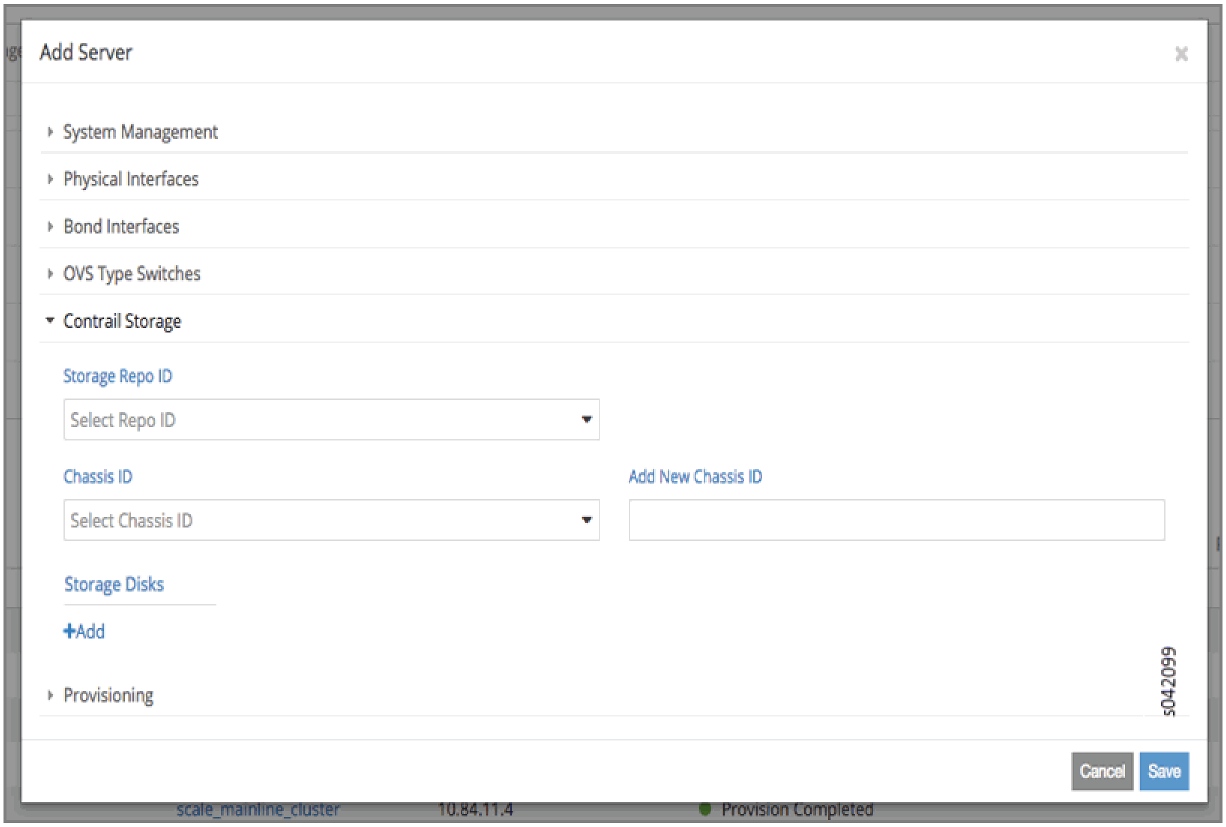
In the following image, the **Physical Interfaces** icon is selected. You can add new interfaces or edit existing interfaces. To enable editing for any field, hover the cursor on any selected field to open it, see Figure 39 on page 127.

Figure 39: Add Server, Physical Interfaces



In the following image, the **Contrail Storage** icon is selected. You can configure parameters for Contrail Storage, including selecting a package and adding storage disks locations, see Figure 40 on page 128.

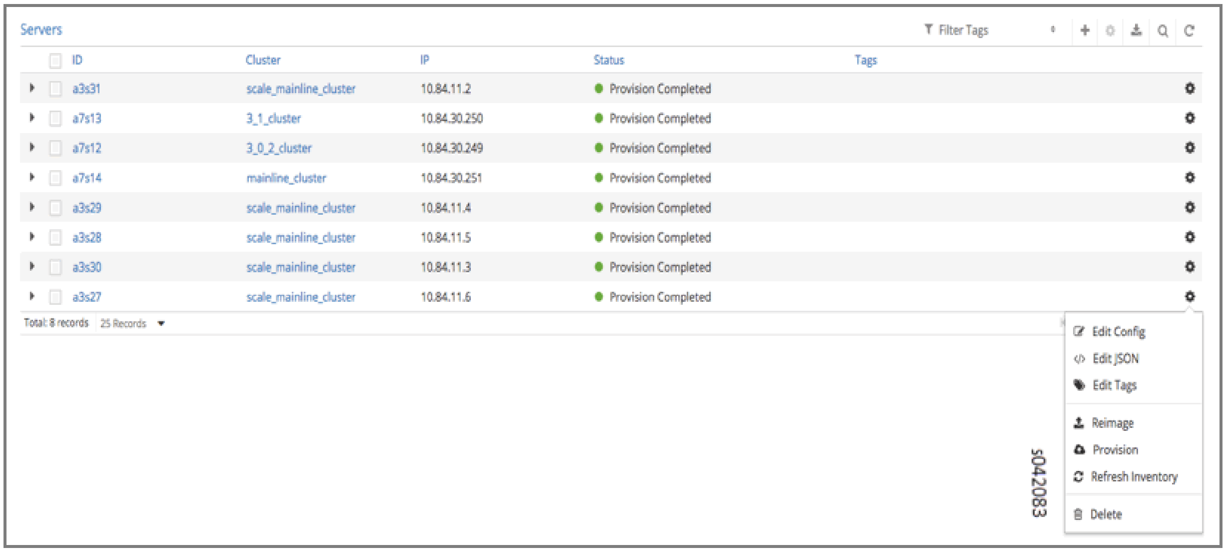
Figure 40: Add Server, Contrail Storage



When you are finished entering new server details in the **Add Server** window, click **Save** to add the new server configuration to the list of servers.

You can change details of the new server by clicking the gear wheel icon to the right side to get a list of actions available, including **Edit Config, Edit JSON, Edit Tags, Reimage, Provision, Refresh Inventory,** and **Delete**, see Figure 41 on page 129.

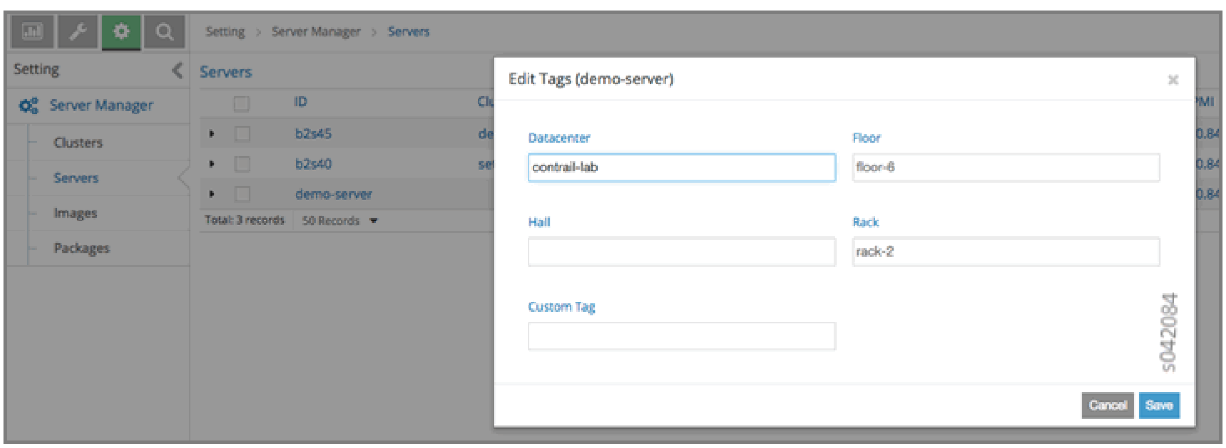
Figure 41: Select Server Actions



Edit Tags for Servers

Select **Edit Tags** from the gear wheel icon menu. The **Edit Tags** window is displayed. Enter any user-defined tags to be associated with the selected server, then click **Save** to add the tags to the server configuration, see Figure 42 on page 129.

Figure 42: Edit Tags



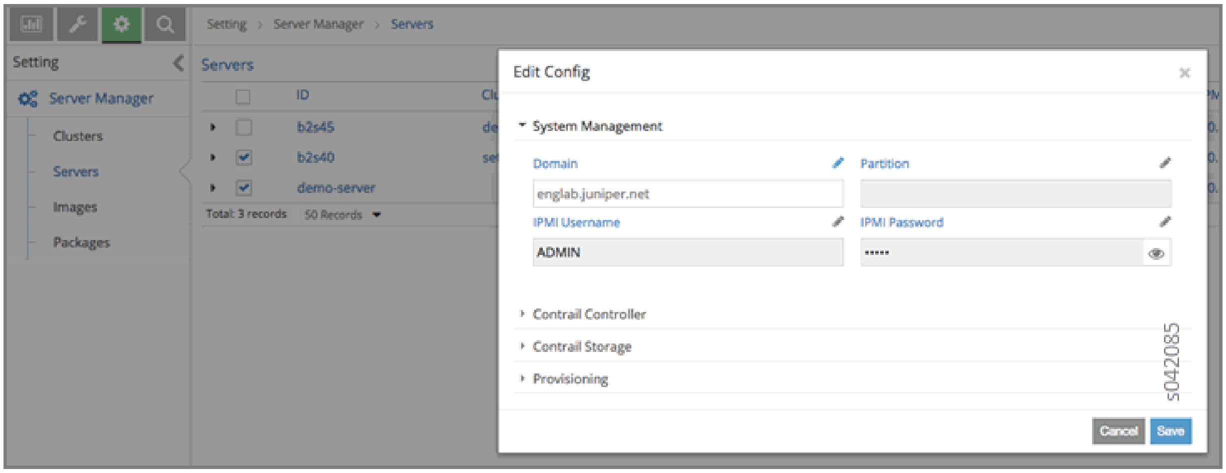
Using the Edit Config Option for Multiple Servers

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

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

Click a pencil icon to open configuration fields that can be edited Fields include **System Management, Contrail Controller, Contrail Storage**, and so on, see Figure 43 on page 130.

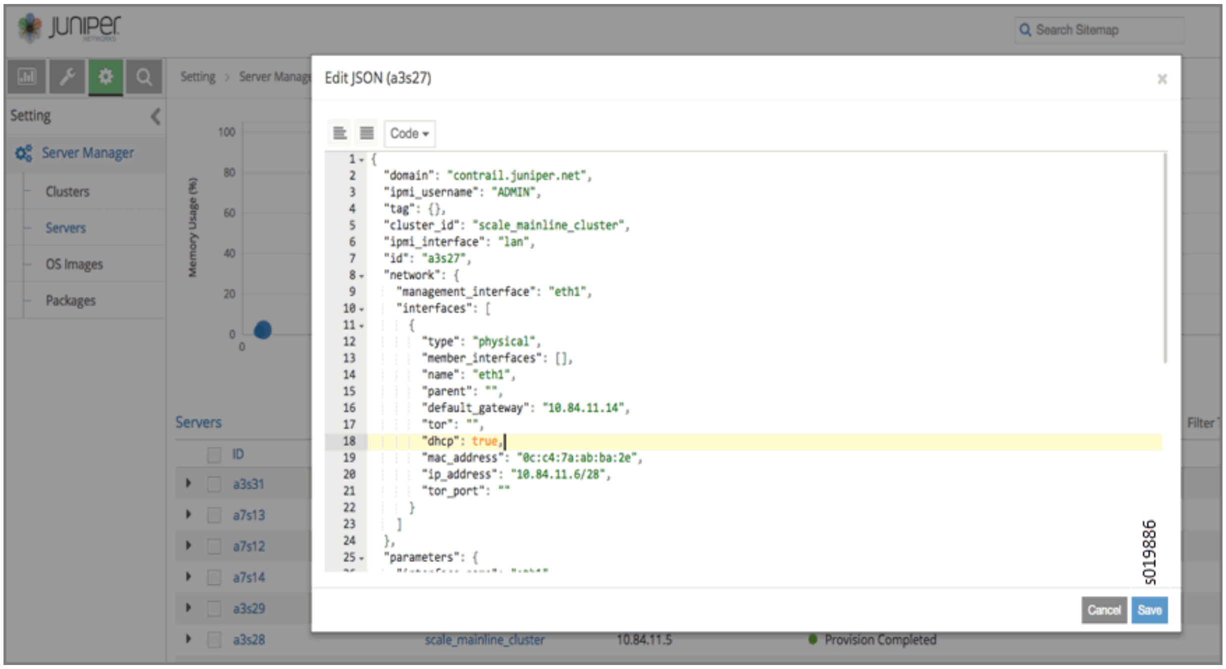
Figure 43: Edit Config, Multiple Servers



Edit a Server through Server Manager, Edit JSON

Select **Edit JSON** to edit the server through JSON file. Make changes to the server details in the JSON, then click **Save**, see Figure 44 on page 131.

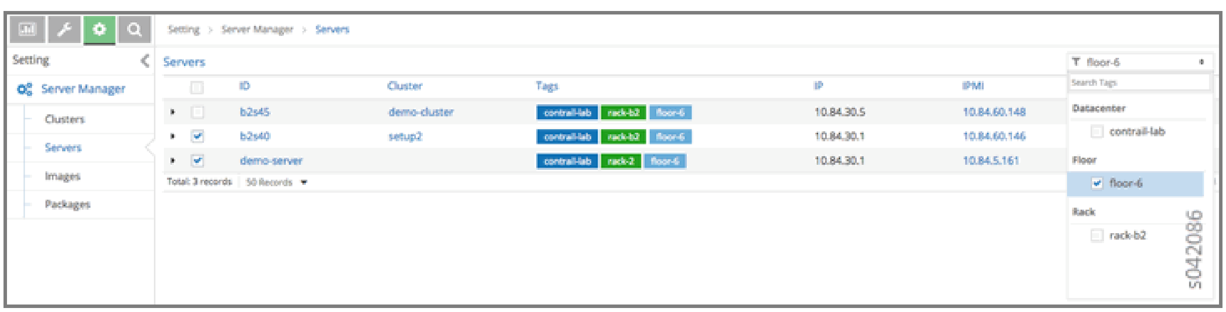
Figure 44: Server Edit JSON



Filter Servers by Tag

You can filter servers according to the tags defined for them. In the **Servers** window, click the **Filter Tags** field in the upper right heading. A list of configured tags is displayed. Select a tag by which to filter the list of servers, see Figure 45 on page 131.

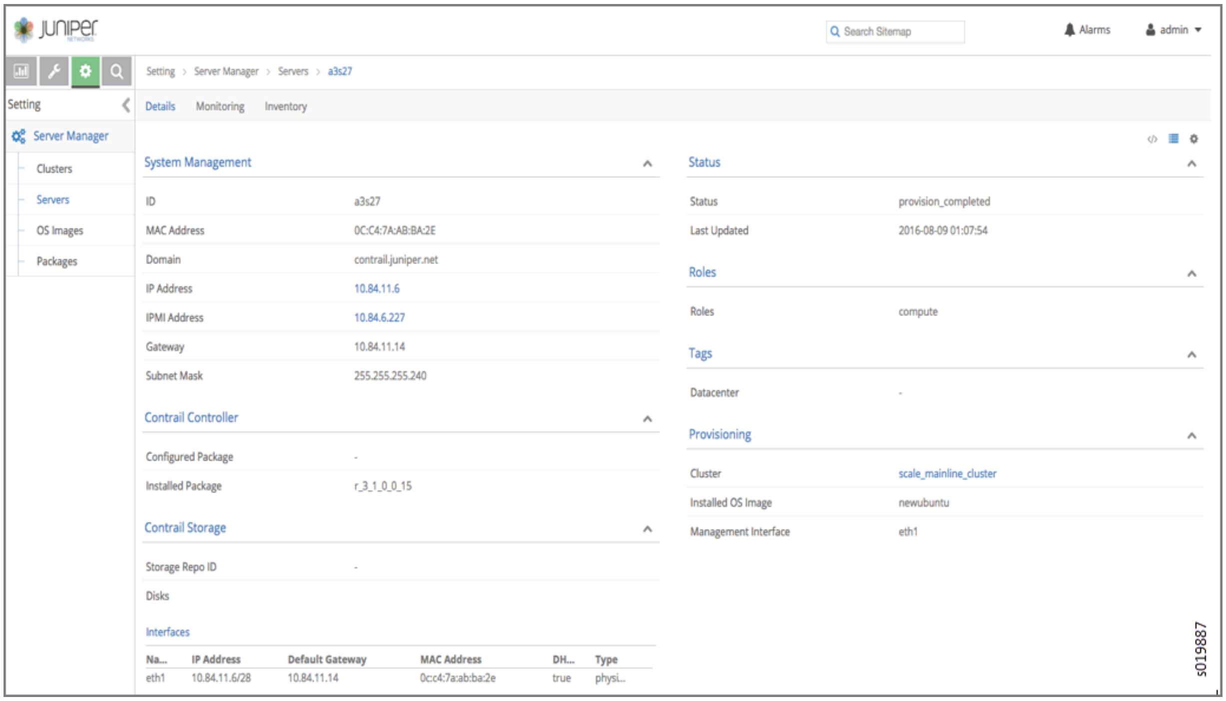
Figure 45: Filter Servers by Tag



Viewing Server Details

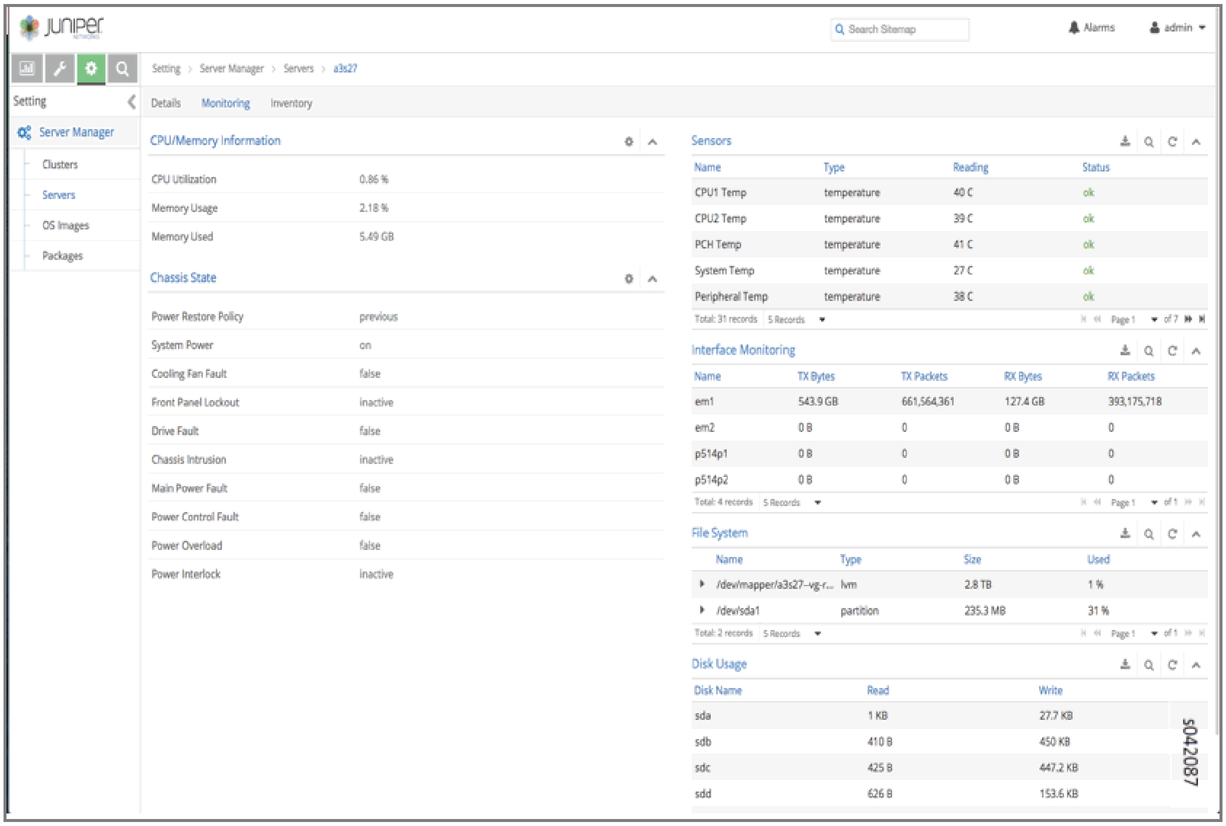
Each server name on the **Servers** page is a link to the details page for that server. Click any server name to open the details for that server, including **System Management** information, **Status, Contrail Controller, Contrail Storage, Roles, Tags**, and **Provisioning**, see Figure 46 on page 132.

Figure 46: View Server Details, System Management



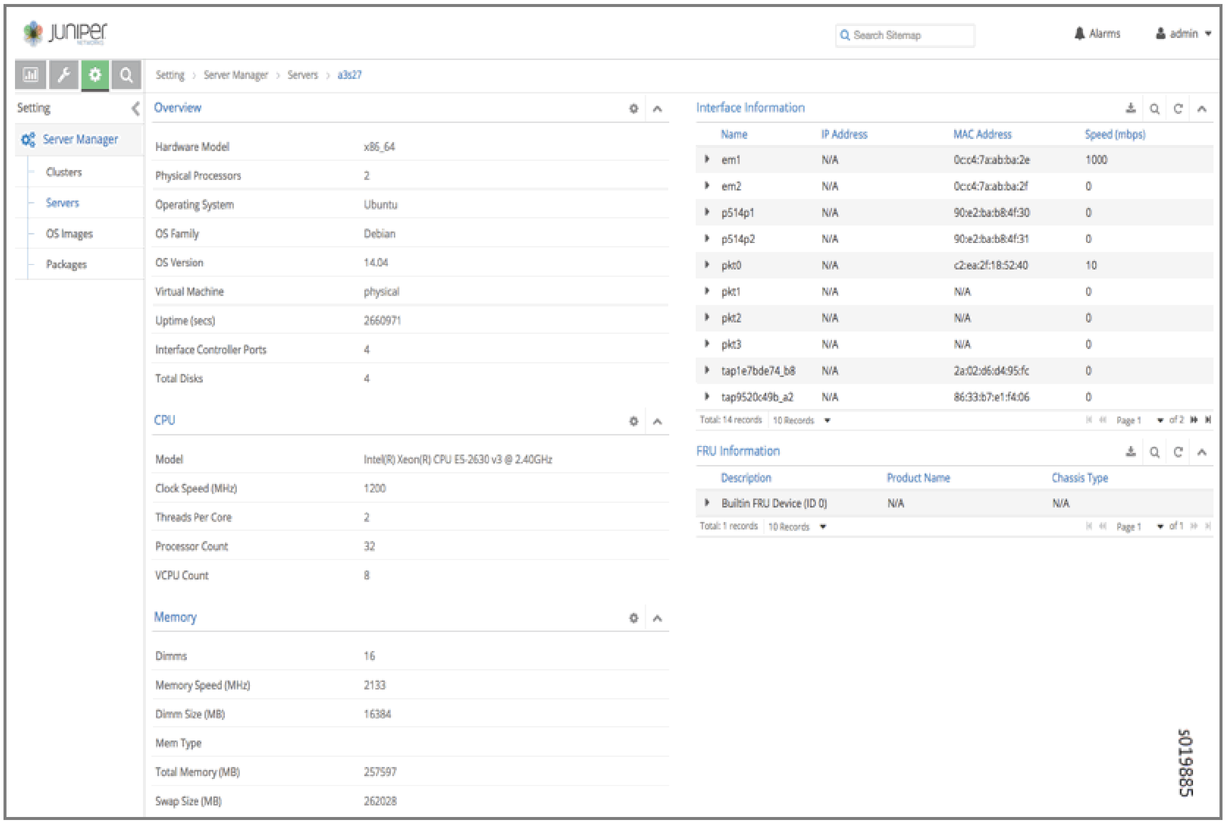
At the **Servers** page, click the **Monitoring** tab to see detailed information regarding **CPU/Memory Information, Chassis State, Sensors, Interface Monitoring, File System**, and **Disk Usage**, see Figure 47 on page 133.

Figure 47: Server Monitoring



At the **Servers** page, click the **Inventory** tab to see detailed information regarding **Overview of the server, Interface Information, CPU information, Memory,** and **FRU Information**, see Figure 48 on page 134.

Figure 48: Server Inventory



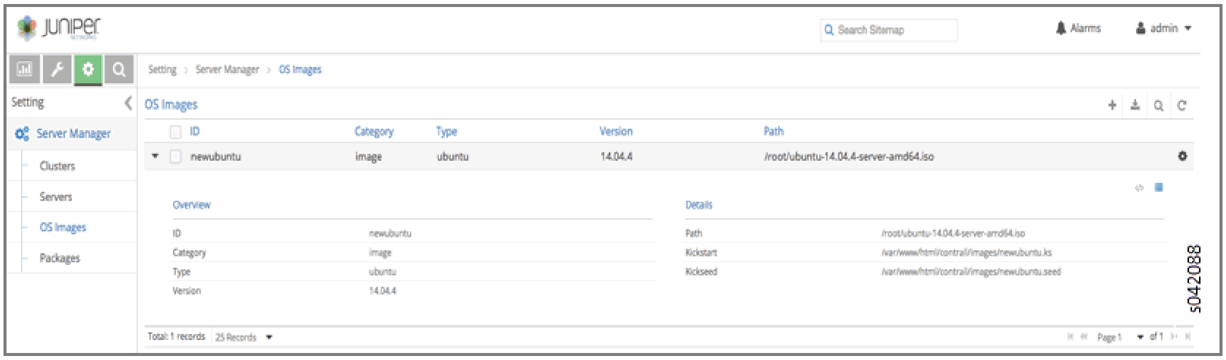
Configuring Images and Packages

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

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

Select **Images**. The Images page is displayed, see Figure 49 on page 135.

Figure 49: Servers OS Images

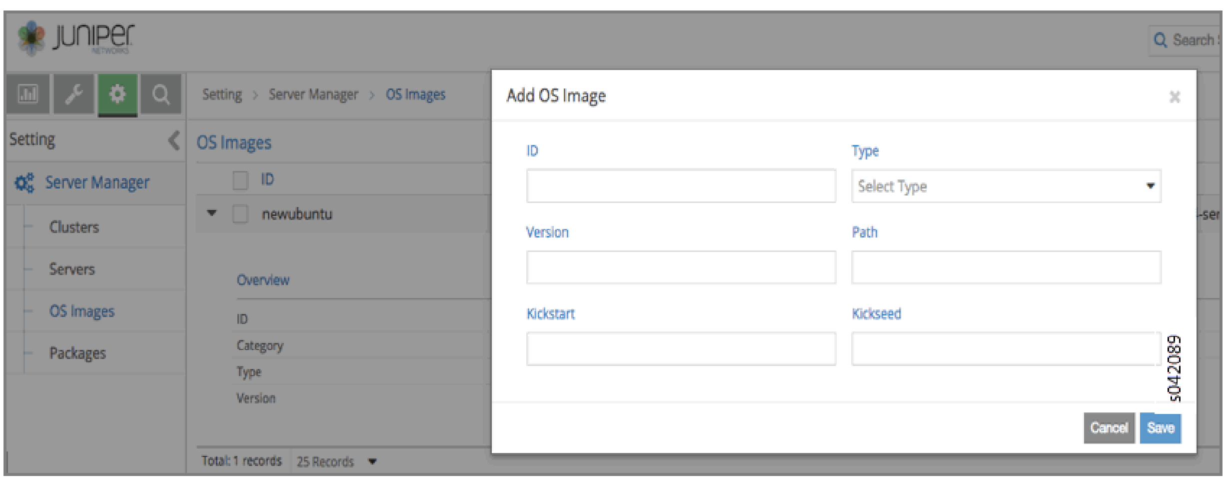


Add New Image or Package

To add a new image or package, on the respective **Images** or **Packages** page, click the plus (+) icon in the upper right header. The **Add Image** window is displayed. Enter the information for the new image (or package) and click **Save** to add the new item to the list of configured items, see Figure 50 on page 135.

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

Figure 50: Add OS Image



Selecting Server Manager Actions for Clusters

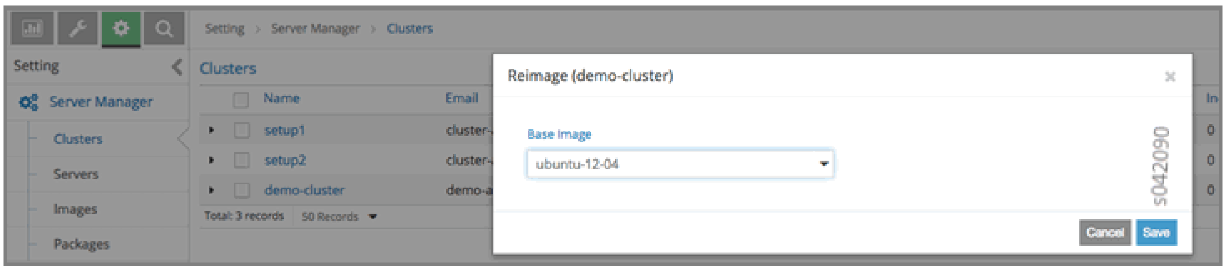
After all aspects of a cluster are configured, you can select actions for the Server Manager to perform on the cluster, such as **Reimage** or **Provision**.

Reimage a Cluster

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

The **Reimage** dialog box is displayed, as shown. Verify that the correct image is selected in the **Default Image** field, then click **Save** to initiate the reimage action, se Figure 51 on page 136.

Figure 51: Reimage Cluster

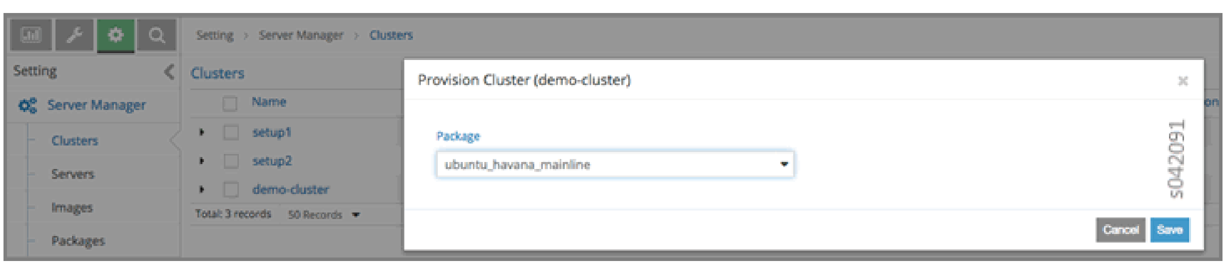


Provision a Cluster

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

The **Provision Cluster** dialog box is displayed, as shown. Verify that the correct package for provisioning is selected in the **Default Package** field, then click **Save** to initiate the provisioning action, see Figure 52 on page 136.

Figure 52: Provision Cluster



SEE ALSO

Using Server Manager to Automate Provisioning | 79

Installing and Using Server Manager Lite

IN THIS SECTION

- Server Manager Lite Overview | 137
- Installing Server Manager Lite | 138
- Provisioning Using SM-Lite with Contrail 4.0 | 138
- Displaying the Cluster Status | 139
- Displaying the SM-Lite Installation and Provisioning Log Files | 139
- Contrail Provisioning Log Files | 139

This topic describes how to install and troubleshoot Server Manager Lite.

Server Manager Lite Overview

Server Manager Lite (SM-Lite), is a streamlined version of the Server Manager software that does not include the reimage function.

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

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

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

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

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

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

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

Installing Server Manager Lite

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

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

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

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

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

Provisioning Using SM-Lite with Contrail 4.0

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

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

1. Installs SM-Lite.

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

2. Prepares the cluster for Contrail provisioning.

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

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

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

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

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

Displaying the Cluster Status

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

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

Displaying the SM-Lite Installation and Provisioning Log Files

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

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

Contrail Provisioning Log Files

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

- **http:*<sm-lite-ip-address>*/logs**
- **/var/log/contrail_server_manager/*<target>*/*<timestamp>*.log**
- **/var/log/contrail/***

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

RELATED DOCUMENTATION

Using Server Manager to Automate Provisioning | 79
Using the Server Manager Web User Interface | 113

CHAPTER 6

Installing and Using Contrail Storage

IN THIS CHAPTER

- Installing and Using Contrail Storage | 140

Installing and Using Contrail Storage

IN THIS SECTION

- Overview of the Contrail Storage Solution | 140
- Basic Storage Functionality with Contrail | 141
- Ceph Block and Object Storage Functionality | 141
- Using the Contrail Storage User Interface | 142
- Hardware Specifications | 143
- Contrail Storage Provisioning | 143

Overview of the Contrail Storage Solution

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

The Contrail storage solution has the following features:

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

- Does not require virtual machines (VMs) to configure mirrors for replication.
- Allows nodes to provide both compute and storage services.
- Provides easy installation of basic storage functionality based on Contrail roles.
- Provides a Contrail-integrated user interface from which the user can monitor Ceph components and drill down for more information about components.
- Provides native live-migration support if the VM is booted with Ceph storage as its root volume.
- Provides object storage support through Swift and S3 APIs.

Basic Storage Functionality with Contrail

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

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

Ceph Block and Object Storage Functionality

In Contrail Release 4.0, installing the Contrail storage solution creates the following Ceph configurations.

- Each disk is configured as a standalone storage device, enhancing optimal performance and creating proper failure boundaries. Ceph allocates and assigns a process called object storage daemon (OSD) to each disk.
- A replication factor of 2 is configured, consisting of one original instance plus one replica copy. Ceph ensures that each replica is on a different storage node.
- A Ceph monitor process (mon) is configured is configured on the contrail-ceph-controller node.
- The correct number of placement groups are automatically configured, based on the number of disk drives in the cluster.
- Properly identified SSD drives are set up for use as Ceph OSD journals to reduce write latencies.
- Multi-pool configuration is set up to segregate the OSD disks into logical pools improving performance and efficiency.

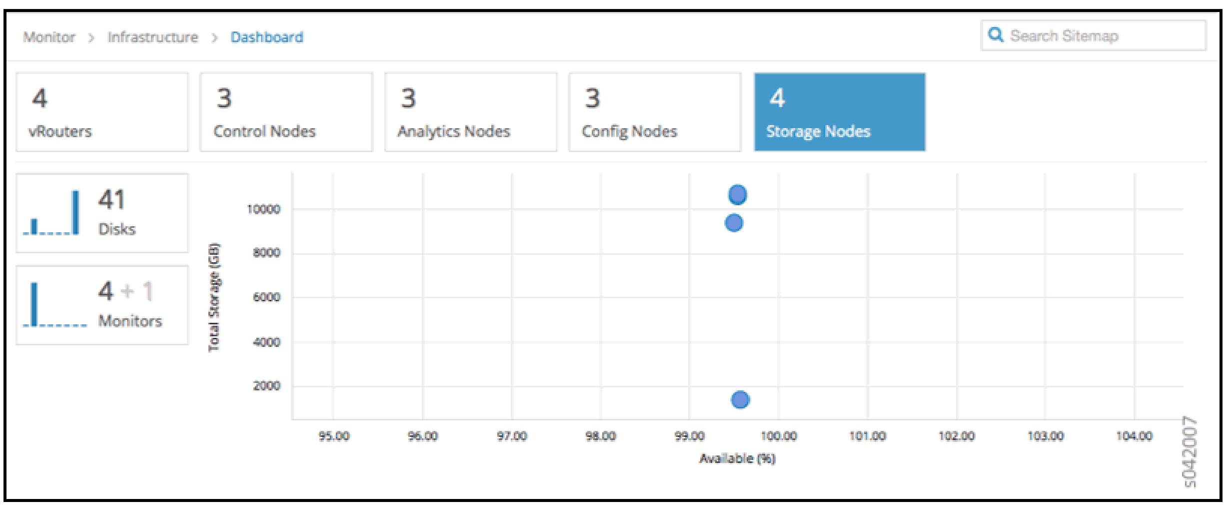
- If multiple storage nodes are in a single chassis, the `chassis` option helps in defining replication of data and also disabling replication of data within the nodes of the same chassis. Replication helps in avoiding data loss during a power failure to the chassis.

Using the Contrail Storage User Interface

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

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

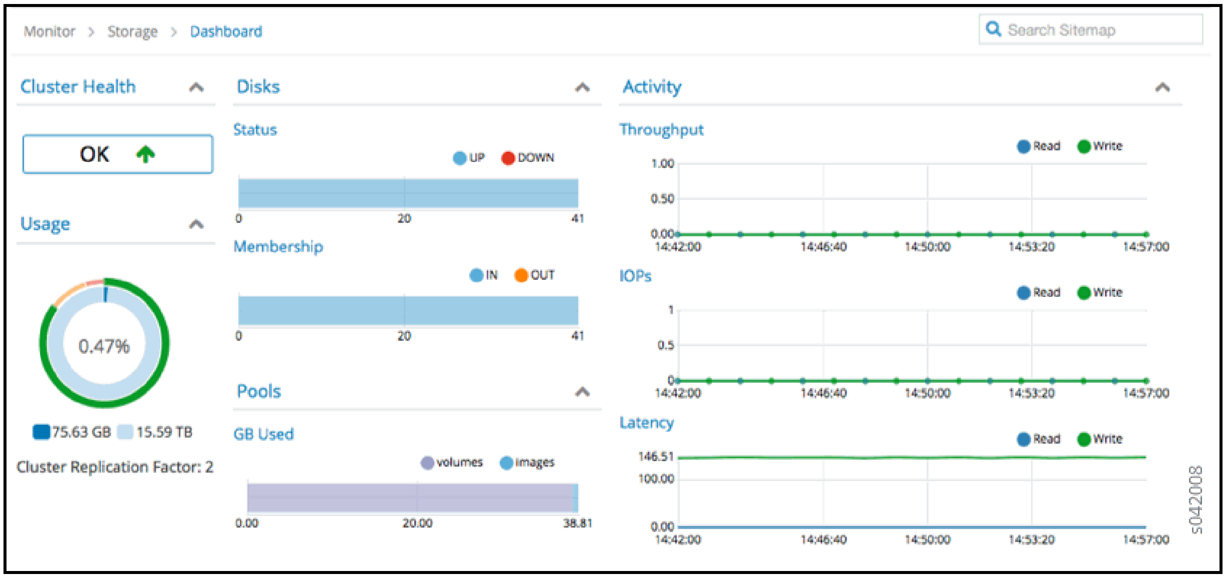
Select **Monitor > Infrastructure > Dashboard** to display an at-a-glance view of the system infrastructure components, including the numbers of virtual routers, control nodes, analytics nodes, config nodes, and storage nodes currently operational, and a bubble chart of storage nodes showing the Available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



Bubble charts use the following color-coding scheme for storage nodes:

- Blue—working as configured.
- Red—error, node is down.
- Yellow—one of the node disks is down.

Select **Monitor > Storage > Dashboard** to see a summary of cluster health, usage, pools, and disk status, and to gain insight into activity statistics for all nodes. See the following figure.



Hardware Specifications

The following are additional hardware specifications needed for the Contrail storage solution.

Additional minimum specifications:

- Two 500 GB, 7200 RPM drives in the server 4 and server 5 cluster positions (those with the compute storage role) in the Contrail installation. This configuration provides 1 TB of clustered, replicated storage.

Recommended compute storage configuration:

- For every 4-5 HDD devices on one compute storage node, use one SSD device to provide the OSD journals for that set of HDD devices.

Contrail Storage Provisioning

The `contrail-ceph-controller` and `contrail-ceph-compute` are two roles required to enable Ceph storage. The `contrail-ceph-controller` role is added to the Ceph monitor servers. The number of mons is limited to three for small clusters and five for large clusters with more than 1000 disks. The `contrail-ceph-compute` role is added to the servers that have the physical disks required for Ceph storage and also to the OpenStack Nova compute nodes that require Ceph storage services.

The following example displays sample **cluster.json** to provide Ceph storage configurations.

```
"parameters": {
                  "provision": {
```

```
                    "contrail_4": {
                        "storage_ceph_config": {
                            "replica_size": 2,
"ceph_object_storage": "True",
"object_store_pool": "volumes"
                        }
                    }
                }
            }
```

The replica_size is added to change the default replica size of 2. The `ceph_object_storage` option enables the Ceph-based object storage to support Swift and S3 APIs and the `object_storage_pool` option specifies the Ceph pool used for the Ceph object storage functionality.

The following example displays sample **server.json** to enable Ceph storage.

```
Server.json :

            "parameters": {
                "provision": {
                    "contrail_4":{
                        "storage":{
                            "storage_osd_disks":[
                                "/dev/sdb:/dev/sdd:Pool_1",
                                "/dev/sdc:/dev/sdd:Pool_2"
                            ],
                            "storage_osd_ssd_disks":[
                                "/dev/sde:Pool_1",
                                "/dev/sdf:Pool_2"
                            ],
                "chassis_id": "chassis_1"
                        }
                    }
                }
            "roles": [
"contrail-ceph-controller", "contrail-ceph-compute"
                   [p0-           ]
```

The `storage_osd_disks` or `storage_osd_ssd_disk` is needed to provision the disks for Ceph. The first disk iss OSD disk and the second optional disk is used as a Journal disk. If a multi-pool configuration is required, the pool name can be added along the OSD disk as shown in the **server.json** to enable Ceph storage. The `chassis_id` option can also be included per server. Pools and the `chassis` option cannot co-exist.

NOTE: The disks added to Ceph are not included in the OS disk. The `partition` parameter in the server JSON lists only the required OS disks.

```
"parameters": {
"partition": "/dev/sda"
}
```

The disks added to Ceph cannot be part of LVM.

CHAPTER 7

Upgrading Contrail Software

IN THIS CHAPTER

- Upgrading Contrail 4.0 to 4.1 | 146
- Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3 | 148
- Upgrade Procedure for Ubuntu-based Contrail 4.1.3 to Contrail 4.1.4 Using Juju with Netronome SmartNIC | 161
- Upgrade Procedure for RHOSP-based Contrail 4.1.3 to Contrail 4.1.4 | 170
- Dynamic Kernel Module Support (DKMS) for vRouter | 185

Upgrading Contrail 4.0 to 4.1

IN THIS SECTION

- Upgrade Assumptions | 146
- Upgrade Procedure | 147

This section provides the process for upgrading an existing Contrail Release 4.0 system to Contrail Release 4.1.

Upgrade Assumptions

This upgrade procedure assumes the following.

- The initial cluster (4.0.x) was provisioned using Server Manager.
- The OpenStack SKU is the same in the “from” and “to” versions.
- A backup has been made of the analytics database, see *Backing Up Contrail Databases Using JSON Format*.

Upgrade Procedure

1. Make a backup of the analytics database, because the upgrade procedure removes the analytics database information, see *Backing Up Contrail Databases Using JSON Format*.

2. Add the new Contrail 4.1 Debian image to the Server Manager JSON used for provisioning.

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

3. Upgrade the cluster by reprovisioning the cluster with the new image.

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

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

   - For a multinode system:

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

4. Monitor progress of the provisioning by observing cluster status or log entries.

   - Cluster status: `server-manager display server --cluster_id <cluster_id> --select "id,ip_address,roles,status"`

   - Log entries: /var/log/contrail-server-manager/debug.log

     NOTE: Log entries from the previous version are lost in the upgrade process.

For more upgrade instructions, see:

- Upgrade Procedure for RHOSP-based Contrail 4.1.3 to Contrail 4.1.4
- Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3
- Upgrade Procedure for RHOSP-based Contrail 4.1.1 to Contrail 4.1.2
- Upgrade Procedure for RHOSP-based Contrail 3.2.x to Contrail 4.1

Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3

IN THIS SECTION

- Prerequisite | 148
- Upgrade the Undercloud | 149
- Update Red Hat Director Image Archives | 151
- Prepare Repositories on all Nodes | 153
- Upgrade the Operating System on Contrail Nodes | 153
- Prepare the Contrail Packages | 154
- Upgrade the Contrail Heat Templates | 155
- Modify the Yum Update Script for TripleO Puppet | 156
- Update the Overcloud Deployment Plan | 157
- Upgrade the Overcloud | 158

This section presents the steps to upgrade an OSP-based Contrail deployment from Contrail version 4.1.2 to Contrail version 4.1.3.

Prerequisite

Before upgrading to Contrail Release 4.1.3, you must update the `net-snmp` package to the `net-snmp #37` version. The following `net-snmp` packages must be available in the upgrade repository and are installed automatically on Contrail Analytics nodes during the upgrade process:

- net-snmp-5.7.2-37.el7.x86_64.rpm
- net-snmp-agent-libs-5.7.2-37.el7.x86_64.rpm
- net-snmp-libs-5.7.2-37.el7.x86_64.rpm
- net-snmp-utils-5.7.2-37.el7.x86_64.rpm

Ensure you have a cloud up and running with RHOSP10 and Contrail 4.1.2 before you proceed with the upgrade procedure.

This procedure has been validated with the following Contrail, Red Hat, and OpenStack versions.

Contrail Version	Red Hat Version	OpenStack Version
3.2.3	RHEL 7.3	RHOSP10 (packages dated Apr. 15, 2017)
3.2.6	RHEL 7.4	RHOSP10 (packages dated Feb. 2, 2018)
4.1	RHEL 7.4	RHOSP10 (packages dated Feb. 27, 2018)
4.1.1	RHEL 7.5	RHOSP10 (packages dated Jun. 4, 2018) RHOSP11 (packages dated Jun. 4, 2018)
4.1.2	RHEL 7.5	RHOSP10 (packages dated Oct 29, 2018)
4.1.3	RHEL 7.5	RHOSP10 (packages dated Oct 29, 2018)

CAUTION: Set the Red Hat Satellite filter end date to October 29, 2018 before proceeding with the upgrade.

Upgrade the Undercloud

Upgrade the undercloud to the most current RHOSP10 version.

1. Log in to the undercloud as the stack user.

```
$ su - stack
```

2. Update the Contrail repositories.

```
$ sudo rm -rf /etc/yum.repos.d/*contrail*
```

```
$ curl http://newrepo.contrail41-dev.repo -o /etc/yum.repos.d/localrepo.repo
```

3. Stop the main OpenStack platform services.

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```

4. Update the `python-tripleoclient` package and its dependencies to ensure you have the most current scripts for the minor version update.

```
$ sudo yum update python-tripleoclient
```

5. Upgrade the undercloud.

```
$ openstack undercloud upgrade
```

6. Reboot the node.

```
$ sudo reboot
```

7. Wait until the node reboots, then check the status of all services.

NOTE: It can take as much as 10 minutes or more for the `openstack-nova-compute` to become active after a reboot.

```
$ sudo systemctl list-units "openstack*" "neutron*" "openvswitch*"
```

8. Verify the version of RHEL after the undercloud upgrade.

NOTE: Contrail does not support undercloud Red Hat version running with RHEL-7.6 as part of Contrail 4.1.3 release.

```
[root@undercloud ~]# cat /etc/redhat-release
Red Hat Enterprise Linux Server release 7.5 (Maipo)
[root@undercloud ~]#
```

9. Verify the existence of the overcloud and its nodes.

```
$ openstack stack list
```

```
$ ironic node-list
```

10. Verify that all OpenStack servers are Active.

```
$ openstack server list
```

Figure 53: Server List

ID	Name	Status	Networks	Image Name
e37480b0-e098-4216-aea4-04b028aa5fb9	overcloud-contrailanalytics-2	ACTIVE	ctlplane=192.0.2.11	overcloud-full
c77dcc49-c039-4855-bc91-e217bd34a51e	overcloud-contrailanalytics-1	ACTIVE	ctlplane=192.0.2.12	overcloud-full
63d147ec-61e0-4d49-b1cb-6b68b7e5f48f	overcloud-contrailanalytics-0	ACTIVE	ctlplane=192.0.2.21	overcloud-full
913d0bcb-9e9b-4987-a858-0389345e7025	overcloud-controller-0	ACTIVE	ctlplane=192.0.2.14	overcloud-full
e66e86b1-df4a-4246-8e24-41461d617c18	overcloud-controller-2	ACTIVE	ctlplane=192.0.2.15	overcloud-full
9bbc1552-9ac1-4f27-846c-7756b14b84de	overcloud-controller-1	ACTIVE	ctlplane=192.0.2.22	overcloud-full
8f8f5b9b-d5e2-4349-8d46-a488f2b7ab56	overcloud-contrailanalyticsdatabase-0	ACTIVE	ctlplane=192.0.2.17	overcloud-full
505c00f5-6715-48a7-a9ea-61ff157aa28a	overcloud-contrailanalyticsdatabase-2	ACTIVE	ctlplane=192.0.2.24	overcloud-full
ba8abe03-071b-4326-8fc0-715e93db3e4d	overcloud-contrailanalyticsdatabase-1	ACTIVE	ctlplane=192.0.2.19	overcloud-full
a7763383-92a0-4304-afc3-8aebf0ed2a05	overcloud-contrailcontroller-2	ACTIVE	ctlplane=192.0.2.20	overcloud-full
e381db99-add0-4887-ac96-1e40d4108fc8	overcloud-contrailcontroller-0	ACTIVE	ctlplane=192.0.2.18	overcloud-full
c808dcf1-4906-491e-8780-1f099c6a776c	overcloud-novacompute-0	ACTIVE	ctlplane=192.0.2.16	overcloud-full
7eabc822-16d6-4fc6-afd7-7ba662ce7e92	overcloud-contrailcontroller-1	ACTIVE	ctlplane=192.0.2.23	overcloud-full

s019929

Update Red Hat Director Image Archives

The undercloud update process might download new image archives from the rhosp-director images and the rhosp-director-ipa packages. You will have to update your existing system with any new image archives.

1. Check the yum log to determine if new image archives are available.

```
$ sudo grep "rhosp-director-images" /var/log/yum.log
```

```
[stack@undercloud]$ sudo grep "rhosp-director-images" /var/log/yum.log

Oct 26 15:09:20 Installed: rhosp-director-images-ipa-10.0-20180821.1.el7ost.noarch
Oct 26 15:10:10 Installed: rhosp-director-images-10.0-20180821.1.el7ost.noarch
```

2. If new image archives are available, replace your current images with the new images. Before deploying any new images, remove any existing images from the images undercloud on the stack user's home (/home/stack/images).

```
$ rm -rf ~/images/*
```

3. Extract the new image archives.

```
mkdir  images
    cd images
for i in /usr/share/rhosp-director-images/overcloud-full-latest-10.0.tar /usr/share/rhosp-
director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

4. Install the Contrail package on the overcloud image by using the virt-customize command.

```
export LIBGUESTFS_BACKEND=direct  /usr/bin/virt-customize  -a /home/stack/images/overcloud-
full.qcow2 \
--copy-in /etc/yum.repos.d/mylocalrepo.repo:/etc/yum.repos.d \
--run-command 'yum -y install puppet-tripleo contrail-tripleo-puppet puppet-contrail‘\
--run-command ‘ cp -r /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/
tripleo/ ‘ \
--run-command 'rm -fr /var/cache/yum/*' \
--run-command 'yum clean all' \  --selinux-relabel
```

5. Import the new image archives into the undercloud and configure nodes to use the new images.

```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

6. Verify that the images are uploaded.

```
$ openstack image list
```

```
+--------------------------------------+------------------------+--------+
| ID                                   | Name                   | Status |
+--------------------------------------+------------------------+--------+
| a518a08c-3c49-47b7-9705-a7da5c90dcc6 | bm-deploy-ramdisk      | active |
| b7d75ff7-926b-426c-aa2e-424f5d9f8328 | bm-deploy-kernel       | active |
| cd7eedc0-2ed2-4a21-8359-9d8ee89374ac | overcloud-full         | active |
| 9c9e46f4-d571-49fb-a485-4653b6a52b44 | overcloud-full-initrd  | active |
| 740b6cb9-5757-475b-896b-49c526807671 | overcloud-full-vmlinuz | active |
+--------------------------------------+------------------------+--------+
```

7. Show the details of the new image that has been created. The new image will be used to add a new node in the overcloud.

```
$ openstack image show overcloud-full
```

```
+------------------+---------------------------------------------------
| Field            | Value                                                                                          |
+------------------+---------------------------------------------------
| checksum         | 13e67f5039dc7e69b5bbc494d8838b8d                                                               |
| container_format | bare                                                                                           |
| created_at       | 2018-10-26T20:02:18.000000                                                                     |
| deleted          | False                                                                                          |
| deleted_at       | None                                                                                           |
| disk_format      | qcow2                                                                                          |
| id               | cd7eedc0-2ed2-4a21-8359-9d8ee89374ac                                                           |
| is_public        | True                                                                                           |
| min_disk         | 0                                                                                              |
| min_ram          | 0                                                                                              |
| name             | overcloud-full                                                                                 |
| owner            | 0cd0e938b0fe46ef9c431715395f9469                                                               |
| properties       | kernel_id='740b6cb9-5757-475b-896b-49c526807671', ramdisk_id='9c9e46f4-d571-49fb-a485-4653b6a52b44' |
| protected        | False                                                                                          |
| size             | 1469448192                                                                                     |
| status           | active                                                                                         |
| updated_at       | 2018-10-26T20:02:52.000000                                                                     |
| virtual_size     | None                                                                                           |
+------------------+--------------------------------------------------
```

8. Verify `contrail-status` on all Contrail nodes. All services in the Contrail nodes, except the controller (OpenStack), should be up and running before proceeding with the upgrade.

```
[stack@undercloud ~]$ for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done
```

Prepare Repositories on all Nodes

1. Delete existing repositories on all overcloud nodes. Verify each deletion.

```
sudo for ipnode in $(nova list | sed '4,$ !d;$d'| awk -F 'ctlplane=' '{print $2}' | tr -d '|'); do echo cleaning yum repolist on $ipnode && ssh heat-admin@$ipnode 'find /etc/yum.repos.d/ ! -name 'contrail-install.repo' -type f -exec sudo rm -f {} +' ; done
```

2. Add new repositories on all overcloud nodes. Verify each addition.

```
sudo for ipnode in $(nova list | sed '4,$ !d;$d'| awk -F 'ctlplane=' '{print $2}' | tr -d '|'); do echo cleaning yum repolist on $ipnode && ssh heat-admin@$ipnode ' curl http://newrepo.contrail41-dev.repo -o /etc/yum.repos.d/localrepo.rep' ; done
```

Upgrade the Operating System on Contrail Nodes

1. Define a list ($iplist) that contains all Contrail nodes. Run the following command on undercloud VM as stack user.

```
Iplist=" @IPcontrailController1 @IPContrailController2 …"
```

> **CAUTION**: Attach the new satellite subscription key on all overcloud nodes before upgrading the overcloud packages. Satellite must be synced with filter end date *29th Oct 2018*. Make sure to clear cache by typing **sudo yum clean all**.

2. Upgrade the operating system for all nodes in the iplist. Run the following command on undercloud VM as stack user

```
sudo for ipnode in $iplist; do echo -e "\n\n\t******upgrade node : $ipnode ******" && ssh heat-admin@$ipnode 'sudo yum update -y --disablerepo=*contrail* --skip-broken && exit' ; done
```

3. Reboot overcloud contrail compute nodes, if there is any change in the kernel version. This needs to be done before installing contrail packages on compute VM.

```
Supported kernel versions: 3.10.0-862.11.6.el7.x86_64 and 3.10.0-957.el7.x86_64
------------------------------- [root@overcloud-novacompute-2 ~]# modinfo vrouter filename: /lib/modules/
```

```
3.10.0-862.11.6.el7.x86_64/extra/net/vrouter/vrouter.ko version: 4.1.3.0 license: GPL retpoline: Y
rhelversion: 7.5
```

Prepare the Contrail Packages

To prepare the Contrail packages for the installation from a local repository:

1. Navigate to the Contrail repository and perform the following tasks:

   - Delete the existing Contrail repositories.

     All existing repositories in the undercloud and overcloud will be deleted during these steps.

   - Access the Contrail update package.

   - Copy the SNMP packages into the repository:

     - net-snmp-5.7.2-37.el7.x86_64.rpm
     - net-snmp-agent-libs-5.7.2-37.el7.x86_64.rpm
     - net-snmp-libs-5.7.2-37.el7.x86_64.rpm
     - net-snmp-utils-5.7.2-37.el7.x86_64.rpm

     In the provided example, all 4 of these files are in the `/mnt/net-snmp/` directory and all files from the directory are copied into the repository.

   - Unsubscribe every node with all registered satellite server repositories.

   - Delete all repositories on undercloud and overcloud nodes, and replace these deleted repositories with a Contrail repository.

   - Clean the yum cache, verify the repository list, and check for yum updates.

   A sample procedure:

```
[stack@undercloud ~]#
sudo su -
cd /var/www/html/contrail
rm -rf /var/www/html/contrail/*
#enter the location of the contrail update package
tar -xzvf /mnt/contrail-install-packages_4.1.3.0-30-newton.tgz
#copy prerequisite snmp packages; in this setup packages are in /mnt/net-snmp/
cp /mnt/net-snmp/* .
rm -rf /var/www/html/contrail/repodata/usr/bin/createrepo /var/www/html/contrail/subscription-
manager repos --disable=*subscription-manager unregister
```

```
rm -f /etc/yum.repos.d/*
#create local repo file
echo -e '[Contrail]\nname=Contrail Repo\nbaseurl=http://192.168.24.1/contrail
\nenabled=1\ngpgcheck=0' > /etc/yum.repos.d/contrail.repo
# disable yum plugins
sed -i 's/plugins=1/plugins=0/g' /etc/yum.conf
yum clean all
rm -rf /var/cache/yum/*
yum check-update
exit
yum repolist

[stack@undercloud ~]#
 . stackrc;for ipnode in $(nova list | sed '4,$ !d;$d'| awk -F 'ctlplane=' '{print $2}' | tr -
d '|');
do echo "Node $ipnode";
echo "sudo subscription-manager repos --disable=*;
sudo subscription-manager unregister;
sudo rm -f /etc/yum.repos.d/*;
sudo echo -e '[Contrail]\nname=Contrail Repo\nbaseurl=http://192.168.24.1/contrail
\nenabled=1\ngpgcheck=0' > /tmp/contrail.repo;
sudo mv /tmp/contrail.repo /etc/yum.repos.d/;
sudo sed -i 's/plugins=1/plugins=0/g' /etc/yum.conf;
sudo yum clean all;sudo rm -rf /var/cache/yum/*;
sudo yum repolist;sudo yum check-update" | ssh heat-admin@$ipnode bash;
done
```

2. Check the undercloud Contrail packages versions for `contrail-tripleo-puppet`, `puppet-contrail`, and `contrail-tripleo-heat-templates`. The newest versions of those packages must be installed before proceeding with the overcloud upgrade. See the following example, with current package versions.

```
[stack@undercloud~]$ rpm -qa | grep contrail

puppet-contrail-4.1.3.0-NN.el7.noarch
contrail-tripleo-heat-templates-4.1.3.0-NN.el7.noarch
contrail-tripleo-puppet-4.1.3.0-NN.el7.noarch
```

Upgrade the Contrail Heat Templates

You must copy the new Contrail Heat templates package to the undercloud node, while retaining a copy of the Heat templates that were used for the existing deployment.

1. Make a copy of all of the Heat templates that were used for deployment and save the copies, because the existing files will be overwritten by the new versions. The templates to copy are of the form contrail-services.yaml, contrail-net.yaml, and so on.

   NOTE: Red Hat does not support changing IP address of the existing cluster as a part of upgrade. Do not change IP address of the cluster while creating new tripleo-heat-templates

2. Copy the new contrail-tripleo-heat templates to the undercloud node.

```
cp /home/stack/tripleo-heat-templates /home/stack/tripleo-heat-templates-bk

sudo cp -r /usr/share/contrail-tripleo-heat-templates/environments/contrail /home/stack/
tripleo-heat-templates/environments/

sudo cp -r /usr/share/contrail-tripleo-heat-templates/puppet/services/network/* /home/stack/
tripleo-heat-templates/puppet/services/network
```

   NOTE: The directory /home/stack/tripleo-heat-templates is user defined, it can be *User Defined-directory>/openstack-tripleo-heat-templates*

Modify the Yum Update Script for TripleO Puppet

Following Puppet commands must be added to the `yum_update` script before starting the upgrade. The script is located at:

/home/stack/tripleo-heat-templates/extraconfig/tasks/yum_update.sh

1. Update the following Puppet commands in the `yum_update.sh` after the line “`echo -n "false" > $heat_outputs_path.update_managed_packages`”.

   Refer to the following patch for details regarding the exact placement of the commands patch: https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum_updates.patch

```
yum install -y contrail-tripleo-puppet puppet-contrail

rsync -a /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/tripleo/
```

2. Update the fields `*contrail version` and `*contrail repo` in `contrail-services.yaml`.

   Default parameter for `contrailVersion` is 4.

Filename:**/home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml**.

Add the following parameters:

ContrailVersion: 4

ContrailRepo : *<location of the contrail-41 repo>*

NOTE: **/home/stack/tripleo-heat-templates**directory is user defined and it can be directory name under stack user.

Update the Overcloud Deployment Plan

1. Make a copy of the existing deploy script to the **update-stack.sh** file by re-running the command used for cloud deployment and adding the suffix - `-update-plan-only`.

```
<openstack overcloud deploy> -update-plan-only
```

Example:

```
openstack overcloud deploy --update-plan-only --templates /home/stack/tripleo-heat-templates/
\
   --roles-file /home/stack/tripleo-heat-templates/environments/contrail/roles_data.yaml \
  -e /home/stack/tripleo-heat-templates/environments/puppet-pacemaker.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
  -e /home/stack/tripleo-heat-templates/environments/network-isolation.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
  -e /home/stack/tripleo-heat-templates/environments/ips-from-pool-all.yaml \
  -e /home/stack/tripleo-heat-templates/environments/network-management.yaml \
  -e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/environment-
rhel-registration.yaml \
  -e /home/stack/tripleo-heat-templates/environments/hostname-map.yaml \
  -e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/rhel-
registration-resource-registry.yaml \
  --libvirt-type qemu
```

2. If you are using a local repository for the update and the `environment-rhel-registration.yaml` and `rhel-registration-resource-registry.yaml` files are present, delete these lines from the deploy script:

```
  -e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/environment-
rhel-registration.yaml \
  -e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/rhel-
registration-resource-registry.yaml \
```

3. Prepare the YAML files for the update:

   - Verify each .yaml template referenced in the update-stack.sh file contains the original settings that match the files that were backed up.

   - In the **contrail-net.yaml** file, adapt all referenced templates from **heat_template_version: newton** to **heat_template_version: 2015-04-30**. Keep all other original installation settings in this file.

4. Update the deployment plan.

   `./update-stack.sh`

   Example

```
[stack@undercloud ~]$ ./update-stack.sh
Removing the current plan files
Uploading new plan files
Started Mistral Workflow. Execution ID: 6c8fb5b7-6eda-4d92-8245-f7ac46bb369d
Plan updated
Deploying templates in the directory /tmp/tripleoclient-CdyN2I/tripleo-heat-templates
Overcloud Endpoint: http://10.87.67.232:5000/v2.0
Overcloud Deployed
[stack@undercloud ~]$
```

Upgrade the Overcloud

CAUTION: The steps in this section are service disrupting, and should only be performed within a maintenance window.

1. Update the overcloud stack.

```
$ openstack overcloud update stack -i overcloud
on_breakpoint: [u'overcloud-contrailanalyticsdatabase-0']
Breakpoint reached, continue? Regexp or Enter=proceed (will clear
4386bdc7-5087-4a4d-865c-0b0181ce9345), no=cancel update, C-c=quit interactive mode:
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
COMPLETE
update finished with status COMPLETE
```

2. Verify the overcloud `stack status`, the `contrail-status`, and the `contrail-version` after the upgrade.

Overcloud Stack Status

```
[stack@undercloud]# heat stack-list
WARNING (shell) "heat stack-list" is deprecated, please use "openstack stack list" instead
+--------------------------------------+------------+-----------------+----------------------
+----------------------+
| id                                   | stack_name | stack_status    | creation_time
| updated_time         |
+--------------------------------------+------------+-----------------+----------------------
+----------------------+
| e873706c-7fb3-44ba-80dc-30b0fdbd519e | overcloud  | UPDATE_COMPLETE | 2019-03-13T19:20:52Z
| 2019-03-13T22:01:05Z |
+--------------------------------------+------------+-----------------+----------------------
+----------------------+
[stack@undercloud ~]$
```

Contrail Stack Status

```
sudo for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-
admin@$i  sudo contrail-status; done
```

Contrail Version Check

```
for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@
$i  sudo contrail-version; done
```

RELATED DOCUMENTATION

Upgrade Procedure for RHOSP-based Contrail 3.2.x to Contrail 4.1

Upgrade Procedure for RHOSP-based Contrail 4.1.1 to Contrail 4.1.2

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

IN THIS SECTION

- Prerequisites | 161
- Acquire the Software | 161
- Attach Contrail Packages using Juju | 162
- Upgrade the Contrail Clusters | 162

This section presents the steps to upgrade from an Ubuntu-16.04-based Contrail deployment from Contrail version 4.1.3 to Contrail version 4.1.4 using Juju charms.

Prerequisites

These instructions assume that these requirements for installing Contrail Release 4.1.3 are already present in your environment:

- MaaS Server—MaaS version 2.3 is installed on this server. This procedure was tested using MaaS version 2.3.5.
- Juju Controller—Juju version 2.3 is installed, and the Juju controller is bootstrapped on a VM or a bare metal server. This procedure was tested using Juju version 2.3.7-xenial-amd64.
- A repository to get Netronome, patched Openstack packages, and Contrail vRouter packages is operational.
- A Contrail Controller using Ubuntu 16.04 xenial is operational.
- A Contrail cluster with one or more compute nodes using Agilio SmartNICs.

Acquire the Software

To acquire the Contrail 4.1.4 software images to perform this procedure:

1. Go to the Juniper Networks Support site for Contrail.
2. Select **OS** as *Contrail* and **Version** as *4.1.4* from the drop-down menus.
3. Download the *contrail-cloud-docker_4.1.4.0-63-ocata_xenial.tgz* file.

4. Extract the following images from the *contrail-cloud-docker_4.1.4.0-63-ocata_xenial.tgz* file:

   - Contrail Analytics package: *contrail-analytics-ubuntu16.04-4.1.4.0-63.tar.gz*.
   - Contrail Analytics Database package: *contrail-analyticsdb-ubuntu16.04-4.1.4.0-63.tar.gz* .
   - Contrail Controller package: *contrail-controller-ubuntu16.04-4.1.4.0-63.tar.gz*

   The images need to be extracted because the Contrail Analytics, Contrail Analytics Database, and Contrail Controller packages must be upgraded individually to perform this upgrade.

Attach Contrail Packages using Juju

The Contrail Controller, Contrail Analytics, and Contrail Analytics DB packages need to be attached using Juju to perform this upgrade.

To attach these software packages into Juju:

1. Attach the Contrail Controller, Contrail Analytics, & Contrail Analytics DB packages into Juju:

   ```
   juju attach contrail-analytics contrail-analytics=/home/jenkins/docker/contrail-analytics-
   ubuntu16.04-4.1.4.0-63.tar.gz
   juju attach contrail-controller contrail-controller=/home/jenkins/docker/contrail-controller-
   ubuntu16.04-4.1.4.0-63.tar.gz
   juju attach contrail-analyticsdb contrail-analyticsdb=/home/jenkins/docker/contrail-
   analyticsdb-ubuntu16.04-4.1.4.0-63.tar.gz
   ```

2. Check status of the software image attachments into Juju using the `juju status` command.

   Wait for the `juju status` command output to indicate that the upgrade is successful. The output in the `juju status` should indicate that all processes are *Active* and all machine states are *started*.

Upgrade the Contrail Clusters

This section provides the steps to update the Contrail clusters for this upgrade.

It includes the following sections:

Upgrade the Contrail Controllers

The Contrail controllers must be upgraded one by one to complete this procedure.

To upgrade the Contrail controllers:

1. SSH into the Contrail controller server and decommission the Contrail controller from the Cassandra cluster:

   ```
   sudo docker exec -it contrail-controller /usr/bin/nodetool decommission
   ```

2. Remove the Contrail Controller container:

   ```
   sudo docker rm -f contrail-controller
   ```

3. Update the hooks to the Contrail Controller from the Juju Controller:

   ```
   juju run --application contrail-controller hooks/update-status
   ```

4. Wait for the Contrail status for all packages on the upgrading node to change to *active*. This step can take up to 10 minutes.

   Enter the **contrail-status** command to check status. All packages in the *Contrail Control* section of the output must move to the *active* state before proceeding.

5. Check Juju status by entering the `juju status` command.

   All Contrail components in this output should be in the *active* state.

6. After each controller update, check the controllers to make sure the databases are consistent across all controllers:

   - Enter the **nodetool describecluster** command. Confirm that the *schema version* output is identical on all 3 controllers.

   - Enter the **echo stat | nc localhost 2181** command. The *node count* output should be identical on all 3 controllers.

   - Ensure that the **contrail-status** output is *active* for all components in all 3 controllers.

   If your upgrade is not successful after 15 minutes, retry steps 1 through 5.

   If you need to decommission a node that is not upgrading successfully, use the **nodetool removenode *node-ID*** command.

7. Repeat steps 1 through 6 for all other Contrail controller nodes.

Upgrade Contrail Analytics Nodes

To upgrade the Contrail Analytics nodes:

1. SSH into the first Contrail Analytics node and remove the Contrail Analytics container:

   ```
   sudo docker rm -f contrail-analytics
   ```

2. Confirm Juju status using the `juju status` command.

   The output in the `juju status` should indicate that all processes are *Active* and all machine states are *started*.

3. From the MaaS server, update hooks to the Contrail Analytics controller:

   ```
   juju run --application contrail-analytics/0 hooks/update-status
   ```

4. Wait for the Contrail status for all packages on the upgrading node to change to *active*. This step can take up to 10 minutes.

   Enter the **contrail-status** command to check status. All packages in the *Contrail Analytics* section of the output must move to the *active* state before proceeding.

5. Repeat steps 1 through 4 for all other Contrail Analytics nodes.

Upgrade Analytics Database Nodes

To upgrade the Contrail Analytics database nodes:

1. SSH into a Contrail analytics database server and decommission the node from the Cassandra cluster:

   ```
   sudo docker exec -it contrail-analyticsdb /usr/bin/nodetool decommission
   ```

2. Remove the AnalyticsDB container:

   ```
   sudo docker rm -f contrail-analyticsdb
   ```

3. From the Juju controller, update the hooks to the Contrail Analytics DB controller:

```
juju run --application contrail-analyticsdb hooks/update-status
```

4. Wait for the Contrail status for all packages on the upgrading node to change to *active*. This step can take up to 10 minutes.

   Enter the **contrail-status** command to check status. All packages in the *Contrail Database* section of the output must move to the *active* state before proceeding.

5. Check Juju status by entering the `juju status` command.

   All Contrail components in this output should be in the *active* state.

6. After each analytics database node update, check the nodes to ensure the databases are consistent inside the contrail analytics database containers:

   - Enter the **nodetool describecluster** command. Confirm that the *schema version* output is identical on all 3 nodes.

   - Enter the **echo stat | nc localhost 2181** command. The *node count* output should be identical on all 3 nodes.

   - Ensure that the **contrail-status** output is *active* for all components in all 3 contrail analytics db nodes.

   If your upgrade is not successful after 15 minutes, retry steps 1 through 5.

   If you need to decommission a node that is not upgrading successfully, use the **nodetool removenode *node-ID*** command.

7. Repeat steps 1 through 6 for all other Contrail Analytics database nodes.

Updating the Neutron Plugin and the vRouter Agent

The process for updating the neutron plugin and the vRouter agent is different for compute nodes than it is for other nodes.

This section covers both procedures and includes these sections:

Updating the Neutron Plugin and the vRouter Agent on Non-Compute Nodes

Use this procedure to update the Neutron Plugin and the vRouter agent on all non-compute nodes in your environment:

NOTE: This procedure assumes that the APT Get repository was created during the previous installation, and that the latest Contrail packages can be placed into the repository.

1. SSH into the Neutron API plugin unit.

2. From the Neutron API plugin unit, get the latest APT Get update:

```
sudo apt-get update
```

3. Upgrade APT GET:

```
sudo apt-get upgrade
```

NOTE: This step shows how to upgrade APT get for all packages. You can also manually update the neutron-plugin-contrail and python-contrail packages to complete this step, if you'd rather not perform the complete upgrade. This procedure does not provide the steps to manually update these packages.

4. Restart the Neutron service:

```
sudo systemctl restart neutron-server.service
```

Updating the Neutron Plugin and the vRouter Agent on Compute Nodes

Use this procedure to update the Neutron Plugin and the vRouter agent on all compute devices in your environment:

NOTE: This procedure assumes that the APT Get repository was created during the previous installation, and that the latest Contrail packages can be placed into the repository.

1. SSH into the Neutron API plugin unit.

2. From the Neutron API plugin unit, get the latest APT Get update:

```
sudo apt-get update
```

3. Upgrade APT GET:

```
sudo apt-get upgrade
```

NOTE: This step shows how to upgrade APT get for all packages. You can also manually update the following packages to complete this step:

- contrail-lib
- contrail-nodemgr
- contrail-setup
- contrail-utils
- contrail-vrouter-agent
- contrail-vrouter-common
- contrail-vrouter-dkms
- contrail-vrouter-init
- contrail-vrouter-utils
- python-contrail
- python-contrail-vrouter-api
- python-opencontrail-vrouter-netns

This procedure does not provide the steps to manually update these packages.

4. Upgrade the vRouter agent and, if using Netronome SmartNICs, the netronome plugin.

   - If you are performing this procedure on a compute node without a Netronome SmartNIC:

NOTE: The network connection over the vhost is down while this procedure is performed. Traffic will be lost.

a. Stop the Contrail vRouter agent:

```
sudo systemctl stop contrail-vrouter-agent
```

b. Remove the Contrail vRouter module:

```
sudo rmmod vrouter
```

c. Insert the vRouter module:

```
sudo insmod /lib/modules/4.4.0-116-generic/updates/dkms/vrouter.ko
```

d. Activate the vhost:

```
sudo ifup vhost0
```

e. Restart the Contrail vRouter agent:

```
sudo systemctl start contrail-vrouter-agent
```

- If you are performing this procedure on a compute node with a Netronome SmartNIC:

NOTE: The network connection over the vhost is down while this procedure is performed. Traffic will be lost.

a. Stop the Contrail vRouter agent:

```
sudo systemctl stop contrail-vrouter-agent
```

b. Stop the Virtio forwarder module:

```
sudo systemctl stop virtio-forwarder
```

c. Stop the vRouter control module:

```
sudo /opt/netronome/bin/ns-vrouter-ctl stop
```

d. Restart the Virtio forwarder module:

```
sudo systemctl start virtio-forwarder
```

e. Restart the Contrail vRouter agent:

```
sudo /opt/netronome/bin/ns-vrouter-ctl start
```

f. Activate the vhost:

```
sudo ifup vhost0
```

g. Restart the Contrail vRouter agent:

```
sudo systemctl start contrail-vrouter-agent
```

5. Verify Contrail status:

```
sudo contrail-status
```

All packages in the *Contrail vRouter* section of the output should be in the *active* state. This step can take several minutes.

RELATED DOCUMENTATION

Deploying Contrail Release 4.1 with Netronome SmartNICs by Using Juju

Upgrade Procedure for RHOSP-based Contrail 4.1.3 to Contrail 4.1.4

IN THIS SECTION

- Prerequisites | 170
- Post-Installation | 171
- Acquire the Software | 172
- Upgrade the Undercloud | 172
- Update Red Hat Director Image Archives | 177
- Upgrade the Operating System on Contrail Nodes | 179
- Prepare the Contrail Packages | 180
- Upgrade the Contrail Heat Templates | 180
- Modify the Yum Update Script for TripleO Puppet | 181
- Update the Overcloud Deployment Plan | 182
- Upgrade the Overcloud | 183
- Upgrade Cautions | 184

This section presents the steps to upgrade a RHOSP-based Contrail deployment from Contrail version 4.1.3 to Contrail version 4.1.4.

Prerequisites

Ensure you have a cloud up and running with RHOSP10 and Contrail 4.1.3 before you proceed with the upgrade procedure.

This procedure has been validated with the following Contrail, Red Hat, and OpenStack versions.

Table 23: Pre-Installation Software Versions

Contrail Version	Red Hat Version	OpenStack Version
4.1.3	RHEL 7.6 (3.10.0-957.el7.x86_64)	RHOSP10 (packages dated October 29, 2018)
4.1.3	RHEL 7.5 (3.10.0-862.11.6.el7.x86_64)	RHOSP10 (packages dated October 29, 2018)

CAUTION: Set the Red Hat Satellite filter end date to December 9, 2019 before proceeding with the upgrade.

Post-Installation

After the installation, you'll have a cloud networking running RHOSP10 and Contrail 4.1.4. The Red Hat Enterprise Linux (RHEL) kernel version updates to 7.7 during this procedure.

Table 24 on page 171 summarizes the post-installation software versions.

Table 24: Post Installation Software Summary

Contrail Version	Red Hat Version	OpenStack Version
4.1.4	RHEL 7.7 (3.10.0-1062.el7.x86_64) RHEL 7.7 (3.10.0-1062.1.2.el7.x86_64) RHEL 7.7 (3.10.0-1062.9.1.el7.x86_64)	RHOSP10 (packages dated December 9, 2019)

Contrail version R4.1.4 supports net-snmp package version 5.7.2-43 to support SNMP. The net-snmp packages come from Red Hat, with the exception of the *net-snmp-python-5.7.2-43.el7.x86_64.rpm* package which is provided in the Contrail repository.

Table 25 on page 171 summarizes the net-snmp depend packages and their associated repository locations.

Table 25: Post Installation Software Summary

Net-SNMP Depend Packages	Repository
net-snmp-5.7.2-43.el7.x86_64.rpm	Red Hat Satellite
net-snmp-agent-libs-5.7.2-43.el7.x86_64.rpm	Red Hat Satellite
net-snmp-libs-5.7.2-43.el7.x86_64.rpm	Red Hat Satellite

Table 25: Post Installation Software Summary *(Continued)*

Net-SNMP Depend Packages	Repository
net-snmp-python-5.7.2-43.el7.x86_64.rpm	Contrail
net-snmp-utils-5.7.2-43.el7.x86_64.rpm	Red Hat Satellite

Acquire the Software

To download the software images for this procedure:

1. Go to the Juniper Networks Support site for Contrail.

2. Select **OS** as *Contrail* and **Version** as *4.1.4*. Download the images that apply to your environment.

Upgrade the Undercloud

1. Log in to the undercloud as the stack user.

   ```
   $ su - stack
   ```

2. Update the Contrail repositories.

   - Backup the Contrail 4.1.3 packages to a repository with a different name. In this example, the packages are moved to a repository named *contrail-R4-1-3*.

     ```
     [stack@undercloud ~]$ cd /var/www/html/
     [stack@undercloud html]$ sudo mv contrail/ contrail-R4-1-3
     ```

   - Create a new repository directory to store the Contrail 4.1.4 packages:

     ```
     [stack@undercloud html]$ sudo mkdir contrail
     ```

3. Copy the downloaded file—in the provided sample, the file is *contrail-install-packages_4.1.4.0-63-newton.tgz*—to the Contrail repository created in Step 2.

NOTE: This step assumes that you've already downloaded the Contrail software. See "Acquire the Software" on page 172.

```
[stack@undercloud contrail]$ ls -lrt
    total 377104
    -rw-r--r--. 1 root root 386151602 Mar 14 06:58 contrail-install-packages_4.1.4.0-63-
newton.tgz
```

4. Untar the downloaded tgz file.

```
[stack@undercloud contrail]$ sudo tar -xvf contrail-install-packages_4.1.4.0-63-newton.tgz
```

5. Create a repository in the new directory:

```
[stack@undercloud contrail]$ pwd
    /var/www/html/contrail

    [stack@undercloud contrail]$ sudo createrepo .
```

   If the **createrepo** command is not available, download the createrepo package from Red Hat (Red Hat subscription required).

6. (Clusters deployed using Swift Puppet files only) If your Contrail 4.1 cluster was deployed using Swift Puppet, perform these steps:

   a. Remove overcloud artifacts from the undercloud:

```
[stack@undercloud ~]$ swift delete overcloud-artifacts
puppet-modules.tgz
overcloud-artifacts
```

   b. Delete the *deployments-artifacts.yaml* file if the file is present.

```
[stack@undercloud ~]$ ls /home/stack/.tripleo/environments/deployment-artifacts.yaml
[stack@undercloud ~]$ rm -rf /home/stack/.tripleo/environments/deployment-artifacts.yaml
```

c. Clean the repositories and confirm that all repositories are available.

```
[stack@undercloud ~]$ sudo yum clean all
[stack@undercloud ~]$ sudo yum repolist
```

7. Stop the main OpenStack platform services.

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```

8. Update the `python-tripleoclient` package and its dependencies to ensure you have the most current scripts for the minor version update.

```
$ sudo yum update python-tripleoclient
```

9. Upgrade the undercloud.

```
$ openstack undercloud upgrade
```

10. Reboot the node.

```
$ sudo reboot
```

Wait for the node to reboot. The reboot process can take 10 or more minutes to complete.

11. Ensure the undercloud has the latest Contrail R4.1.4 contrail packages:

```
[stack@undercloud ~]$ rpm -qa  | grep contrail

puppet-contrail-4.1.4.0-X.el7.noarch
contrail-tripleo-heat-templates-4.1.4.0-x.el7.noarch
contrail-tripleo-puppet-4.1.4.0-x.el7.noarch
python-gevent-1.1rc5-1contrail1.el7.x86_64
```

12. Ensure the undercloud has the latest RHOSP images:

```
[stack@undercloud]$ rpm -qa  | grep direct

rhosp-director-images-10.0-20180821.1.el7ost.noarch
rhosp-director-images-10.0-20190829.1.el7ost.noarch
rhosp-director-images-10.0-20190918.1.el7ost.noarch
rhosp-director-images-ipa-10.0-20190829.1.el7ost.noarch
rhosp-director-images-ipa-10.0-20180821.1.el7ost.noarch
rhosp-director-images-ipa-10.0-20190918.1.el7ost.noarch
```

13. Review the **ironic node-list** output to confirm the following statuses for each package::

- **Power state** is *power on*.
- **Provision State** is *active*.
- **Maintenance** is *False*.

```
[stack@undercloud ~]$ ironic node-list
+------------------------------+----------+-------------+-------------+
| Name                         | Power    | Provisioning | Maintenance |
|                              | State    | State        |             |
+------------------------------+----------+-------------+-------------+
| controller-3                 | power on | active       | False       |
| compute-5c5s35               | power on | active       | False       |
| contrail-controller1         | power on | active       | False       |
| contrail-analytics1          | power on | active       | False       |
| contrail-controller-3        | power on | active       | False       |
| contrail-controller-2        | power on | active       | False       |
| contrail-analytics-database1 | power on | active       | False       |
| controller-2                 | power on | active       | False       |
| controller1                  | power on | active       | False       |
| compute-5c5s37               | power on | active       | False       |
| compute-5c5s36               | power on | active       | False       |
| contrail-analytics-2         | power on | active       | False       |
| contrail-analytics-3         | power on | active       | False       |
| compute-5c5s38               | power on | active       | False       |
| contrail-analytics-database-3 | power on | active      | False       |
| contrail-analytics-database-2 | power on | active      | False       |
+------------------------------+----------+-------------+-------------+
```

NOTE: This output presentation has been modified for readability. The *UUID* and *Instance UUID* fields were removed as part of this modification.

14. Verify that all OpenStack servers are in the Active state.

```
[stack@undercloud ~]$ openstack server list
+-----------------------------------------------+--------+
| Name                                          | Status |
+-----------------------------------------------+--------+
```

```
| overcloud-contrailanalytics-2-4-1-4-7-7         | ACTIVE |
| overcloud-controller-0-4-1-4-7-7                | ACTIVE |
| overcloud-contrailanalytics-0-4-1-4-7-7         | ACTIVE |
| overcloud-contrailanalyticsdatabase-2-4-1-4-7-7 | ACTIVE |
| overcloud-contrailanalytics-1-4-1-4-7-7         | ACTIVE |
| overcloud-contrailanalyticsdatabase-0-4-1-4-7-7 | ACTIVE |
| overcloud-contrailcontroller-1-4-1-4-7-7        | ACTIVE |
| overcloud-contrailanalyticsdatabase-1-4-1-4-7-7 | ACTIVE |
| overcloud-contrailcontroller-2-4-1-4-7-7        | ACTIVE |
| overcloud-contrailcontroller-0-4-1-4-7-7        | ACTIVE |
| compute-0-4-1-4-rhel-7-7                        | ACTIVE |
| overcloud-contraildpdk-0-4-1-4-7-7              | ACTIVE |
| overcloud-contraildpdk-1-4-1-4-7-7              | ACTIVE |
| compute-1-4-1-4-rhel-7-7                        | ACTIVE |
+-------------------------------------------------+--------+
```

NOTE: This output presentation has been modified for readability. The *ID*, *Image Name*, and *Networks* fields were removed as part of this modification.

15. If new image archives are available, replace your current images with the new images.

    Before uploading the new images onto the undercloud node, move any existing images from the images directory on the stack user's home directory (/home/stack/images).

```
$ mv /home/stack/images /home/stack/images-old
```

16. Extract the new image archives.

```
mkdir images
cd images
for i in /usr/share/rhosp-director-images/overcloud-full-latest-10.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

17. Import the new image archives into the undercloud and configure the nodes to use the new images.

```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

18. Verify that the images are uploaded:

```
$ glance image-list
```

19. Observe the contrail-status on all Contrail nodes. All services in the Contrail nodes, except the controller (OpenStack), should be up and running before proceeding with the upgrade.

```
[stack@undercloud ~]$ source stackrc
[stack@undercloud ~]$ for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '='
-f2); do ssh heat-admin@$i sudo contrail-status; done
```

20. Ensure that all overcloud node contrail repository pointers are properly pointing to the contrail repository.

    Contrail Analytics Example:

```
[root@overcloud-contrailanalytics-0 heat-admin]# cat     /etc/yum.repos.d/contrail.repo
    [Contrail]
    name=Contrail Repo
    baseurl=http://192.168.24.1/contrail
    enabled=1
    gpgcheck=0
    protect=1
    metadata_expire=30
```

Update Red Hat Director Image Archives

The undercloud update process might download new image archives from the rhosp-director images and the rhosp-director-ipa packages. You will have to update your existing system with any new image archives.

1. Check the yum log to determine if new image archives are available.

   $ sudo grep "rhosp-director-images" /var/log/yum.log

```
[stack@undercloud]$ sudo grep "rhosp-director-images" /var/log/yum.log
```

```
Dec 12 15:09:20 Installed: rhosp-director-images-ipa-10.0-20190918.1.el7ost.noarch
Dec 12 15:10:10 Installed: rhosp-director-images-10.0-20190918.1.el7ost.noarch
```

2. If new image archives are available, replace your current images with the new images. Before deploying any new images, remove any existing images from the images undercloud on the stack user's home (/home/stack/images).

```
$ rm -rf ~/images/*
```

3. Extract the new image archives.

```
mkdir  images
    cd images
for i in /usr/share/rhosp-director-images/overcloud-full-latest-10.0.tar /usr/share/rhosp-
director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

4. Install the Contrail package on the overcloud image by using the virt-customize command.

```
export LIBGUESTFS_BACKEND=direct  /usr/bin/virt-customize  -a /home/stack/images/overcloud-
full.qcow2 \
-copy-in /etc/yum.repos.d/mylocalrepo.repo:/etc/yum.repos.d \
-run-command 'yum -y install puppet-tripleo contrail-tripleo-puppet puppet-contrail'\
-run-command ' cp -r /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/
tripleo/ ' \
-run-command 'rm -fr /var/cache/yum/*' \
-run-command 'yum clean all' \  -selinux-relabel
```

5. Import the new image archives into the undercloud and configure nodes to use the new images.

```
$ openstack overcloud image upload -update-existing -image-path /home/stack/images/
```

6. Verify that the images are uploaded.

```
$ openstack image list
```

```
+--------------------------------------+------------------------+--------+
| ID                                   | Name                   | Status |
+--------------------------------------+------------------------+--------+
| a518a08c-3c49-47b7-9705-a7da5c90dcc6 | bm-deploy-ramdisk      | active |
| b7d75ff7-926b-426c-aa2e-424f5d9f8328 | bm-deploy-kernel       | active |
| cd7eedc0-2ed2-4a21-8359-9d8ee89374ac | overcloud-full         | active |
| 9c9e46f4-d571-49fb-a485-4653b6a52b44 | overcloud-full-initrd  | active |
| 740b6cb9-5757-475b-896b-49c526807671 | overcloud-full-vmlinuz | active |
+--------------------------------------+------------------------+--------+
```

7. Show the details of the new image that has been created. The new image will be used to add a new node in the overcloud.

```
$ openstack image show overcloud-full
```

8. Verify `contrail-status` on all Contrail nodes. All services in the Contrail nodes, except the controller (OpenStack), should be up and running before proceeding with the upgrade.

```
[stack@undercloud ~]$ for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done
```

Upgrade the Operating System on Contrail Nodes

To upgrade the operating system on Contrail nodes:

1. Define a list ($iplist) that contains all Contrail nodes. Run the following command on undercloud VM as a stack user.

```
Iplist=" @IPcontrailController1 @IPContrailController2 …"
```

> **CAUTION**: Attach the new satellite subscription key on all overcloud nodes before upgrading the overcloud packages. Satellite must be synced with filter end date *9th Dec 2019*. Make sure to clear cache by typing **sudo yum clean all**.

2. Upgrade the operating system for all nodes in the iplist.

   Run the following command on undercloud VM as a stack user:

```
sudo for ipnode in $iplist; do echo -e "\n\n\t******upgrade node : $ipnode ******" && ssh heat-admin@$ipnode 'sudo yum update -y --disablerepo=*contrail* --skip-broken && exit' ; done
```

3. (Compute nodes only) Reboot overcloud contrail compute nodes. After the reboot, stop the supervisor-vrouter service.

   This step needs to be performed before installing contrail packages on the compute VM.

   Compute services may go down after rebooting with the latest kernel. These services return later in this procedure during the **openstack overcloud deploy** process.

   Reboot Procedure:

```
[root@compute-1-7-6 modules]# sudo reboot
Connection to 192.0.2.16 closed by remote host.
Connection to 192.0.2.16 closed.
```

Post-Reboot:

```
[stack@undercloud-R4-1-2-b22 ~]$ ssh heat-admin@192.0.2.16
Warning: Permanently added '192.0.2.16' (ECDSA) to the list of known hosts.
Last login: Sat Dec  7 03:46:07 2019 from gateway
[heat-admin@compute-1-7-6 ~]$ sudo su
[root@compute-1-7-6 heat-admin]# contrail-status
vRouter is NOT PRESENT

== Contrail vRouter ==
supervisor-vrouter:          active
contrail-vrouter-agent       initializing
contrail-vrouter-nodemgr     initializing
```

Stop the supervisor-vrouter service:

```
[root@compute-1-7-6 heat-admin]# service supervisor-vrouter stop
Stopping supervisor-vrouter (via systemctl): [  OK  ]

[root@compute-1-7-6 heat-admin]# contrail-status
vRouter is NOT PRESENT

== Contrail vRouter ==
supervisor-vrouter: inactive
unix:///var/run/supervisord_vrouter.sockno
```

Prepare the Contrail Packages

Check the undercloud Contrail packages versions for `contrail-tripleo-puppet`, `puppet-contrail`, and `contrail-tripleo-heat-templates`.

```
[stack@undercloud~]$ rpm -qa | grep contrail
```

Upgrade the Contrail Heat Templates

You must copy the new Contrail Heat templates package to the undercloud node, while retaining a copy of the Heat templates that were used for the existing deployment.

1. Make a copy of all of the Heat templates that were used for deployment and save the copies, because the existing files will be overwritten by the new versions. The templates to copy are of the form contrail-services.yaml, contrail-net.yaml, and so on.

   NOTE: Red Hat does not support changing IP address of the existing cluster as a part of upgrade. Do not change IP address of the cluster while creating new tripleo-heat-templates

2. Copy the new contrail-tripleo-heat templates to the undercloud node.

```
sudo cp -r /usr/share/contrail-tripleo-heat-templates/environments/contrail /home/stack/
tripleo-heat-templates/environments/

sudo cp -r /usr/share/contrail-tripleo-heat-templates/puppet/services/network/* /home/stack/
tripleo-heat-templates/puppet/services/network
```

   NOTE: The directory /home/stack/tripleo-heat-templates is user defined, it can be *User Defined-directory>/openstack-tripleo-heat-templates*

Modify the Yum Update Script for TripleO Puppet

Following Puppet commands must be added to the `yum_update` script before starting the upgrade. The script is located at:

/home/stack/tripleo-heat-templates/extraconfig/tasks/yum_update.sh

1. Update the following Puppet commands in the `yum_update.sh` after the line “`echo -n "false" > $heat_outputs_path.update_managed_packages`”.

   Refer to the following patch for details regarding the exact placement of the commands patch: https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum_updates.patch

```
yum install -y contrail-tripleo-puppet puppet-contrail

rsync -a /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/tripleo/
```

2. Update the fields `*contrail version` and `*contrail repo` in `contrail-services.yaml`.

   Default parameter for `contrailVersion` is 4.

Filename:**/home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml**.

Add the following parameters:

ContrailVersion: 4

ContrailRepo : *<location of the contrail-41 repo>*

> **NOTE**: **/home/stack/tripleo-heat-templates**directory is user defined and it can be directory name under stack user.

Update the Overcloud Deployment Plan

1. Update the current plan by re-running the command used for cloud deployment and adding the suffix `- -update-plan-only`.

   ```
   openstack overcloud deploy -update-plan-only
   ```

   Example:

   ```
   openstack overcloud deploy --update-plan-only --templates /home/stack/tripleo-heat-templates/
   \
     --roles-file /home/stack/tripleo-heat-templates/environments/contrail/roles_data.yaml \
     -e /home/stack/tripleo-heat-templates/environments/puppet-pacemaker.yaml \
     -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
     -e /home/stack/tripleo-heat-templates/environments/contrail/network-isolation.yaml \
     -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
     -e /home/stack/tripleo-heat-templates/environments/ips-from-pool-all.yaml \
     -e /home/stack/tripleo-heat-templates/environments/network-management.yaml \
     -e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/environment-
   rhel-registration.yaml \
     -e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/rhel-
   registration-resource-registry.yaml \
     --libvirt-type qemu
   ```

2. Make a copy of the existing deploy script to the `update-stack.sh`. The `update-stack.sh` is the script used to update the overcloud plan, and it references the same templates that were used to deploy the stack. All files used for the overcloud update should be identical to the files used for deployment, except `contrail-services` file that was updated with the latest `contrail-version` and `contrail-repo`.

   ```
   cp deploy.sh update-stack.sh
   ```

3. Update the deployment plan.

   `./update-stack.sh`

   Example:

```
[stack@undercloud ~]$ ./update-stack.sh
      nRemoving the current plan files
      Uploading new plan files
      Started Mistral Workflow. Execution ID: 998a1b40--a034-8cff453acfb1
      Plan updated
      Deploying templates in the directory /tmp/tripleoclient-JulIDe/tripleo-    heat-
templates
      Overcloud Endpoint: http://10.0.0.35:5000/v2.0
      Overcloud Deployed
```

Upgrade the Overcloud

> **CAUTION**: The steps in this section are service disrupting, and should only be performed within a maintenance window.

1. Update the overcloud stack.

```
$ openstack overcloud update stack -i overcloud
on_breakpoint: [u'overcloud-contrailanalyticsdatabase-0']
Breakpoint reached, continue? Regexp or Enter=proceed (will clear
4386bdc7-5087-4a4d-865c-0b0181ce9345), no=cancel update, C-c=quit interactive mode:
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
COMPLETE
update finished with status COMPLETE
```

2. Verify the overcloud `stack status`, the `contrail-status`, and the `contrail-version` after the upgrade.

Overcloud Stack Status

```
[stack@undercloud ~]$ openstack stack list
+------------+-----------------+---------------------+----------------------+
| Stack Name | Stack Status    | Creation Time        | Updated Time         |
+------------+-----------------+---------------------+----------------------+
| overcloud  | UPDATE_COMPLETE | 2019-12-06T23:30:26Z | 2019-12-09T22:40:01Z |
+------------+-----------------+---------------------+----------------------+
```

NOTE: The **openstack stack list** output presentation has been modified for readability. The *ID* field was removed as part of this modification.

Contrail Stack Status

```
sudo for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-
admin@$i sudo contrail-status; done
```

Contrail Version Check

```
for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@
$i sudo contrail-version; ssh heat-admin@$i sudo contrail-status -d ;  done
```

Upgrade Cautions

CAUTION: The steps to perform the overcloud upgrade are service disrupting, and should only be performed within a maintenance window.

The upgrade procedure may fail due to packages conflicts in Contrail analytics nodes. Some observed failures due to packages conflicts are detailed in this section. Continue with the deployment after applying the recommended solution.

Analytics Node snmp-lib Version Conflict

Error message: `Protected multilib versions: 1:net-snmp-libs-5.7.2-37.el7.x86_64 != 1:net-snmp-libs-5.7.2-33.el7_5.2.i686`

Solution:

```
rpm -e --nodeps net-snmp-libs
```

Services Need Manual Restart After Upgrade

Services may need to be restarted after performing this upgrade. The services might continue to run using Contrail 4.1.3-related processes for a period of time.

Enter the **contrail-status** command to see if the processes continued to run through the upgrade, and monitor the warning messages that appear.

Manually restart the services if you run into this issue.

In the following example, this issue is seen for the Contrail Analytics services immediately after the upgrade:

```
[heat-admin@overcloud-contrailanalytics ~]$ sudo contrail-status -d
Warning: supervisor-analytics.service changed on disk. Run 'systemctl daemon-reload' to reload
units.
== Contrail Analytics ==
supervisor-analytics:      active
contrail-alarm-gen         active      pid 975462, uptime 15 days, 19:07:11
contrail-analytics-api     active      pid 127224, uptime 20 days, 19:48:28
contrail-analytics-nodemgr active      pid 127219, uptime 20 days, 19:48:28
contrail-collector         active      pid 127222, uptime 20 days, 19:48:28
contrail-query-engine      active      pid 127223, uptime 20 days, 19:48:28
contrail-snmp-collector    active      pid 127220, uptime 20 days, 19:48:28
contrail-topology          active      pid 127221, uptime 20 days, 19:48:28
```

Dynamic Kernel Module Support (DKMS) for vRouter

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

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

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

For more information about DKMS, refer to:

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

3

PART

Configuring Contrail

Configuring Virtual Networks | 188

Example of Deploying a Multi-Tier Web Application Using Contrail | 220

Configuring Services | 235

Configuring Service Chaining | 256

Examples: Configuring Service Chaining | 303

CHAPTER 8

Configuring Virtual Networks

IN THIS CHAPTER

- Creating Projects in OpenStack for Configuring Tenants in Contrail | **188**
- Creating a Virtual Network with Juniper Networks Contrail | **190**
- Creating a Virtual Network with OpenStack Contrail | **194**
- Creating an Image for a Project in OpenStack Contrail | **196**
- Creating a Floating IP Address Pool | **200**
- Using Security Groups with Virtual Machines (Instances) | **202**
- Support for IPv6 Networks in Contrail | **206**
- Configuring EVPN and VXLAN | **210**

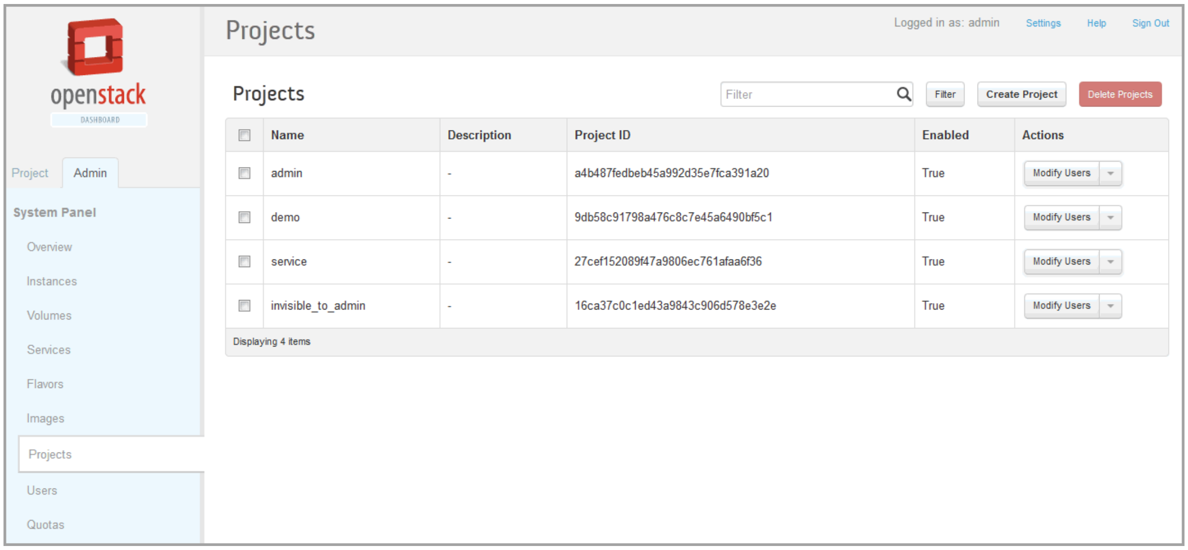
Creating Projects in OpenStack for Configuring Tenants in Contrail

In Contrail, a tenant configuration is called a project. A project is created for each set of virtual machines (VMs) and virtual networks (VNs) that are configured as a discrete entity for the tenant.

Projects are created, managed, and edited at the OpenStack **Projects** page.

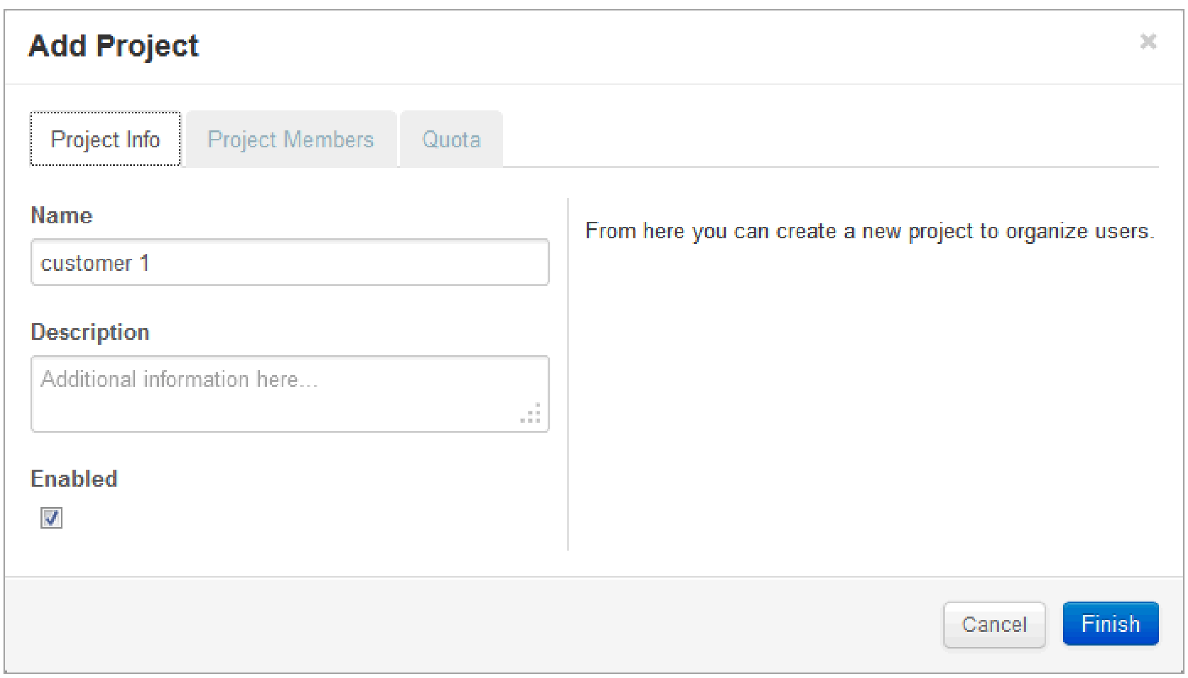
1. Click the **Admin** tab on the OpenStack dashboard, then click the **Projects** link to access the **Projects** page; see Figure 54 on page 189.

Figure 54: OpenStack Projects



2. In the upper right, click the **Create Project** button to access the **Add Project** window; see Figure 55 on page 189.

Figure 55: Add Project



3. In the **Add Project** window, on the **Project Info** tab, enter a **Name** and a **Description** for the new project, and select the **Enabled** check box to activate this project.

4. In the **Add Project** window, select the **Project Members** tab, and assign users to this project. Designate each user as **admin** or as **Member**.

   As a general rule, one person should be a super user in the **admin** role for all projects and a user with a **Member** role should be used for general configuration purposes.

5. Click **Finish** to create the project.

Refer to OpenStack documentation for more information about creating and managing projects.

RELATED DOCUMENTATION

Creating a Virtual Network with Juniper Networks Contrail | **190**
Creating a Virtual Network with OpenStack Contrail | **194**
OpenStack documentation

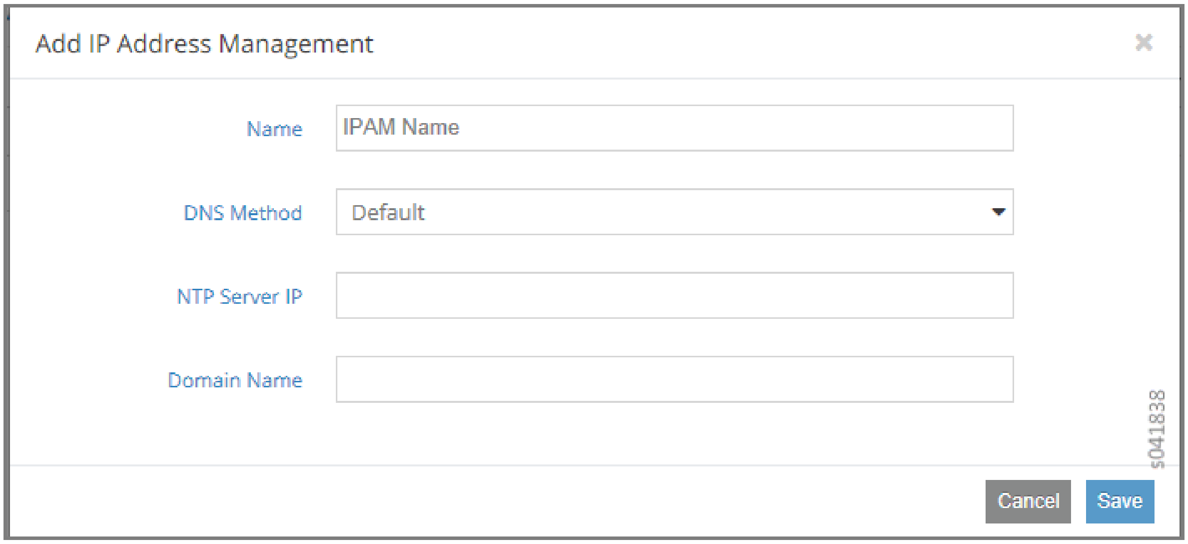
Creating a Virtual Network with Juniper Networks Contrail

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

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

   The **Add IP Address Management** window appears, see Figure 56 on page 191.

Figure 56: Add IP Address Management



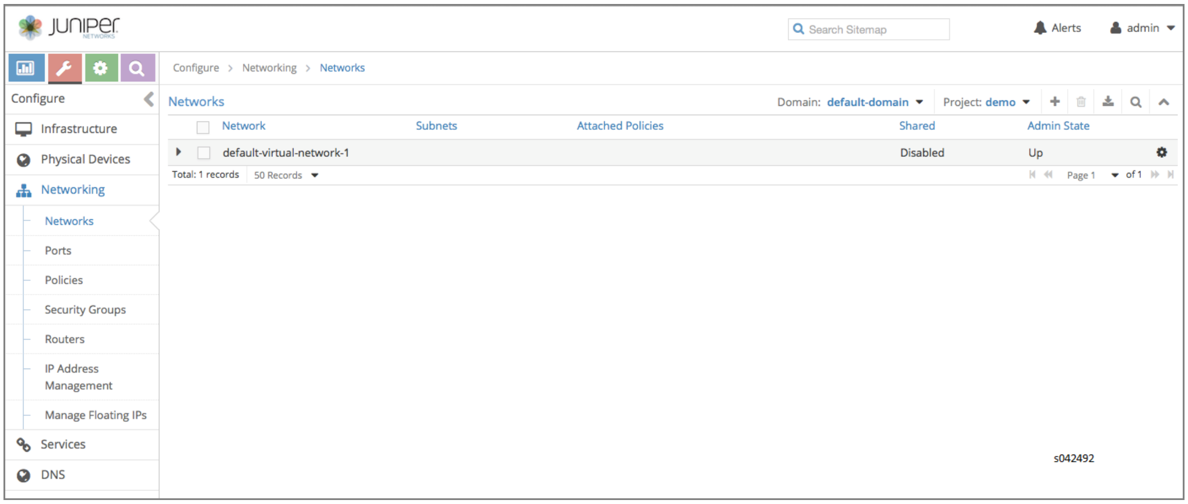
2. Complete the fields in **Add IP Address Management**: The fields are described in Table 26 on page 191.

Table 26: Add IP Address Management Fields

Field	Description
Name	Enter a name for the IPAM you are creating.
DNS Method	Select from a list the domain name server method for this IPAM: **Default, Virtual DNS, Tenant**, or **None.**
NTP Server IP	Enter the IP address of an NTP server to be used for this IPAM.
Domain Name	Enter a domain name to be used for this IPAM.

3. Select **Configure** > **Networking** > **Networks** to access the **Configure Networks** page; see Figure 57 on page 192.

Figure 57: Configure Networks

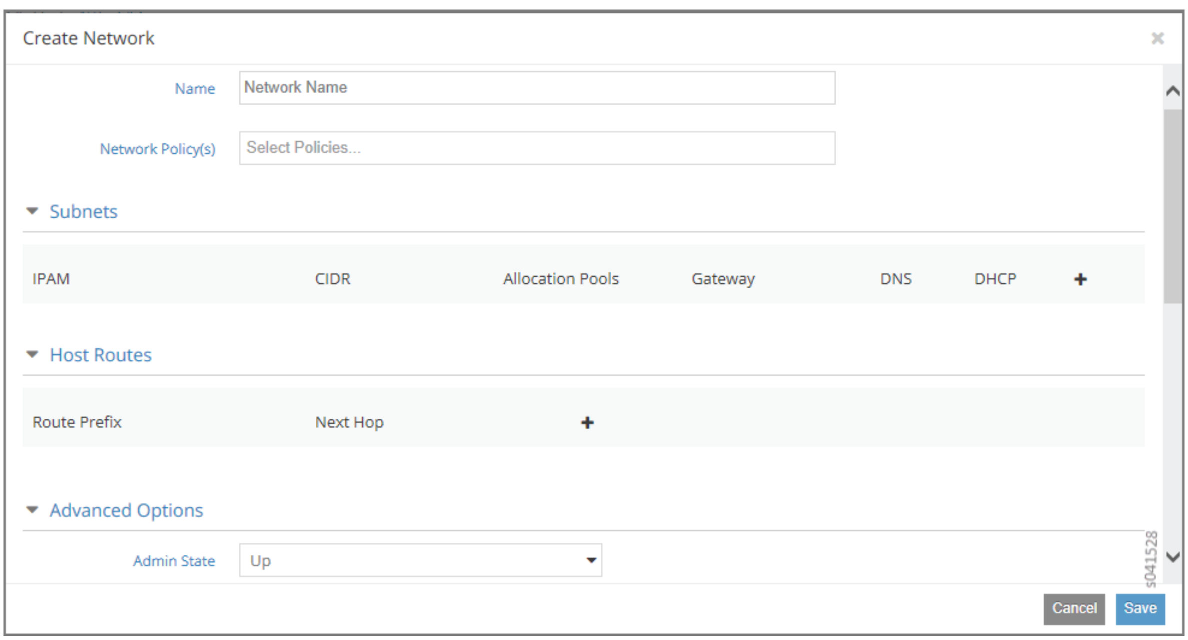


4. Verify that your project is displayed as active in the upper-right field, then click the

   ➕

   icon. The **Create Network** window is displayed. See Figure 58 on page 192. Use the scroll bar to access all sections of this window.

Figure 58: Create Network



5. Complete the fields in the **Create Network** window with values that identify the network name, network policy, and IP options as needed. See field descriptions in Table 27 on page 193.

Table 27: Create Network Fields

Field	Description
Name	Enter a name for the virtual network you are creating.
Network Policy	Select the policy to be applied to this network from the list of available policies. You can select more than one policy by clicking each one needed.
Subnets	Use this area to identify and manage subnets for this virtual network. Click the + icon to open fields for IPAM, CIDR, Allocation Pools, Gateway, DNS, and DHCP. Select the subnet to be added from a drop down list in the IPAM field. Complete the remaining fields as necessary. You can add multiple subnets to a network. When finished, click the + icon to add the selections into the columns below the fields. Alternatively, click the - icon to remove the selections.
Host Routes	Use this area to add or remove host routes for this network. Click the + icon to open fields where you can enter the Route Prefix and the Next Hop. Click the + icon to add the information, or click the - icon to remove the information.
Advanced Options	Use this area to add or remove advanced options, including identifying the Admin State as Up or Down, to identify the network as Shared or External, to add DNS servers, or to define a VxLAN Identifier.
Floating IP Pools	Use this area to identify and manage the floating IP address pools for this virtual network. Click the + icon to open fields where you can enter the Pool Name and Projects. Click the + icon to add the information, or click the - icon to remove the information.
Route Target	Move the scroll bar down to access this area, then specify one or more route targets for this virtual network. Click the + icon to open fields where you can enter route target identifiers. Click the + icon to add the information, or click the - icon to remove the information.

6. To save your network, click the **Save** button, or click **Cancel** to discard your work and start over.

Now you can create a network policy, see Creating a Network Policy—Juniper Networks Contrail.

RELATED DOCUMENTATION

Creating an Image for a Project in OpenStack Contrail | 196

Creating a Virtual Network with OpenStack Contrail

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

1. To create a virtual network when using OpenStack Contrail, select **Project** > **Other** > **Networking**. The **Networks** window is displayed. See Figure 59 on page 194.

Figure 59: Networks Window

2. Verify that the correct project is displayed in the **Current Project** box, then click **Create Network**. The **Create Network** window is displayed. See Figure 60 on page 194 and Figure 61 on page 195.

Figure 60: Create Network Window

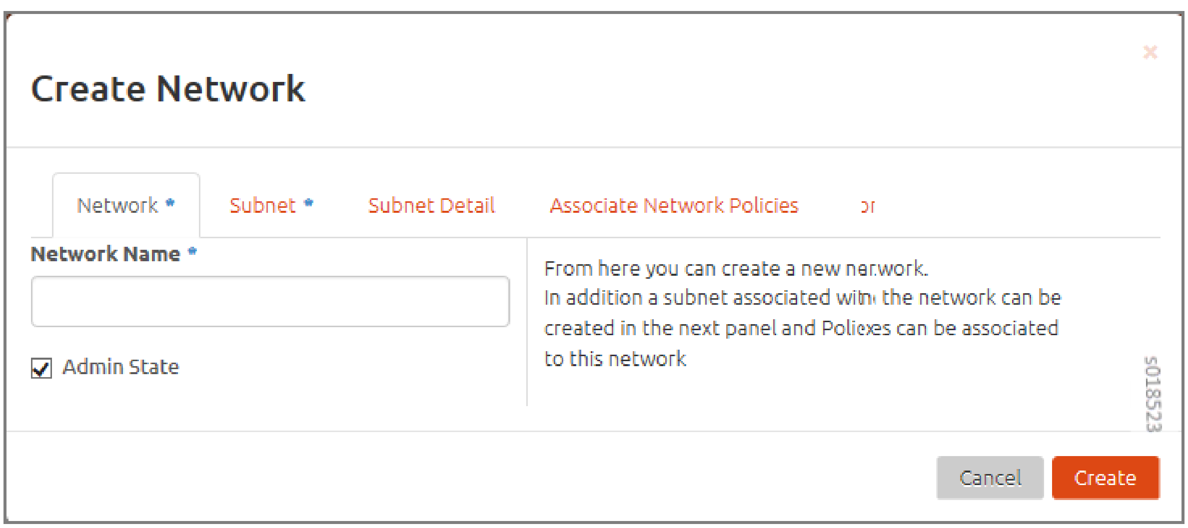
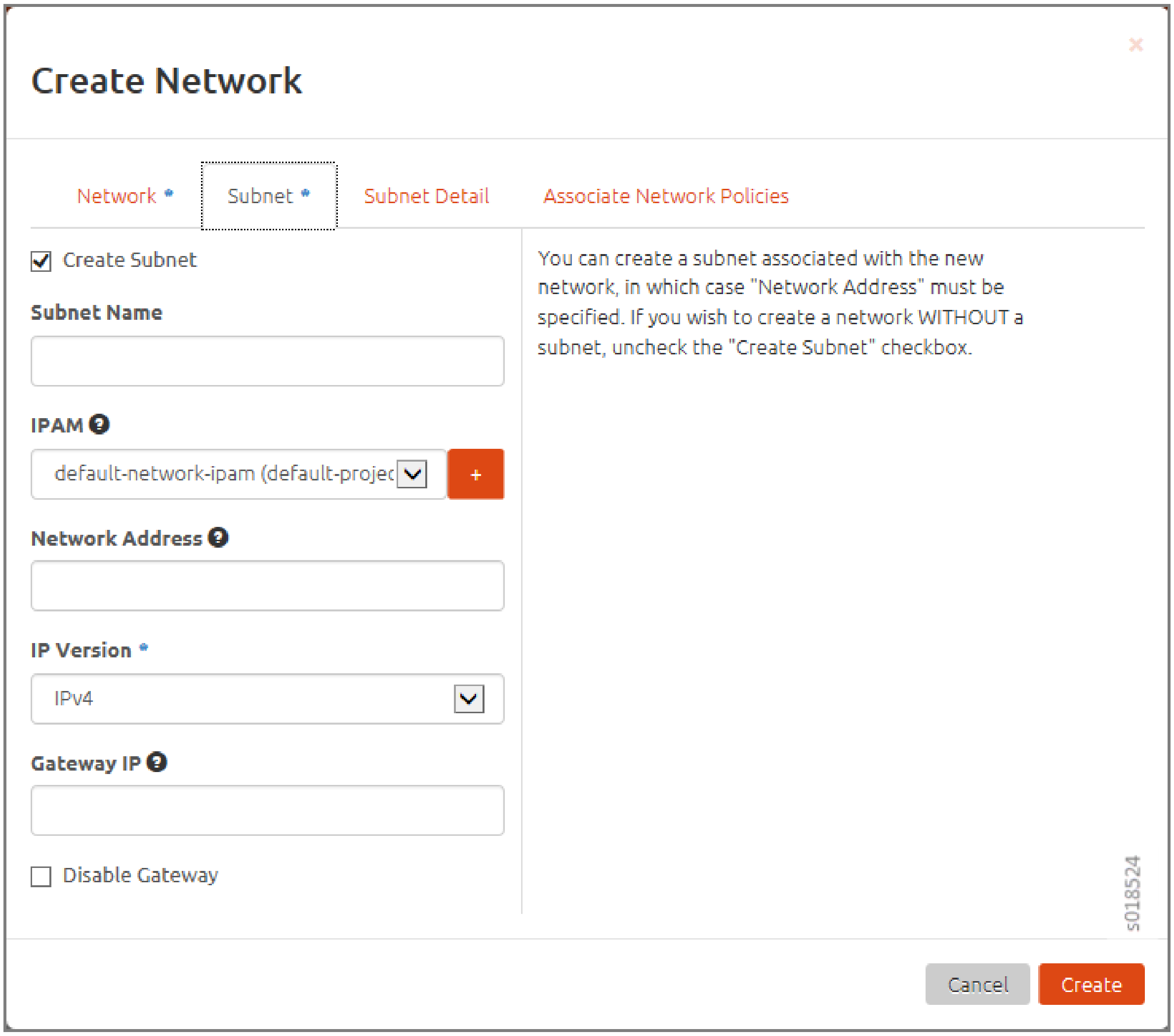


Figure 61: Create Network Window Subnet Tab



3. Click the **Network**, **Subnet**, **Subnet Detail**, and **Associate Network Policies** tabs to complete the fields in the **Create Network** window. See field descriptions in Table 28 on page 195.

Table 28: Create Network Fields

Field	Description
Network Name	Enter a name for the network.
Subnet Name	Enter a name for the subnetwork.

Table 28: Create Network Fields *(Continued)*

Field	Description
IPAM	Select the IPAM associated with the IP block. For new projects, an IPAM can be added while creating the virtual network. VM instances created in this virtual network are assigned an address from this address block automatically by the system when a VM is launched.
Network Address	Enter the network address in CIDR format.
IP Version*	Select IPv4 or IPv6.
Gateway IP	Optionally, enter an explicit gateway IP address for the IP address block. Check the Disable Gateway box if no gateway is to be used.
Network Policy	Any policies already created are listed. To select a policy, click the check box for the policy.

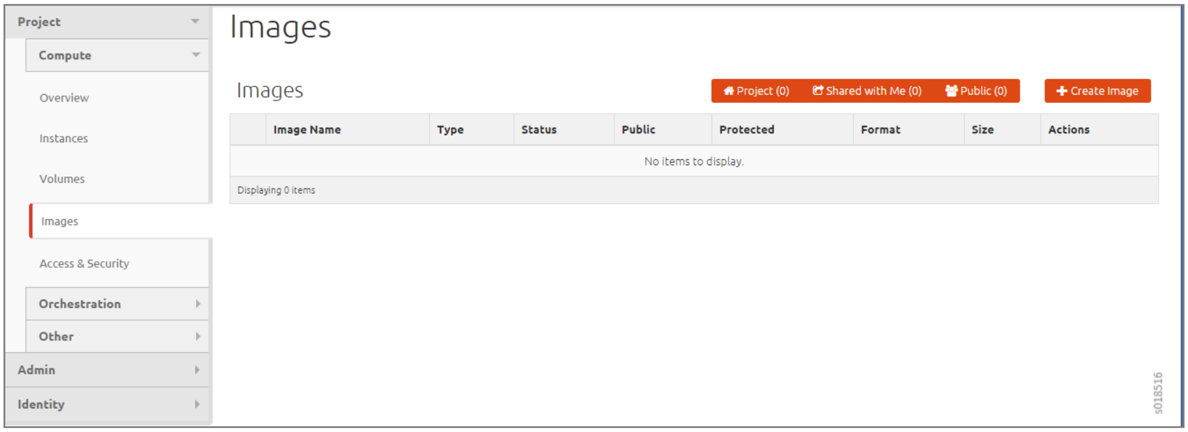
4. Click the **Subnet Details** tab to specify the Allocation Pool, DNS Name Servers, and Host Routes.
5. Click the **Associate Network Policies** tab to associate policies to the network.
6. To save your network, click **Create Network**, or click **Cancel** to discard your work and start over.

Creating an Image for a Project in OpenStack Contrail

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

1. In OpenStack, select **Project > Compute > Images**. The Images window is displayed. See Figure 62 on page 197.

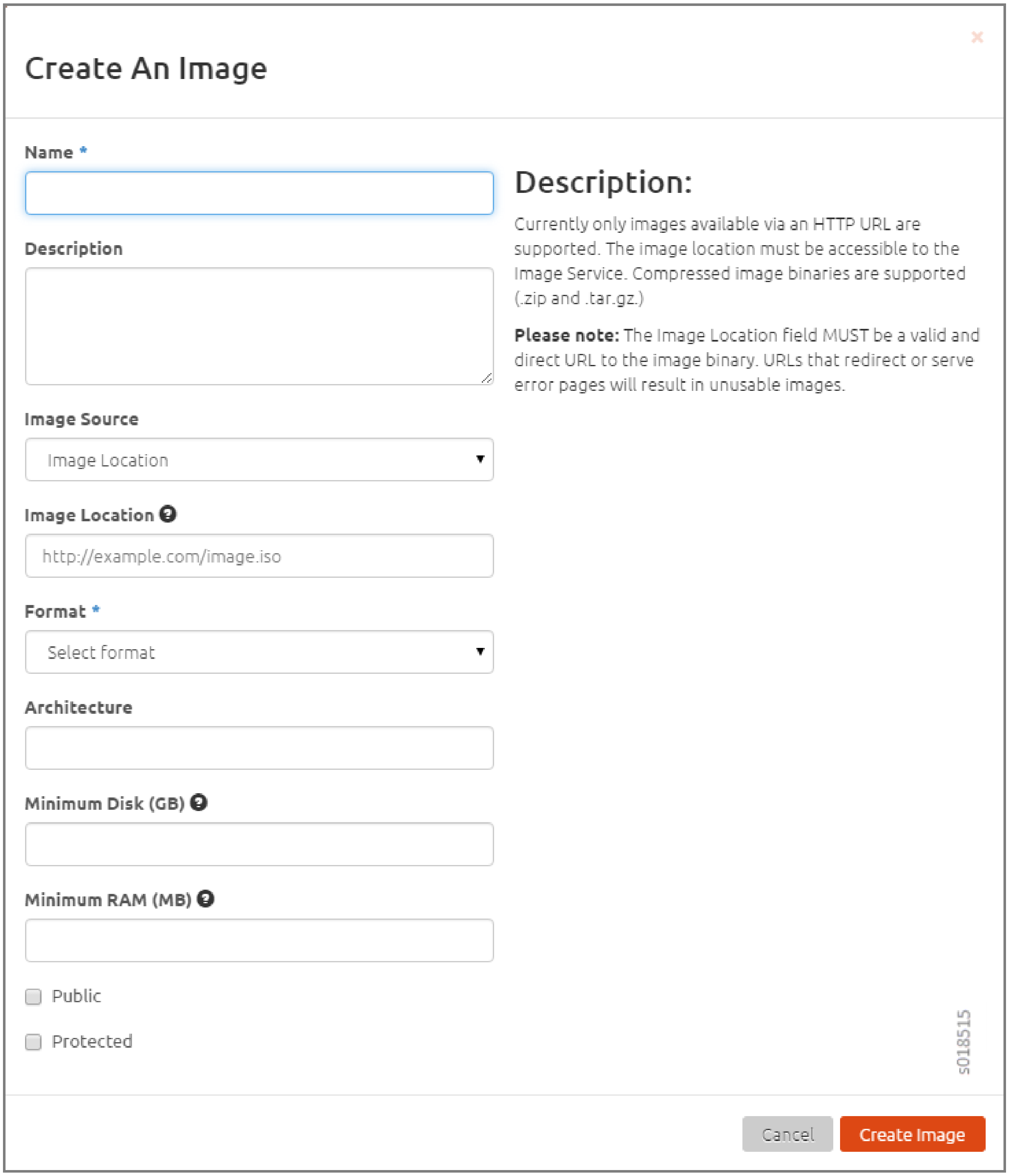
Figure 62: OpenStack Images Window



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

   The **Create An Image** window is displayed. See Figure 63 on page 198.

Figure 63: OpenStack Create An Image Window



4. Complete the fields to specify your image. Table 29 on page 199 describes each of the fields on the window.

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

Table 29: Create an Image Fields

Field	Description
Name	Enter a name for this image.
Description	Enter a description for the image.
Image Source	Select **Image File** or **Image Location**. If you select **Image File**, you are prompted to browse to the local location of the file.
Image Location	Enter an external HTTP URL from which to load the image. The URL must be a valid and direct URL to the image binary. URLs that redirect or serve error pages result in unusable images.
Format	Required field. Select the format of the image from a list: AKI– Amazon Kernel Image AMI– Amazon Machine Image ARI– Amazon Ramdisk Image ISO– Optical Disk Image QCOW2– QEMU Emulator Raw– An unstructured image format VDI– Virtual Disk Imade VHD– Virtual Hard Disk VMDK– Virtual Machine Disk
Architecture	Enter the architecture.

Table 29: 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	Select this check box if this is a public image. Leave unselected for a private image.
Protected	Select this check box for a protected image.

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

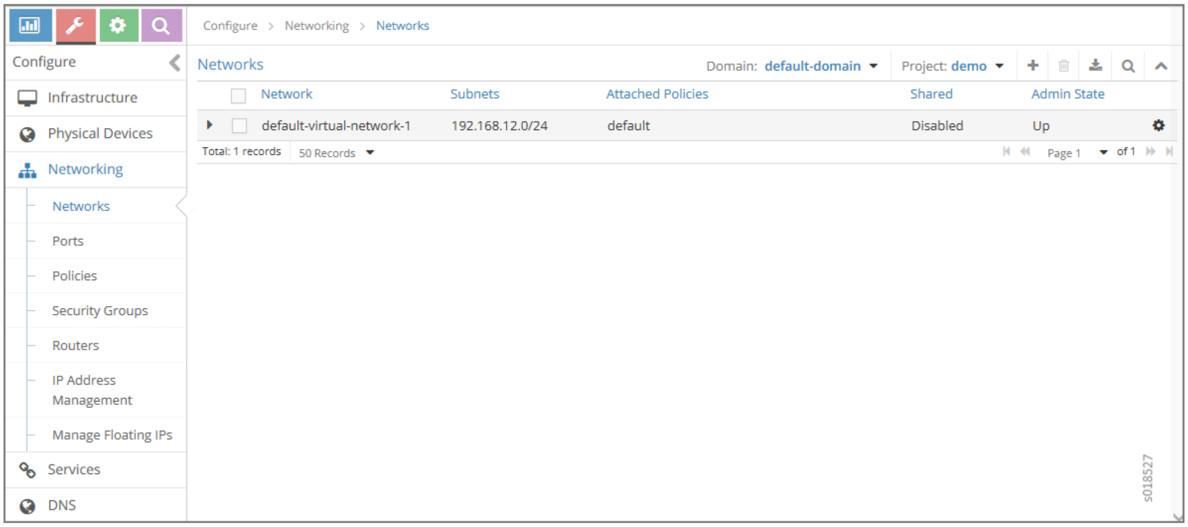
Creating a Floating IP Address Pool

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

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

1. Select **Configure > Networking > Networks**; see Figure 64 on page 201. Make sure your project is the active project in the upper right.

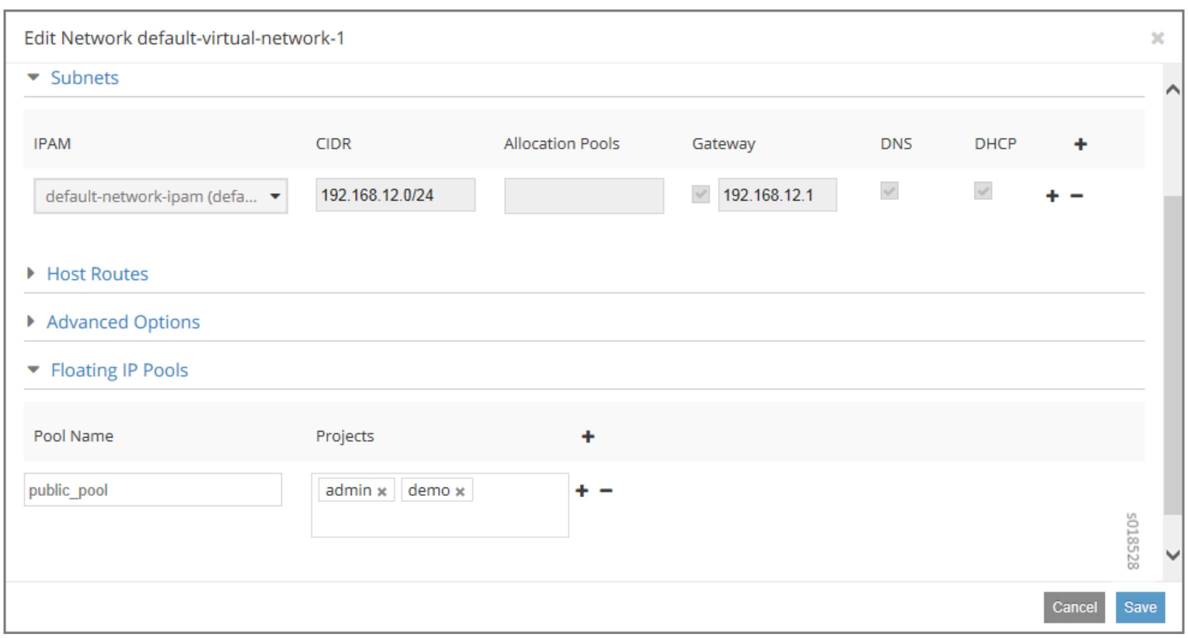
Figure 64: Configure > Networking > Networks



2. Click the network you want to associate with a floating IP pool, then in the **Action** column, click the action icon and select **Edit**.

   The **Edit Network** window for the selected network is displayed; see Figure 65 on page 201.

Figure 65: Edit Network



3. In the **Floating IP Pools** section, click the **Pool Name** field, enter a name for your floating IP pool, and click the **+** (plus sign) to add the IP pool to the table below the field.

   - Multiple floating IP pools can be created at the same time.

- A floating IP pool can be associated with multiple projects.

4. Click **Save** to create the floating IP address pool, or click **Cancel** to remove your work and start over.

Using Security Groups with Virtual Machines (Instances)

IN THIS SECTION

- Security Groups Overview | 202
- Creating Security Groups and Adding Rules | 202

Security Groups Overview

A **security group** is a container for security group rules. Security groups and security group rules allow administrators to specify the type of traffic that is allowed to pass through a port. When a virtual machine (VM) is created in a virtual network (VN), a security group can be associated with the VM when it is launched. If a security group is not specified, a port is associated with a default security group. The default security group allows both ingress and egress traffic. Security rules can be added to the default security group to change the traffic behavior.

Creating Security Groups and Adding Rules

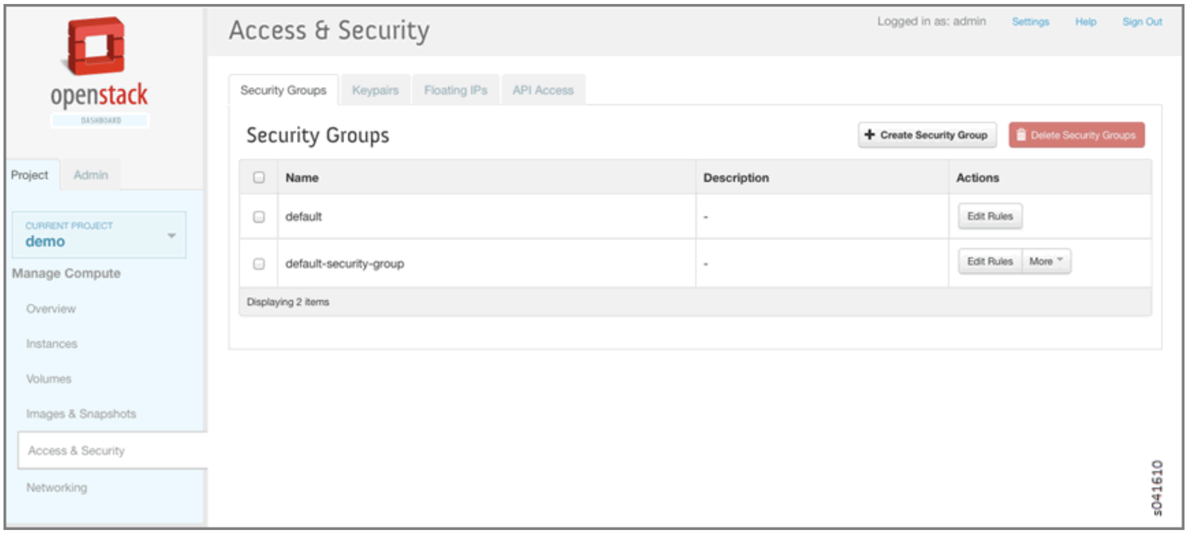
A default security group is created for each project. You can add security rules to the default security group and you can create additional security groups and add rules to them. The security groups are then associated with a VM, when the VM is launched or at a later date.

To add rules to a security group:

1. From the OpenStack interface, click the **Project** tab, select **Access & Security**, and click the **Security Groups** tab.

   Any existing security groups are listed under the **Security Groups** tab, including the default security group; see Figure 66 on page 203.

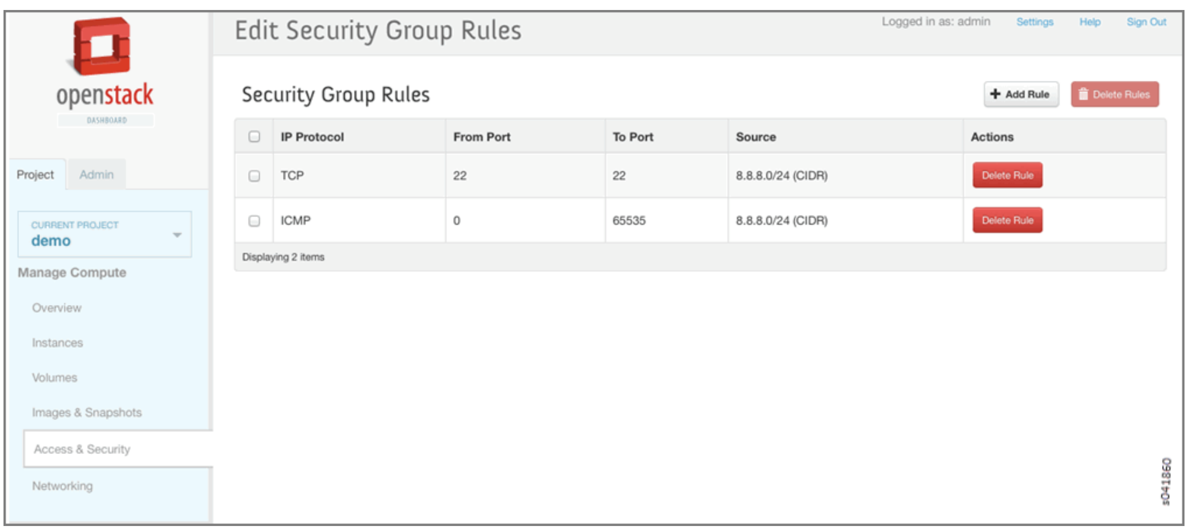
Figure 66: Security Groups



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

   The **Edit Security Group Rules** window is displayed; see Figure 67 on page 203. Any rules already associated with the security group are listed.

Figure 67: Edit Security Group Rules



3. Click **Add Rule** to add a new rule; see Figure 68 on page 204.

Figure 68: Add Rule

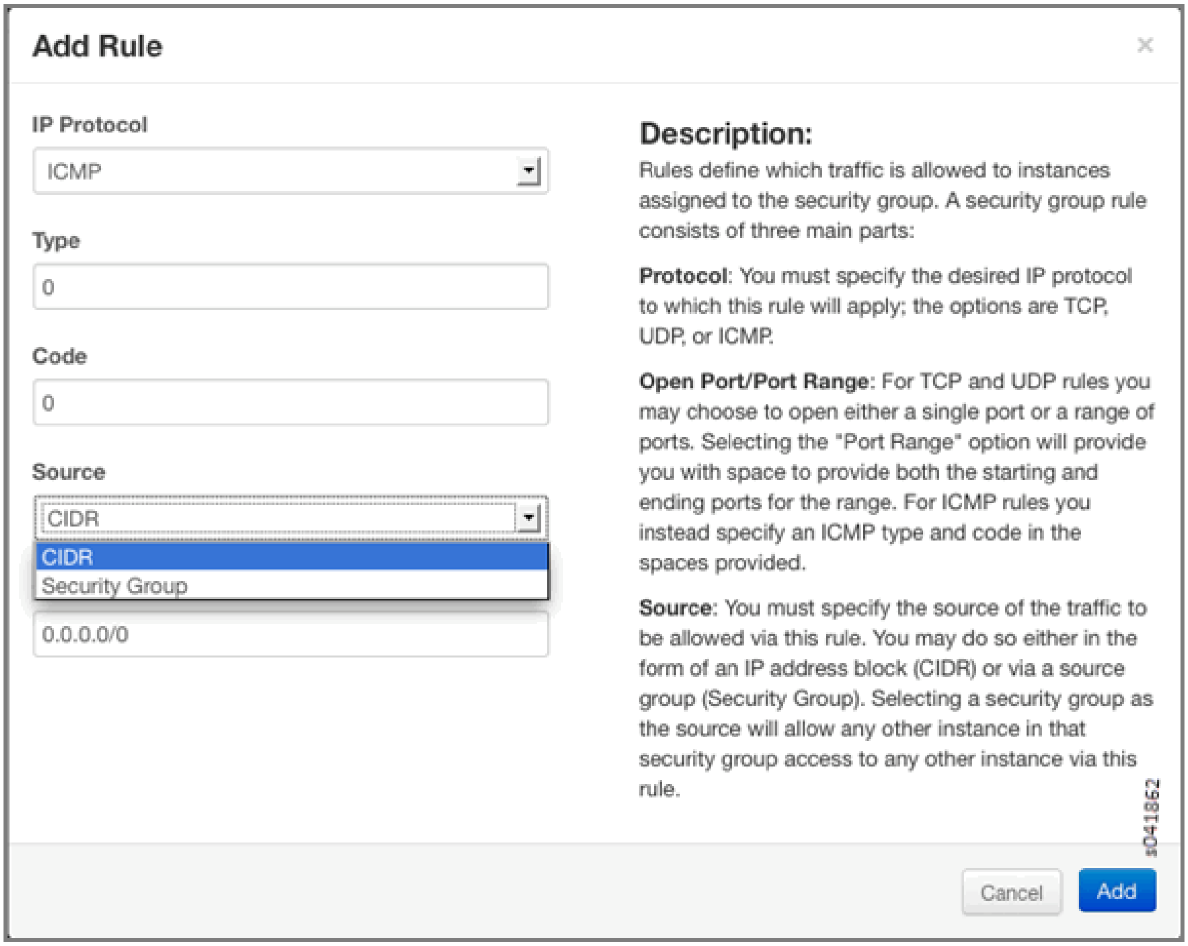


Table 30: Add Rule Fields

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.

Table 30: Add Rule Fields *(Continued)*

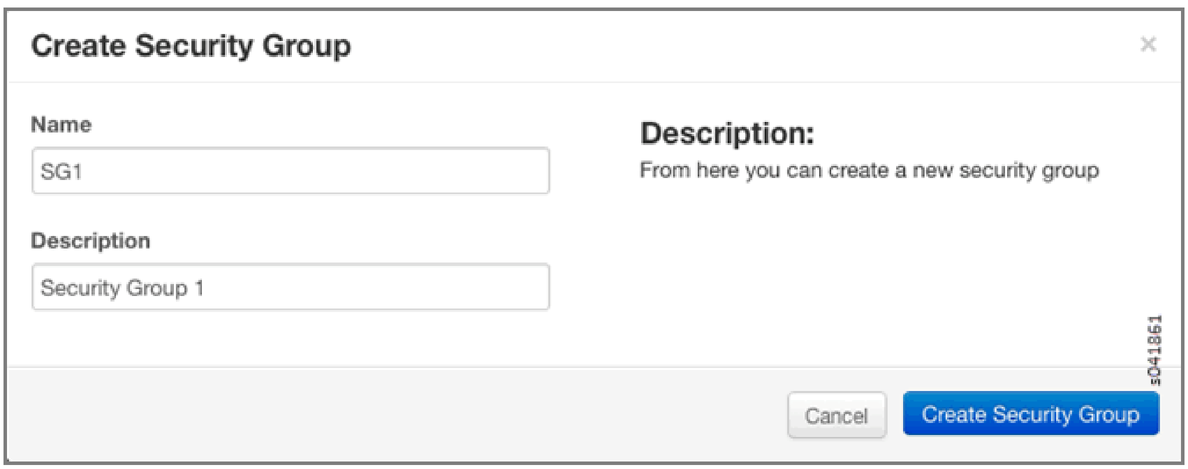
Column	Description
Source	Select the source of traffic to be allowed by this rule. Specify subnet—the CIDR IP address or address block of the inter-domain source of the traffic that applies to this rule, or you can choose security group as source. Selecting security group as source allows any other instance in that security group access to any other instance via this rule.

4. Click **Create Security Group** to create additional security groups.

   The **Create Security Group** window is displayed; see Figure 69 on page 205.

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

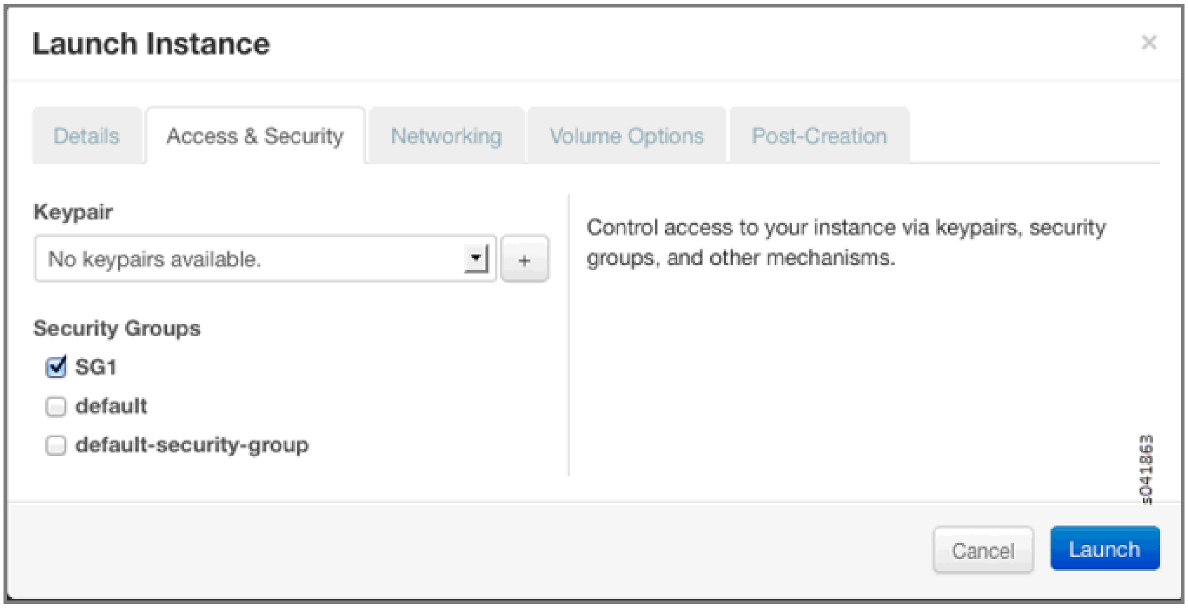
   Figure 69: Create Security Group

5. When an instance is launched, there is an opportunity to associate a security group; see Figure 70 on page 206.

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

Figure 70: Associate Security Group at Launch Instance



6. You can verify that security groups are attached by viewing the `SgListReq` and `IntfReq` associated with the `agent.xml`.

Support for IPv6 Networks in Contrail

IN THIS SECTION

- Overview: IPv6 Networks in Contrail | 206
- Creating IPv6 Virtual Networks in Contrail | 207
- Adding IPv6 Peers | 209

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

Overview: IPv6 Networks in Contrail

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

- Virtual machines with IPv6 and IPv4 interfaces

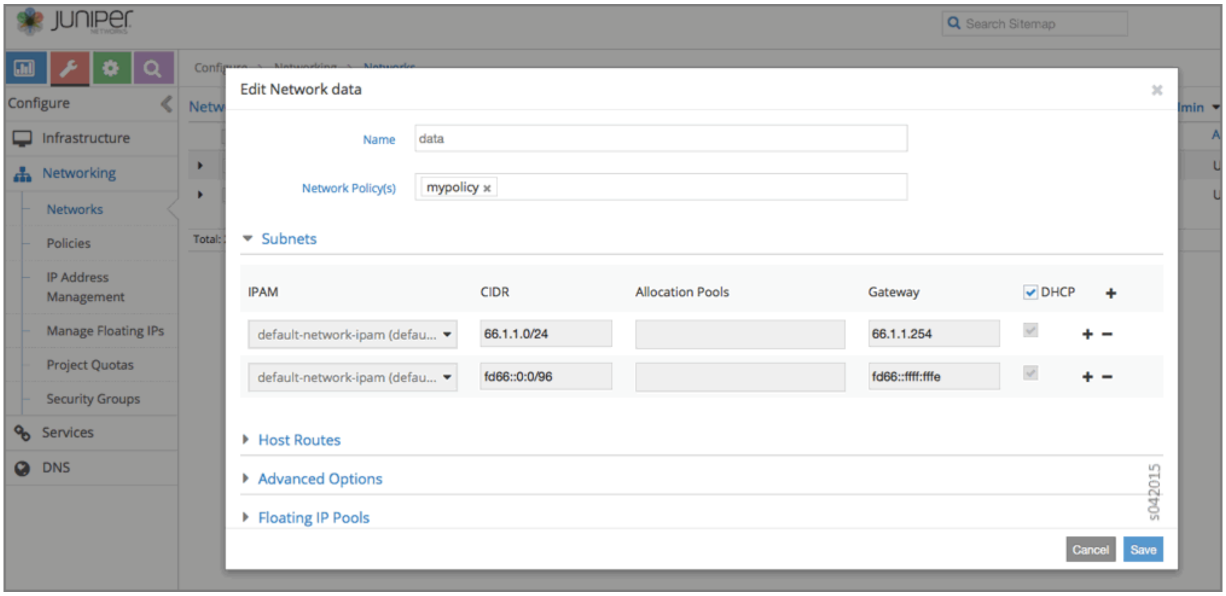
- Virtual machines with IPv6-only interfaces
- DHCPv6 and neighbor discovery
- Policy and Security groups
- IPv6 flow set up, tear down, and aging
- Flow set up and tear down based on TCP state machine
- Protocol-based flow aging
- Fat flow
- Allowed address pair configuration with IPv6 addresses
- IPv6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

NOT present is support for the following:

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

Creating IPv6 Virtual Networks in Contrail

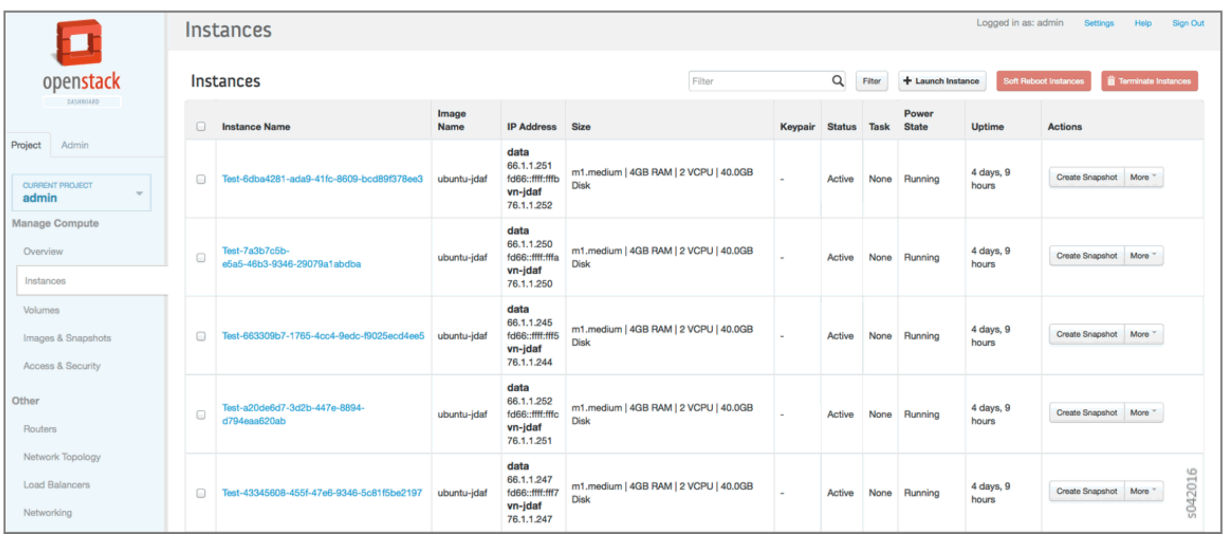
You can create an IPv6 virtual network from the Contrail user interface in the same way you create an IPv4 virtual network. When you create a new virtual network by selecting **Configure > Networking > Networks**, the Edit fields accept IPv6 addresses, as shown in the following image.



Address Assignments

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

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



Enabling DHCPv6 In Virtual Machines

To allow IPv6 address assignment using DHCPv6, the virtual machine network interface configuration must be updated appropriately.

For example, to enable DHCPv6 for Ubuntu-based virtual machines, add the following line in the **/etc/network/interfaces** file:

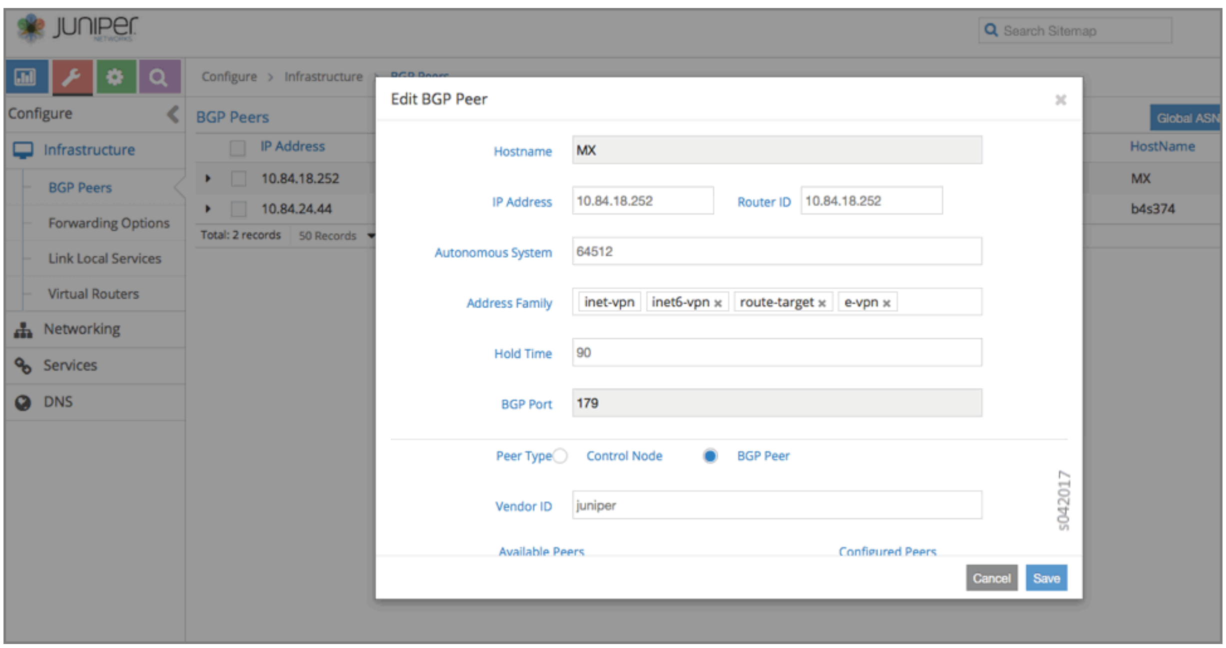
```
iface eht0 inet6 dhcp
```

Also, `dhclient -6` can be run from within the virtual machine to get IPv6 addresses using DHCPv6.

Adding IPv6 Peers

The procedure to add an IPv6 BGP peer in Contrail is similar to adding an IPv4 peer. Select **Configure** > **Infrastructure** > **BGP Peers**, include `inet6-vpn` in the Address Family list to allow advertisement of IPv6 addresses.

A sample is shown in the following.



NOTE: Additional configuration is required on the peer router to allow inet6-vpn peering.

Configuring EVPN and VXLAN

IN THIS SECTION

- Configuring the VXLAN Identifier Mode | 212
- Configuring Forwarding | 214
- Configuring the VXLAN Identifier | 215
- Configuring Encapsulation Methods | 216

Contrail supports Ethernet VPNs (EVPN) and Virtual Extensible Local Area Networks (VXLAN).

EVPN is a flexible solution that uses Layer 2 overlays to interconnect multiple edges (virtual machines) within a data center. Traditionally, the data center is built as a flat Layer 2 network with issues such as flooding, limitations in redundancy and provisioning, and high volumes of MAC address learning, which cause churn during node failures. EVPNs are designed to address these issues without disturbing flat MAC connectivity.

In EVPNs, MAC address learning is driven by the control plane, rather than by the data plane, which helps control learned MAC addresses across virtual forwarders, thus avoiding flooding. The forwarders advertise locally learned MAC addresses to the controllers. The controllers use MP-BGP to communicate with peers. The peering of controllers using BGP for EVPN results in better and faster convergence.

With EVPN, MAC learning is confined to the virtual networks to which the virtual machine belongs, thus isolating traffic between multiple virtual networks. In this manner, virtual networks can share the same MAC addresses without any traffic crossover.

Unicast in EVPNs

Unicast forwarding is based on MAC addresses where traffic can terminate on a local endpoint or is encapsulated to reach the remote endpoint. Encapsulation can be MPLS/UDP, MPLS/GRE, or VXLAN.

BUM Traffic in EVPN

Multicast and broadcast traffic is flooded in a virtual network. The replication tree is built by the control plane, based on the advertisements of end nodes (virtual machines) sent by forwarders. Each virtual network has one distribution tree, a method that avoids maintaining multicast states at fabric nodes, so the nodes are unaffected by multicast. The replication happens at the edge forwarders. Per-group subscription is not provided. Broadcast, unknown unicast, and multicast (BUM) traffic is handled the same way, and gets flooded in the virtual network to which the virtual machine belongs.

VXLAN

VXLAN is an overlay technology that encapsulates MAC frames into a UDP header at Layer 2. Communication is established between two virtual tunnel endpoints (VTEPs). VTEPs encapsulate the virtual machine traffic into a VXLAN header, as well as strip off the encapsulation. Virtual machines can only communicate with each other when they belong to the same VXLAN segment. A 24-bit virtual network identifier (VNID) uniquely identifies the VXLAN segment. This enables having the same MAC frames across multiple VXLAN segments without traffic crossover. Multicast in VXLAN is implemented as Layer 3 multicast, in which endpoints subscribe to groups.

Design Details of EVPN and VXLAN

In Contrail Release 1.03 and later, EVPN is enabled by default. The supported forwarding modes include:

- Fallback bridging—IPv4 traffic lookup is performed using the IP FIB. All non-IPv4 traffic is directed to a MAC FIB.
- Layer 2-only— All traffic is forwarded using a MAC FIB lookup.

You can configure the forwarding mode individually on each virtual network.

EVPN is used to share MAC addresses across different control planes in both forwarding models. The result of a MAC address lookup is a next hop, which, similar to IP forwarding, points to a local virtual machine or a tunnel to reach the virtual machine on a remote server. The tunnel encapsulation methods supported for EVPN are MPLSoGRE, MPLSoUDP, and VXLAN. The encapsulation method selected is based on a user-configured priority.

In VXLAN, the VNID is assigned uniquely for every virtual network carried in the VXLAN header. The VNID uniquely identifies a virtual network. When the VXLAN header is received from the fabric at a remote server, the VNID lookup provides the VRF of the virtual machine. This VRF is used for the MAC lookup from the inner header, which then provides the destination virtual machine.

Non-IP multicast traffic uses the same multicast tree as for IP multicast (255.255.255.255). The multicast is matched against the all-broadcast prefix in the bridging table (FF:FF:FF:FF:FF:FF). VXLAN is not supported for IP/non-IP multicast traffic.

The following table summarizes the traffic and encapsulation types supported for EVPN.

		Encapsulation		
		MPLS-GRE	**MPLS-UDP**	**VXLAN**
Traffic Type	**IP unicast**	Yes	Yes	No

	IP-BUM	Yes	Yes	No
	non IP unicast	Yes	Yes	Yes
	non IP-BUM	Yes	Yes	No

Configuring the VXLAN Identifier Mode

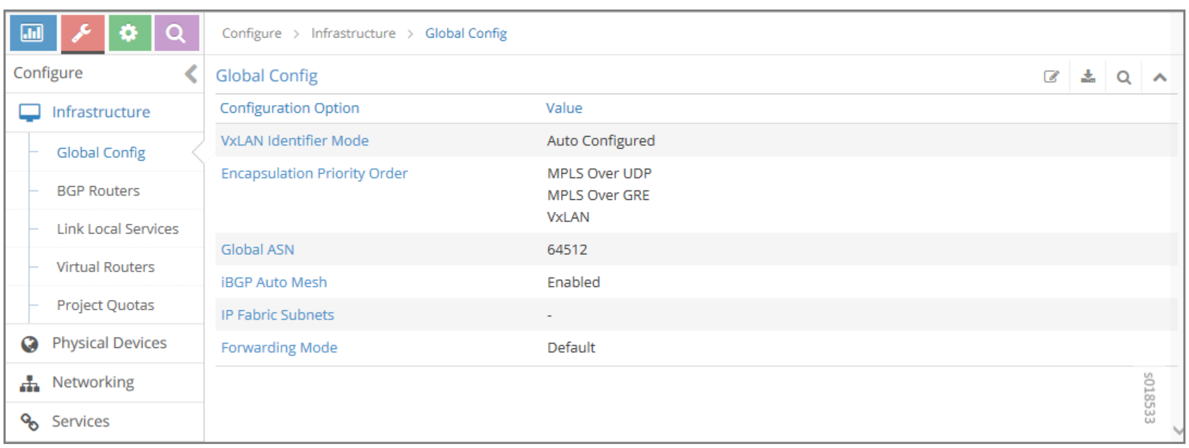
You can configure the global VXLAN identifier mode to select an auto-generated VNID or a user-generated VXLAN ID, either through the Contrail Web UI or by modifying a python file.

To configure the global VXLAN identifier mode:

1. From the Contrail Web UI, select **Configure > Infrastucture > Global Config**.

   The Global Config options and values are displayed in the Global Config window.

   Figure 71: Global Config Window for VXLAN ID

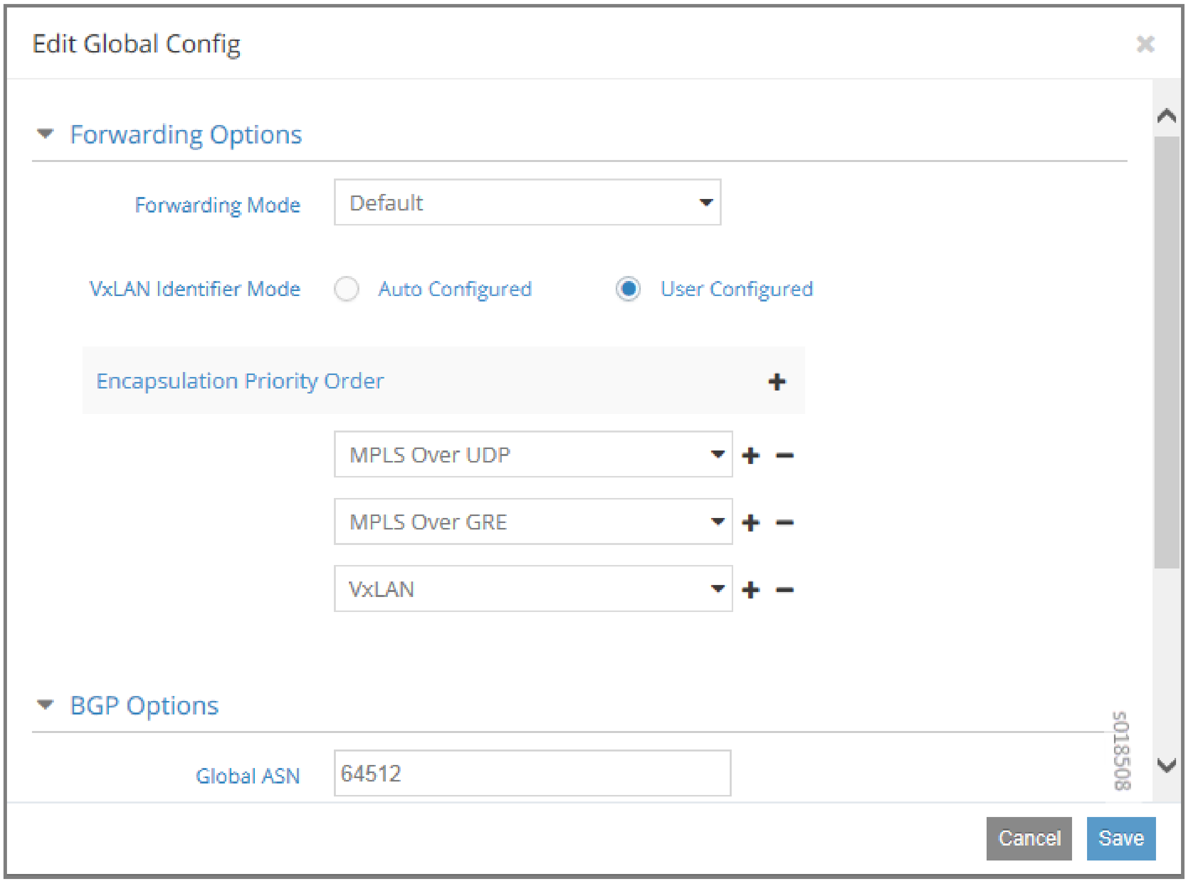
2. Click the edit icon

   .

   The Edit Global Config window is displayed as shown in Figure 72 on page 213.

Figure 72: Edit Global Config Window for VXLAN Identifier Mode



3. Select one of the following:

   - **Auto Configured**— The VXLAN identifier is automatically assigned for the virtual network.

   - **User Configured**– You must provide the VXLAN identifier for the virtual network.

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

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

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

Configuring Forwarding

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

To change the forwarding mode:

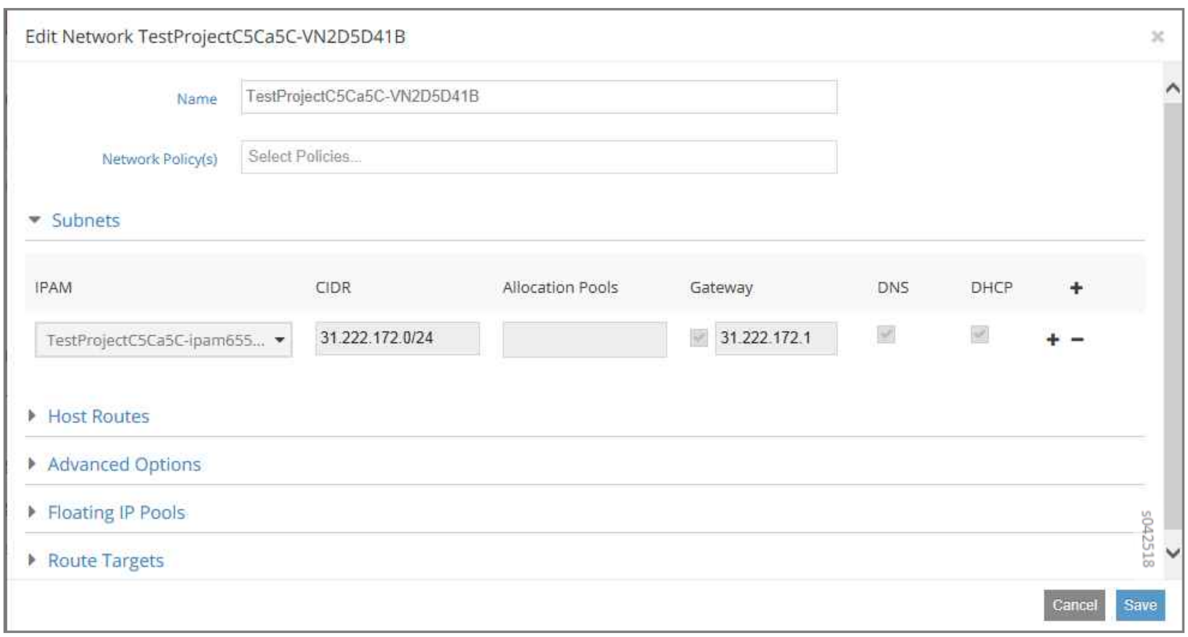
1. From the Contrail Web UI, select **Configure** > **Networking** > **Networks**.
2. Select the virtual network that you want to change the forwarding mode for.
3. Click the gear icon

   and select **Edit**.

   The Edit Network window is displayed as shown in Figure 73 on page 214.

   Figure 73: Edit Network Window

   Under the Advanced Options select the forwarding mode from the following choices:

   - Select **Default** to enable the default forwarding mode.
   - Select **L2 and L3** to enable IP and MAC FIB (fallback bridging).
   - Select **L2 Only** to enable only MAC FIB.
   - Select **L3 Only** to enable only IP.

NOTE: The full list of forwarding modes are only displayed if you change entries in the **/usr/src/contrail/contrail-web-core/config/config.global.js** file. For example:

1. To make the **L2** selection available locate the following:

```
config.network = {};
config.network.L2_enable = false;
```

2. Change the entry to the following:

```
config.network = {};
config.network.L2_enable = true;
```

3. To make the other selections available, modify the corresponding entries.

4. Save the file and quit the editor.

5. Restart the Contrail Web user interface process (webui).

Alternatively, you can use the following python provisioning command to change the forwarding mode:

```
python provisioning_forwarding_mode --project_fq_name 'defaultdomain: admin' --vn_name vn1 --forwarding_mode <
l2_l3| l2 >
```

Options:

`l2_l3` = Enable IP FIB and MAC FIB (fallback bridging)

`l2` = Enable MAC FIB only (Layer 2 only)

Configuring the VXLAN Identifier

The VXLAN identifier can be set only if the VXLAN network identifier mode has been set to User Configured. You can then set the VXLAN ID by either using the Contrail Web UI or by using Python commands.

To configure the global VXLAN identifier:

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

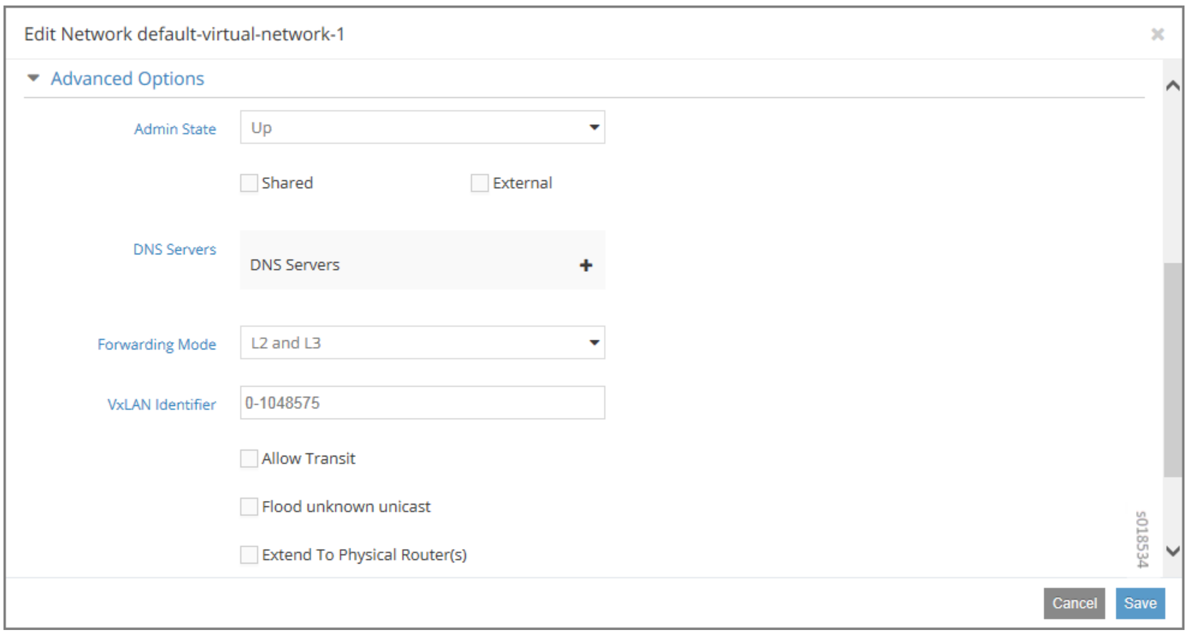
3. Click the gear icon

   and select **Edit**.

   The Edit Network window is displayed. Select the **Advanced Options** as shown in Figure 74 on page 216.

Figure 74: Edit Network Window for VXLAN Identifier



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

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

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

Configuring Encapsulation Methods

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

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

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

To configure the global VXLAN identifier mode:

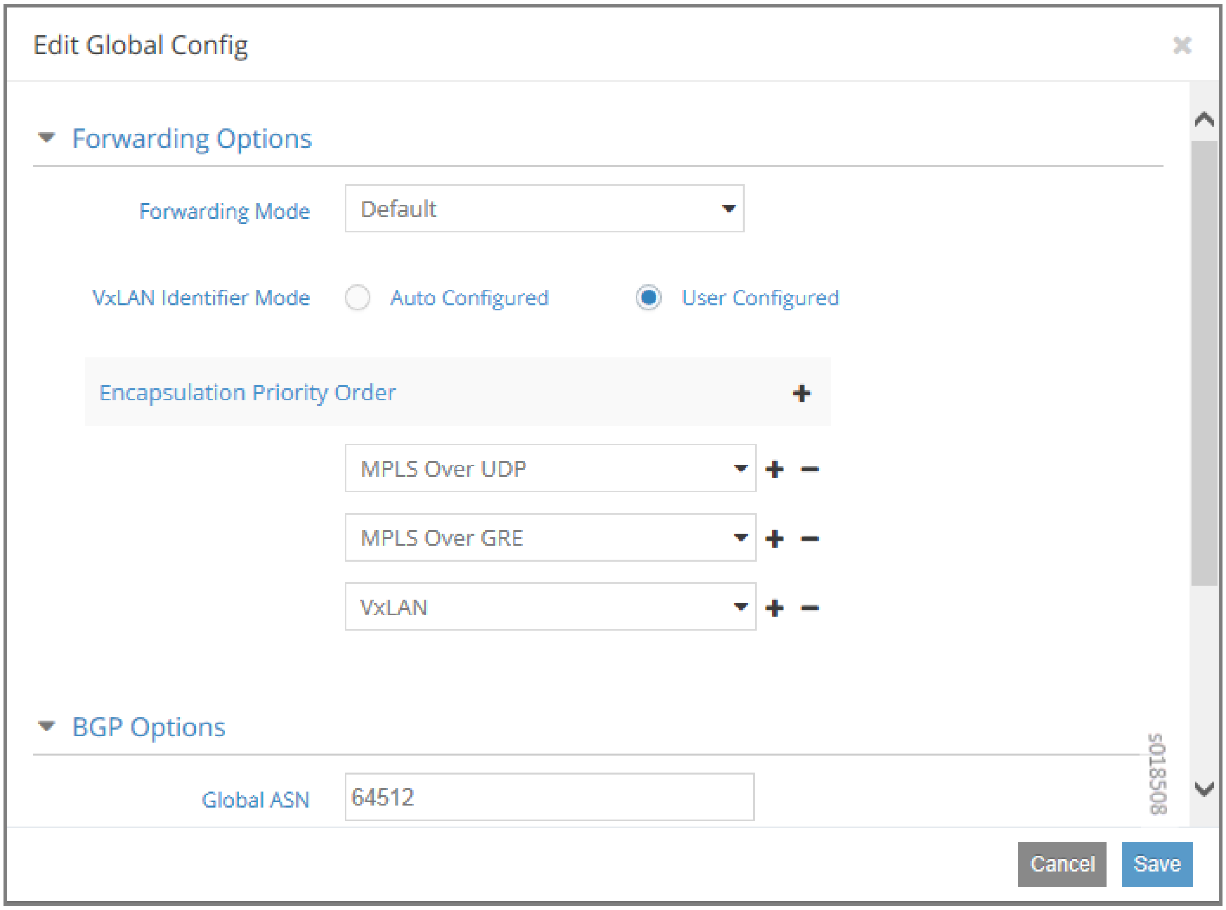
- From the Contrail Web UI, select **Configure > Infrastucture > Global Config**.
- The Global Config options are displayed.
- Click the edit icon

  .

The Edit Global Config window is displayed as shown in Figure 75 on page 218.

Figure 75: Edit Global Config Window for Encapsulation Priority Order



Under Encapsulation Priority Order select one of the following:

- MPLS over UDP
- MPLS over GRE
- VxLAN

Click the + plus symbol to the right of the first priority to add a second priority or third priority.

Use the following procedure to change the default encapsulation method to VXLAN by editing the **encap.py** file.

> **NOTE**: VXLAN is *only* supported for EVPN unicast. It is not supported for IP traffic or multicast traffic. VXLAN priority and presence in the **encap.py** file or configured in the Web UI is ignored for traffic not supported by VXLAN.

To set the priority of encapsulation methods to VXLAN:

1. Modify the **encap.py** file found in the **/opt/contrail/utils/** directory.

   The default encapsulation line is:

   ```
   encap_obj=EncapsulationPrioritiesType(encapsulation=['MPLSoUDP','M PLSoGRE'])
   ```

   Modify the line to:

   ```
   encap_obj=EncapsulationPrioritiesType(encapsulation=['VXLAN', 'MPLSoUDP','MPLSoGRE'])
   ```

2. After the status is modified, execute the following script:

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

   The configuration is applied globally for all virtual networks.

CHAPTER 9

Example of Deploying a Multi-Tier Web Application Using Contrail

IN THIS CHAPTER

- Example: Deploying a Multi-Tier Web Application | 220
- Sample Network Configuration for Devices for Simple Tiered Web Application | 228

Example: Deploying a Multi-Tier Web Application

IN THIS SECTION

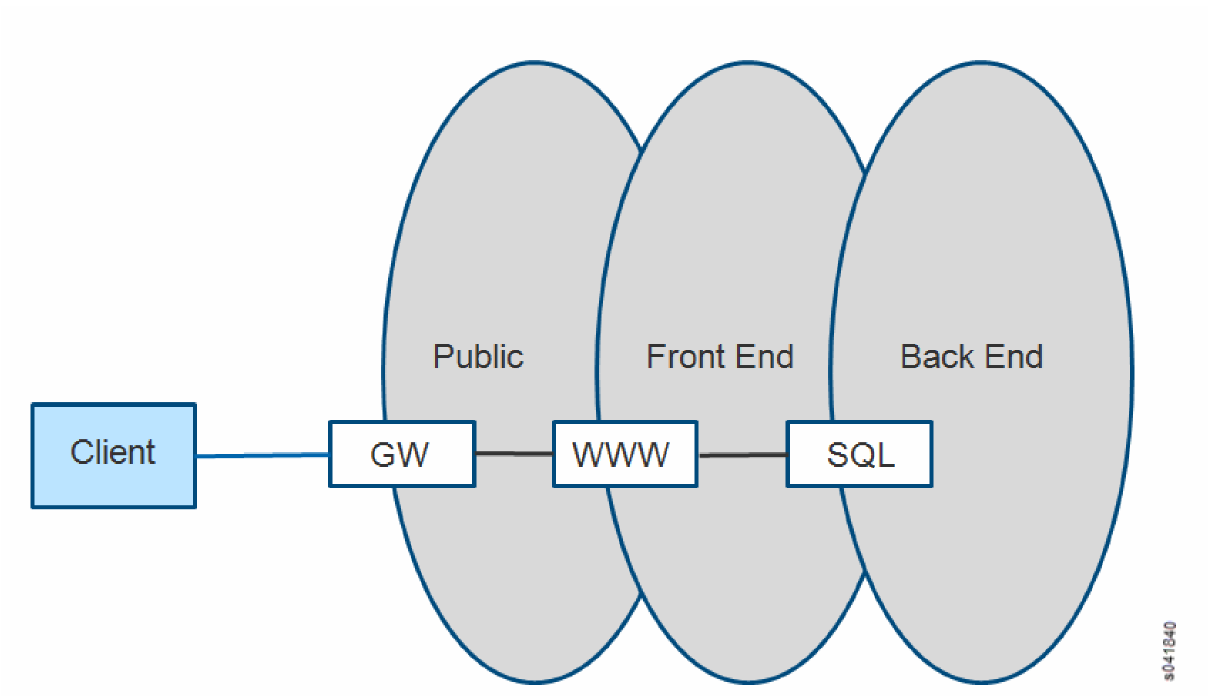
- Multi-Tier Web Application Overview | 220
- Example: Setting Up Virtual Networks for a Simple Tiered Web Application | 221
- Verifying the Multi-Tier Web Application | 224
- Sample Addressing Scheme for Simple Tiered Web Application | 224
- Sample Physical Topology for Simple Tiered Web Application | 225
- Sample Physical Topology Addressing | 226

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 76 on page 221.

Figure 76: 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 188.

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

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

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

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

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

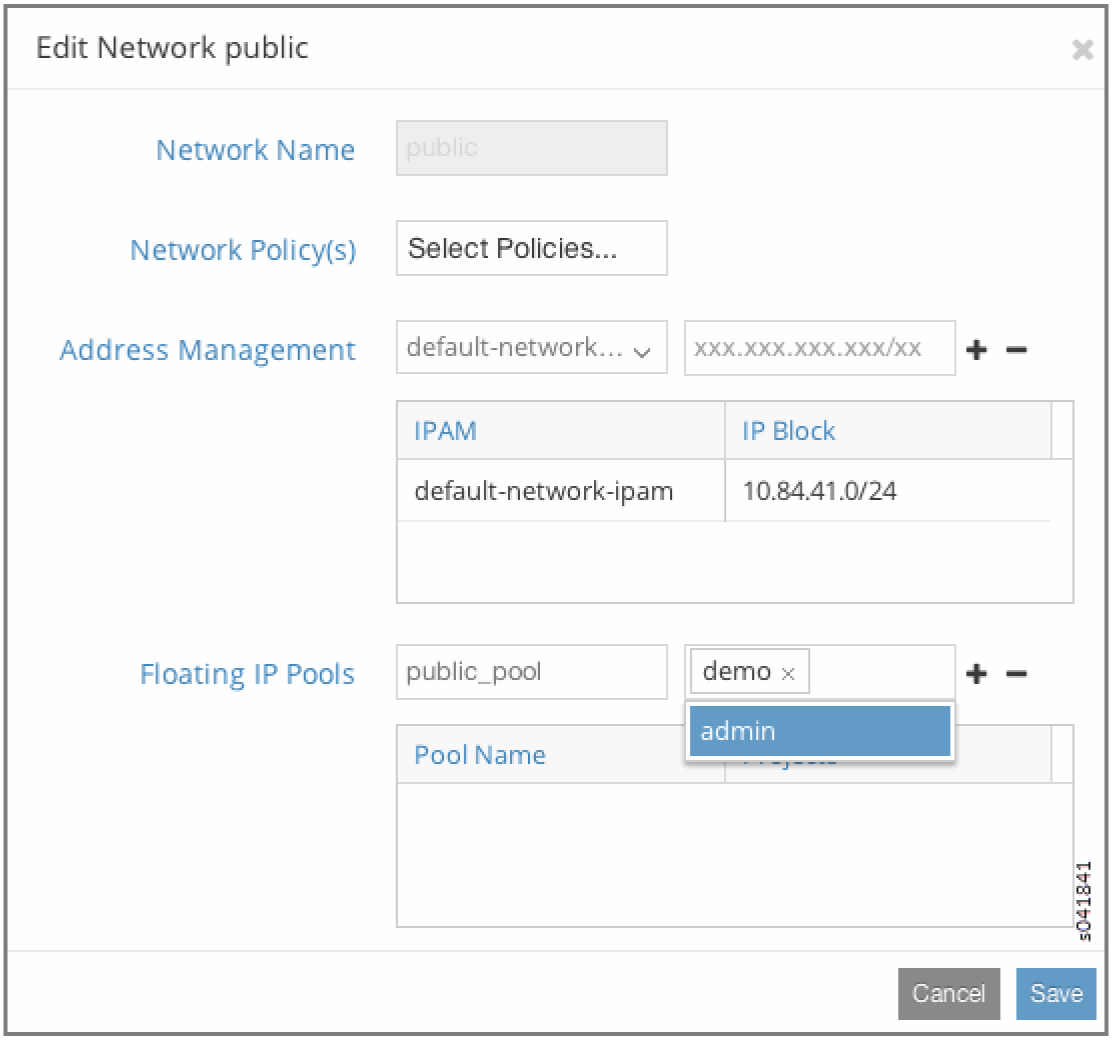
c. A network named **backend** with IP address **192.168.2.0/24**

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

For more information; see "Creating a Virtual Network with OpenStack Contrail" on page 194 or "Creating a Virtual Network with Juniper Networks Contrail" on page 190.

3. Create a floating IP pool named **public_pool** for the **public** network within the **demo** project; see Figure 77 on page 223.

Figure 77: Create Floating IP Pool



4. Allocate the floating IP pool **public_pool** to the **demo** project; see Figure 78 on page 223.

Figure 78: Allocate Floating IP

5. Verify that the floating IP pool has been allocated; see **Configure > Networking > Allocate Floating IPs.**
6. Create a policy that allows any host to talk to any host using any IP address, protocol, and port, and apply this policy between the **frontend** network and the **backend** network.

   This now allows communication between the web servers in the front-end network and the database servers in the back-end network.
7. Launch the virtual machine instances that represent the web server and the database server.

   > **NOTE**: Your installation might not include the virtual machines needed for the web server and the database server. Contact your account team if you need to download the VMs for this setup.

   On the **Instances** tab for this project, select **Launch Instance** and for each instance that you launch, complete the fields to make the following associations:

   - Web server VM: select **frontend** network and the policy created to allow communication between **frontend** and **backend** networks. Apply the floating IP address pool to the web server.
   - Database server VM: select **backend** network and the policy created to allow communication between **frontend** and **backend** networks.

Verifying the Multi-Tier Web Application

Verify your web setup.

- To demonstrate this web application setup, go to the client machine, open a browser, and navigate to the address in the **public** network that is assigned to the web server in the **frontend** network.

  The result will display the Contrail interface with various data populated, verifying that the web server is communicating with the database server in the **backend** network and retrieving data.

  The client machine only has access to the public IP address. Attempts to browse to any of the addresses assigned to the **frontend** network or to the **backend** network should fail.

Sample Addressing Scheme for Simple Tiered Web Application

Use the information in Table 31 on page 225 as a guide for addressing devices in the simple tiered web example.

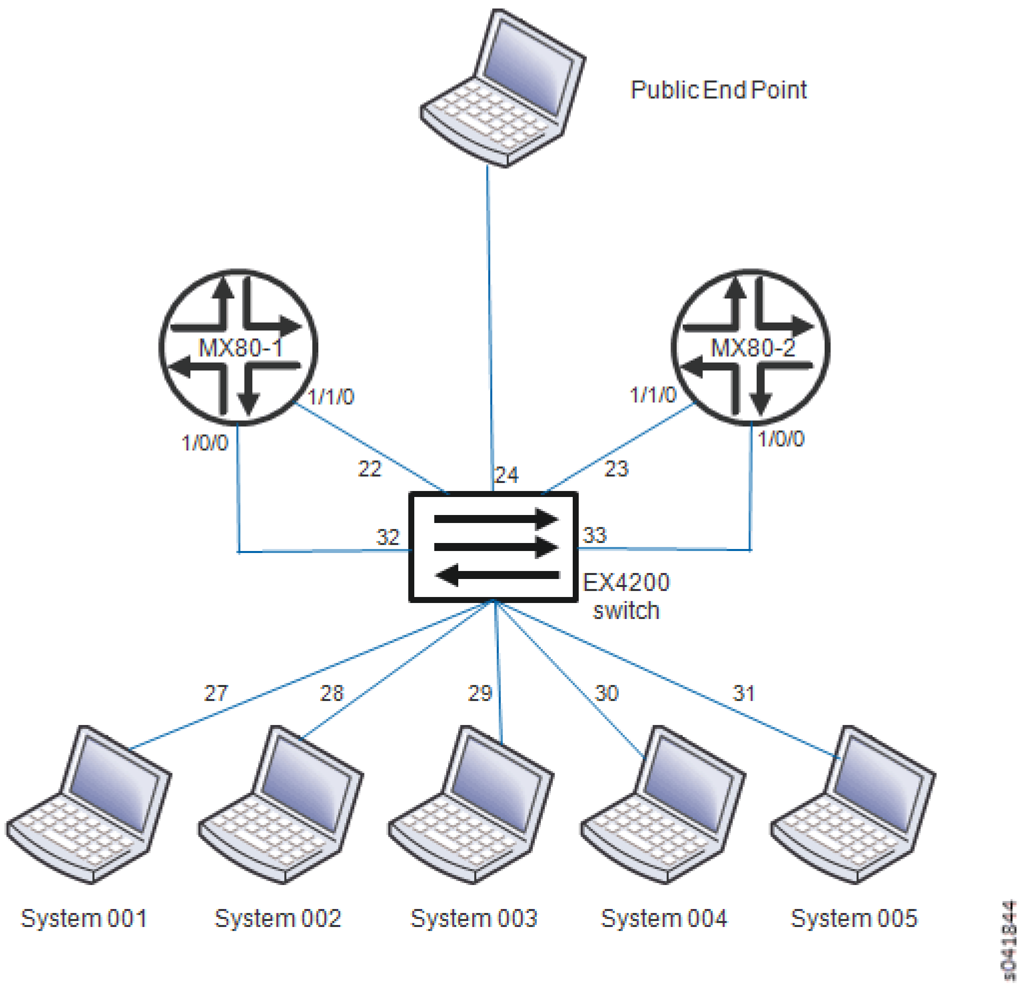
Table 31: Sample Addressing Scheme for Example

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 79 on page 226 provides a guideline diagram for the physical topology for the simple tiered web application example.

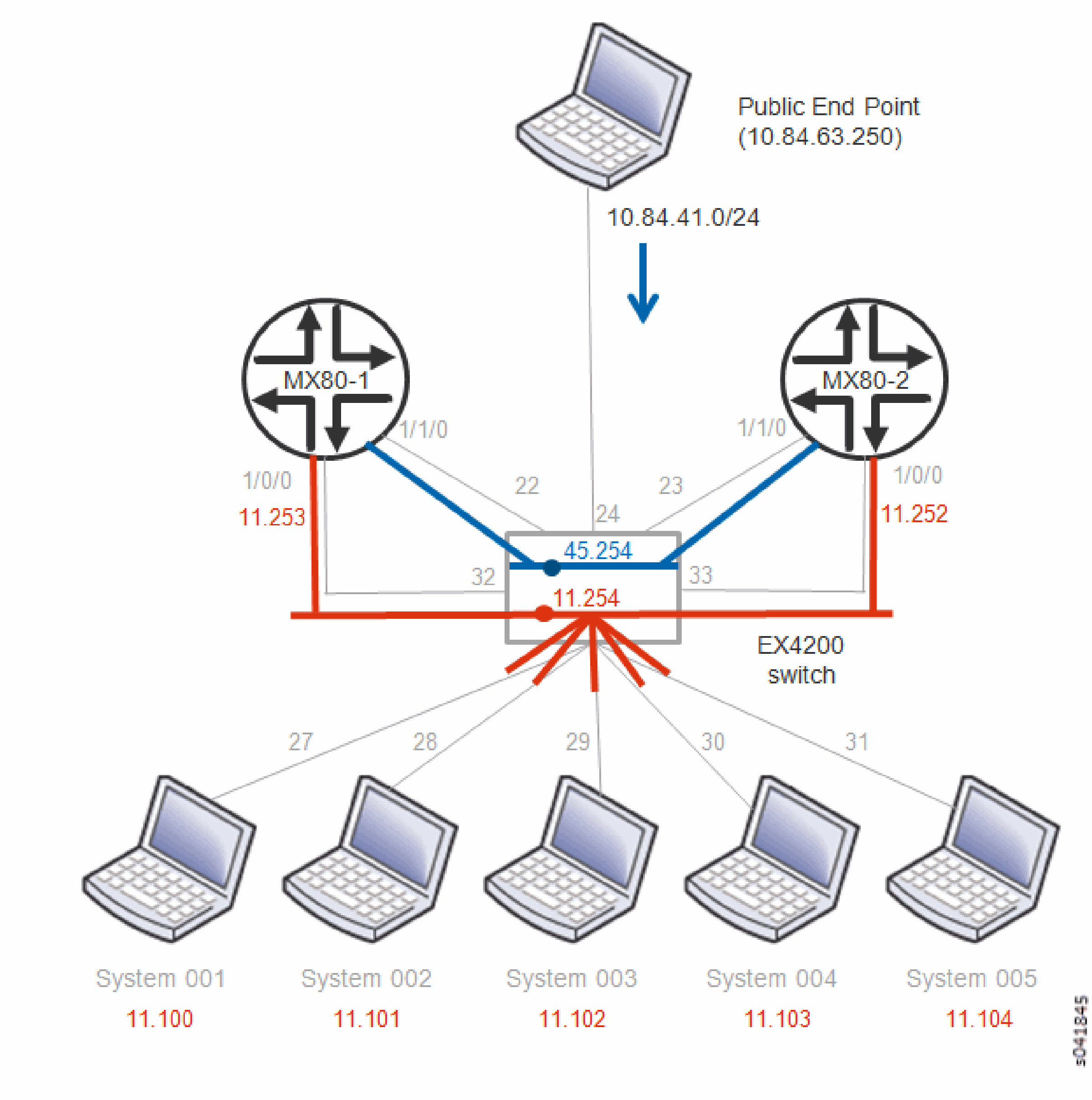
Figure 79: Sample Physical Topology for Simple Tiered Web Application



Sample Physical Topology Addressing

Figure 80 on page 227 provides a guideline diagram for addressing the physical topology for the simple tiered web application example.

Figure 80: Sample Physical Topology Addressing



SEE ALSO

Sample Network Configuration for Devices for Simple Tiered Web Application | 228

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 220. Configurations are shown for Juniper Networks devices: two MX80s and one EX4200.

MX80-1 Configuration

```
version 12.2R1.3;
system {
    root-authentication {
        encrypted-password "xxxxxxxxxx"; ## SECRET-DATA
    }
    services {
        ssh {
            root-login allow;
        }
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
    }
}
chassis {
    fpc 1 {
        pic 0 {
            tunnel-services;
        }
    }
}
interfaces {
    ge-1/0/0 {
        unit 0 {
            family inet {
                address 10.84.11.253/24;
```

```
            }
        }
    }
    ge-1/1/0 {
        description "IP Fabric interface";
        unit 0 {
            family inet {
                address 10.84.45.1/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 127.0.0.1/32;
            }
        }
    }
}
routing-options {
    static {
        route 0.0.0.0/0 next-hop 10.84.45.254;
    }
    route-distinguisher-id 10.84.11.253;
    autonomous-system 64512;
    dynamic-tunnels {
        setup1 {
            source-address 10.84.11.253;
            gre;
            destination-networks {
                10.84.11.0/24;
            }
        }
    }
}
protocols {
    bgp {
        group mx {
            type internal;
            local-address 10.84.11.253;
            family inet-vpn {
                unicast;
            }
```

```
            neighbor 10.84.11.252;
        }
        group contrail-controller {
            type internal;
            local-address 10.84.11.253;
            family inet-vpn {
                unicast;
            }
            neighbor 10.84.11.101;
            neighbor 10.84.11.102;
        }

    }
}
routing-instances {
    customer-public {
        instance-type vrf;
        interface ge-1/1/0.0;
        vrf-target target:64512:10000;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 10.84.45.254;
            }
        }
    }
}
```

MX80-2 Configuration

```
version 12.2R1.3;
system {
    root-authentication {
        encrypted-password "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 "xxxxxxxxxxxxx"; ## SECRET-DATA
    }
    login {
        class read {
            permissions [ clear interface view view-configuration ];
        }
        user admin {
            uid 2000;
            class super-user;
            authentication {
                encrypted-password "xxxxxxxxxxxx"; ## SECRET-DATA
            }
        }
        user user1 {
            uid 2002;
            class read;
            authentication {
                encrypted-password "xxxxxxxxxxxxxx"; ## 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 10

Configuring Services

IN THIS CHAPTER

- Configuring DNS Servers | 235
- Support for Multicast | 247
- Using Static Routes with Services | 250
- Configuring Metadata Service | 254

Configuring DNS Servers

IN THIS SECTION

- DNS Overview | 235
- Defining Multiple Virtual Domain Name Servers | 236
- IPAM and Virtual DNS | 237
- DNS Record Types | 237
- Configuring DNS Using the Interface | 238
- Configuring DNS Using Scripts | 246

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.

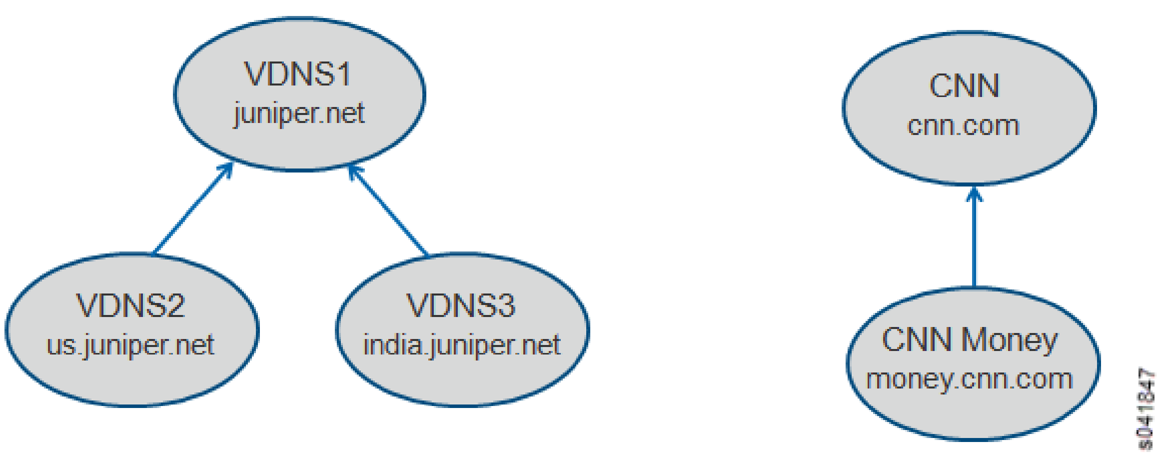
The following are the key attributes of domain name service in a virtual world:

- 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 81 on page 236 shows examples of virtual DNS servers defined in **default-domain**, providing the name service for the DNS domains indicated.

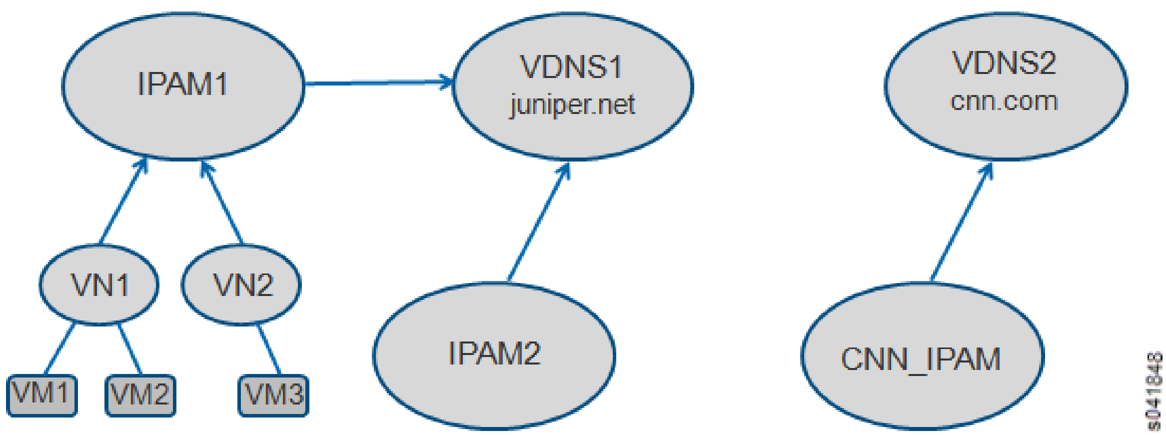
Figure 81: DNS Servers Examples



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 82 on page 237

Figure 82: 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 32 on page 237 for descriptions of the record types.

Table 32: DNS Record Types Supported

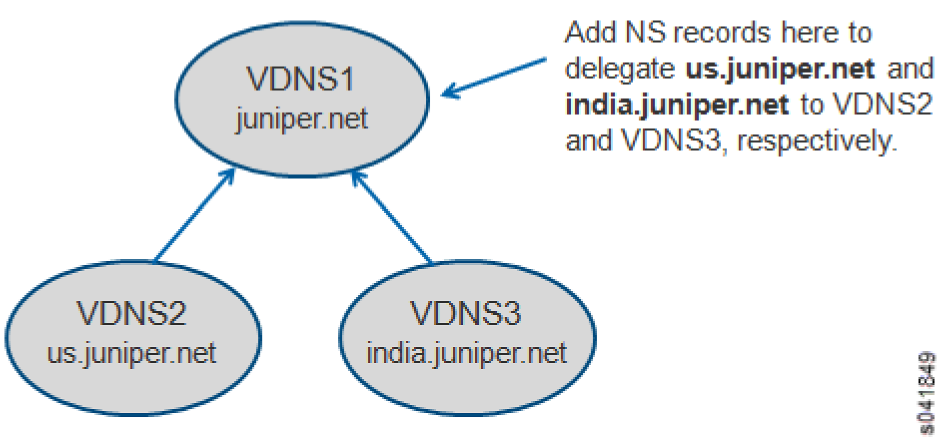
DNS Record Type	Description
A	Used for mapping hostnames to IPv4 addresses. Name refers to the name of the virtual machine, and data is the IPv4 address of the virtual machine.
CNAME	Provides an alias to a name. Name refers to the name of the virtual machine, and data is the new name (alias) for the virtual machine.

Table 32: DNS Record Types Supported *(Continued)*

DNS Record Type	Description
PTR	A pointer to a record, it provides reverse mapping from an IP address to a name. Name refers to the IP address, and data is the name for the virtual machine. The address in the PTR record should be part of a subnet configured for a VN within one of the IPAMs referring to this virtual DNS server.
NS	Used to delegate a subdomain to another DNS server. The DNS server could be another virtual DNS server defined in the system or the IP address of an external DNS server reachable via the infrastructure. Name refers to the subdomain being delegated, and data is the name of the virtual DNS server or IP address of an external server.

Figure 83 on page 238 shows an example usage for the DNS record type of **NS**.

Figure 83: Example Usage for NS Record Type



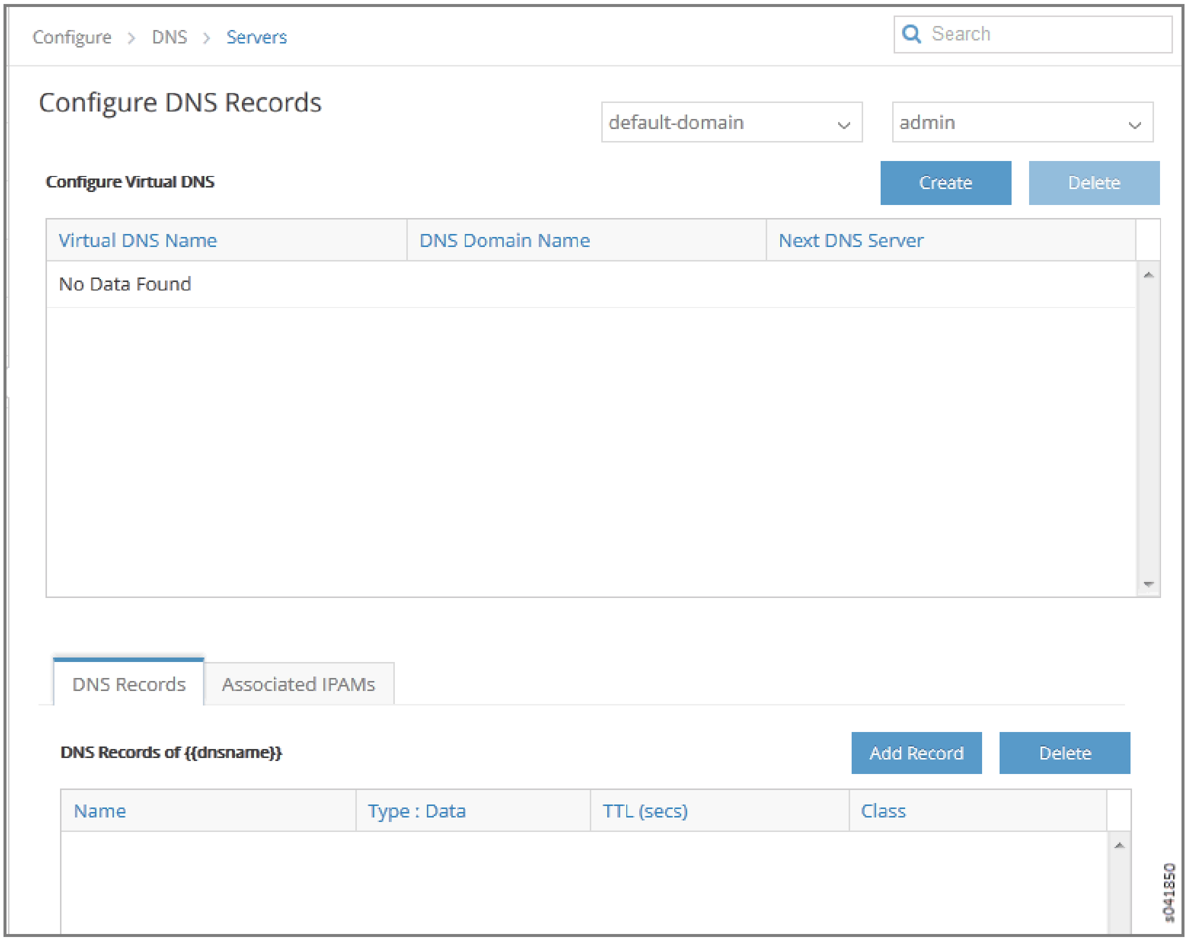
Configuring DNS Using the Interface

DNS can be configured by using the user interface or by using scripts. The following procedure shows how to configure DNS through the Juniper Networks Contrail interface.

1. Access **Configure** > **DNS** > **Servers** to create or delete virtual DNS servers and records.

   The **Configure DNS Records** page appears; see Figure 84 on page 239.

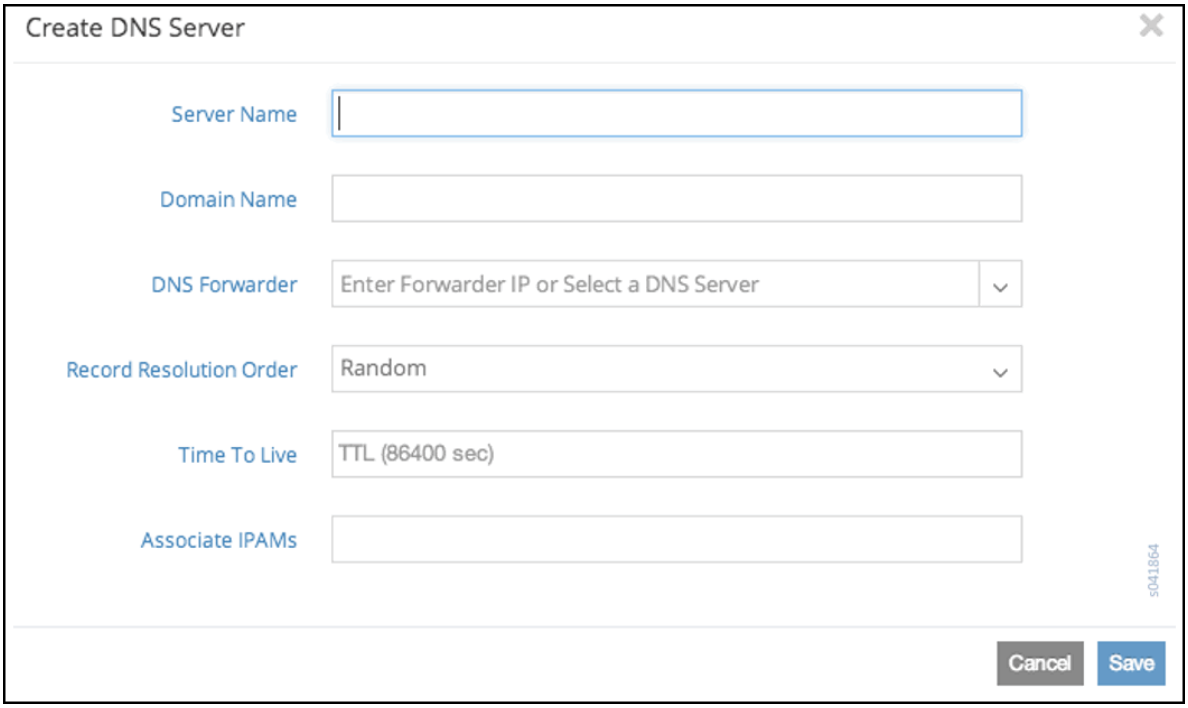
Figure 84: Configure DNS Records



2. To add a new DNS server, click the **Create** button.

   Enter DNS server information in the **Add DNS** window; see Figure 85 on page 240

Figure 85: Add DNS



Complete the fields for the new server; see Table 33 on page 240.

Table 33: Add DNS Fields

Field	Description
Server Name	Enter a name for this server.
Domain Name	Enter the name of the domain for this server.
Time To Live	Enter the **TTL** in seconds.
Next DNS Server	Select from a list the name of the next DNS server to process DNS requests if they cannot be processed at this server, or **None**.

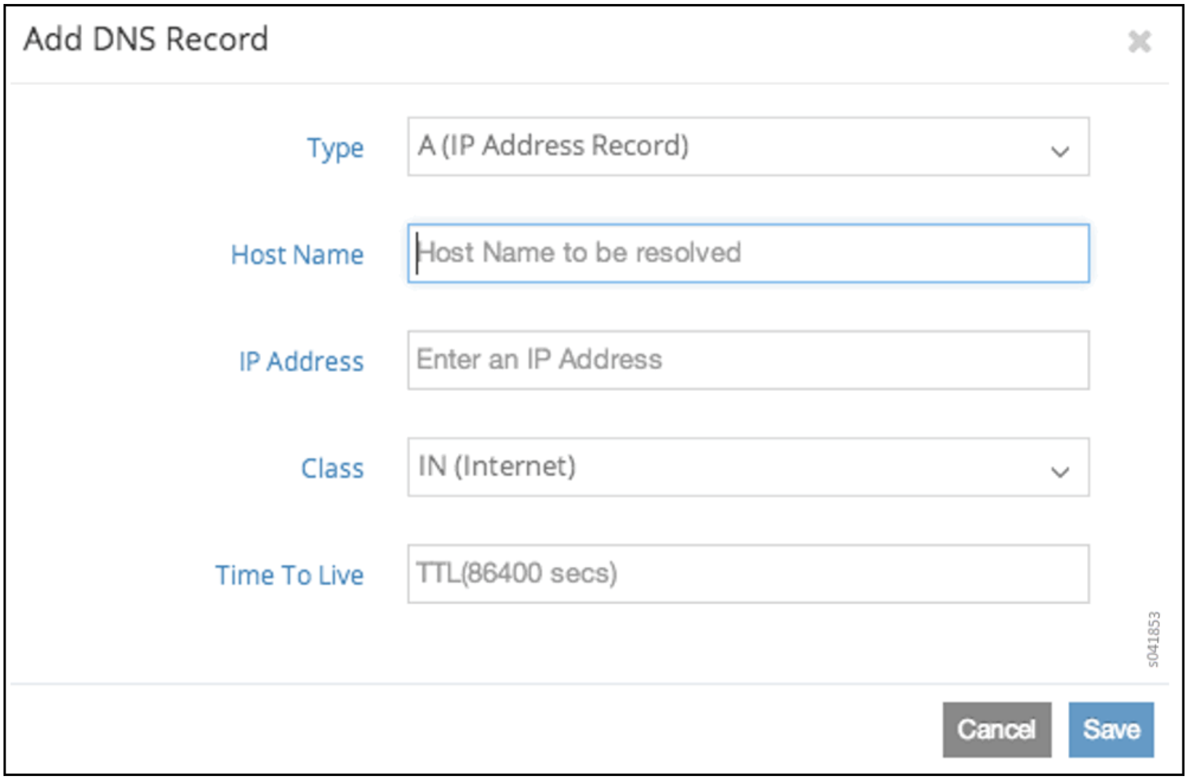
Table 33: Add DNS Fields *(Continued)*

Field	Description
Load Balancing Order	Select the load-balancing order from a list—**Random, Fixed, Round Robin**. When a name has multiple records matching, the configured record order determines the order in which the records are sent in the response. Select **Random** to have the records sent in random order. Select **Fixed** to have records sent in the order of creation. Select **Round Robin** to have the record order cycled for each request to the record.
OK	Click **OK** to create the record.
Cancel	Click **Cancel** to clear the fields and start over.

3. To add a new DNS record, from the **Configure DNS Records** page, click the **Add Record** button in the lower right portion of the screen.

   The **Add DNS Record** window appears; see Figure 86 on page 242.

Figure 86: Add DNS Record



4. Complete the fields for the new record; see Table 34 on page 242.

Table 34: Add DNS Record Fields

Field	Description
Record Name	Enter a name for this record.
Type	Select the record type from a list—**A, CNAME, PTR, NS.**
IP Address	Enter the IP address for the location for this record.
Class	Select the record class from a list—**IN** is the default.
Time To Live	Enter the **TTL** in seconds.
OK	Click **OK** to create the record.

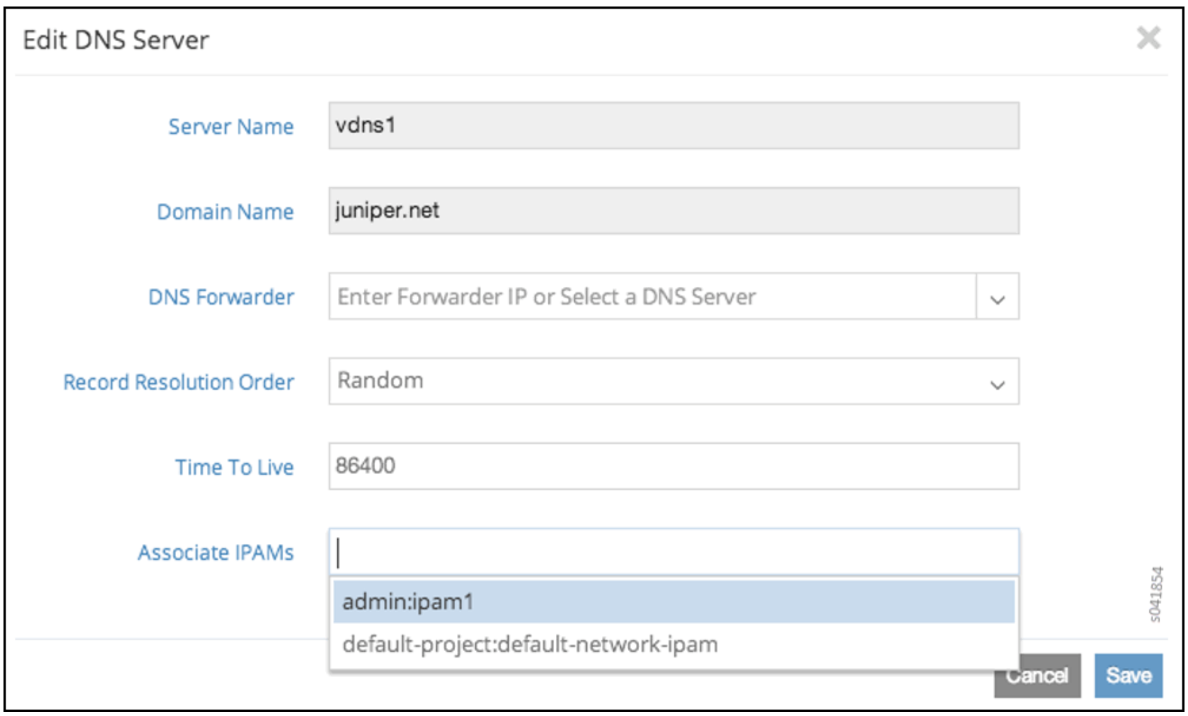
Table 34: Add DNS Record Fields ***(Continued)***

Field	Description
Cancel	Click **Cancel** to clear the fields and start over.

5. To associate an IPAM to a virtual DNS server, from the **Configure DNS Records** page, select the **Associated IPAMs** tab in the lower right portion of the screen and click the **Edit** button.

   The **Associate IPAMs to DNS** window appears; see Figure 87 on page 243.

Figure 87: Associate IPAMs to DNS



Complete the IPAM associations, using the field descriptions in Table 35 on page 243.

Table 35: Associate IPAMs to DNS Fields

Field	Description
Associate to All IPAMs	Select this box to associate the selected DNS server to all available IPAMs.

Table 35: Associate IPAMs to DNS Fields *(Continued)*

Field	Description
Available IPAMs	This column displays the currently available IPAMs.
Associated IPAMs	This column displays the IPAMs currently associated with the selected DNS server.
>>	Use this button to associate an available IPAM to the selected DNS server, by selecting an available IPAM in the left column and clicking this button to move it to the Associated IPAMs column. The selected IPAM is now associated with the selected DNS server.
<<	Use this button to disassociate an IPAM from the selected DNS server, by selecting an associated IPAM in the right column and clicking this button to move it to the left column (Available IPAMs). The selected IPAM is now disassociated from the selected DNS server.
OK	Click **OK** to commit the changes indicated in the window.
Cancel	Click **Cancel** to clear all entries and start over.

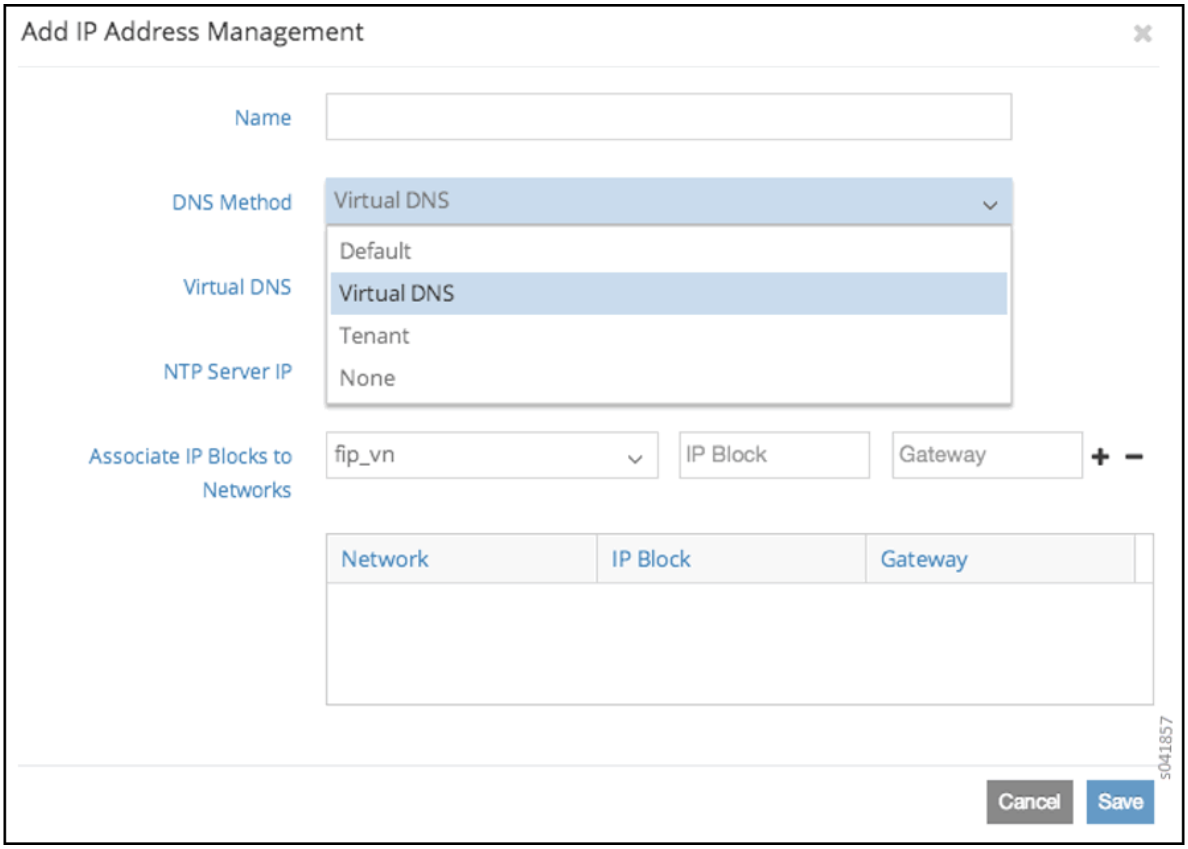
6. Use the **IP Address Management** page (**Configure > Networking > IP Address Management**); see Figure 88 on page 244) to configure the DNS mode for any DNS server and to associate an IPAM to DNS servers of any mode or to tenants' IP addresses.

Figure 88: Configure IP Address Management

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

   The **Edit IP Address Management** window appears; see Figure 89 on page 245.

Figure 89: DNS Server



8. In the first field, select the **DNS Method** from a list (**None, Default DNS, Tenant DNS, Virtual DNS**; see Table 36 on page 245.

Table 36: DNS Modes

DNS Mode	Description
None	Select **None** when no DNS support is required for the VMs.
Default	In default mode, DNS resolution for VMs is performed based on the name server configuration in the server infrastructure. The subnet default gateway is configured as the DNS server for the VM, and the DHCP response to the VM has this DNS server option. DNS requests sent by a VM to the default gateway are sent to the name servers configured on the respective compute nodes. The responses are sent back to the VM.

Table 36: DNS Modes *(Continued)*

DNS Mode	Description
Tenant	Configure this mode when a tenant wants to use its own DNS servers. Configure the list of servers in the IPAM. The server list is sent in the DHCP response to the VM as DNS servers. DNS requests sent by the VMs are routed the same as any other data packet based on the available routing information.
Virtual DNS	Configure this mode to support virtual DNS servers (VDNS) to resolve the DNS requests from the VMs. Each IPAM can have a virtual DNS server configured in this mode.

9. Complete the remaining fields on this page, and click **OK** to commit the changes, or click **Cancel** to clear the fields and start over.

Configuring DNS Using Scripts

DNS can be configured via the user interface or by using scripts that are available in the **opt/contrail/utils** directory. The scripts are described in Table 37 on page 246.

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 37: DNS Scripts

Action	Script
Add a virtual DNS server	Script: `add_virtual_dns.py` Sample usage: `python add_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --name vdns1 --domain_name default-domain --dns_domain juniper.net --dyn_updates --record_order random --ttl 1200 --next_vdns default-domain:vdns2`

Table 37: DNS Scripts *(Continued)*

Action	Script
Delete a virtual DNS server	Script: `del_virtual_dns_record.py` Sample usage: `python del_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --fq_name default-domain:vdns1`
Add a DNS record	Script: `add_virtual_dns_record.py` Sample usage: `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`
Delete a DNS record	Script: `del_virtual_dns_record.py` Sample usage: `python del_virtual_dns_record.py --api_server_ip 10.204.216.21 --api_server_port 8082 --fq_name default-domain:vdns1:rec1`
Associate a virtual DNS server with an IPAM	Script: `associate_virtual_dns.py` Sample usage: `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`
Disassociate a virtual DNS server with an IPAM	Script: `disassociate_virtual_dns.py` Sample usage: `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`

Support for Multicast

IN THIS SECTION

- Subnet Broadcast | **248**

- All-Broadcast/Limited-Broadcast and Link-Local Multicast | 249
- Host Broadcast | 249

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

Subnet Broadcast

Multiple subnets can be attached to a virtual network when it is spawned. Each of the subnets has one subnet broadcast route installed in the unicast routing table assigned to that virtual network. The recipient list for the subnet broadcast route includes all of the virtual machines that belong to that subnet. Packets originating from any VM in that subnet are replicated to all members of the recipient list, except the originator. Because the next hop is the list of recipients, it is called a composite next hop.

If there is no virtual machine spawned under a subnet, the subnet routing entry discards the packets received. If all of the virtual machines in a subnet are turned off, the routing entry points to discard. If the IPAM is deleted, the subnet route corresponding to that IPAM is deleted. If the virtual network is turned off, all of the subnet routes associated with the virtual network are removed.

Subnet Broadcast Example

The following configuration is made:

1. Virtual network name – **vn1**
2. Unicast routing instance – `vn1.uc.inet`
3. Subnets (IPAM) allocated – `1.1.1.0/24; 2.2.0.0/16; 3.3.0.0/16`
4. Virtual machines spawned – `vm1 (1.1.1.253); vm2 (1.1.1.252); vm3 (1.1.1.251); vm4 (3.3.1.253)`

The following subnet route additions are made to the routing instance `vn1.uc.inet.0`:

1. `1.1.1.255` -> forward to NH1 (composite next hop)
2. `2.2.255.255` -> DROP
3. `3.3.255.255` -> forward to NH2
4.
5. The following entries are made to the next-hop table:
6. NH1 – `1.1.1.253; 1.1.1.252; 1.1.1.251`
7. NH2 – `3.3.1.253`

If traffic originates for `1.1.1.255` from `vm1 (1.1.1.253)`, it will be forwarded to `vm2 (1.1.1.252)` and `vm3 (1.1.1.251)`. The originator `vm1 (1.1.1.253)` will not receive the traffic even though it is listed as a recipient in the next hop.

All-Broadcast/Limited-Broadcast and Link-Local Multicast

The address group `255.255.255.255` is used with all-broadcast (limited-broadcast) and multicast traffic. The route is installed in the multicast routing instance. The source address is recorded as ANY, so the route is `ANY/255.255.255.255 (*,G)`. It is unique per routing instance, and is associated with its corresponding virtual network. When a virtual network is spawned, it usually contains multiple subnets, in which virtual machines are added. All of the virtual machines, regardless of their subnets, are part of the recipient list for `ANY/255.255.255.255`. The replication is sent to every recipient except the originator.

Link-local multicast also uses the all-broadcast method for replication. The route is deleted when all virtual machines in this virtual network are turned off or the virtual network itself is deleted.

All-Broadcast Example

The following configuration is made:

1. Virtual network name – `vn1`
2. Unicast routing instance – `vn1.uc.inet`
3. Subnets (IPAM) allocated – `1.1.1.0/24; 2.2.0.0/16; 3.3.0.0/16`
4. Virtual machines spawned – `vm1 (1.1.1.253); vm2 (1.1.1.252); vm3 (1.1.1.251); vm4 (3.3.1.253)`

The following subnet route addition is made to the routing instance `vn1.uc.inet.0`:

1. `255.255.255.255/*` -> NH1
2.

The following entries are made to the next-hop table:

1. NH1 – `1.1.1.253; 1.1.1.252; 1.1.1.251; 3.3.1.253`

If traffic originates for `1.1.1.255` from `vm1 (1.1.1.253)`, the traffic is forwarded to `vm2 (1.1.1.252), vm3 (1.1.1.251)`, and `vm4 (3.3.1.253)`. The originator `vm1 (1.1.1.253)` will not receive the traffic even though it is listed as a recipient in the next hop.

Host Broadcast

The host broadcast route is present in the host routing instance so that the host operating system can send a subnet broadcast/all-broadcast (limited-broadcast). This type of broadcast is sent to the fabric by means of a **vhost** interface. Additionally, any subnet broadcast/all-broadcast received from the fabric will be handed over to the host operating system.

Using Static Routes with Services

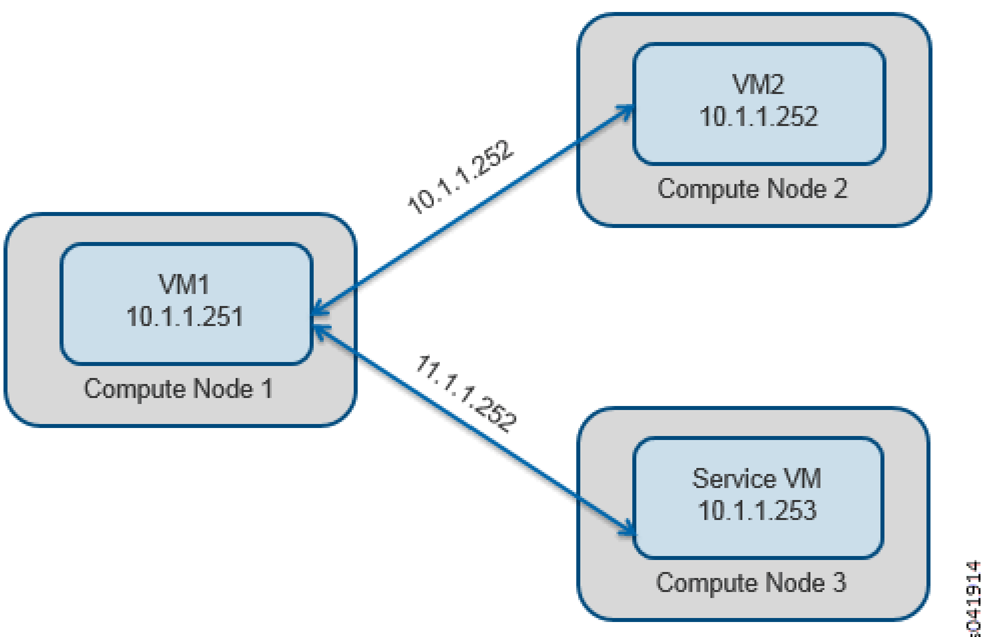
IN THIS SECTION

- Static Routes for Service Instances | 250
- Configuring Static Routes on a Service Instance | 251
- Configuring Static Routes on Service Instance Interfaces | 252
- Configuring Static Routes as Host Routes | 253

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.



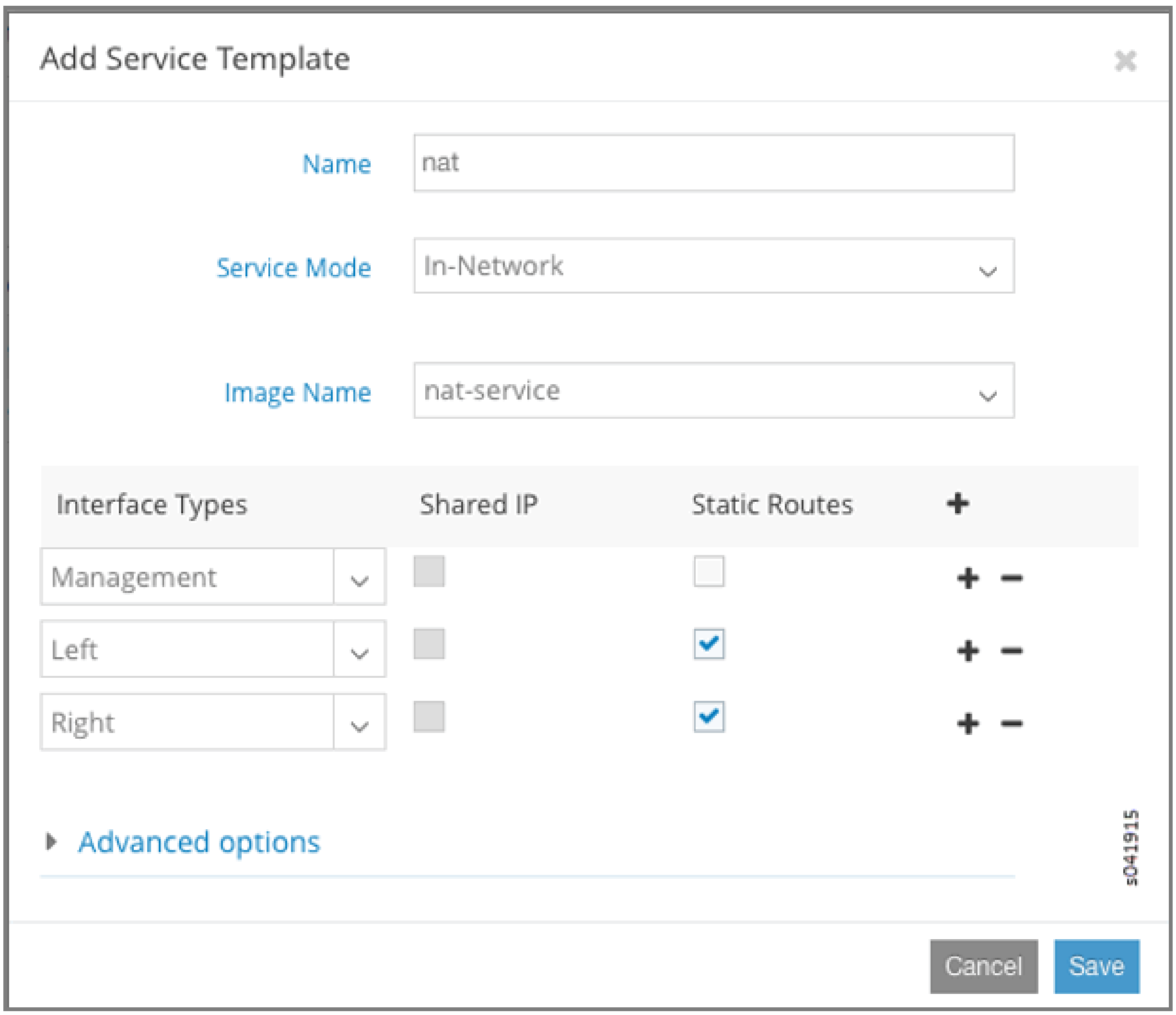
Configuring Static Routes on a Service Instance

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.



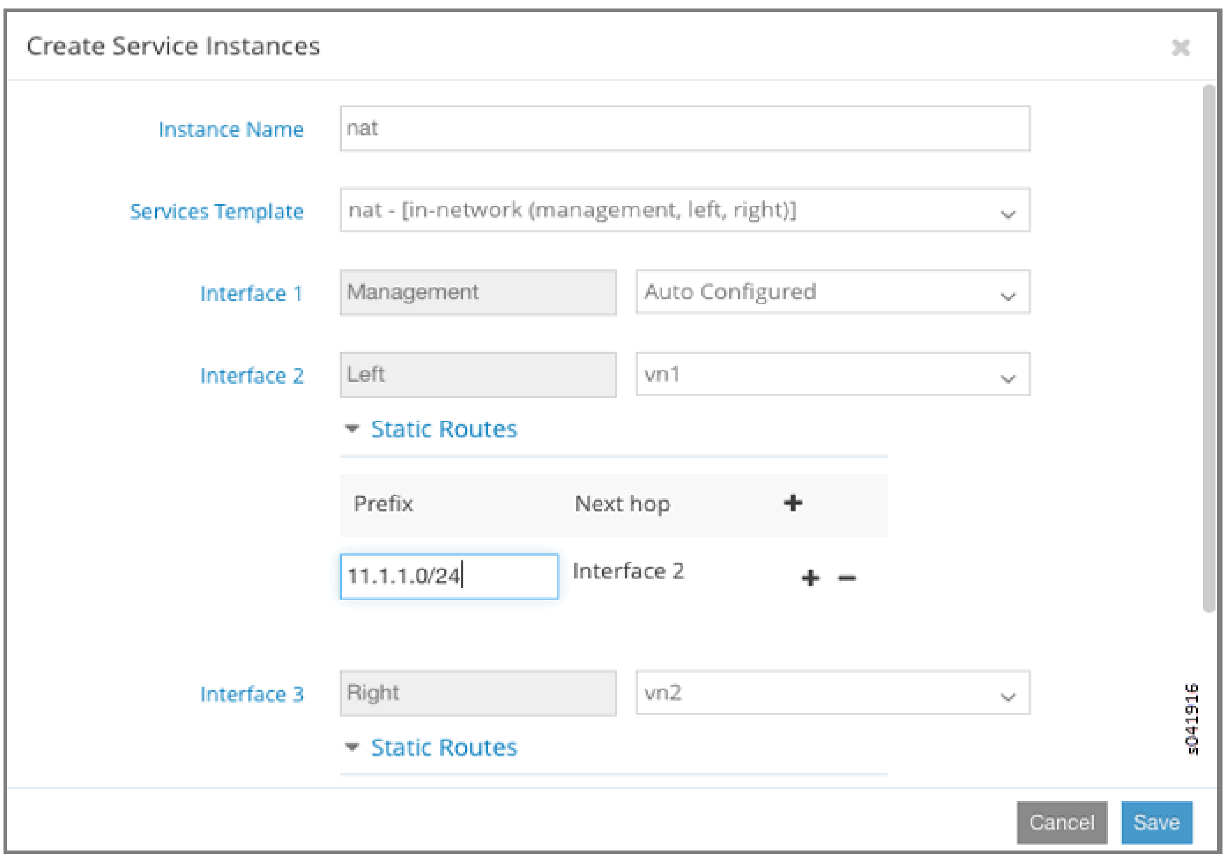
Configuring Static Routes on Service Instance Interfaces

To configure static routes on a service instance interface:

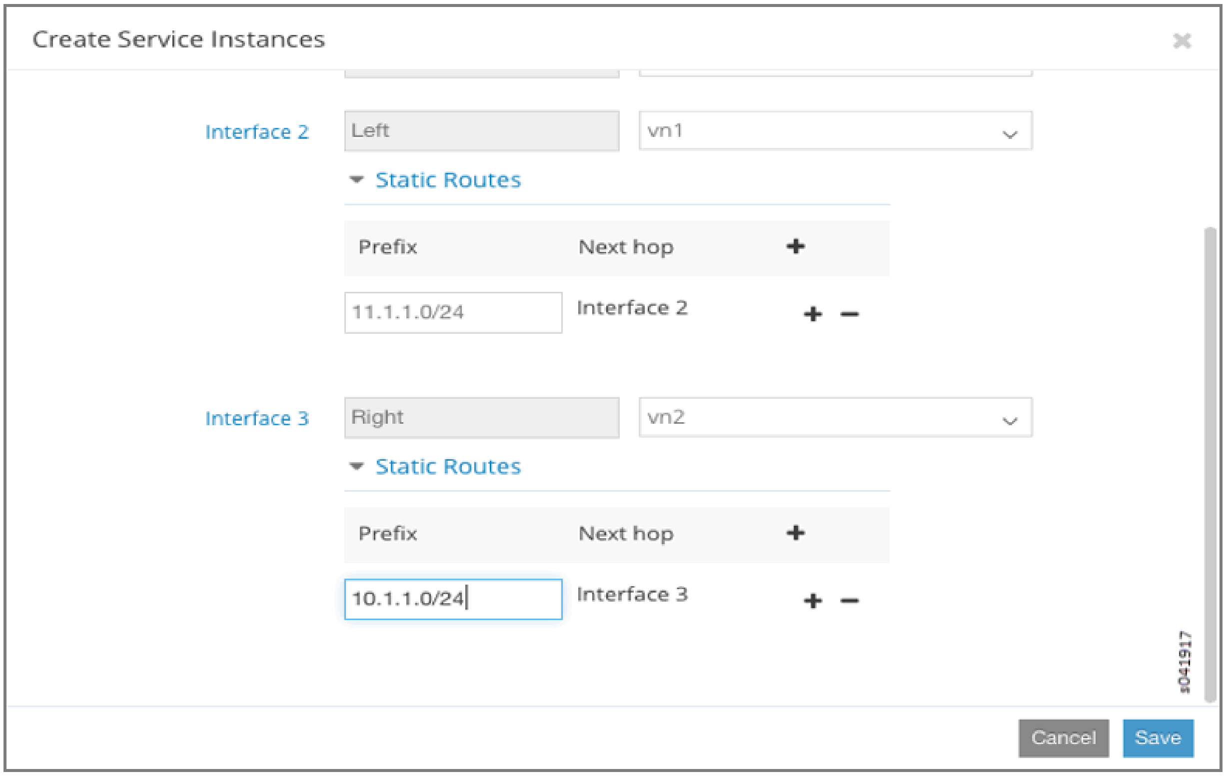
1. Go to **Configure** > **Services** > **Service Instances** and click **Create**.
2. At **Create Service Instances**, complete the fields for **Instance Name** and **Services Template**.
3. Select the virtual network for each of the interfaces
4. Click the **Static Routes** dropdown menu under each interface field for which the static routes option is enabled to open the **Static Routes** menu and configure the static routes in the fields provided.

> **NOTE**: If the **Auto Configured** option is selected, traffic destined to the static route subnet is load balanced across service instances.

The following figure shows a configuration to apply a service instance between VN1 (10.1.1.0/24) and VN2 (11.1.1.0/24). The left interface of the service instance is configured with VN1 and the right interface is configured to be VN2 (11.1.1.0/24). The static route 11.1.1.0/24 is configured on the left interface, so that all traffic from VN1 that is destined to VN2 reaches the left interface of the service instance.



The 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.



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

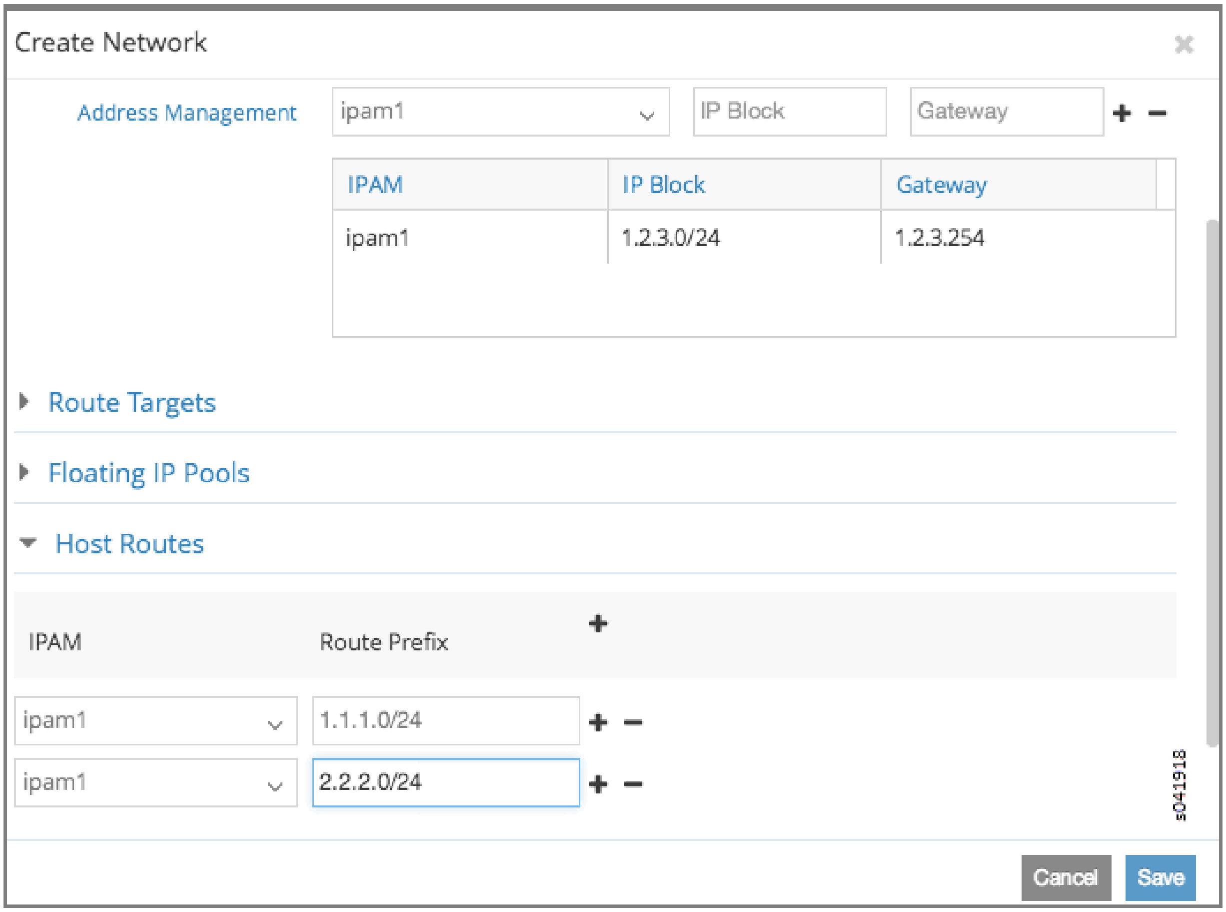
You can also use static routes for host routes for a virtual machine, by using the classless static routes option in the DHCP server response that is sent to the virtual machine.

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.



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 11

Configuring Service Chaining

IN THIS CHAPTER

- Service Chaining | 256
- Service Chaining MX Series Configuration | 260
- ECMP Load Balancing in the Service Chain | 262
- Customized Hash Field Selection for ECMP Load Balancing | 263
- Using the Contrail Heat Template | 268
- Service Chain Route Reorigination | 273
- Service Instance Health Checks | 295

Service Chaining

IN THIS SECTION

- Service Chaining Basics | 256
- Service Chaining Configuration Elements | 258

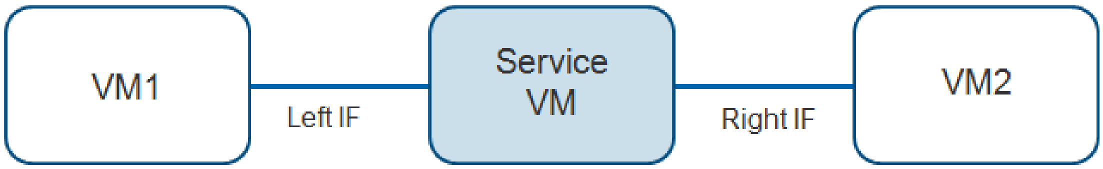
Contrail Controller supports chaining of various Layer 2 through Layer 7 services such as firewall, NAT, IDP, and so on.

Service Chaining Basics

Services are offered by instantiating service virtual machines to dynamically apply single or multiple services to virtual machine (VM) traffic. It is also possible to chain physical appliance-based services.

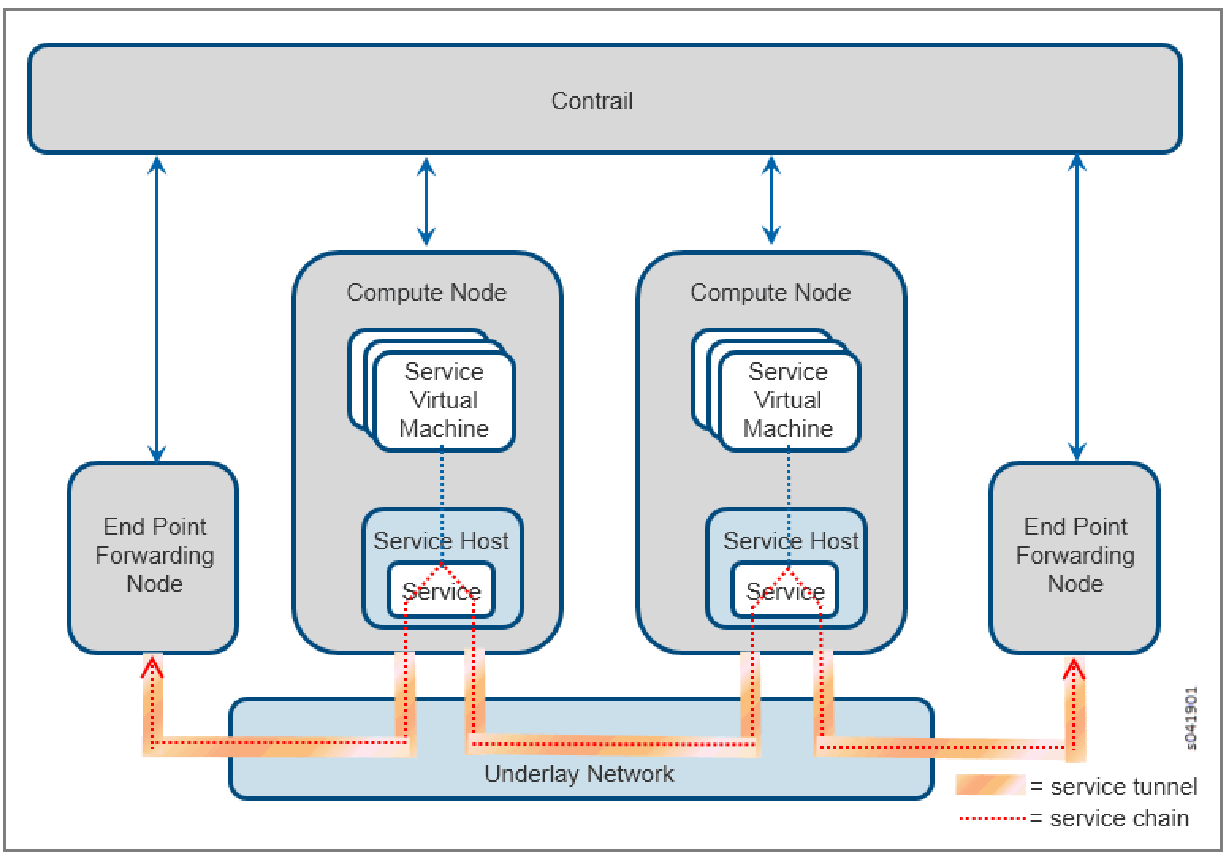
Figure 90 on page 257 shows the basic service chain schema, with a single service. The service VM spawns the service, using the convention of left interface (left IF) and right interface (right IF). Multiple services can also be chained together.

Figure 90: 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 91 on page 257 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 91: Contrail Service Chain



The following are the modes of services that can be configured.

1. *Transparent or bridge mode*

   a. Used for services that do not modify the packet. Also known as bump-in-the-wire or Layer 2 mode. Examples include Layer 2 firewall, IDP, and so on.

2. *In-network or routed mode*

   a. Provides a gateway service where packets are routed between the service instance interfaces. Examples include NAT, Layer 3 firewall, load balancer, HTTP proxy, and so on.

3. *In-network-nat mode*

   a. Similar to in-network mode, however, return traffic does not need to be routed to the source network. In-network-nat mode is particularly useful for NAT service.

Service Chaining Configuration Elements

Service chaining requires the following configuration elements in the solution:

- Service template
- Service instance
- Service policy

Service Template

Service templates are always configured in the scope of a domain, and the templates can be used on all projects within a domain. A template can be used to launch multiple service instances in different projects within a domain.

The following are the parameters to be configured for a service template:

- Service template name
- Domain name
- Service mode
  - Transparent
  - In-Network
  - In-Network NAT
- Image name (for virtual service)
  - If the service is a virtual service, then the name of the image to be used must be included in the service template. In an OpenStack setup, the image must be added to the setup by using Glance.

- Interface list
  - Ordered list of interfaces---this determines the order in which Interfaces will be created on the service instance.
  - Most service templates will have management, left, and right interfaces. For service instances requiring more interfaces, “other” interfaces can be added to the interface list.
  - Shared IP attribute, per interface
  - Static routes enabled attribute, per interface
- Advanced options
  - Service scaling— use this attribute to enable a service instance to have more than one instance of the service instance virtual machine.
  - Flavor—assign an OpenStack flavor to be used while launching the service instance. Flavors are defined in OpenStack Nova with attributes such as assignments of CPU cores, memory, and disk space.

Service Instance

A service instance is always maintained within the scope of a project. A service instance is launched using a specified service template from the domain to which the project belongs.

The following are the parameters to be configured for a service instance:

- Service instance name
- Project name
- Service template name
- Number of virtual machines that will be spawned
  - Enable service scaling in the service template for multiple virtual machines
- Ordered virtual network list
  - Interfaces listed in the order specified in the service template
  - Identify virtual network for each interface
  - Assign static routes for virtual networks that have static route enabled in the service template for their interface
    - Traffic that matches an assigned static route is directed to the service instance on the interface created for the corresponding virtual network

Service Policy

The following are the parameters to be configured for a service policy:

- Policy name
- Source network name
- Destination network name
- Other policy match conditions, for example direction and source and destination ports
- Policy configured in “routed/in-network” or “bridged/” mode
- An action type called **apply_service** is used:

  1. Example: **'apply_service'**: **[DomainName:ProjectName:ServiceInstanceName]**

RELATED DOCUMENTATION

Example: Creating an In-Network Service Chain by Using Contrail Command
Example: Creating an In-Network-NAT Service Chain by Using Contrail Command
Example: Creating a Transparent Service Chain by Using Contrail Command
ECMP Load Balancing in the Service Chain | 262

Service Chaining MX Series Configuration

This topic shows how to extend service chaining to the MX Series routers.

To configure service chaining for MX Series routers, extend the virtual networks to the MX Series router and program routes so that traffic generated from a host connected to the router can be routed through the service.

1. The following configuration snippet for an MX Series router has a left virtual network called `enterprise` and a right virtual network called `public`. The configuration creates two routing instances with loopback interfaces and route targets.

```
routing-instances {
     enterprise {
         instance-type vrf;
         interface lo0.1;
         vrf-target target:100:20000;
```

```
        }
        public {
            instance-type vrf;
            interface lo0.2;
            vrf-target target:100:10000;
routing-options {
        static {
 route 0.0.0.0/0 next-hop 10.84.20.1
        }
 }
 interface xe-0/0/0.0;
        }
 }
```

2. The following configuration snippet shows the configuration for the loopback interfaces.

```
interfaces {
    lo0 {
        unit 1 {
            family inet {
                address 2.1.1.100/32;
            }
        }
        unit 2 {
            family inet {
                address 200.1.1.1/32;
            }
        }
    }
}
```

3. The following configuration snippet shows the configuration to enable BGP. The `neighbor 10.84.20.39` and `neighbor 10.84.20.40` are control nodes.

```
protocols {
    bgp {
        group demo_contrail {
            type internal;
            description "To Contrail Control Nodes & other MX";
            local-address 10.84.20.252;
            keep all;
            family inet-vpn {
```

```
                unicast;
            }
            neighbor 10.84.20.39;
            neighbor 10.84.20.40;
        }
}
```

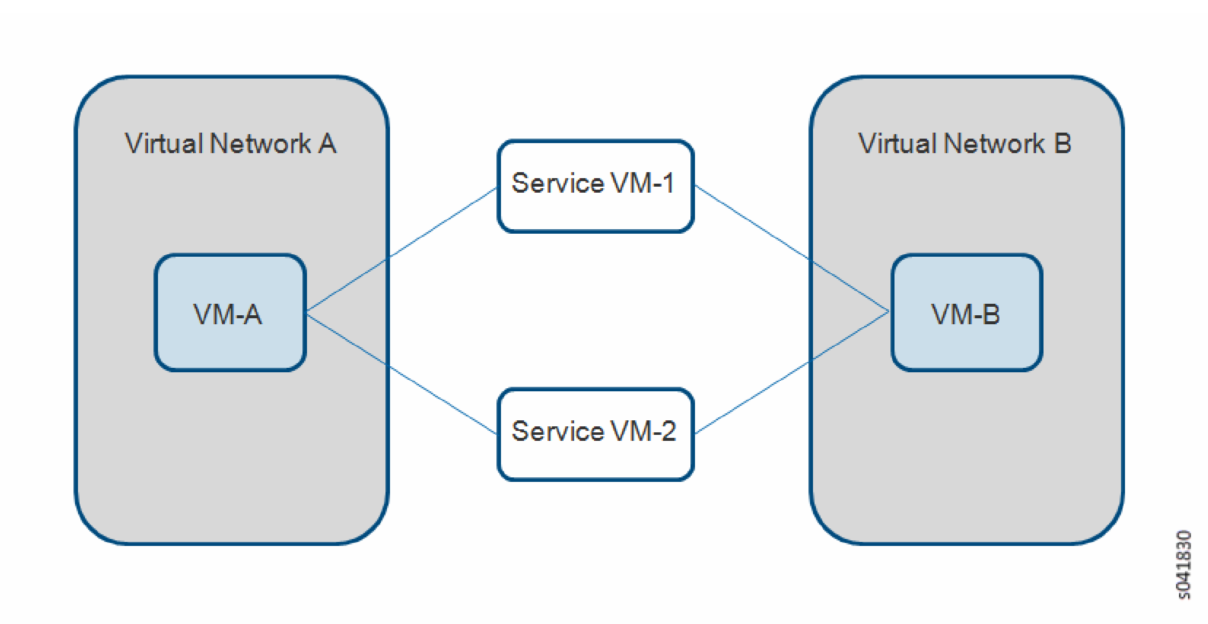
4. The final step is to add `target:100:10000` to the public virtual network and `target:100:20000` to the enterprise virtual network, using the Contrail Juniper Networks interface.

A full MX Series router configuration for Contrail can be seen in "Sample Network Configuration for Devices for Simple Tiered Web Application" on page 228.

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 92 on page 262, 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 92: 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 | 256
Customized Hash Field Selection for ECMP Load Balancing | 263

Customized Hash Field Selection for ECMP Load Balancing

IN THIS SECTION

- Overview: Custom Hash Feature | 263
- Using ECMP Hash Fields Selection | 265
- Sample Flows | 266

Overview: Custom Hash Feature

Starting with Contrail Release 3.0, it is possible to configure the set of fields used to hash upon during equal-cost multipath (ECMP) load balancing.

Earlier versions of Contrail had this set of fields fixed to the standard 5-tuple set of: source L3 address, destination L3 address, L4 protocol, L4 SourcePort, and L4 DestinationPort.

With the custom hash feature, users can configure an exact subset of fields to hash upon when choosing the forwarding path among a set of eligible ECMP candidates.

The custom hash configuration can be applied in the following ways:

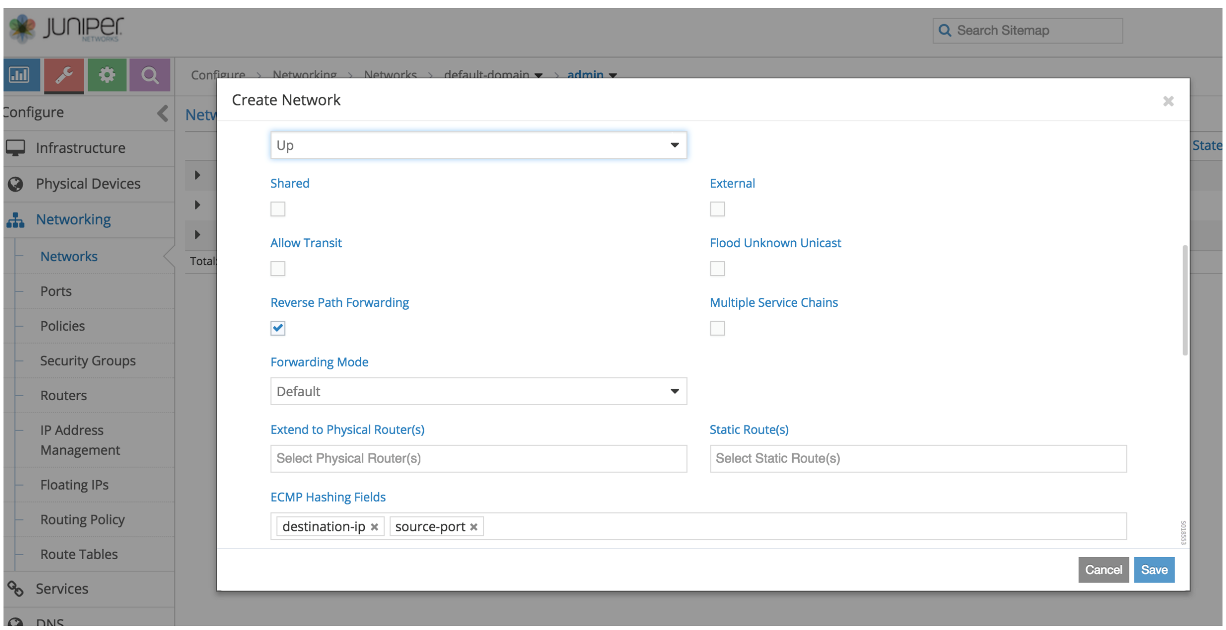
- globally

- per virtual network (VN)
- per virtual network interface (VNI)

VNI configurations take precedence over VN configurations, and VN configurations take precedence over global level configuration (if present).

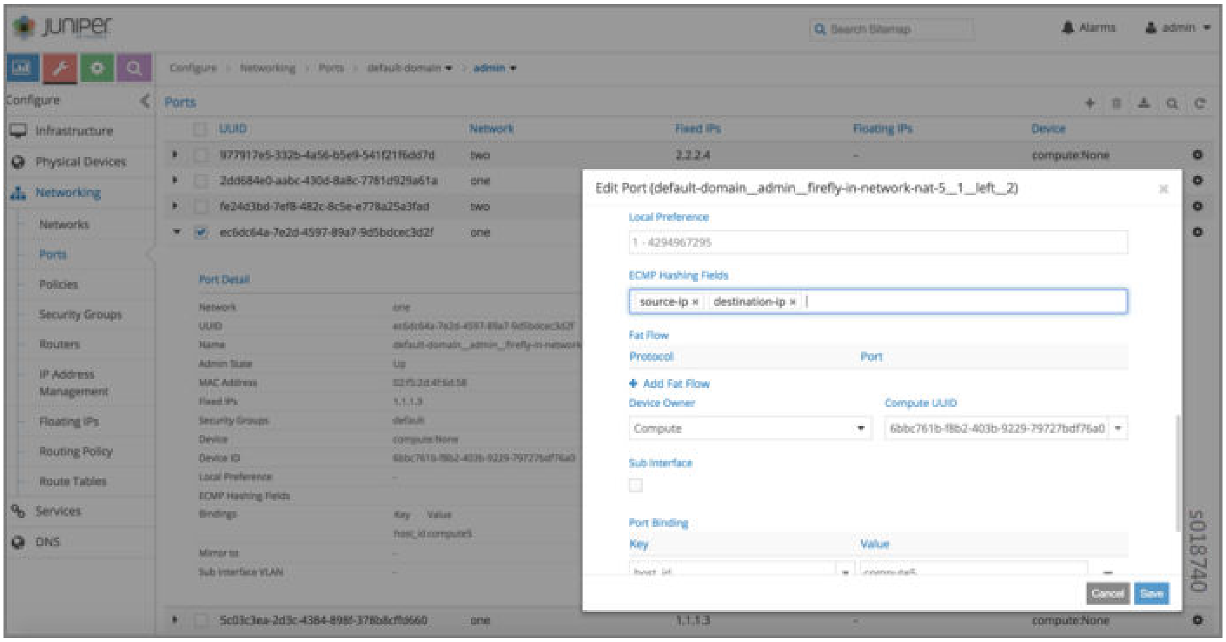
Custom hash is useful whenever packets originating from a particular source and addressed to a particular destination must go through the same set of service instances during transit. This might be required if source, destination, or transit nodes maintain a certain state based on the flow, and the state behavior could also be used for subsequent new flowsl, between the same pair of source and destination addresses. In such cases, subsequent flows must follow the same set of service nodes followed by the initial flow.

You can use the Contrail UI to identify specific fields in the network upon which to hash at the **Configure > Networking > Network, Create Network** window, in the **ECMP Hashing Fields** section as shown in the following figure.



If the hashing fields are configured for a virtual network, all traffic destined to that VN will be subject to the customized hash field selection during forwarding over ECMP paths by vRouters. This may not be desirable in all cases, as it could potentially skew all traffic to the destination network over a smaller set of paths across the IP fabric.

A more practical scenario is one in which flows between a source and destination must go through the same service instance in between, where one could configure customized ECMP fields for the virtual machine interface (VMI) of the service instance. Then, each service chain route originating from that VMI would get the desired ECMP field selection applied as its path attribute, and eventually get propagated to the ingress vRouter node. See the following example.



Using ECMP Hash Fields Selection

Custom hash fields selection is most useful in scenarios where multiple ECMP paths exist for a destination. Typically, the multiple ECMP paths point to ingress service instance nodes, which could be running anywhere in the Contrail cloud.

Configuring ECMP Hash Fields Over Service Chains

Use the following steps to create customized hash fields with ECMP over service chains.

1. Create the virtual networks needed to interconnect using service chaining, with ECMP load-balancing.

2. Create a service template and enable scaling.

3. Create a service instance, and using the service template, configure by selecting:

   - the desired number of instances for scale-out

   - the left and right virtual network to connect

   - the shared address space, to make sure that instantiated services come up with the same IP address for left and right, respectively

   This configuration enables ECMP among all those service instances during forwarding.

4. Create a policy, then select the service instance previously created and apply the policy to to the desired VMIs or VNs.

5. After the service VMs are instantiated, the ports of the left and right interfaces are available for further configuration. At the Contrail UI Ports section under Networking, select the left port (VMI) of the service instance and apply the desired ECMP hash field configuration.

> **NOTE**: Currently the ECMP field selection configuration for the service instance left or right interface must be applied by using the Ports (VMIs) section under Networking and explicitly configuring the ECMP fields selection for each of the instantiated service instances' VMIs. This must be done for all service interfaces of the group, to ensure the end result is as expected, because the load balance attribute of only the best path is carried over to the ingress vRouter. If the load balance attribute is not configured, it is not propagated to the ingress vRouter, even if other paths have that configuration.

When the configuration is finished, the vRouters get programmed with routing tables with the ECMP paths to the various service instances. The vRouters are also programmed with the desired ECMP hash fields to be used during load balancing of the traffic.

Sample Flows

This section provides sample flows with and without ECMP custom hash field selection.

Sample Traffic Flow Path Without Custom ECMP Hash Fields

The following is an example of a traffic flow path without using a customized ECMP hash fields selection configuration. The flow is configured with standard 5-tuple flow fields.

```
tcpdump -i eth0 'port 1023 and tcp[tcpflags] & (tcp-syn) != 0 and tcp[tcpflags] & (tcp-ack) == 0'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
14:55:10.115122 IP 2.2.2.5.18337 > 2.2.2.100.1023: Flags [S], seq 2276852196, win 29200, options
[mss 1398,sackOK,TS val 25208882 ecr 0,nop,wscale 7], length 0
14:55:10.132753 IP 2.2.2.4.21193 > 2.2.2.100.1023: Flags [S], seq 4161487314, win 29200, options
[mss 1398,sackOK,TS val 25208886 ecr 0,nop,wscale 7], length 0
14:55:10.152053 IP 2.2.2.5.24230 > 2.2.2.100.1023: Flags [S], seq 2466454857, win 29200, options
[mss 1398,sackOK,TS val 25208892 ecr 0,nop,wscale 7], length 0
14:55:11.146029 IP 2.2.2.5.24230 > 2.2.2.100.1023: Flags [S], seq 2466454857, win 29200, options
[mss 1398,sackOK,TS val 25209142 ecr 0,nop,wscale 7], length 0
14:55:13.147616 IP 2.2.2.5.24230 > 2.2.2.100.1023: Flags [S], seq 2466454857, win 29200, options
[mss 1398,sackOK,TS val 25209643 ecr 0,nop,wscale 7], length 0
14:55:13.164367 IP 2.2.2.3.25582 > 2.2.2.100.1023: Flags [S], seq 2259034580, win 29200, options
[mss 1398,sackOK,TS val 25209644 ecr 0,nop,wscale 7], length 0
```

```
14:55:13.179939 IP 2.2.2.5.24895 > 2.2.2.100.1023: Flags [S], seq 2174031724, win 29200, options
[mss 1398,sackOK,TS val 25209648 ecr 0,nop,wscale 7], length 0
14:55:14.168282 IP 2.2.2.5.24895 > 2.2.2.100.1023: Flags [S], seq 2174031724, win 29200, options
[mss 1398,sackOK,TS val 25209898 ecr 0,nop,wscale 7], length 0
14:55:16.172384 IP 2.2.2.5.24895 > 2.2.2.100.1023: Flags [S], seq 2174031724, win 29200, options
[mss 1398,sackOK,TS val 25210399 ecr 0,nop,wscale 7], length 0
14:55:16.189864 IP 2.2.2.5.22952 > 2.2.2.100.1023: Flags [S], seq 3099816842, win 29200, options
[mss 1398,sackOK,TS val 25210401 ecr 0,nop,wscale 7], length 0
14:55:16.205142 IP 2.2.2.4.16487 > 2.2.2.100.1023: Flags [S], seq 3961114202, win 29200, options
[mss 1398,sackOK,TS val 25210405 ecr 0,nop,wscale 7], length 0
14:55:17.196763 IP 2.2.2.4.16487 > 2.2.2.100.1023: Flags [S], seq 3961114202, win 29200, options
[mss 1398,sackOK,TS val 25210655 ecr 0,nop,wscale 7], length 0
14:55:19.200623 IP 2.2.2.4.16487 > 2.2.2.100.1023: Flags [S], seq 3961114202, win 29200, options
[mss 1398,sackOK,TS val 25211156 ecr 0,nop,wscale 7], length 0
14:55:19.215809 IP 2.2.2.3.18914 > 2.2.2.100.1023: Flags [S], seq 3157557440, win 29200, options
[mss 1398,sackOK,TS val 25211158 ecr 0,nop,wscale 7], length 0
14:55:19.228405 IP 2.2.2.7.15569 > 2.2.2.100.1023: Flags [S], seq 3850648420, win 29200, options
[mss 1398,sackOK,TS val 25211161 ecr 0,nop,wscale 7], length 0
14:55:20.223482 IP 2.2.2.7.15569 > 2.2.2.100.1023: Flags [S], seq 3850648420, win 29200, options
[mss 1398,sackOK,TS val 25211412 ecr 0,nop,wscale 7], length 0
14:55:22.232068 IP 2.2.2.7.15569 > 2.2.2.100.1023: Flags [S], seq 3850648420, win 29200, options
[mss 1398,sackOK,TS val 25211913 ecr 0,nop,wscale 7], length 0
14:55:22.247325 IP 2.2.2.4.28388 > 2.2.2.100.1023: Flags [S], seq 3609240658, win 29200, options
[mss 1398,sackOK,TS val 25211915 ecr 0,nop,wscale 7], length 0
```

Sample Traffic Flow Path With Custom ECMP Hash Fields

The following is an example of a traffic flow path using a customized ECMP hash fields selection configuration, for `source-ip` and `destination-ip` only.

```
tcpdump -i eth0 'port 1023 and tcp[tcpflags] & (tcp-syn) != 0 and tcp[tcpflags] & (tcp-ack) == 0'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
15:57:18.680853 IP 2.2.2.4.21718 > 2.2.2.100.1023: Flags [S], seq 2052086108, win 29200, options
[mss 1398,sackOK,TS val 26141024 ecr 0,nop,wscale 7], length 0
15:57:18.696114 IP 2.2.2.4.13585 > 2.2.2.100.1023: Flags [S], seq 2039627277, win 29200, options
[mss 1398,sackOK,TS val 26141028 ecr 0,nop,wscale 7], length 0
15:57:18.714846 IP 2.2.2.4.16414 > 2.2.2.100.1023: Flags [S], seq 3252526560, win 29200, options
[mss 1398,sackOK,TS val 26141033 ecr 0,nop,wscale 7], length 0
15:57:18.731281 IP 2.2.2.4.32499 > 2.2.2.100.1023: Flags [S], seq 1389133175, win 29200, options
[mss 1398,sackOK,TS val 26141037 ecr 0,nop,wscale 7], length 0
```

```
15:57:18.747051 IP 2.2.2.4.6081 > 2.2.2.100.1023: Flags [S], seq 427936299, win 29200, options
[mss 1398,sackOK,TS val 26141041 ecr 0,nop,wscale 7], length 0
15:57:19.740204 IP 2.2.2.4.6081 > 2.2.2.100.1023: Flags [S], seq 427936299, win 29200, options
[mss 1398,sackOK,TS val 26141291 ecr 0,nop,wscale 7], length 0
15:57:21.743951 IP 2.2.2.4.6081 > 2.2.2.100.1023: Flags [S], seq 427936299, win 29200, options
[mss 1398,sackOK,TS val 26141792 ecr 0,nop,wscale 7], length 0
15:57:21.758532 IP 2.2.2.4.13800 > 2.2.2.100.1023: Flags [S], seq 3020971712, win 29200, options
[mss 1398,sackOK,TS val 26141794 ecr 0,nop,wscale 7], length 0
15:57:21.772646 IP 2.2.2.4.23894 > 2.2.2.100.1023: Flags [S], seq 3373734307, win 29200, options
[mss 1398,sackOK,TS val 26141797 ecr 0,nop,wscale 7], length 0
15:57:22.764469 IP 2.2.2.4.23894 > 2.2.2.100.1023: Flags [S], seq 3373734307, win 29200, options
[mss 1398,sackOK,TS val 26142047 ecr 0,nop,wscale 7], length 0
15:57:24.768511 IP 2.2.2.4.23894 > 2.2.2.100.1023: Flags [S], seq 3373734307, win 29200, options
[mss 1398,sackOK,TS val 26142548 ecr 0,nop,wscale 7], length 0
15:57:24.784119 IP 2.2.2.4.21858 > 2.2.2.100.1023: Flags [S], seq 2212369297, win 29200, options
[mss 1398,sackOK,TS val 26142550 ecr 0,nop,wscale 7], length 0
15:57:24.797149 IP 2.2.2.4.29440 > 2.2.2.100.1023: Flags [S], seq 2007897735, win 29200, options
[mss 1398,sackOK,TS val 26142554 ecr 0,nop,wscale 7], length 0
15:57:25.792816 IP 2.2.2.4.29440 > 2.2.2.100.1023: Flags [S], seq 2007897735, win 29200, options
[mss 1398,sackOK,TS val 26142804 ecr 0,nop,wscale 7], length 0
15:57:27.797538 IP 2.2.2.4.29440 > 2.2.2.100.1023: Flags [S], seq 2007897735, win 29200, options
[mss 1398,sackOK,TS val 26143305 ecr 0,nop,wscale 7], length 0
15:57:27.814002 IP 2.2.2.4.23452 > 2.2.2.100.1023: Flags [S], seq 1659332655, win 29200, options
[mss 1398,sackOK,TS val 26143307 ecr 0,nop,wscale 7], length 0
```

Using the Contrail Heat Template

IN THIS SECTION

- Introduction to Heat | 269
- Heat Architecture | 269
- Support for Heat Version 2 Resources | 269
- Heat Version 2 with Service Chaining and Port Tuple Sample Workflow | 270
- Example: Creating a Service Template Using Heat | 271

Heat is the orchestration engine of the OpenStack program. Heat enables launching multiple cloud applications based on templates that are comprised of text files.

Introduction to Heat

A Heat template describes the infrastructure for a cloud application, such as networks, servers, floating IP addresses, and the like, and can be used to manage the entire life cycle of that application.

When the application infrastructure changes, the Heat templates can be modified to automatically reflect those changes. Heat can also delete all application resources if the system is finished with an application.

Heat templates can record the relationships between resources, for example, which networks are connected by means of policy enforcements, and consequently call OpenStack REST APIs that create the necessary infrastructure, in the correct order, needed to launch the application managed by the Heat template.

Heat Architecture

Heat is implemented by means of Python applications, including the following:

- `heat-client`—The CLI tool that communicates with the `heat-api` application to run Heat APIs.
- `heat-api`—Provides an OpenStack native REST API that processes API requests by sending them to the Heat engine over remote procedure calls (RPCs).
- `heat-engine`—Responsible for orchestrating the launch of templates and providing events back to the API consumer.

Support for Heat Version 2 Resources

Starting with Contrail Release 3.0.2, Contrail Heat resources and templates are autogenerated from the Contrail schema, using Heat Version 2 resources. Contrail Release 3.0.2 is the minimum required version for using Heat with Contrail in 3.x releases. The Contrail Heat Version 2 resources are of the following hierarchy: `OS::ContrailV2::<ResourceName>`.

The generated resources and templates are part of the Contrail Python package, and are located in the following directory in the target installation:

/usr/lib/python2.7/dist-packages/vnc_api/gen/heat/

The **heat/** directory has the following subdirectories:

- **resources/**—Contains all the resources for the contrail-heat plugin, which runs in the context of the Heat engine service.

- **templates/**—Contains sample templates for each resource. Each sample template presents every possible parameter in the schema. Use the sample templates as a reference when you build up more complex templates for your network design.

- **env/**—Contains the environment for input to each template.

The following contains a list of all the generated plug-in resources that are supported by `contrail-heat` in Contrail Release 3.0.2 and greater:

https://github.com/Juniper/contrail-heat/tree/master/generated/resources

The following contains a list of new example templates:

https://github.com/Juniper/contrail-heat/tree/master/contrail_heat/new_templates

Deprecation of Heat Version 1 Resources

Heat Version 1 resources within the hierarchy `OS::Contrail::<ResourceName>` are being deprecated, and you should not create new service chains using the Heat Version 1 templates.

Heat Version 2 with Service Chaining and Port Tuple Sample Workflow

With Contrail service templates Version 2, the user can create ports and bind them to a virtual machine (VM)-based service instance, by means of a port-tuple object. All objects created with the Version 2 service template are directly visible to the Contrail Heat engine, and are directly managed by Heat.

The following shows the basic workflow steps for creating a port tuple and service instance that will be managed by Heat:

1. Create a service template. Select 2 in the Version field.

2. Create a service instance for the service template just created.

3. Create a port-tuple object.

4. Create ports, using Nova VM launch or without a VM launch.

5. Label each port as left, right, mgmt, and so on, and add the ports to the port-tuple object.

   Use a unique label for each of the ports in a single port tuple. The labels named left and right are used for forwarding.

6. Link the port tuple to a service instance.

7. Launch the service instance.

Example: Creating a Service Template Using Heat

The following is an example of how to create a service template using Heat.

1. Define a template to create the service template.

```
service_template.yaml
heat_template_version:  2013--05--23
description: >
   HOT template to create a service template
parameters:
   name:
      type: string
      description: Name of service template
   mode:
      type: string
      description: service mode
   type:
      type: string
      description: service type
   image:
      type: string
      description: Name of the image
   flavor:
      type: string
      description: Flavor
   service_interface_type_list:
      type: string
      description: List of interface types
   shared_ip_list:
      type: string
      description: List of shared ip enabled--disabled
   static_routes_list:
      type: string
      description: List of static routes enabled--disabled

resources:
   service_template:
      type: OS::ContrailV2::ServiceTemplate
      properties:
         name: { get_param: name }
         service_mode: { get_param: mode }
```

```
         service_type: { get_param: type }
         image_name: { get_param: image }
         flavor: { get_param: flavor }
         service_interface_type_list: { "Fn::Split" : [ ",", Ref:
service_interface_type_list ] }
         shared_ip_list: { "Fn::Split" : [ ",", Ref: shared_ip_list ] }
         static_routes_list: { "Fn::Split" : [ ",", Ref: static_routes_list ] }
   outputs:
      service_template_fq_name:
         description: FQ name of the service template
         value: { get_attr: [ service_template, fq_name] }

}
```

2. Create an environment file to define the values to put in the variables in the template file.

```
service_template.env

parameters:

   name: contrail_svc_temp

   mode: transparent

   type: firewall

   image: cirros

   flavor: m1.tiny

   service_interface_type_list: management,left,right,other

   shared_ip_list: True,True,False,False

   static_routes_list: False,True,False,False
```

3. Create the Heat stack by launching the template and the environment file, using the following command:

```
heat stack create stack1 -f service_template.yaml -e service_template.env
```

OR use this command for recent versions of OpenStack

```
openstack stack create -e <env-file-name> -t <template-file-name> <stack-name>
```

RELATED DOCUMENTATION

Service Chain Version 2 with Port Tuple

Service Chain Route Reorigination

IN THIS SECTION

- Overview: Service Chaining in Contrail | 273
- Route Aggregation | 275
- Routing Policy | 282
- Control for Route Reorigination | 292

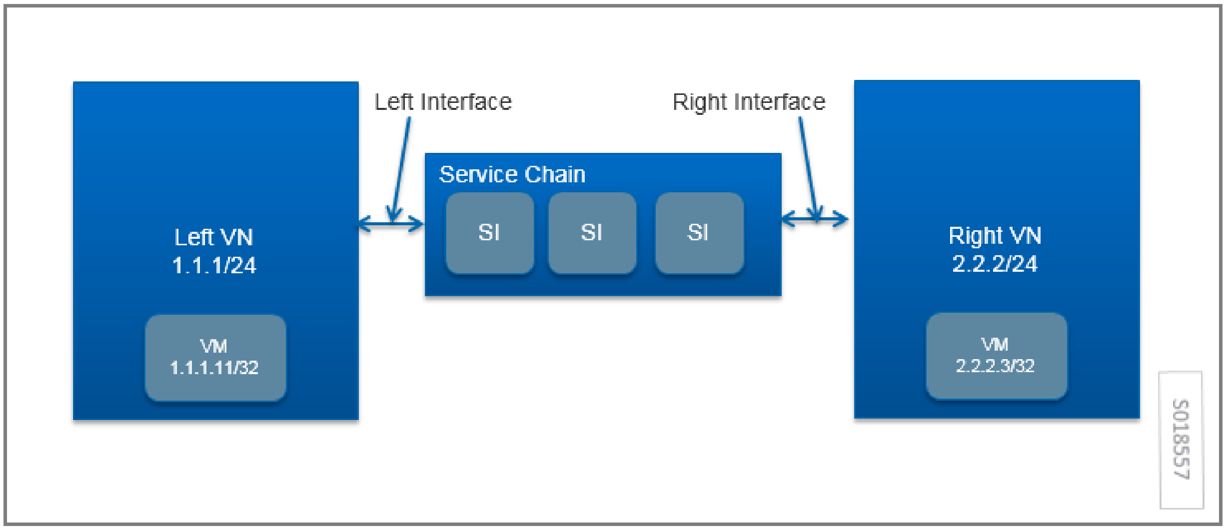
Overview: Service Chaining in Contrail

In Contrail, the service chaining feature allows the operator to insert dynamic services to control the traffic between two virtual networks. The service chaining works on a basic rule of next-hop stitching.

In Figure 93 on page 274, the service chain is inserted between the Left VN and the Right VN. The service chain contains one or more service instances to achieve a required network policy.

In the example, the route for the VM in the Right VN is added to the routing table for the Left VN, with the next hop modified to ensure that the traffic is sent by means of the left interface of the service chain. This is an example of route reorigination.

Figure 93: Route Reorigination



Using reorigination of routes for service chaining (for example, putting the route for the right network in the left routing table) requires the following features:

- **Route aggregation**

  For scaling purposes, it is useful to publish an aggregated route as the service chain route, rather than publishing every route of each VM (/32). This reduces the memory footprint for the route table in the gateway router and also reduces route exchanges between control nodes and the gateway router. The route can be aggregated to the default route (0/0), to the VN subnet prefix, or to any arbitrary route prefix.

- **Path attribute modification for reoriginated routes**

  There are cases where the `BgpPath` attribute for the service chain route needs to be modified. An example is the case of service chain failover, in which there are two service chains with identical services that are connected between the same two VNs. The operator needs to control which service chain is used for traffic between two networks, in addition to ensuring redundancy and high availability by providing failover support. Path attribute modification for reoriginated routes is implemented by means of routing policy, by providing an option to alter the MED (multi-exit discriminator) or `local-pref` of the reoriginated service chain route.

- **Control to enable and disable reorigination of the route**

  In some scenarios, the operator needs a control to stop reorigination of the route as the service chain route, for example, when static routes are configured on service VM interfaces. Control to enable or disable reorigination of the route is implemented by tagging the routes with the `no-reoriginate` community. Routes with the `no-reoriginate` community tag are skipped for route reorigination.

Route Aggregation

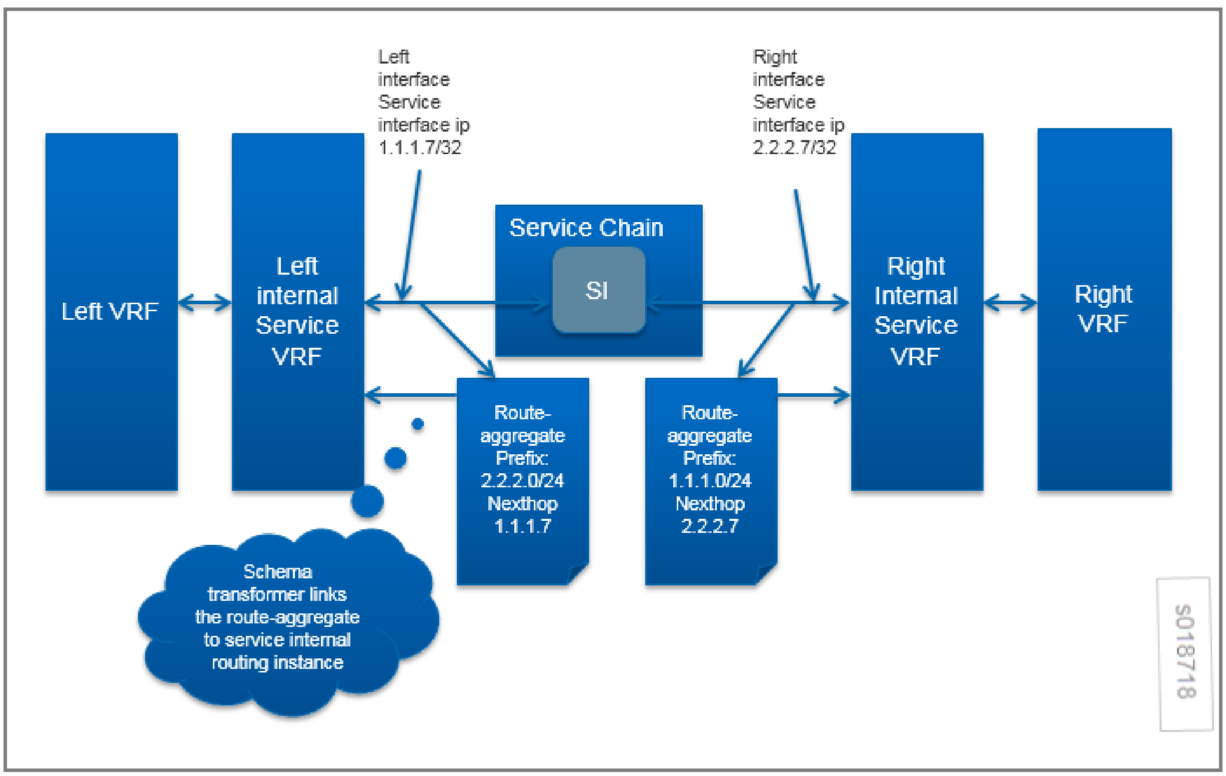
The route aggregation configuration object contains a list of prefixes to aggregate. The next-hop field in the route aggregate object contains the address of the route whose next hop is stitched as a next hop of the aggregate route.

Route aggregation is configured on the service instance. The operator can attach multiple route aggregation objects to a service instance. For example, if routes from the Right VN need to be aggregated and reoriginated in the route table of the Left VN, the route aggregate object is created with a prefix of the Right VN's subnet prefix and attached to the left interface of the service instance.

If the service chain has multiple service instances, the route aggregate object is attached to the left interface of the left-most service instance and to the right interface of the right-most service instance.

The relationships are shown in Figure 94 on page 275.

Figure 94: Route Aggregate Relationships



The schema transformer sets the next-hop field of the route aggregate object to the service chain interface address. The schema transformer also links the route aggregate object to the internal routing instance created for the service instance.

Using the configuration as described, the Contrail control service reads the route aggregation object on the routing instance. When the first, more specific route or contributing route is launched (when the first VM is launched on the right VN), the aggregate route is published. Similarly, the aggregated route is deleted when the last, more specific route or contributing route is deleted (when the last VM is deleted in the right VN). The aggregated route is published when the next hop for the aggregated route gets resolved.

By default, in BGP or XMPP route exchanges, the control node will not publish contributing routes of an aggregate route.

Schema for Route Aggregation

Route Aggregate Object

The following is the schema for route aggregate objects. Multiple prefixes can be specified in a single route aggregate object.

```
<xsd:element name="route-aggregate" type="ifmap:IdentityType"/>
<xsd:complexType name="RouteListType">
    <xsd:element name="route" type="xsd:string" maxOccurs="unbounded"/>
</xsd:complexType>

<xsd:element name='aggregate-route-entries' type='RouteListType'/>
<!--#IFMAP-SEMANTICS-IDL
    Property('aggregate-route-entries', 'route-aggregate') -->

<xsd:element name='aggregate-route-nexthop' type='xsd:string'/>
<!--#IFMAP-SEMANTICS-IDL
        Property('aggregate-route-nexthop', 'route-aggregate') -->
```

Service Instance Link to Route Aggregate Object

The following is the schema for the service instance link to route aggregation objects. The operator can link multiple route aggregate objects to a single service interface.

```
<xsd:element name="route-aggregate" type="ifmap:IdentityType"/>
<xsd:complexType name="RouteListType">
    <xsd:element name="route" type="xsd:string" maxOccurs="unbounded"/>
</xsd:complexType>
```

```
<xsd:element name='aggregate-route-entries' type='RouteListType'/>
<!--#IFMAP-SEMANTICS-IDL
    Property('aggregate-route-entries', 'route-aggregate') -->

<xsd:element name='aggregate-route-nexthop' type='xsd:string'/>
<!--#IFMAP-SEMANTICS-IDL
        Property('aggregate-route-nexthop', 'route-aggregate') -->

<xsd:simpleType name="ServiceInterfaceType">
        <xsd:restriction base="xsd:string">
        <xsd:pattern value="management|left|right|other[0-9]*"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name='ServiceInterfaceTag'>
        <xsd:element name="interface-type" type="ServiceInterfaceType"/>
</xsd:complexType>

<xsd:element name="route-aggregate-service-instance" type="ServiceInterfaceTag"/>
<!--#IFMAP-SEMANTICS-IDL
        Link('route-aggregate-service-instance',
                   'bgp:route-aggregate', 'service-instance', ['ref']) -->
```

Routing Instance Link to Route Aggregate Object

The following is the schema for the routing instance link to the route aggregation object. A routing instance can be linked to multiple route aggregate objects to perform route aggregation for multiple route prefixes.

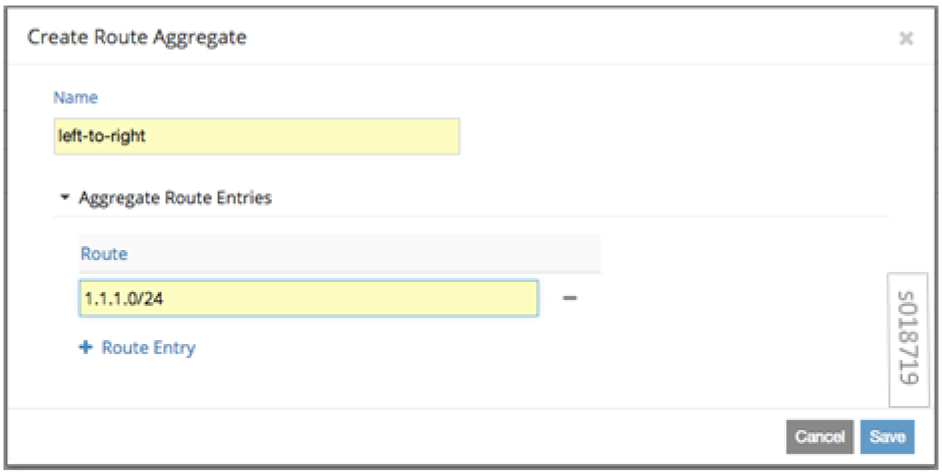
```
<xsd:element name="route-aggregate-routing-instance"/>
<!--#IFMAP-SEMANTICS-IDL
        Link('route-aggregate-routing-instance',
                  'route-aggregate', 'routing-instance', ['ref']) -->
```

Configuring and Troubleshooting Route Aggregation

Configure Route Aggregate Object

You can use the Contrail UI, **Configure > Networking > Routing > Create >Route Aggregate** screen to name the route aggregate object and identify the routes to aggregate. See Figure 95 on page 278.

Figure 95: Create Route Aggregate



Example VNC Script to Create a Route Aggregate Object

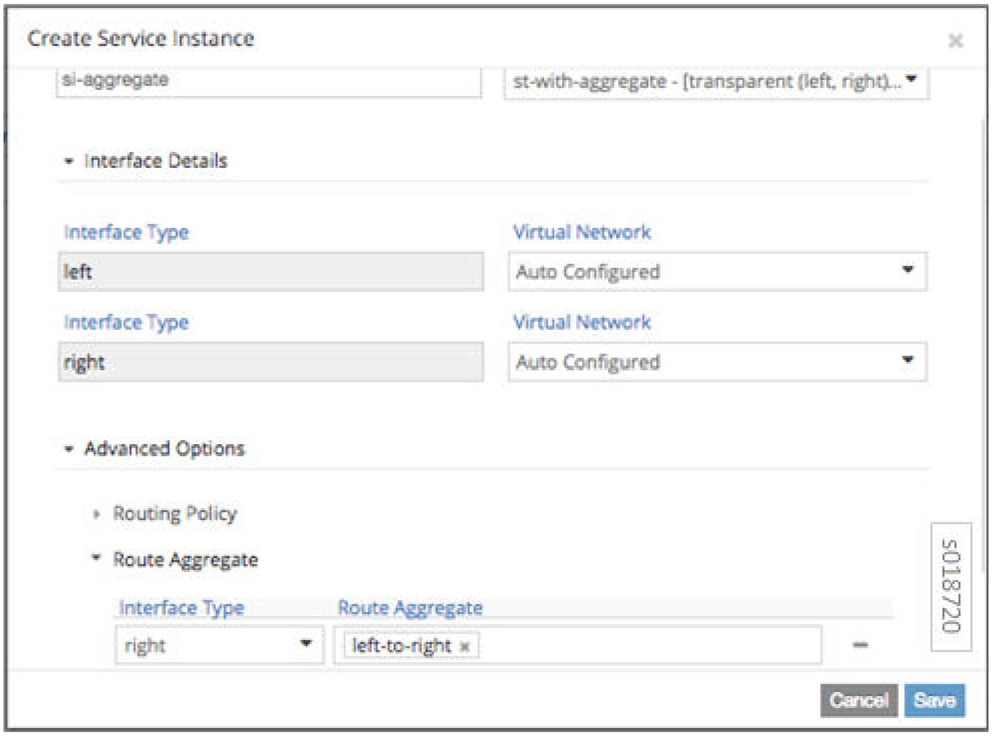
You can use a VNC script to create a route aggregate object, as in the following example:

```
from vnc_api.vnc_api import *
vnc_lib = VncApi("admin", "<password>.", "admin")
project=vnc_lib.project_read(fq_name=["default-domain", "admin"])
route_aggregate=RouteAggregate(name="left_to_right", parent_obj=project)
route_list=RouteListType(["<ip address>"])
route_aggregate.set_aggregate_route_entries(route_list)
vnc_lib.route_aggregate_create(route_aggregate)
```

Configuring a Service Instance

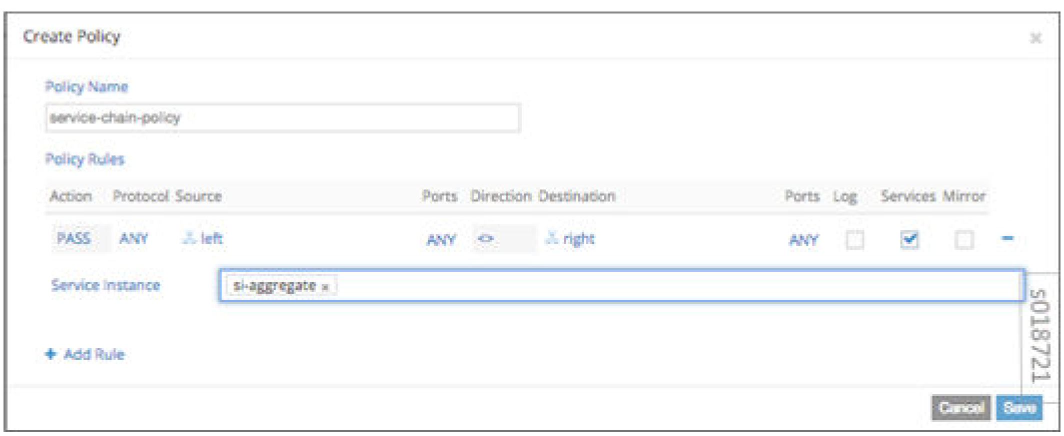
Create a service instance with the route aggregate object linked to the aggregate left network subnet prefix in the right virtual network. See the example in Figure 96 on page 279.

Figure 96: Create Service Instance

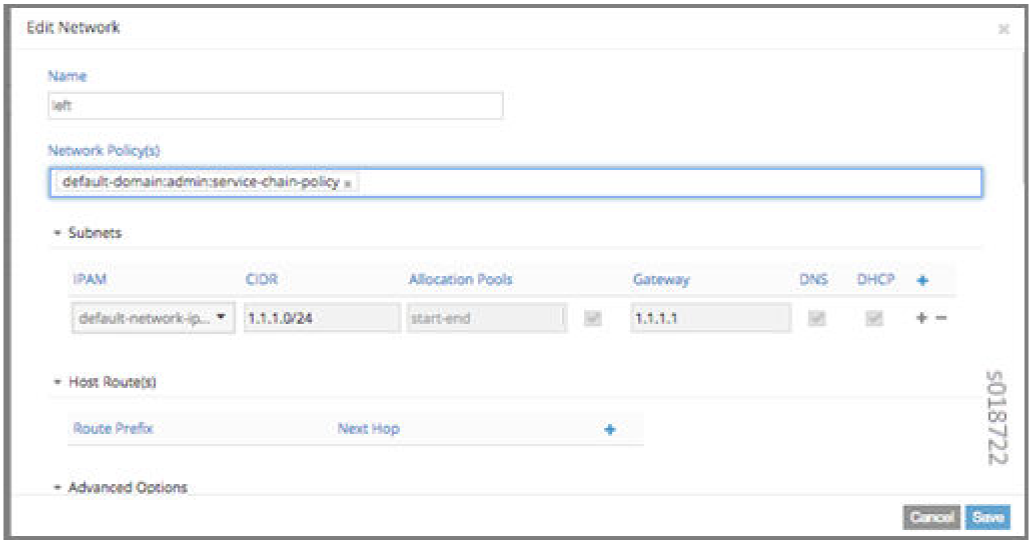


Create a Virtual Network and Network Policy

Create a left and right virtual network with the subnets 1.1.1/24 and 2.2.2/24, respectively. Create a network policy to apply a service chain between the left VN and the right VN. See the following example.



Attach the network policy to create the service chain between the left and right VNs. See the following example.



Validate the Route Aggregate Object in the API Server

Validate the route aggregate object in the API server configuration database. Verify the routing instance reference and the service instance reference for the aggregate object. The `aggregate_route_nexthop` field in the route aggregate object is initialized by the schema transformer to the service chain address. See the following example.

Validate the Route Aggregate Object in the Control Node

Validate the instance configurations of the route aggregate by checking the control node introspect for the service instance internal routing instance. For example:

`http://<control-node>:8083/Snh_ShowBgpInstanceConfigReq?search_string=default- domain:admin:right:service-ace7ae00-56e3-42d1-96ec-7fe77088d97f-default- domain_admin_si-aggregate`

See the following example.

To check the state of the route aggregate object on the control node, point your browser to:

`http://<control-node>:8083/Snh_ShowRouteAggregateReq`

See the following example.

You can also check the route table for the aggregate route in the right VN BGP able. For example:

`http://<control-node>:8083/Snh_ShowRouteReq?x=default-domain:admin:right:right.inet.0`

See the following example.

Routing Policy

Contrail uses routing policy infrastructure to manipulate the route and path attribute dynamically. Contrail also supports attaching the import routing policy on the service instances.

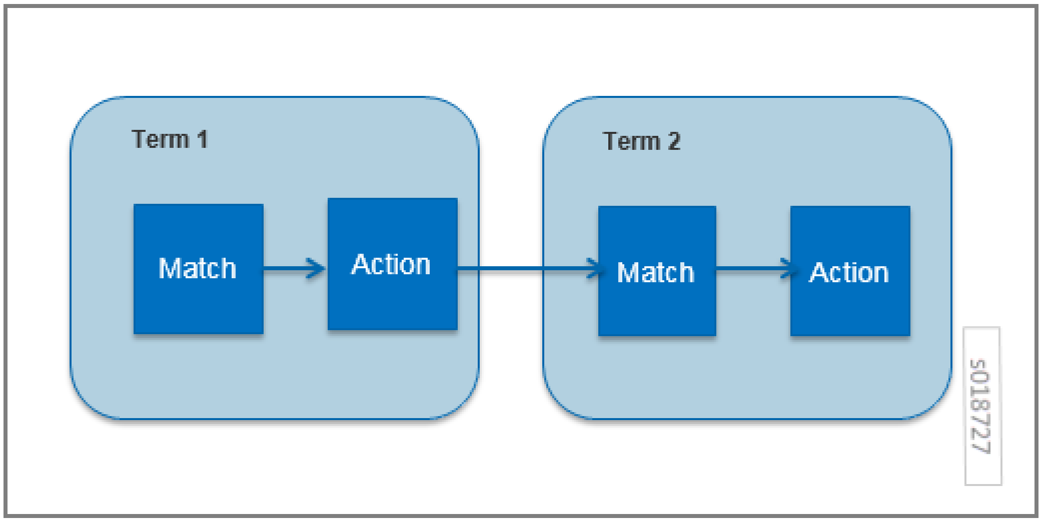
The routing policy contains list terms. A term can be a terminal rule, meaning that upon a match on the specified term, no further terms are evaluated and the route is dropped or accepted, based on the action in that term.

If the term is not a terminal rule, subsequent terms are evaluated for the given route.

The list terms are structured as in the following example.

```
Policy {
     Term-1
     Term-2
}
```

The matches and actions of the policy term lists operate similarly to the Junos language match and actions operations. A visual representation is the following.



Each term is represented as in the following:

```
from {
    match-condition-1
    match-condition-2
    ..
    ..
}
then {
    action
    update-action-1
    update-action-2
    ..
    ..
}
```

The term should not contain an `any` match condition, for example, an empty `from` should not be present.

If an `any` match condition is present, all routes are considered as matching the term.

However, the `then` condition can be empty or the action can be unspecified.

Applying Routing Policy

The routing policy evaluation has the following key points:

- If the term of a routing policy consists of multiple match conditions, a route must satisfy all match conditions to apply the action specified in the term.
- If a term in the policy does not specify a match condition, all routes are evaluated against the match.
- If a match occurs but the policy does not specify an accept, reject, or next term action, one of the following occurs:
  - The next term, if present, is evaluated.
  - If no other terms are present, the next policy is evaluated.
  - If no other policies are present, the route is accepted. The default routing policy action is “accept”.
- If a match does not occur with a term in a policy, and subsequent terms in the same policy exist, the next term is evaluated.
- If a match does not occur with any terms in a policy, and subsequent policies exist, the next policy is evaluated.
- If a match does not occur by the end of a policy or all policies, the route is accepted.

A routing policy can consist of multiple terms. Each term consists of match conditions and actions to apply to matching routes.

Each route is evaluated against the policy as follows:

1. The route is evaluated against the first term. If it matches, the specified action is taken. If the action is to accept or reject the route, that action is taken and the evaluation of the route ends. If the next term action is specified or if no action is specified, or if the route does not match, the evaluation continues as described above to subsequent terms.
2. Upon hitting the last non-terminal term of the given routing policy, the route is evaluated against the next policy, if present, in the same manner as described in step 1.

Match Condition: From

The match condition `from` contains a list of match conditions to be satisfied for applying the action specified in the term. It is possible that the term doesn’t have any match condition. This indicates that all routes match this term and action is applied according to the action specified in the term.

The following table describes the match conditions supported by Contrail.

Match Condition	User Input	Description
Prefix	List of prefixes to match	Each prefix in the list is represented as prefix and match type, where the prefix match type can be: • `exact` • `orlonger` • `longer` Example: 1.1.0.0/16 `orlonger` A route matches this condition if its prefix matches any of the prefixes in the list.
Community	Community string to match	Represented as either a well-known community string with `no export` or `no reoriginate`, or a string representation of a community (64512:11).
Protocol	Array of path source or path protocol to match	BGP | XMPP | StaticRoute | ServiceChain | Aggregate. A path is considered as matching this condition if the path protocol is one of protocols in the list.

Routing Policy Action and Update Action

The policy action contains two parts, action and update action.

The following table describes `action` as supported by Contrail.

Action	Terminal?	Description
Reject	Yes	Reject the route that matches this term. No more terms are evaluated after hitting this term.
Accept	Yes	Accept the route that matches this term. No more terms are evaluated after hitting this term. The route is updated using the update specified in the policy action.

(Continued)

Action	Terminal?	Description
Next Term	No	This is the default action taken upon matching the policy term. The route is updated according to the update specified in the policy action. Next terms present in the routing policy are processed on the route. If there are no more terms in the policy, the next routing policy is processed, if present.

The update action section specifies the route modification to be performed on the matching route.

The following table describes `update action` as supported by Contrail.

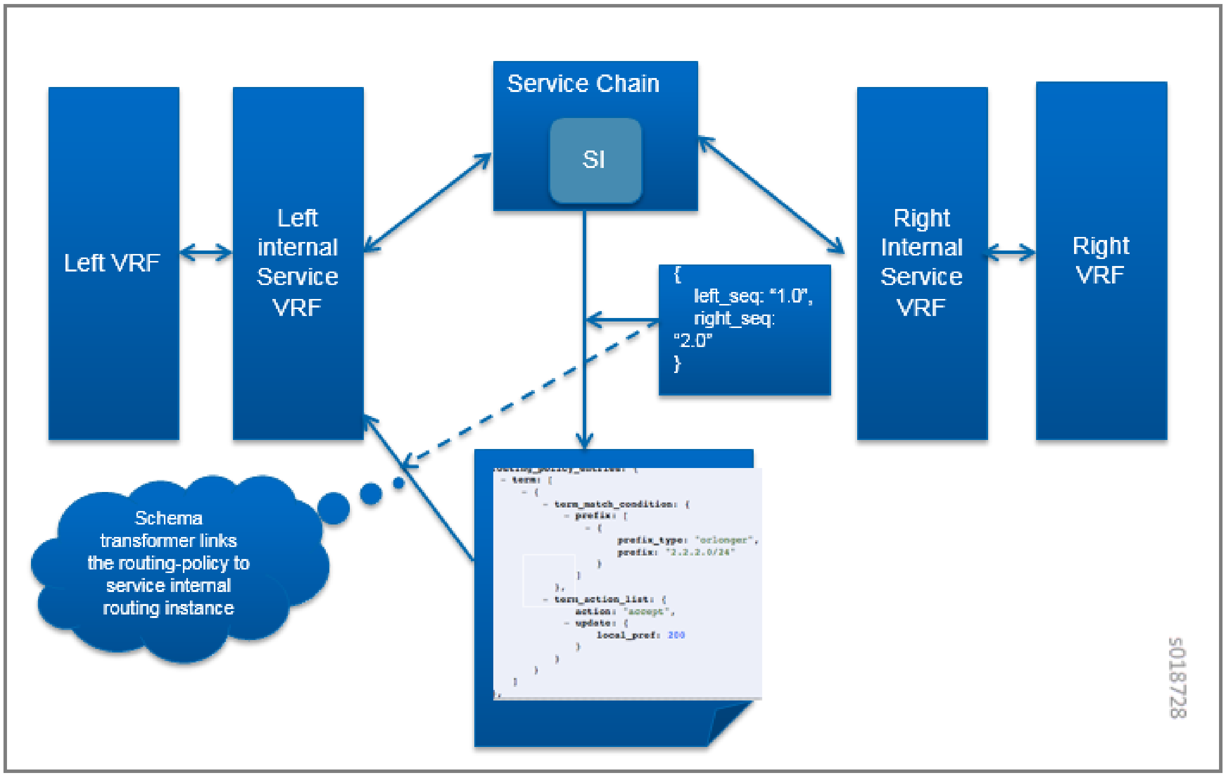
Update Action	User Input	Description
community	List of community	As part of the policy update, the following actions can be taken for community: • Add a list of community to the existing community. • Set a list of community. • Remove a list of community (if present) from the existing community.
MED	Update the MED of the BgpPath	Unsigned integer representing the MED
local-pref	Update the local-pref of the BgpPath	Unsigned integer representing local-pref

Routing Policy Configuration

Routing policy is configured on the service instance. Multiple routing policies can be attached to a single service instance interface.

When the policy is applied on the left interface, the policy is evaluated for all the routes that are reoriginated in the left VN for routes belonging to the right VN. Similarly, the routing policy attached to the right interface influences the route reorigination in the right VN, for routes belonging to the left VN.

The following figure illustrates a routing policy configuration.



The policy sequence number specified in the routing policy link data determines the order in which the routing policy is evaluated. The routing policy link data on the service instance also specifies whether the policy needs to be applied to the left service interface, to the right service interface, or to both interfaces.

It is possible to attach the same routing policy to both the left and right interfaces for a service instance, in a different order of policy evaluation. Consequently, the routing policy link data contains the sequence number for policy evaluation separately for the left and right interfaces.

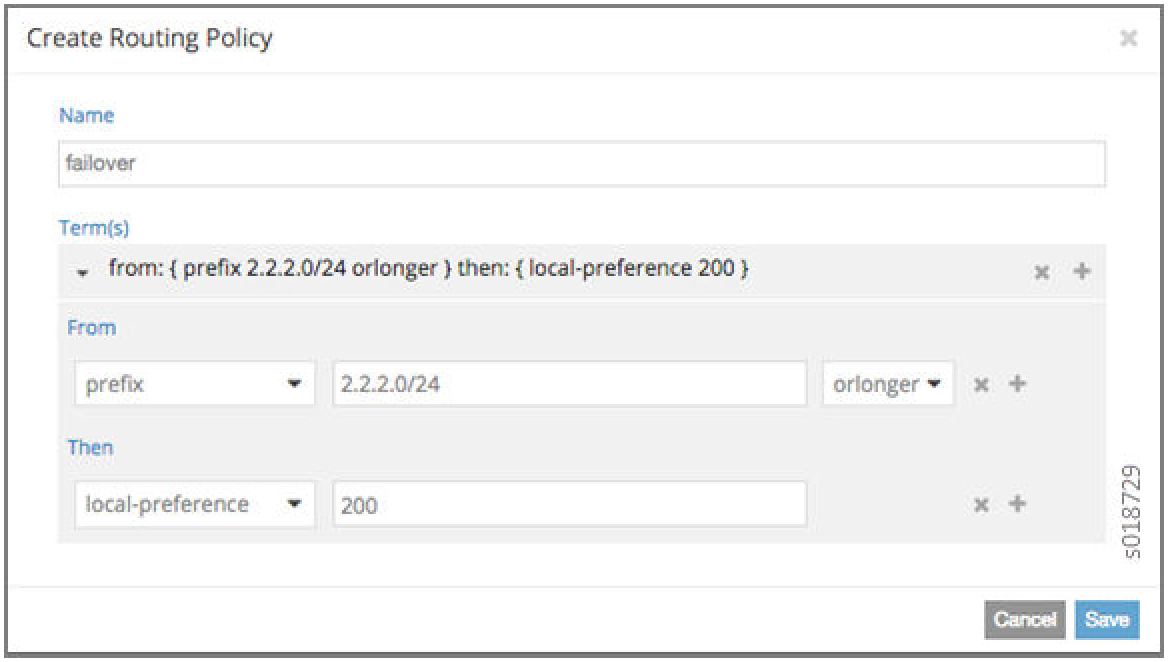
The schema transformer links the routing policy object to the internal routing instance created for the service instance. The transformer also copies the routing policy link data to ensure the same policy order.

Configuring and Troubleshooting Routing Policy

This section shows how to create a routing policy for service chains and how to validate the policy.

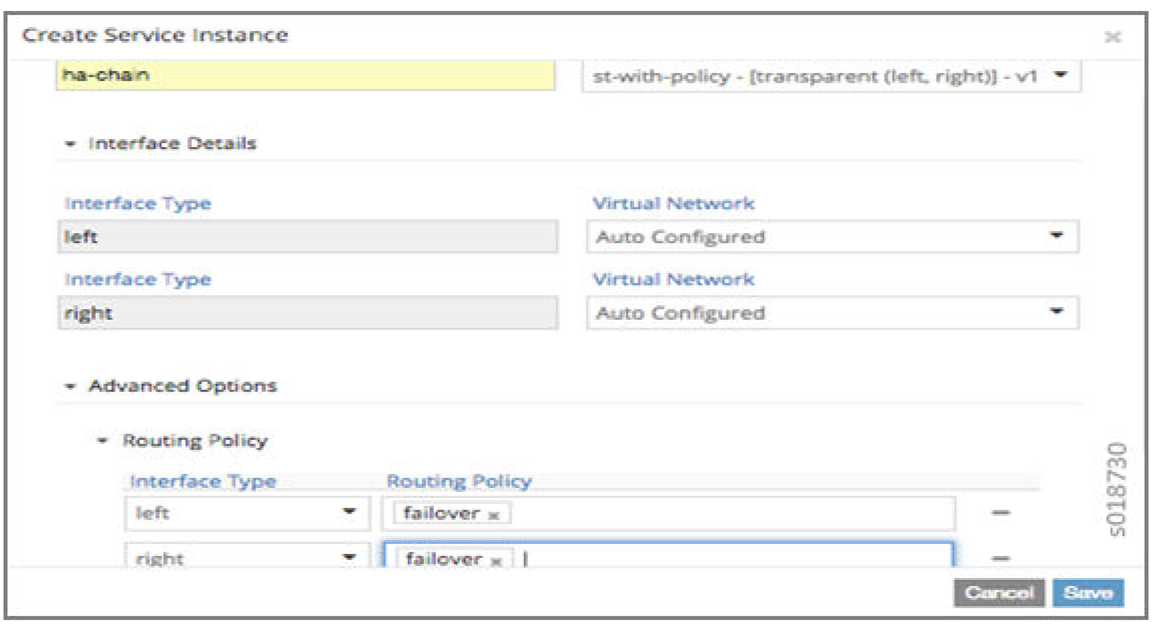
Create Routing Policy

First, create the routing policy, **Configure > Networking > Routing > Create >Routing Policy**. See the following example.



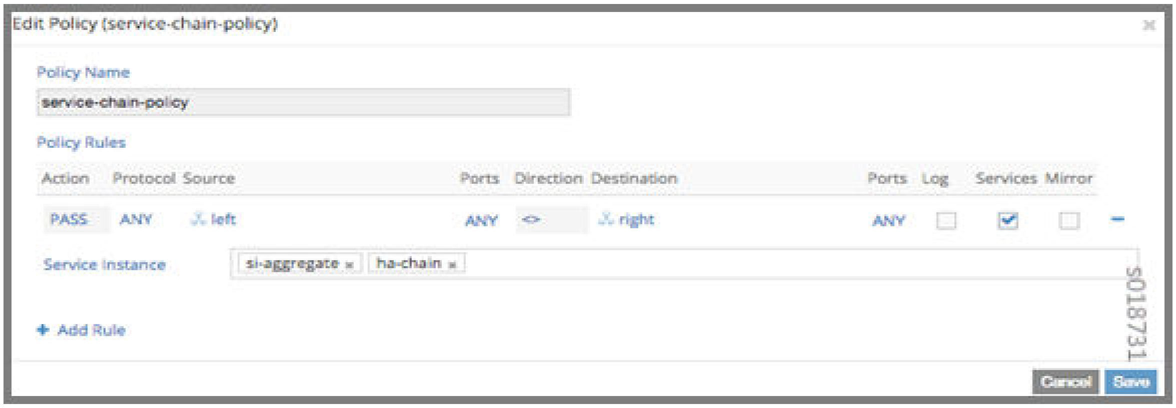
Configure Service Instance

Create a service instance and attach the routing policy to both the left and right interfaces. The order of the policy is calculated by the UI, based on the order of the policy specified in the list.



Configure the Network Policy for the Service Chain

At **Edit Policy**, create a policy for the service chain, see the following example.



Using a VNC Script to Create Routing Policy

The following example shows use of a VNC API script to create a routing policy.

```
from vnc_api.vnc_api import *
vnc_lib = VncApi("admin", "<password>", "admin")
project=vnc_lib.project_read(fq_name=["default-domain", "admin"])
routing_policy=RoutingPolicy(name="vnc_3", parent_obj=project)
policy_term=PolicyTermType()
policy_statement=PolicyStatementType()

match_condition=TermMatchConditionType(protocol=["bgp"], community="22:33")
prefix_match=PrefixMatchType(prefix="1.1.1.0/24", prefix_type="orlonger")
match_condition.set_prefix([prefix_match])

term_action=TermActionListType(action="accept")
action_update=ActionUpdateType(local_pref=101, med=10)
add_community=ActionCommunityType()
comm_list=CommunityListType(["11:22"])
add_community.set_add(comm_list)
action_update.set_community(add_community)
term_action.set_update(action_update)

policy_term.set_term_action_list(term_action)
policy_term.set_term_match_condition(match_condition)

policy_statement.add_term(policy_term)
```

```
routing_policy.set_routing_policy_entries(policy_statement)
vnc_lib.routing_policy_create(routing_policy)
```

Verify Routing Policy in API Server

You can verify the service instance references and the routing instance references for the routing policy by looking in the API server configuration database. See the following example.

- routing_policy_entries: {
  - term: [
    - {
      - term_match_condition: {
        - prefix: [
          - {
                prefix_type: "orlonger",
                prefix: "2.2.2.0/24"
            }
          ]
        },
      - term_action_list: {
            action: "accept",
          - update: {
                local_pref: 200
            }
        }
      }
    ]
  },
+ id_perms: {…},
- routing_instance_refs: [
  - {
      - to: [
            "default-domain",
            "admin",
            "right",
            "service-ace7ae00-56e3-42d1-96ec-7fe77088d97f-default-domain_admin_ha-chain"
        ],
        href: "http://nodea27.englab.juniper.net:8082/routing-instance/32b7eed4-57ce-4c44-bbb0-513f78db6068",
      - attr: {
            sequence: "1"
        },
        uuid: "32b7eed4-57ce-4c44-bbb0-513f78db6068"
    },
  - {
      - to: [
            "default-domain",
            "admin",
            "left",
            "service-ace7ae00-56e3-42d1-96ec-7fe77088d97f-default-domain_admin_ha-chain"
        ],
        href: "http://nodea27.englab.juniper.net:8082/routing-instance/6ad868d1-a412-4765-b8c4-f93ec5d9f4b2",
      - attr: {
            sequence: "1"
        },
        uuid: "6ad868d1-a412-4765-b8c4-f93ec5d9f4b2"
    }
  ],
- service_instance_refs: [
  - {
      - to: [
            "default-domain",
            "admin",
            "ha-chain"
        ],
        href: "http://nodea27.englab.juniper.net:8082/service-instance/983bb90b-b3f4-4d6c-be54-33a474eee7de",
      - attr: {
            left_sequence: "1",
            right_sequence: "1"
        },
        uuid: "983bb90b-b3f4-4d6c-be54-33a474eee7de"
    }
  ],
  name: "failover"

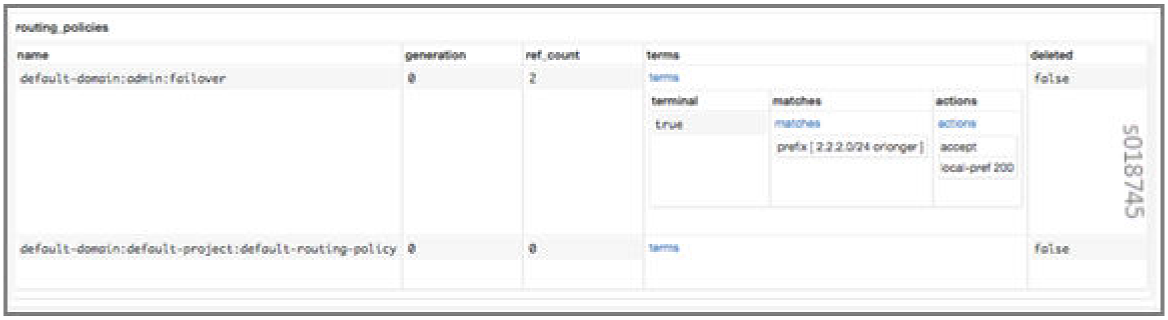
Verify Routing Policy in the Control Node

You can verify the routing policy in the control node.

Point your browser to:

`http://<control-node>:8083/Snh_ShowRoutingPolicyReq?search_string=failover`

See the following example.



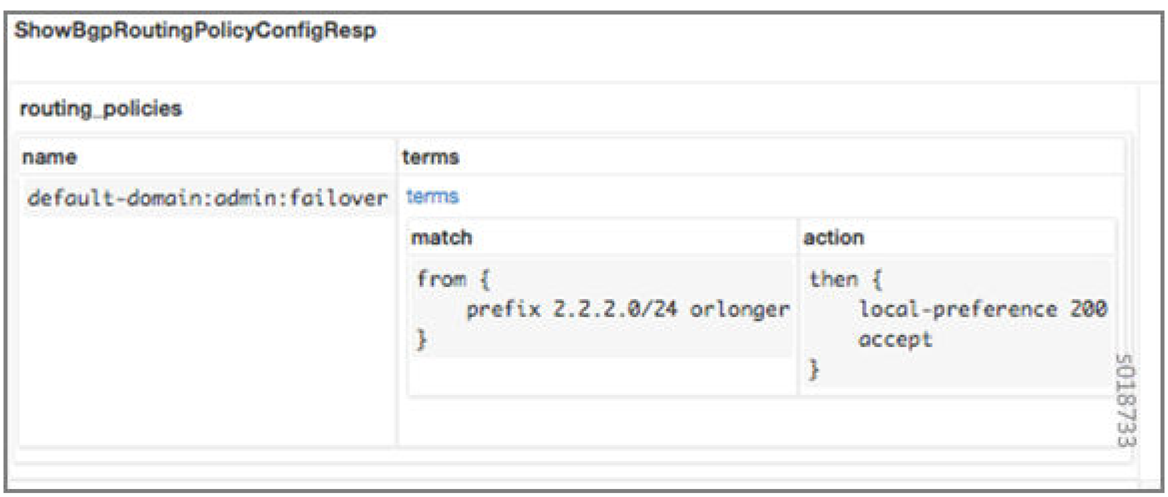
Verify Routing Policy Configuration in the Control Node

You can verify the routing policy configuration in the control node.

Point your browser to:

`http://<control-node>:8083/Snh_ShowBgpRoutingPolicyConfigReq?search_string=failover`

See the following example.



Verify Routing Policy Configuration on the Routing Instance

You can verify the routing policy configuration on the internal routing instance.

Point your browser to:

```
http://<control-node>:8083/Snh_ShowBgpInstanceConfigReq?search_string=<name-of-internal-vrf>
```

See the following example.

You can also verify the routing policy on the routing instance operational object.

Point your browser to:

```
http://<control-node>:8083/Snh_ShowRoutingInstanceReq?x=<name-of-internal-vrf>
```

See the following example.

Control for Route Reorigination

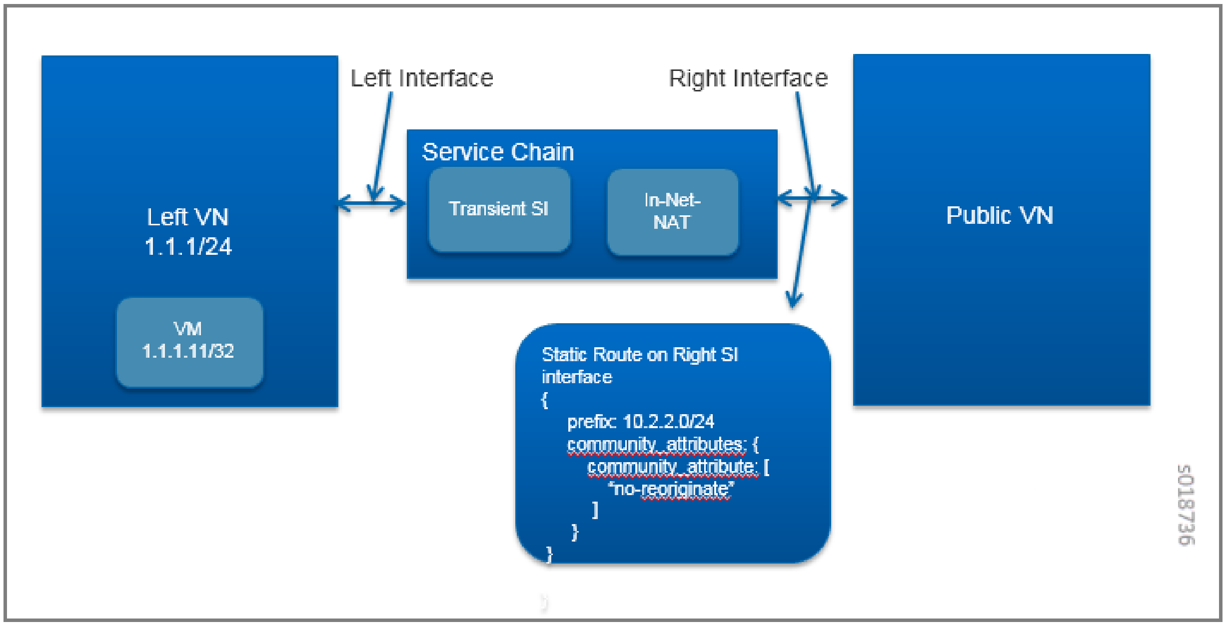
The ability to prevent reorigination of interface static routes is typically required when routes are configured on an interface that belongs to a service VM.

As an example, the following image shows a service chain that has multiple service instances, with an `in-net-nat` service instance as the last service VM, also with the right VN as the public VN.

The last service instance performs NAT by using a NAT pool. The right interface of the service VM must be configured with an interface static route for the NAT pool so that the destination in the right VN knows how to reach addresses in the NAT pool. However, the NAT pool prefix should not be reoriginated into the left VN.

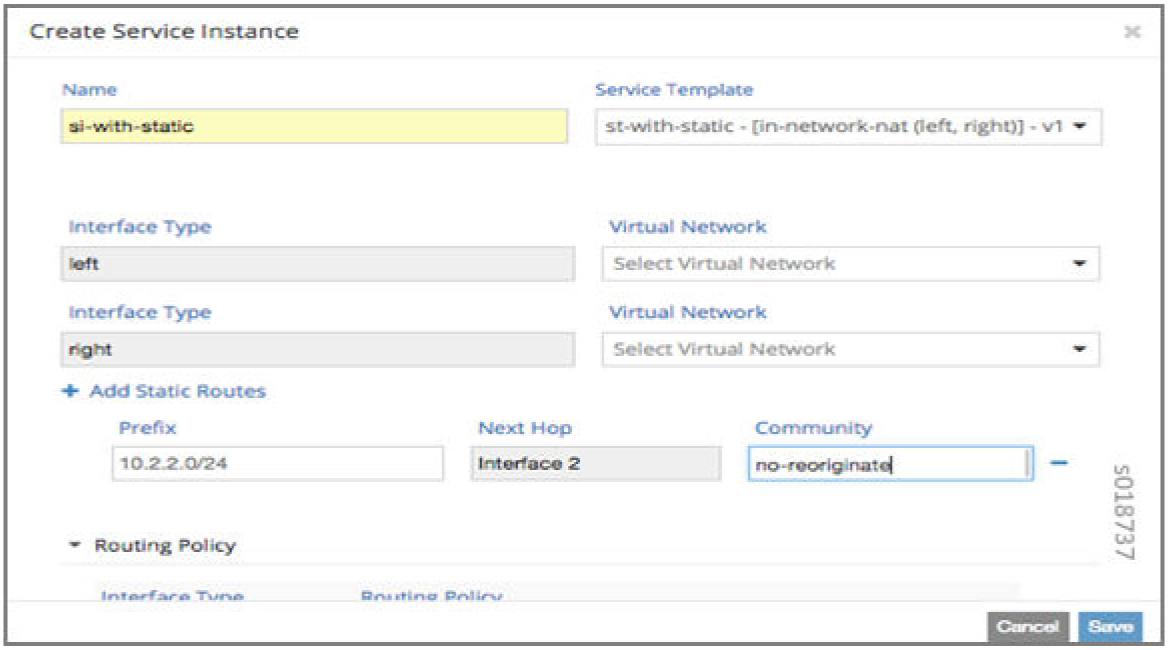
To prevent route reorigination, the interface static route is tagged with a well-known BGP community called `no-reoriginate`.

When the control node is reoriginating the route, it skips the routes that are tagged with the BGP community.



Configuring and Troubleshooting Reorigination Control

The community attribute on the static routes for the interface static route of the service instance is specified during creation of the service instance. See the following example.



Use the following example to verify that the service instance configuration object in the API server has the correct community set for the static route. See the following example.

Service Instance Health Checks

IN THIS SECTION

- Health Check Object | 295
- Bidirectional Forwarding and Detection Health Check over Virtual Machine Interfaces | 300
- Bidirectional Forwarding and Detection Health Check for BGPaaS | 300
- Health Check of Transparent Service Chain | 301
- Service Instance Fate Sharing | 301

In Contrail Release 3.0 and greater, a service instance health check can be used to determine the liveliness of a service provided by a virtual machine (VM).

Health Check Object

IN THIS SECTION

- Health Check Overview | 295
- Health Check Object Configuration | 296
- Creating a Health Check with the Contrail User Interface | 297
- Using the Health Check | 299
- Health Check Process | 299

Health Check Overview

The service instance health check is used to determine the liveliness of a service provided by a VM, checking whether the service is operationally up or down. The vRouter agent uses ping and an HTTP URL to the link-local address to check the liveliness of the interface.

If the health check determines that a service is no longer operational, it removes the routes for the VM, thereby disabling packet forwarding to the VM.

The service instance health check is used with service template version 2.

Health Check Object Configuration

Table 38 on page 296 shows the configurable properties of the health check object.

Table 38: Health Check Configurable Parameters

Field	Description
`- enabled`	Indicates that health check is enabled. The default is `False`.
`- health-check-type`	Indicates the health check type: `link-local`, `end-to-end`, `bgp-as-a-service`, and so on.. The default is `link-local`.
`- monitor-type`	The protocol type to be used: `PING` or `HTTP`.
`- delay`	The delay, in seconds, to repeat the health check.
`- timeout`	The number of seconds to wait for a response.
`- max-retries`	The number of retries to attempt before declaring an instance health down.
`- http-method`	When the monitor protocol is HTTP, the type of HTTP method used, such as GET, PUT, POST, and so on.
`- url-path`	When the monitor protocol is HTTP, the URL to be used. For all other cases, such as ICMP, the destination IP address.
`- expected-codes`	When the monitor protocol is HTTP, the expected return code for HTTP operations.

Health Check Modes

The following modes are supported for the service instance health check:

- `link-local`—A local check for the service VM on the vRouter where the VM is running. In this case, the source IP of the packet is the service chain IP.

- `end-to-end`—A remote address or URL is provided for a service health check through a chain of services. The destination of the health check probe is allowed to be outside the service instance. However, the health check probe must be reachable through the interface of the service instance where the health check is attached. The end-to-end health check probe is transmitted all the way to the actual destination outside the service instance. The response to the health check probe is received and processed by the service health check to evaluate the status.

  Restrictions include:

  - This check is applicable for a chain where the services are not scaled out.
  - When this mode is configured, a new health check IP is allocated and used as the source IP of the packet.
  - The health check IP is allocated per `virtual-machine-interface` of the service VM where the health check is attached.
  - The agent relies on the `service-health-check-ip` flag to use as the source IP.

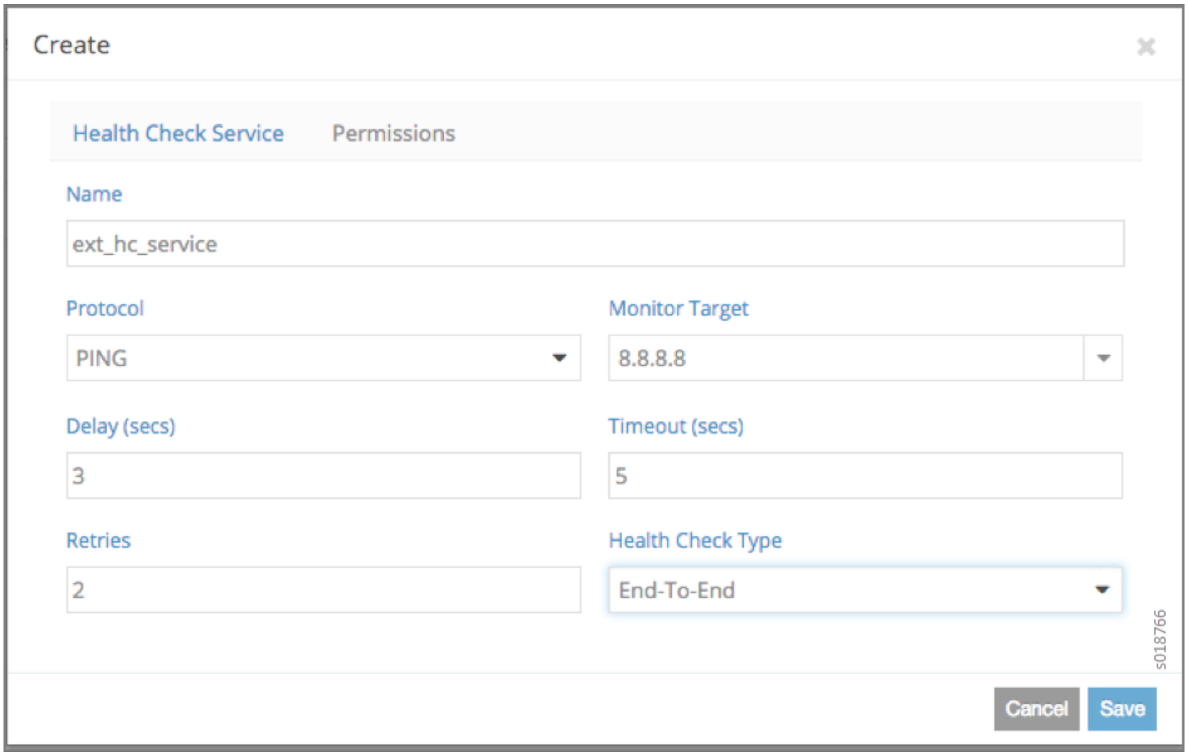
  > **NOTE**: In versions prior to Contrail 4.1, end-to-end health check is not supported on a transparent service chain. However, a link-local health check is possible on a transparent service instance if the corresponding service instance interface is configured with its IP address. Contrail 4.1 supports a segment-based health check for transparent service chain.

Creating a Health Check with the Contrail User Interface

To create a health check with the Contrail Web UI:

1. Navigate to **Configure** > **Services** > **Health Check Service**, and click to open the **Create** screen. See Figure 97 on page 298.

Figure 97: Create Health Check Screen



2. Complete the fields to define the permissions for the health check, see Table 39 on page 298.

Table 39: Create Health Check Fields

Field	Description
Name	Enter a name for the health check service you are creating.
Protocol	Select from the list the protocol to use for the health check, PING, HTTP, BFD, and so on.
Monitor Target	Select from the list the address of the target to be monitored by the health check.
Delay (secs)	The delay, in seconds, to repeat the health check.
Timeout (secs)	The number of seconds to wait for a response.

Table 39: Create Health Check Fields *(Continued)*

Field	Description
Retries	The number of retries to attempt before declaring an instance health down.
Health Check Type	Select from the list the type of health check—link-local, end-to-end, segment-based, bgp-as-a-service, and so on.

Using the Health Check

A REST API can be used to create a health check object and define its associated properties, then a link is added to the VM interface.

The health check object can be linked to multiple VM interfaces. Additionally, a VM interface can be associated with multiple health check objects. The following is an example:

```
HealthCheckObject 1 ---------------- VirtualMachineInterface 1 ----------------
HealthCheckObject 2
          |
          |
VirtualMachineInterface 2
```

Health Check Process

The Contrail vRouter agent is responsible for providing the health check service. The agent spawns a Python script to monitor the status of a service hosted on a VM on the same compute node, and the script updates the status to the vRouter agent.

The vRouter agent acts on the status provided by the script to withdraw or restore the exported interface routes. It is also responsible for providing a link-local metadata IP for allowing the script to communicate with the destination IP from the underlay network, using appropriate NAT translations. In a running system, this information is displayed in the vRouter agent introspect at:

`http://<compute-node-ip>:8085/Snh_HealthCheckSandeshReq?uuid=`

NOTE: Running health check creates flow entries to perform translation from underlay to overlay. Consequently, in a heavily loaded environment with a full flow table, it is possible to observe false failures.

Bidirectional Forwarding and Detection Health Check over Virtual Machine Interfaces

Contrail Networking Release 4.1 and later support for BFD-based health checks for VMIs.

Health check for VMIs is already supported as poll-based checks with ping and curl commands. When enabled, these health checks run periodically, once every few seconds. Consequently, failure detection times can be quite large, always in seconds.

Health checks based on the BFD protocol provide failure detection and recovery in sub-second intervals, because applications are notified immediately upon BFD session state changes.

If BFD-based health check is configured, whenever a BFD session status is detected as `Up` or `Down` by the health-checker, corresponding logs are generated.

Logging is enabled in the contrail-vrouter-agent.conf file with the log severity level `SYS_NOTICE`.

You can view the log file in the location **/var/log/contrail/contrail-vrouter-agent.log**

Snippet of sample log message related to BFD session events

```
2019-02-26 Tue 14:38:49:417.479 SYS_NOTICE BFD session Down  interface: test-bfd-hc-vmi.st2
vrf: default-domain:admin:VN.hc.st2:VN.hc.st2
2019-02-26 Tue 14:38:49:479.733 PST SYS_NOTICE BFD session Up  interface: test-bfd-hc-vmi.st2
vrf: default-domain:admin:VN.hc.st2:VN.hc.st2
```

Bidirectional Forwarding and Detection Health Check for BGPaaS

Contrail Release 4.1 adds support for BFD-based health check for BGP as a Service (BGPaaS) sessions.

This health check should not be confused with the BFD-based health check over VMIs feature, also introduced in Release 4.1. The BFD-based health check for VMIs cannot be used for a BGPaaS session, because the session shares a tenant destination address over a set of VMIs, with only one VMI active at any given time.

When the BFD-based health check for BGP as a Service (BGPaaS) is configured, any time a BFD-for-BGP session is detected as down by the health-checker, corresponding logs and alarms are generated.

To enable this health check, configure the `ServiceHealthCheckType` property and associate it with a bgp-as-a-service configuration object. This can also be accomplished in the Contrail WebUI.

Health Check of Transparent Service Chain

Contrail 4.1 enhances service chain redundancy by implementing an end-to-end health check for the transparent service chain. The service health check monitors the status of the service chain and if there is a failure, the control node no longer considers the service chain as a valid next hop, triggering traffic failover.

A segment-based health check is used to verify the health of a single instance in a transparent service chain. The user creates a service-health-check object, with type segment-based, and attaches it to either the left or right interface of the service instance. The service health check packet is injected to the interface to which it is attached. When the packet comes out of the other interface, a reply packet is injected on that interface. If health check requests fail after 30-second retries, the service instance is considered unhealthy and the service VLAN routes of the left and right interfaces are removed. When the agent receives health check replies successfully, it adds the retracted routes back onto both interfaces, which triggers the control node to start reoriginating routes to other service instances on that service chain.

For more information, see https://github.com/Juniper/contrail-specs/blob/master/transparent_sc_health_check.md

Service Instance Fate Sharing

A service chain contains multiple service instances (SI) and the failure of a single SI can cause a traffic black hole. In Contrail Release 4.1 and earlier, when an SI fails, the service chain continues to forward packets and routes reoriginate on both sides of the service chain. The packets are dropped in the SI or by the vRouter causing a black hole.

Starting in Contrail Release 4.1, **segment-based** health check type is used to verify the health of a SI in a service chain. To identify a failure of an SI, segment-based health check is configured either on the egress or ingress interface of the SI. When SI health check fails, the vRouter agent drops an SI route or a connected route. A connected route is also dropped if the vRouter agent restarts due to a software failure, when a compute node reboots, or when long-lived graceful restart (LLGR) is not enabled. You can detect an SI failure by keeping track of corresponding connected routes of the service chain address.

> **NOTE**: When an SI is scaled out, the connected route for an SI interface goes down only when all associated VMs have failed.

The control node uses the `service-chain-id` in `ServiceChainInfo` to link all SIs in a service chain. When the control node detects that any SI of the same service-chain-id is down, it stops reoriginating routes in egress and ingress directions for all SIs. The control node reoriginates routes only when the connected routes of all the SIs are up.

CHAPTER 12

Examples: Configuring Service Chaining

IN THIS CHAPTER

- Example: Creating an In-Network or In-Network-NAT Service Chain | **303**
- Example: Creating a Transparent Service Chain | **313**
- Example: Creating a Service Chain With the CLI | **319**

Example: Creating an In-Network or In-Network-NAT Service Chain

IN THIS SECTION

- Creating an In-Network or In-Network-NAT Service Chain | **303**

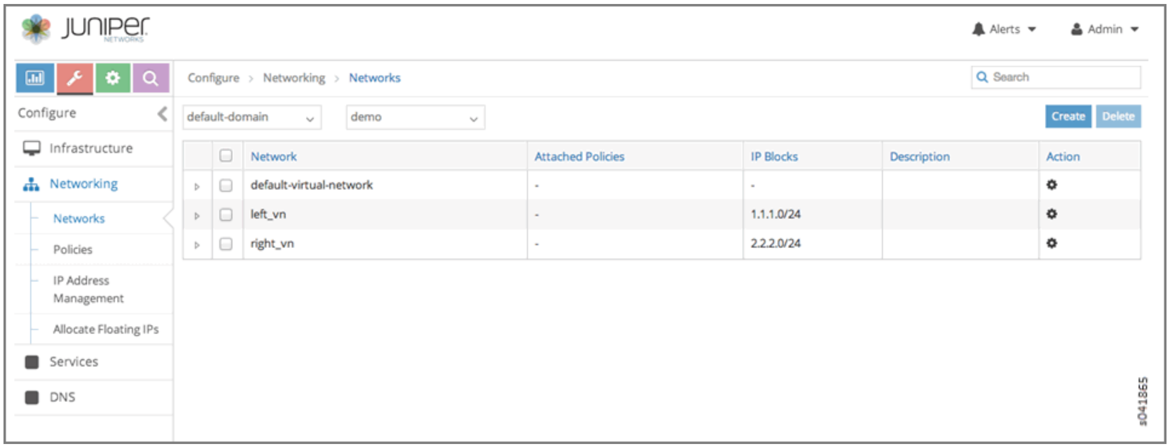
This section provides an example of creating an `in-network` service chain and an `in-network-nat` service chain using the Juniper Networks Contrail user interface. This service chain example also shows scaling of service instances.

Creating an In-Network or In-Network-NAT Service Chain

To create an **in-network** or **in-network-nat** service chain:

1. Create a left and a right virtual network. Select **Configure** > **Networking** > **Networks** and create **left_vn** and **right_vn**; see Figure 98 on page 304.

Figure 98: Create Networks



2. Configure a service template for an in-network service template for NAT. Navigate to **Configure > Services > Service Templates** and click the **Create** button on **Service Templates**. The **Add Service Template** window appears; see Figure 99 on page 305.

Figure 99: Add Service Template

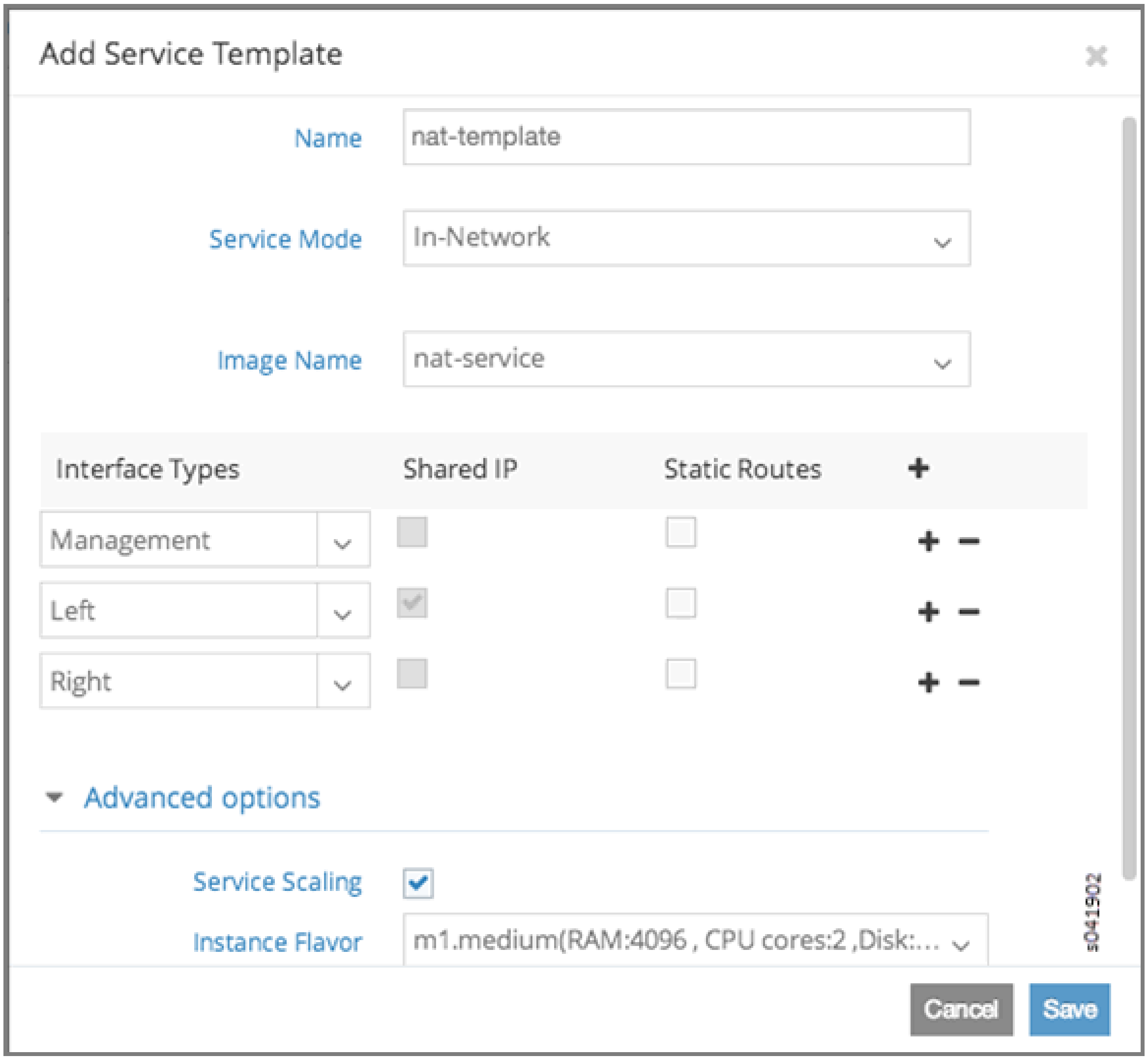


Table 40: Add Service Template Fields

Field	Description
Name	Enter a name for the service template.
Service Mode	Select the service mode: **In-Network** (for firewall service), **In-Network-NAT** (for NAT service), or **Transparent**.

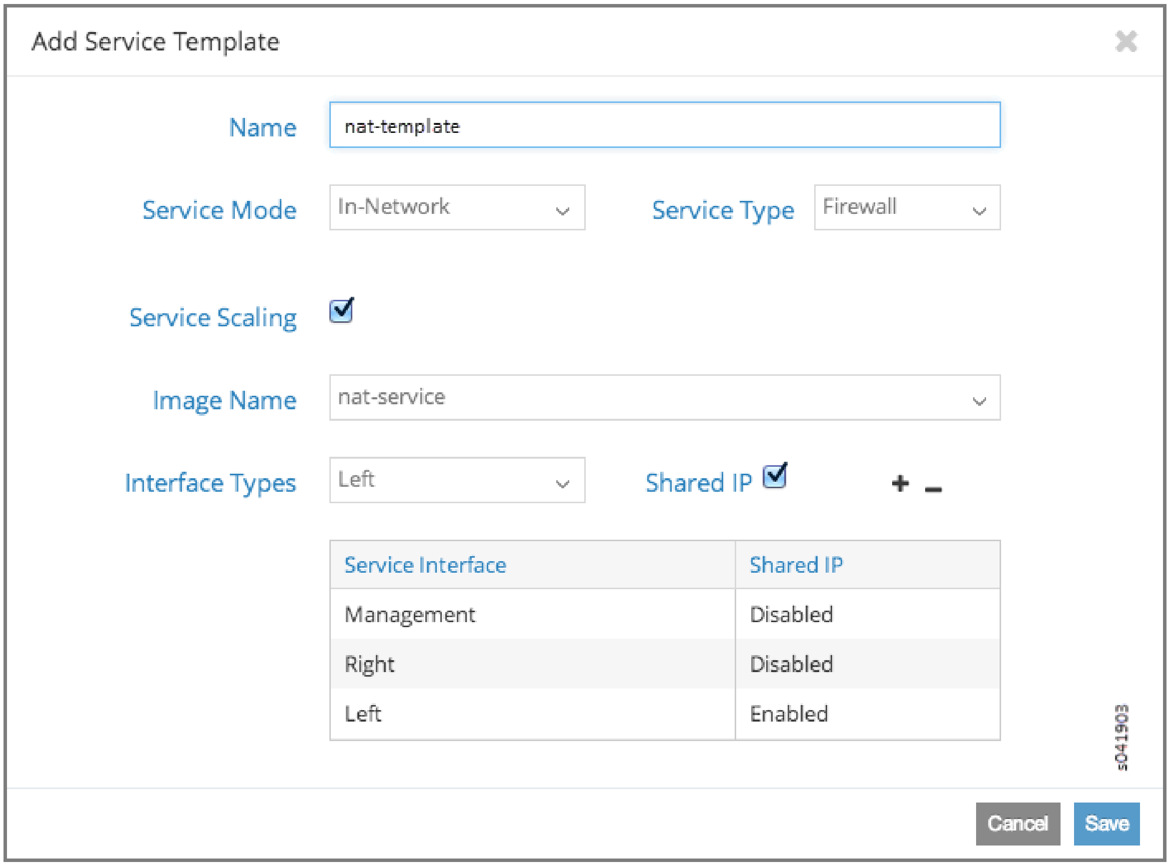
Table 40: Add Service Template Fields *(Continued)*

Field	Description
Service Scaling	If you will be using multiple virtual machines for a single service instance to scale out the service, select the **Service Scaling** check box. When scaling is selected, you can choose to use the same IP address for a particular interface on each virtual machine interface or to allocate new addresses for each virtual machine. For a NAT service, the left (inner) interface should have the same IP address, and the right (outer) interface should have a different IP address.
Image Name	Select from a list of available images the image for the service. **NOTE**: Only images that have been tagged as public in Glance will appear in the drop-down list.
Interface Types	Select the interface type or types for this service: • For firewall or NAT services, both **Left Interface** and **Right Interface** are required. • For an analyzer service, only a **Left Interface** is required. • For Juniper Networks virtual images, **Management Interface** is also required, in addition to any left or right requirement.

3. On **Add Service Template**, complete the following for the in-network service template:
   - **Name**: nat-template
   - **Service Mode**: In-Network
   - **Service Scaling**: Select from Advanced
   - **Image Name**: nat-service
   - **Interface Types**: Select Left Interface and Right Interface. For Juniper Networks virtual images, select Management Interface as the first interface.
   - The Left Interface will be automatically marked for sharing the same IP address
4. If multiple instances are to be launched for a particular service instance, select the **Service Scaling** check box, which enables the **Shared IP** feature. Figure 100 on page 307 shows the **Left** interface selected, with the **Shared IP** check box selected, so the left interface will share the IP address.

NOTE: The **Shared IP** for **Service Scaling** is an internal infrastructure feature used only for service scaling, it cannot be used for other features.

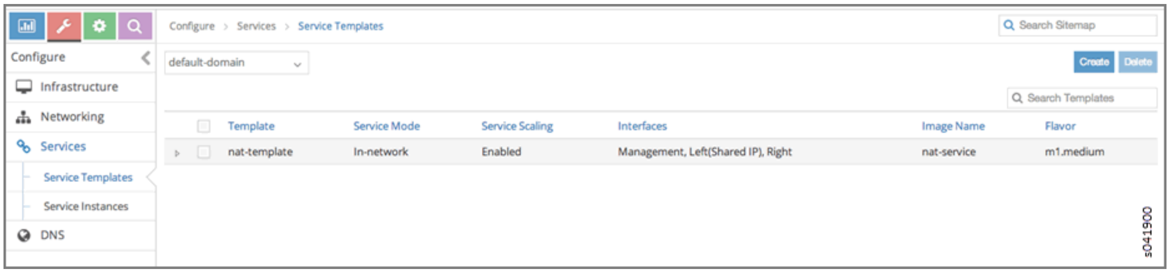
Figure 100: Add Service Template Shared IP



5. Click **Save**.

   The service template is created and appears on the **Service Templates** screen, see Figure 101 on page 308.

Figure 101: Service Templates



6. Create the service instance. Navigate to **Configure** > **Services** > **Service Instances**, and click **Create**, then select the template to use and select the corresponding left, right, or management networks; see Figure 102 on page 308.

Figure 102: Create Service Instances

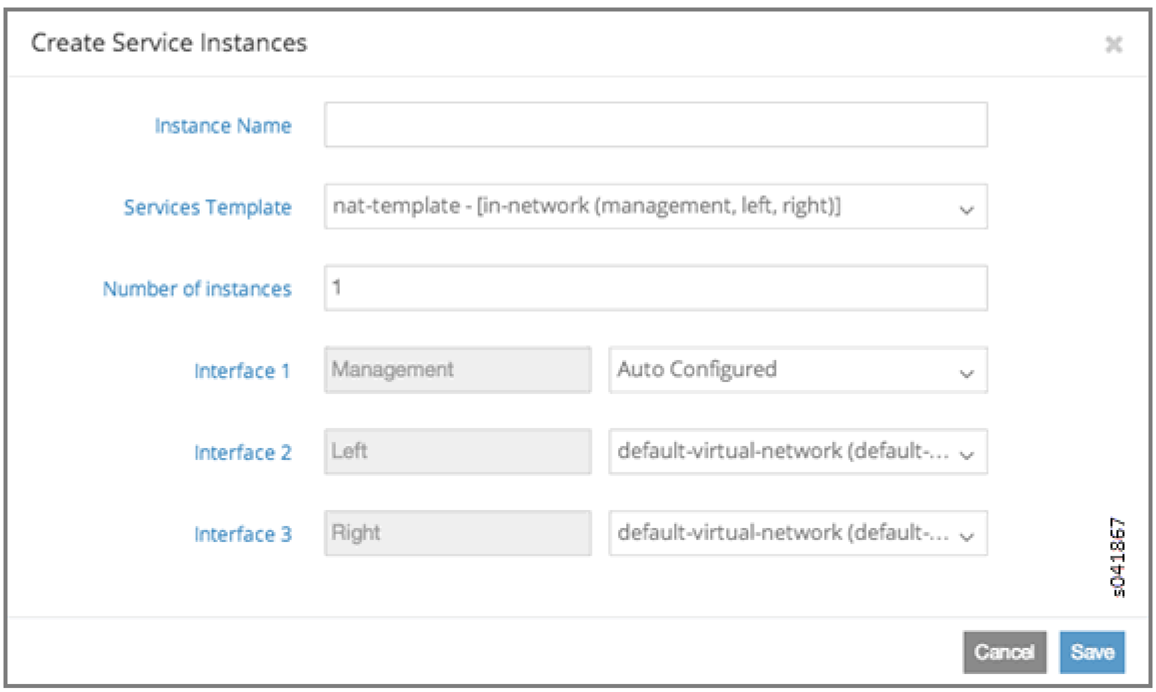


Table 41: Create Service Instances Fields

Field	Description
Instance Name	Enter a name for the service instance.

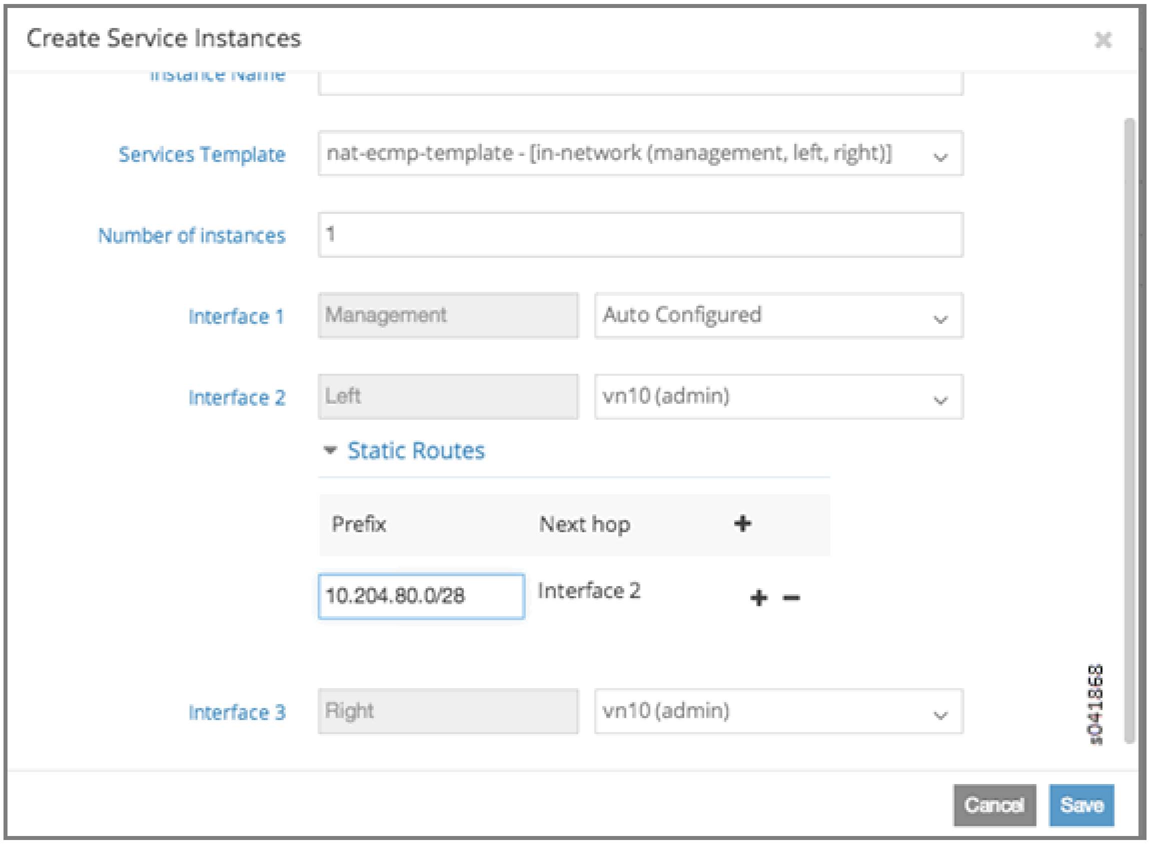
Table 41: Create Service Instances Fields *(Continued)*

Field	Description
Services Template	Select from a list of available service templates the service template to use for this instance.
Number of Instances	If scaling is enabled, enter a value in the **Number of Instances** field to define the number of instances of service virtual machines to launch.
Interface List and Virtual Networks	An ordered list of interfaces as defined in the Service Template. If you are using the **Management Interface**, select **Auto Configured**. The software will use an internally-created virtual network. For **Left Interface** , select **left_vn** and for **Right Interface**, select **right_vn**.

Table 41: Create Service Instances Fields *(Continued)*

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 103 on page 310.

Figure 103: 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 104 on page 310 and Figure 105 on page 311.

Figure 104: Service Instance Details

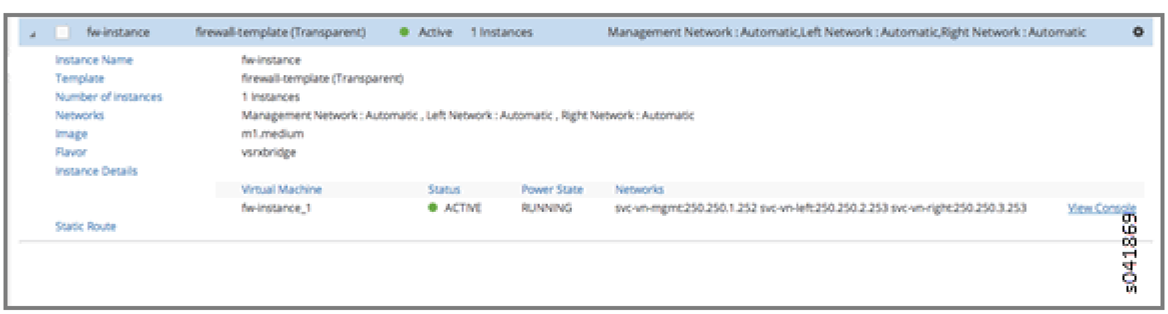
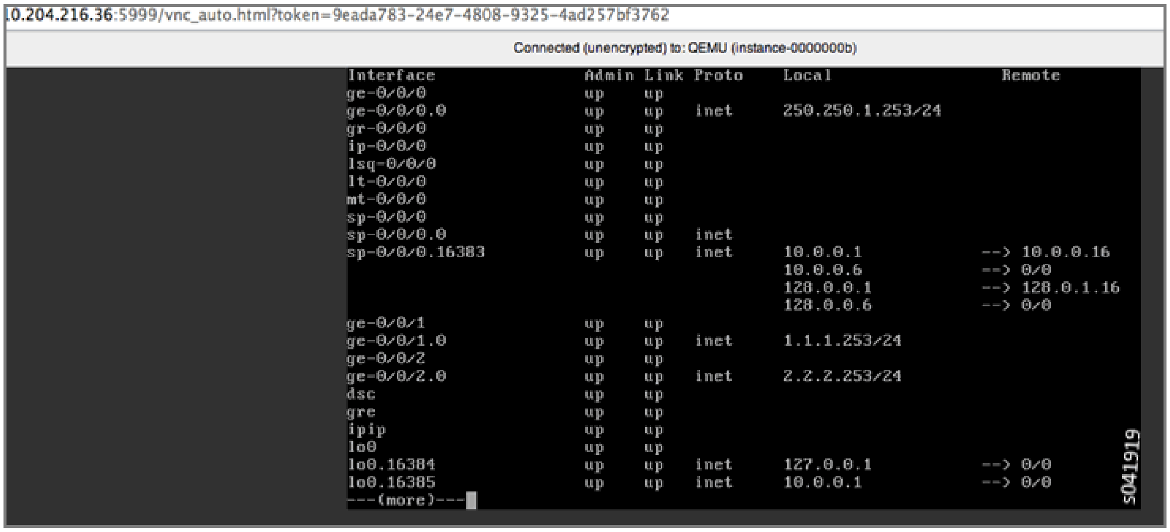


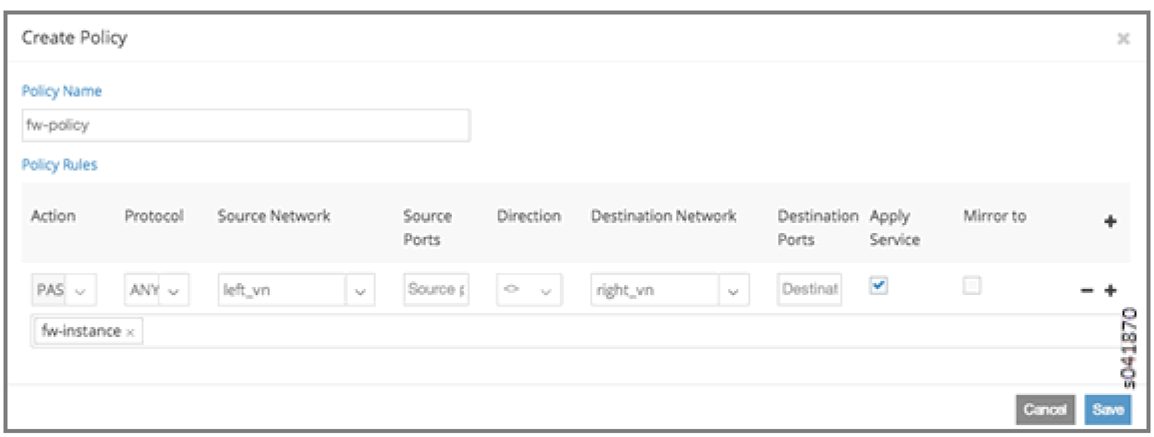
Figure 105: Service Instance Console



9. Configure the network policy. Navigate to **Configure** > **Networking** > **Policies**.

   - Name the policy and associate it with the networks created earlier: **left_vn** and **right_vn**.
   - Set source network as **left_vn** and destination network as **right_vn**.
   - Select **Apply Service** and select the service (**nat-ecmp**).

Figure 106: Create Policy

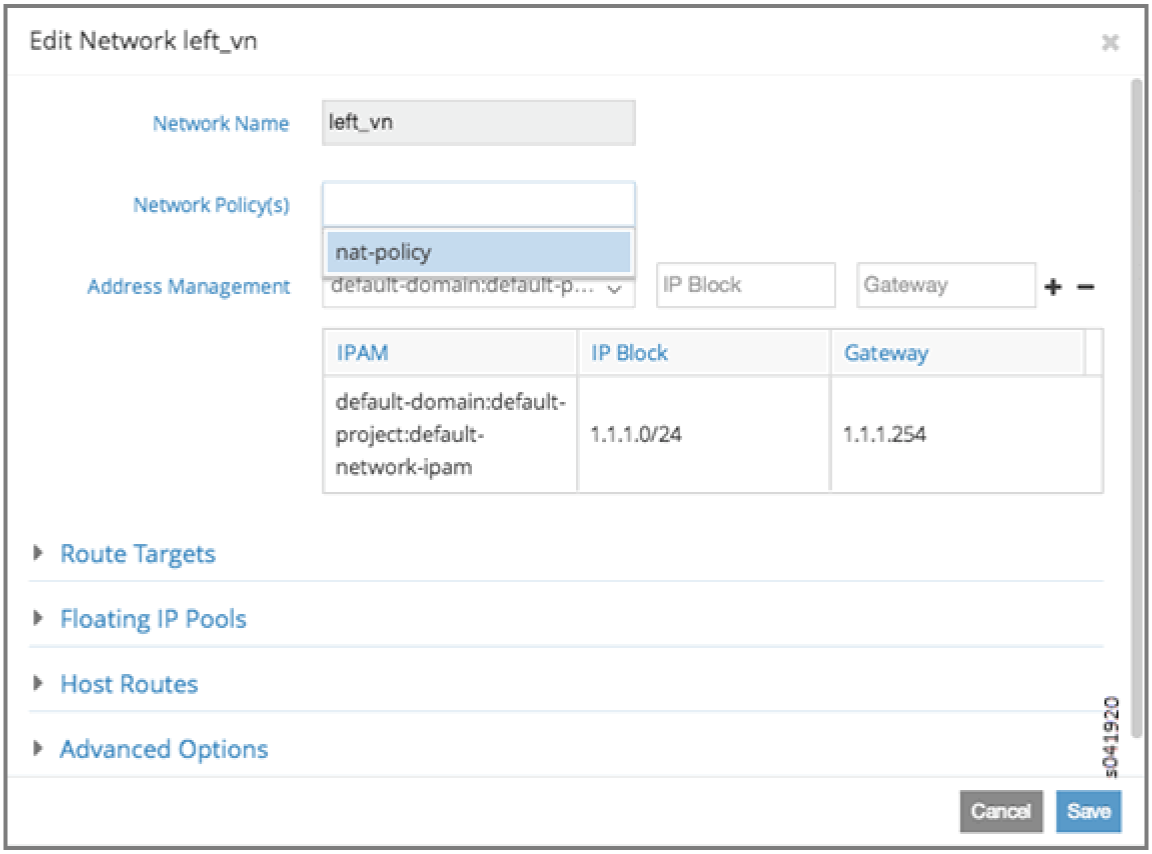


10. Associate the policy with both the **left_vn** and the **right_vn**. Navigate to **Configure** > **Networking** > **Network**.

    - On the right side of **left_vn**, click the gear icon to enable **Edit Network**.
    - In the **Edit Network** dialog box for **left_vn**, select **nat-policy** in the **Network Policy(s)** field.

- Repeat the same process for the **right_vn**.

Figure 107: Edit Network



11. Launch virtual machines (from OpenStack) and test the traffic through the service chain by doing the following:

    a. Navigate to **Configure > Networking > Policies**.

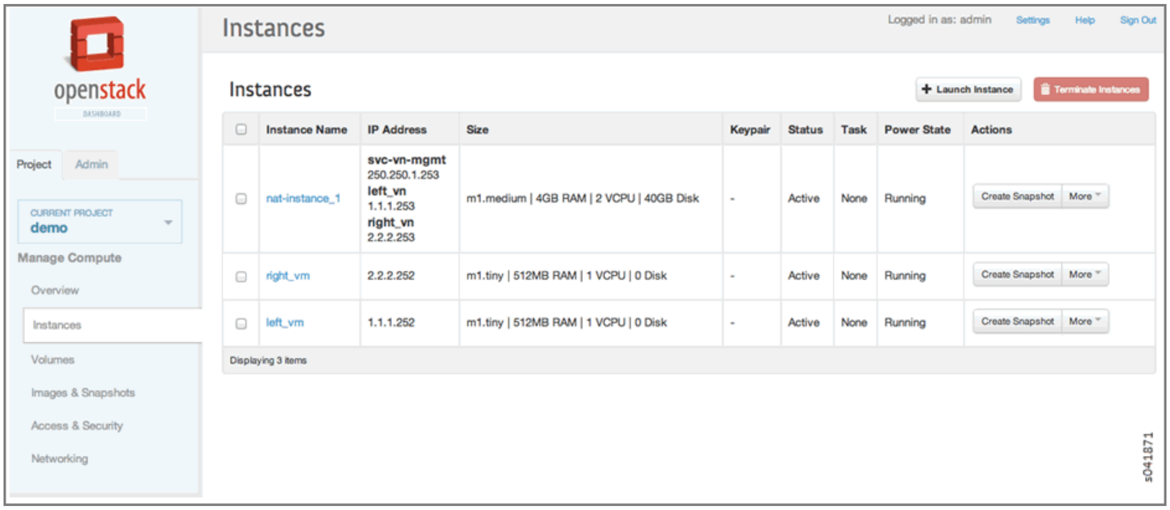
    b. Launch **left_vm** in virtual network **left_vn**.

    c. Launch **right_vm** in virtual network **right_vn**.

    d. Ping from **left_vm** to **right_vm** IP address (**2.2.2.252** in Figure 108 on page 313).

    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 108: Launch Instances



RELATED DOCUMENTATION

Service Chaining | 256

Example: Creating a Transparent Service Chain | 313

ECMP Load Balancing in the Service Chain | 262

Example: Creating a Transparent Service Chain

IN THIS SECTION

- Creating a Transparent Mode Service Chain | 314

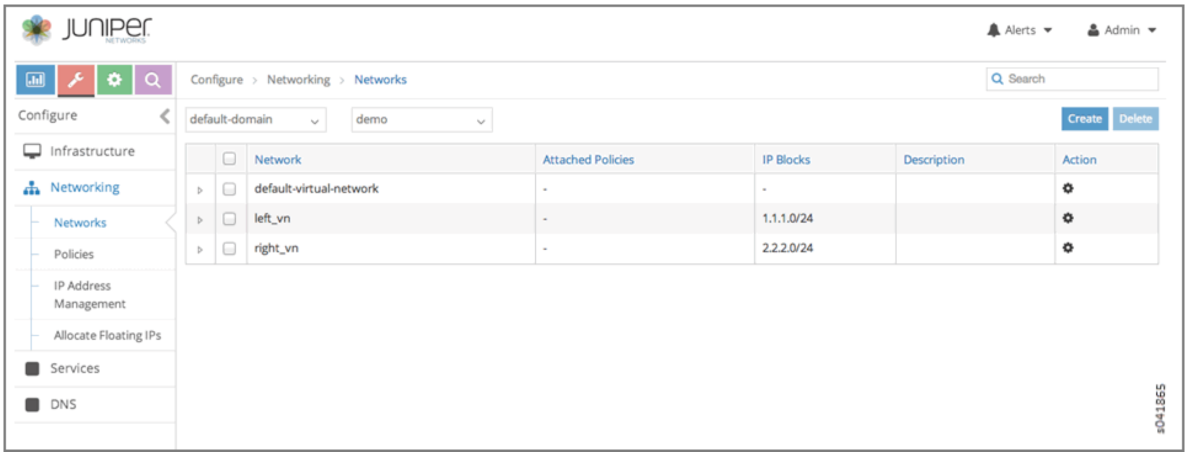
This section provides an example of creating a transparent mode service chain using the Juniper Networks Contrail user interface. Also called bridge mode, transparent mode is used for services that do not modify the packet, such as Layer 2 firewall, Intrusion Detection and Prevention (IDP), and so on. The following service chain example also shows scaling of service instances.

Creating a Transparent Mode Service Chain

To create a transparent mode service chain:

1. Create a left and a right virtual network. Select **Configure** > **Networking** > **Networks** and create **left_vn** and **right_vn**; see Figure 109 on page 314.

Figure 109: Create Networks



2. Configure a service template for a transparent mode. Navigate to **Configure** > **Services** > **Service Templates** and click the **Create** button on **Service Templates**. The **Add Service Template** window appears; see Figure 110 on page 315.

Figure 110: Add Service Template

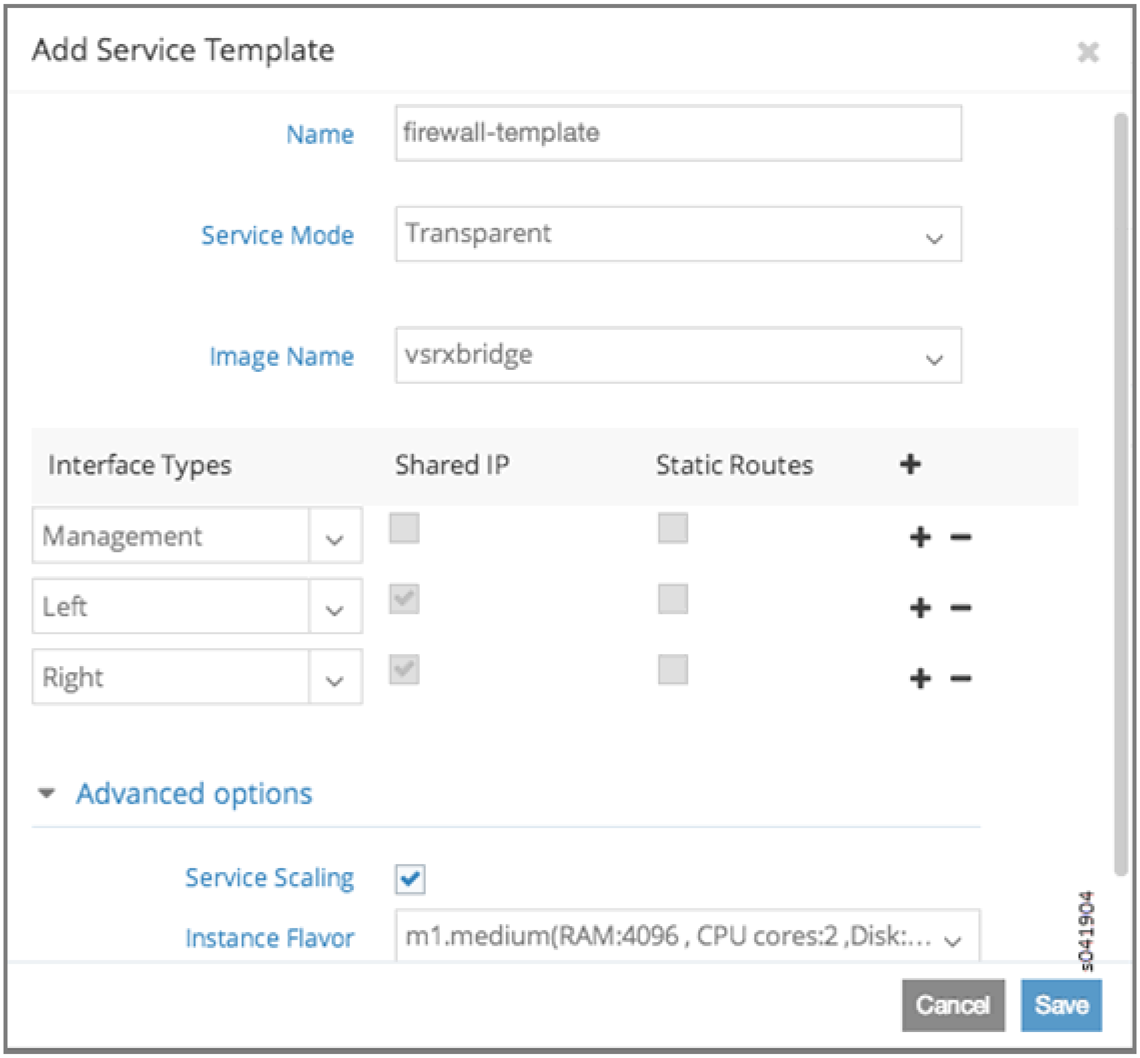


Table 42: Add Service Template Fields

Field	Description
Name	Enter a name for the service template.
Service Mode	Select the service mode: **In-Network** or **Transparent**.

Table 42: Add Service Template Fields *(Continued)*

Field	Description
Service Scaling	If you will be using multiple virtual machines for a single service instance to scale out the service, select the **Service Scaling** check box. When scaling is selected, you can choose to use the same IP address for a particular interface on each virtual machine interface or to allocate new addresses for each virtual machine. For a NAT service, the left (inner) interface should have the same IP address, and the right (outer) interface should have a different IP address.
Image Name	Select from a list of available images the image for the service.
Interface Types	Select the interface type or types for this service: • For firewall or NAT services, both **Left Interface** and **Right Interface** are required. • For an analyzer service, only **Left Interface** is required. • For Juniper Networks virtual images, **Management Interface** is also required, in addition to any left or right requirement.

3. On **Add Service Template**, complete the following for the transparent mode service template:

   - **Name**: firewall-template
   - **Service Mode**: Transparent
   - **Service Scaling**: Select this.
   - **Image Name**: vsrx-bridge
   - **Interface Types**: Select Left Interface, Right Interface, and Management Interface.

   If multiple instances are to be launched for a particular service instance, select the **Service Scaling** check box, which enables the **Shared IP** feature.

4. Click **Save**.

5. Create the service instance. Navigate to **Configure** > **Services** > **Service Instances**, and click **Create**, then select the template to use and select the corresponding left, right, or management networks; see Figure 111 on page 317.

Figure 111: Create Service Instances

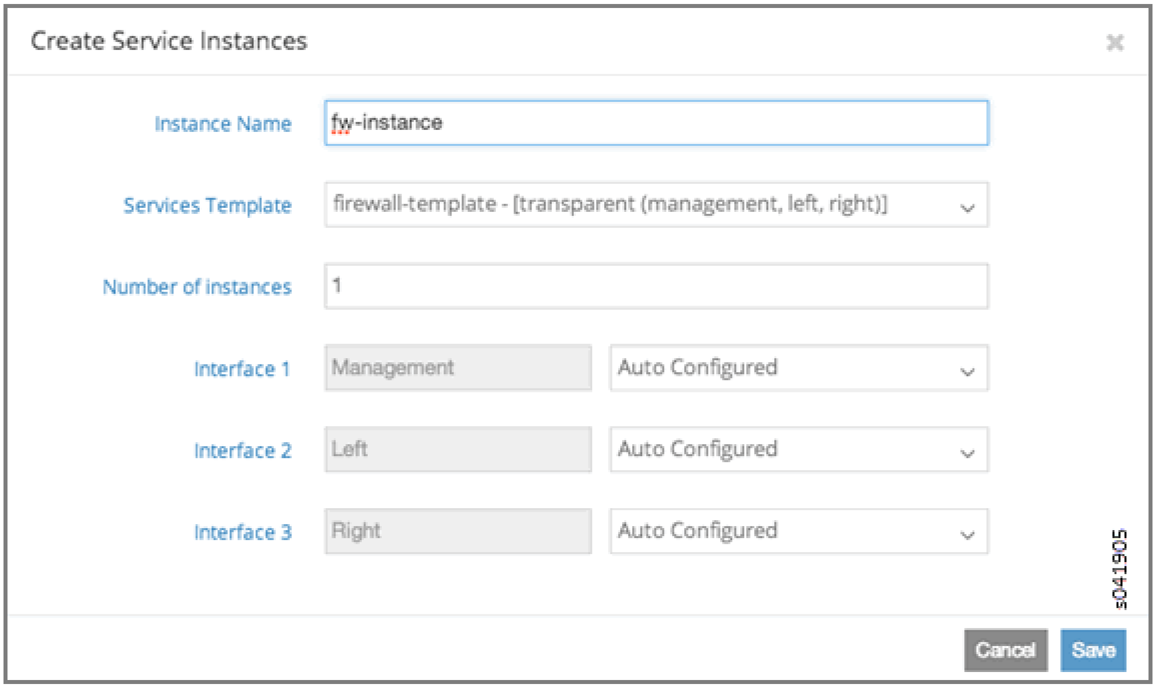
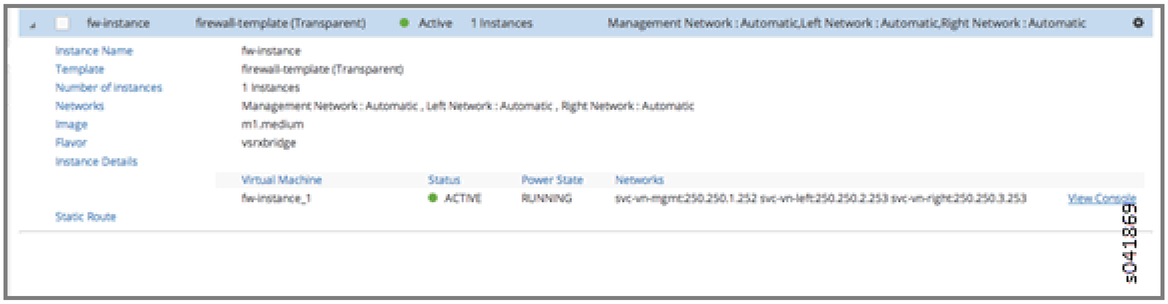


Table 43: Create Service Instances Fields

Field	Description
Instance Name	Enter a name for the service instance.
Services Template	Select from a list of available service templates the service template to use for this instance.
Left Network	Select from a list of available virtual networks the network to use for the left interface. For transparent mode, select **Auto Configured**.
Right Network	Select from a list of available virtual networks the network to use for the right interface. For transparent mode, select **Auto Configured**
Management Network	If you are using the **Management Interface**, select **Auto Configured.** The software will use an internally-created virtual network. For transparent mode, select **Auto Configured**

6. If scaling is enabled, enter a value in the **Number of Instances** field to define the number of instances of service virtual machines to launch; see Figure 112 on page 318.

Figure 112: Service Instance Details



7. Next, configure the network policy. Navigate to **Configure > Networking > Policies**.

- Name the policy **fw-policy**.
- Set source network as **left_vn** and destination network as **right_vn**.
- Check **Apply Service** and select the service (**fw-instance**).

Figure 113: Create Policy

8. Next, associate it to the networks created earlier – **left_vn** and **right_vn**. Navigate to **Configure > Networking > Policies**.

- On the right side of **left_vn**, click the gear icon to enable **Edit Network**.
- In the **Edit Network** dialog box for **left_vn**, select **nat-policy** in the **Network Policy(s)** field.
- Repeat the process for the **right_vn**.

9. Next, launch virtual machines (from OpenStack) and test the traffic through the service chain by doing the following:

   a. Navigate to **Configure > Networking > Policies**.

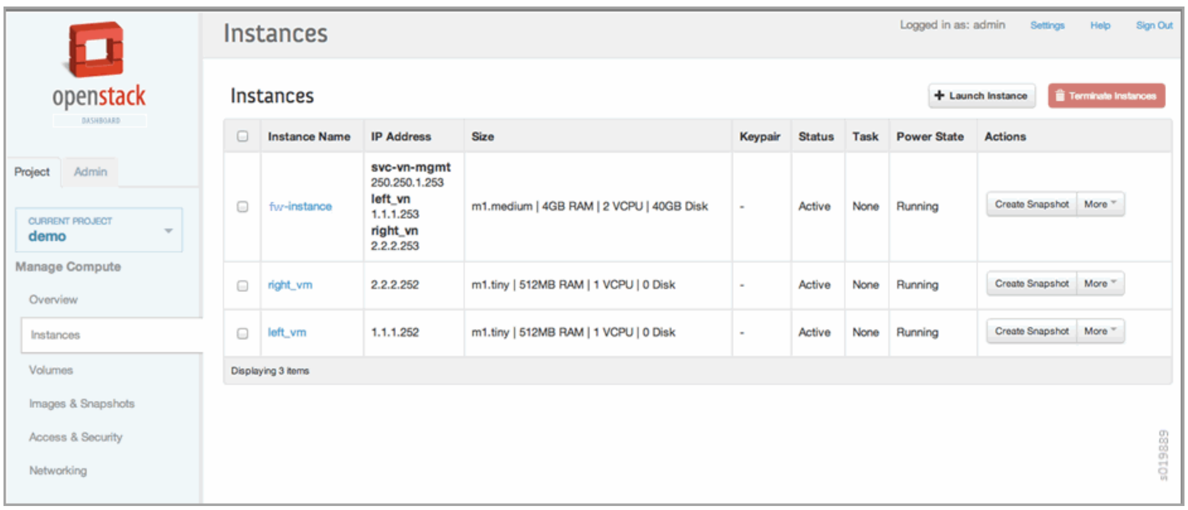
   b. Launch **left_vm** in virtual network **left_vn**.

   c. Launch **right_vm** in virtual network **right_vn**.

   d. Ping from **left_vm** to **right_vm** IP address (**2.2.2.252** in Figure 114 on page 319).

   e. A **TCPDUMP** on the **right_vm** should show that packets have the source IP set to **2.2.2.253**.

Figure 114: Launch Instances



RELATED DOCUMENTATION

Service Chaining | 256

Example: Creating a Service Chain With the CLI

IN THIS SECTION

- CLI for Creating a Service Chain | 320

- CLI for Creating a Service Template | 320
- CLI for Creating a Service Instance | 320
- CLI for Creating a Service Policy | 321
- Example: Creating a Service Chain with VSRX and In-Network or Routed Mode | 321

This section provides syntax and examples for creating service chaining objects for Contrail Controller.

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:

      i. `precise-server-cloudimg-amd64-disk1.img`

      ii. `junos-vsrx-12.1-nat.img`

   b. Add the images to Glance, using the names `ubuntu` and `vsrx`.

      i. `(source /etc/contrail/openstackrc; glance add name='ubuntu' is_public=true container_format=ovf disk_format=qcow2 < precise-server-cloudimg-amd64-disk1.img)`

      ii. `(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`.

   a. `./service-template.py add test_template --svc_type firewall --image_name vsrx`

3. Create virtual networks.

   a. VN1

   b. VN2

4. Create a service template.

   a. `./service-template.py add --svc_scaling ecmp-template`

5. Create a service instance.

   a. `./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.

   a. `./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.

   a. **VM1** (attached to **VN1**)—use **ubuntu** image

   b. **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 | **256**

4

PART

Monitoring and Troubleshooting the Network Using Contrail Analytics

Understanding Contrail Analytics | 325

Configuring Contrail Analytics | 353

Using Contrail Analytics to Monitor and Troubleshoot the Network | 367

CHAPTER 13

Understanding Contrail Analytics

IN THIS CHAPTER

- Understanding Contrail Analytics | **325**
- Contrail Alerts | **326**
- Underlay Overlay Mapping in Contrail | **330**

Understanding Contrail Analytics

Contrail is a distributed system of compute nodes, control nodes, configuration nodes, database nodes, web UI nodes, and analytics nodes.

The analytics nodes are responsible for the collection of system state information, usage statistics, and debug information from all of the software modules across all of the nodes of the system. The analytics nodes store the data gathered across the system in a database that is based on the Apache Cassandra open source distributed database management system. The database is queried by means of an SQL-like language and representational state transfer (REST) APIs.

System state information collected by the analytics nodes is aggregated across all of the nodes, and comprehensive graphical views allow the user to get up-to-date system usage information easily.

Debug information collected by the analytics nodes includes the following types:

- System log (syslog) messages—informational and debug messages generated by system software components.
- Object log messages—records of changes made to system objects such as virtual machines, virtual networks, service instances, virtual routers, BGP peers, routing instances, and the like.
- Trace messages—records of activities collected locally by software components and sent to analytics nodes only on demand.

Statistics information related to flows, CPU and memory usage, and the like is also collected by the analytics nodes and can be queried at the user interface to provide historical analytics and time-series information. The queries are performed using REST APIs.

Analytics data is written to a database in Contrail. The data expires after the default time-to-live (TTL) period of 48 hours. This default TTL time can be changed as needed by changing the value of the `database_ttl` value in the cluster configuration.

RELATED DOCUMENTATION

Contrail Alerts | 326

Analytics Scalability | 353

High Availability for Analytics | 354

Ceilometer Support in a Contrail Cloud | 359

Underlay Overlay Mapping in Contrail | 330

Monitoring the System | 367

Debugging Processes Using the Contrail Introspect Feature | 371

Monitor > Infrastructure > Dashboard | 376

Monitor > Infrastructure > Control Nodes | 380

Monitor > Infrastructure > Virtual Routers | 391

Monitor > Infrastructure > Analytics Nodes | 405

Monitor > Infrastructure > Config Nodes | 413

Monitor > Networking | 417

Understanding Flow Sampling

Query > Flows | 429

Query > Logs | 439

System Log Receiver in Contrail Analytics | 356

Example: Debugging Connectivity Using Monitoring for Troubleshooting | 446

Contrail Alerts

IN THIS SECTION

- Alert API Format | 327
- Analytics APIs for Alerts | 328
- Analytics APIs for SSE Streaming | 329

• Built-in Node Alerts | 329

Starting with Contrail 3.0 and greater, Contrail alerts are provided on a per-user visible entity (UVE) basis.

Contrail analytics raise or clear alerts using Python-coded rules that examine the contents of the UVE and the configuration of the object. Some rules are built in. Others can be added using Python *stevedore* plugins.

This topic describes Contrail alerts capabilities.

Alert API Format

The Contrail alert analytics API provides the following:

- Read access to the alerts as part of the UVE GET APIs.
- Alert acknowledgement using POST requests.
- UVE and alert streaming using server-sent events (SSEs).

For example:

GET http://*<analytics-ip>*:8081/analytics/uves/control-node/a6s40?flat

```
{
    NodeStatus:  {…},
    ControlCpuState:  {…},
    UVEAlarms:  {
    alarms:  [
        {
            description:  [
                {
                value: "0 != 2",
                rule: "BgpRouterState.num_up_bgp_peer != BgpRouterState.num_bgp_peer"
                }
            ],
            ack: false,
            timestamp: 1442995349253178,
            token: "eyJ0aW1lc3RhbXAiOiAxNDQyOTk1MzQ5MjUzMTc4LCAiaHR0cF9wb3J0Ijog
NTk5NSwgImhvc3RfaXAiOiAiMTAuODQuMTMuNDAifQ==",
```

```
            type: "BgpConnectivity",
            severity: 4
        }
    ]
    },
    BgpRouterState:  {…}
}
```

In the example:

- Alerts are raised on a per-UVE basis and can be retrieved by a GET on a UVE.
- An `ack` indicates if the alert has been acknowledged or not.
- A `token` is used by clients when requesting acknowledgements

Analytics APIs for Alerts

The following examples show the API to use to display alerts and alarms and to acknowledge alarms.

- To retrieve a list of alerts raised against the control node named `aXXsYY`.

  ```
  GET http://<analytics-ip>:<rest-api-port>/analytics/uves/control-node/aXXsYY&cfilt=UVEAlarms
  ```

  This is available for all UVE table types.

- To retrieve a list of all alarms in the system.

  ```
  GET http://<analytics-ip>:<rest-api-port>/analytics/alarms
  ```

- To acknowledge an alarm.

  ```
  POST http://<analytics-ip>:<rest-api-port>/analytics/alarms/acknowledge
  Body: {“table”: <object-type>,“name”: <key>, “type”: <alarm type>, “token”: <token>}
  ```

  Acknowledged and unacknowledged alarms can be queried specifically using the following URL query parameters along with the GET operations listed previously.

  ```
  ackFilt=True
  ackFilt=False
  ```

Analytics APIs for SSE Streaming

The following examples show the API to use to retrieve all or portions of SE streams.

- To retrieve an SSE-based stream of UVE updates for the control node alarms.

  ```
  GET http://<analytics-ip>:<rest-api-port>/analytics/uve-stream?tablefilt=control-node
  ```

  This is available for all UVE table types. If the `tablefilt` URL query parameter is not provided, all UVEs are retrieved.

- To retrieve only the alerts portion of the SSE-based stream of UVE updates instead of the entire content.

  ```
  GET http://<analytics-ip>:<rest-api-port>/analytics/alarm-stream?tablefilt=control-node
  ```

  This is available for all UVE table types. If the `tablefilt` URL query parameter is not provided, all UVEs are retrieved.

Built-in Node Alerts

The following built-in node alerts can be retrieved using the APIs listed in *Analytics APIs for Alerts*.

```
control-node: {
PartialSysinfoControl: "Basic System Information is absent for this node in
BgpRouterState.build_info",
ProcessStatus: "NodeMgr reports abnormal status for process(es) in NodeStatus.process_info",
XmppConnectivity: "Not enough XMPP peers are up in BgpRouterState.num_up_bgp_peer",
BgpConnectivity: "Not enough BGP peers are up in BgpRouterState.num_up_bgp_peer",
AddressMismatch: "Mismatch between configured IP Address and operational IP Address",
ProcessConnectivity: "Process(es) are reporting non-functional components in
NodeStatus.process_status"
},

vrouter: {
PartialSysinfoCompute: "Basic System Information is absent for this node in
VrouterAgent.build_info",
ProcessStatus: "NodeMgr reports abnormal status for process(es) in NodeStatus.process_info",
ProcessConnectivity: "Process(es) are reporting non-functional components in
NodeStatus.process_status",
VrouterInterface: "VrouterAgent has interfaces in error state in VrouterAgent.error_intf_list",
```

```
VrouterConfigAbsent: “Vrouter is not present in Configuration”,
},

config-node: {
PartialSysinfoConfig: "Basic System Information is absent for this node in
ModuleCpuState.build_info",
ProcessStatus: "NodeMgr reports abnormal status for process(es) in NodeStatus.process_info",
ProcessConnectivity: "Process(es) are reporting non-functional components in
NodeStatus.process_status"
},

analytics-node: {
ProcessStatus: "NodeMgr reports abnormal status for process(es) in NodeStatus.process_info"
PartialSysinfoAnalytics: "Basic System Information is absent for this node in
CollectorState.build_info",
ProcessConnectivity: "Process(es) are reporting non-functional components in
NodeStatus.process_status"
},

database-node: {
ProcessStatus: "NodeMgr reports abnormal status for process(es) in NodeStatus.process_info",
ProcessConnectivity: "Process(es) are reporting non-functional components in
NodeStatus.process_status"
},
```

Underlay Overlay Mapping in Contrail

IN THIS SECTION

- Overview: Underlay Overlay Mapping using Contrail Analytics | 331
- Underlay Overlay Analytics Available in Contrail | 331
- Architecture and Data Collection | 332
- New Processes/Services for Underlay Overlay Mapping | 332
- External Interfaces Configuration for Underlay Overlay Mapping | 333
- Physical Topology | 333
- SNMP Configuration | 334

- Link Layer Discovery Protocol (LLDP) Configuration | 334
- IPFIX and sFlow Configuration | 334
- Sending pRouter Information to the SNMP Collector in Contrail | 337
- pRouter UVEs | 337
- Contrail User Interface for Underlay Overlay Analytics | 339
- Enabling Physical Topology on the Web UI | 340
- Viewing Topology to the Virtual Machine Level | 340
- Viewing the Traffic of any Link | 340
- Trace Flows | 341
- Search Flows and Map Flows | 342
- Overlay to Underlay Flow Map Schemas | 343
- Module Operations for Overlay Underlay Mapping | 346
- SNMP Collector Operation | 346
- Topology Module Operation | 348
- IPFIX and sFlow Collector Operation | 349
- Troubleshooting Underlay Overlay Mapping | 350
- Script to add pRouter Objects | 350

Overview: Underlay Overlay Mapping using Contrail Analytics

Today's cloud data centers consist of large collections of interconnected servers that provide computing and storage capacity to run a variety of applications. The servers are connected with redundant TOR switches, which in turn, are connected to spine routers. The cloud deployment is typically shared by multiple tenants, each of whom usually needs multiple isolated networks. Multiple isolated networks can be provided by overlay networks that are created by forming tunnels (for example, gre, ip-in-ip, mac-in-mac) over the underlay or physical connectivity.

As data flows in the overlay network, Contrail can provide statistics and visualization of the traffic in the underlay network.

Underlay Overlay Analytics Available in Contrail

Starting with Contrail Release 2.20, you can view a variety of analytics related to underlay and overlay traffic in the Contrail Web user interface. The following are some of the analytics that Contrail provides for statistics and visualization of overlay underlay traffic.

- View the topology of the underlay network.

A user interface view of the physical underlay network with a drill down mechanism to show connected servers (contrail computes) and virtual machines on the servers.

- View the details of any element in the topology.

  You can view details of a pRouter, vRouter, or virtual machine link between two elements. You can also view traffic statistics in a graphical view corresponding to the selected element.

- View the underlay path of an overlay flow.

  Given an overlay flow, you can get the underlay path used for that flow and map the path in the topology view.

Architecture and Data Collection

Accumulation of the data to map an overlay flow to its underlay path is performed in several steps across Contrail modules.

The following outlines the essential steps:

1. The SNMP collector module polls physical routers.

   The SNMP collector module receives the authorizations and configurations of the physical routers from the Contrail config module, and polls all of the physical routers, using SNMP protocol. The collector uploads the data to the Contrail analytics collectors. The SNMP information is stored in the pRouter UVEs (physical router user visible entities).
2. IPFIX and sFlow protocols are used to collect the flow statistics.

   The physical router is configured to send flow statistics to the collector, using one of the collection protocols: Internet Protocol Flow Information Export (IPFIX) or sFlow (an industry standard for sampled flow of packet export at Layer 2).
3. The topology module reads the SNMP information.

   The Contrail topology module reads SNMP information from the pRouter UVEs from the analytics API, computes the neighbor list, and writes the neighbor information into the pRouter UVEs. This neighbor list is used by the Contrail WebUI to display the physical topology.
4. The Contrail user interface reads and displays the topology and statistics.

   The Contrail user interface module reads the topology information from the Contrail analytics and displays the physical topology. It also uses information stored in the analytics to display graphs for link statistics, and to show the map of the overlay flows on the underlay network.

New Processes/Services for Underlay Overlay Mapping

The `contrail-snmp-collector` and the `contrail-topology` are new daemons that are both added to the `contrail-analytics` node. The `contrail-analytics` package contains these new features and their associated files. The contrail-status displays the new services.

Example: contrail-status

The following is an example of using `contrail-status` to show the status of the new process and service for underlay overlay mapping.

```
user@host:~# contrail-status

== Contrail Control ==

supervisor-control:      active

contrail-control       active

…

== Contrail Analytics ==

supervisor-analytics:     active

…

contrail-query-engine     active

contrail-snmp-collector    active

contrail-topology       active
```

Example: Service Command

The `service` command can be used to start, stop, and restart the new services. See the following example.

```
user@host:~# service contrail-snmp-collector status

contrail-snmp-collector     RUNNING  pid 12179, uptime 1 day, 14:59:11
```

External Interfaces Configuration for Underlay Overlay Mapping

This section outlines the external interface configurations necessary for successful underlay overlay mapping for Contrail analytics.

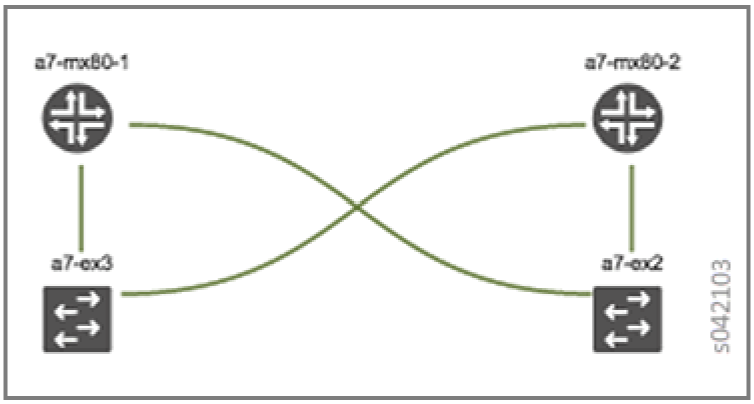
Physical Topology

The typical physical topology includes:

- Servers connected to the ToR switches.
- ToR switches connected to spine switches.
- Spine switches connected to core switches.

The following is an example of how the topology is depicted in the Contrail WebUI analytics.

Figure 115: Analytics Topology



SNMP Configuration

Configure SNMP on the physical devices so that the `contrail-snmp-collector` can read SNMP data.

The following shows an example SNMP configuration from a Juniper Networks device.

```
set snmp community public authorization read-only
```

Link Layer Discovery Protocol (LLDP) Configuration

Configure LLDP on the physical device so that the `contrail-snmp-collector` can read the neighbor information of the routers.

The following is an example of LLDP configuration on a Juniper Networks device.

```
set protocols lldp interface all
```

```
set protocols lldp-med interface all
```

IPFIX and sFlow Configuration

Flow samples are sent to the `contrail-collector` by the physical devices. Because the `contrail-collector` supports the sFlow and IPFIX protocols for receiving flow samples, the physical devices, such as MX Series devices or ToR switches, must be configured to send samples using one of those protocols.

Example: sFlow Configuration

The following shows a sample sFlow configuration. In the sample, the IP variable *<source ip>*refers to the loopback or IP that can be reachable of the device that acts as an sflow source, and the other IP variable *<collector_IP_data>* is the address of the collector device.

```
root@host> show configuration protocols sflow | display set

set protocols sflow polling-interval 0

set protocols sflow sample-rate ingress 10

set protocols sflow source-ip <source ip>4

set protocols sflow collector <collector_IP_data>  udp-port 6343

set protocols sflow interfaces ge-0/0/0.0

set protocols sflow interfaces ge-0/0/1.0

set protocols sflow interfaces ge-0/0/2.0

set protocols sflow interfaces ge-0/0/3.0

set protocols sflow interfaces ge-0/0/4.0
```

Example: IPFIX Configuration

The following is a sample IPFIX configuration from a Juniper Networks device. The IP address variable *<ip_sflow collector>* represents the sflow collector (control-collector analytics node) and *<source ip>* represents the source (outgoing) interface on the router/switch device used for sending flow data to the collector. This could also be the lo0 address, if it s reachable from the Contrail cluster.

```
root@host> show configuration chassis | display set

set chassis tfeb slot 0 sampling-instance sample-ins1

set chassis network-services



root@host> show configuration chassis tfeb | display set
```

```
set chassis tfeb slot 0 sampling-instance sample-ins1


root@host > show configuration services flow-monitoring | display set

set services flow-monitoring version-ipfix template t1 flow-active-timeout 30

set services flow-monitoring version-ipfix template t1 flow-inactive-timeout 30

set services flow-monitoring version-ipfix template t1 template-refresh-rate packets 10

set services flow-monitoring version-ipfix template t1 ipv4-template


root@host > show configuration interfaces | display set | match sampling

set interfaces ge-1/0/0 unit 0 family inet sampling input

set interfaces ge-1/0/1 unit 0 family inet sampling input


root@host> show configuration forwarding-options sampling | display set

set forwarding-options sampling instance sample-ins1 input rate 1

set forwarding-options sampling instance sample-ins1 family inet output flow-server <ip_sflow
collector> port 4739

set forwarding-options sampling instance sample-ins1 family inet output flow-server <ip_sflow
collector> version-ipfix template t1

set forwarding-options sampling instance sample-ins1 family inet output inline-jflow source-
address <source ip>
```

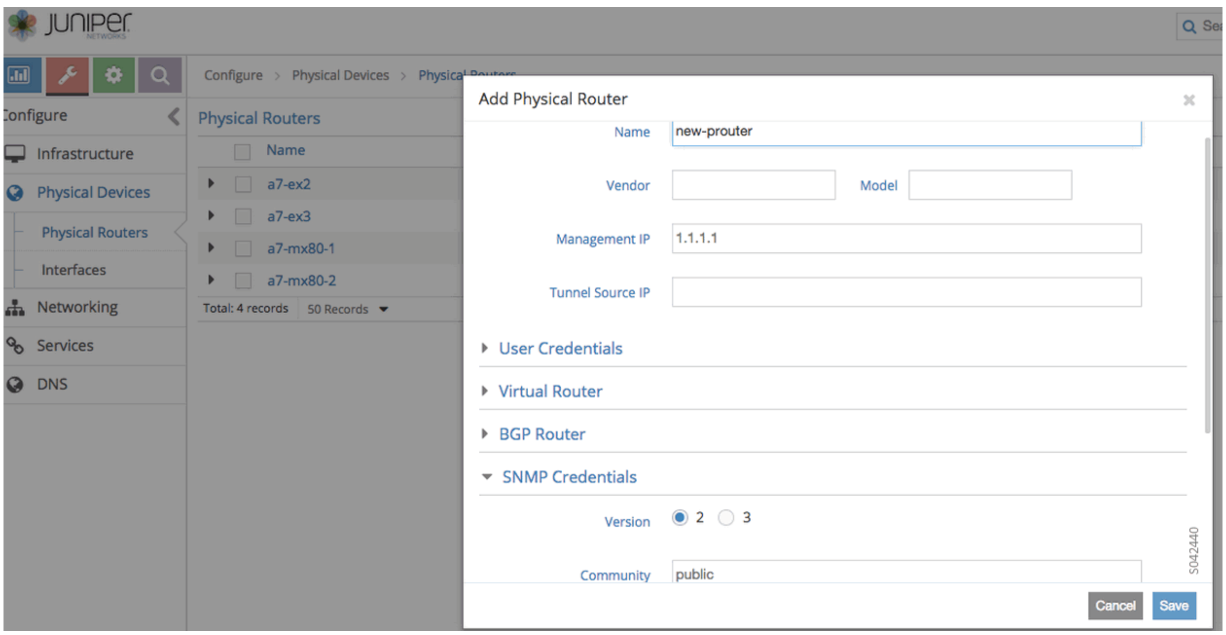
Sending pRouter Information to the SNMP Collector in Contrail

Information about the physical devices must be sent to the SNMP collector before the full analytics information can be read and displayed. Typically, the pRouter information is taken from the `contrail-config` file.

SNMP collector getting pRouter information from contrail-config file

The physical routers are added to the `contrail-config` by using the Contrail user interface or by using direct API, by means of provisioning or other scripts. Once the configuration is in the `contrail-config`, the `contrail-snmp-collector` gets the physical router information from `contrail-config`. The SNMP collector uses this list and the other configuration parameters to perform SNMP queries and to populate pRouter UVEs.

Figure 116: Add Physical Router Window



pRouter UVEs

pRouter UVEs are accessed from the REST APIs on your system from `contrail-analytics-api`, using a URL of the form:

```
http://<host ip>:8081/analytics/uves/prouters
```

The following is sample output from a pRouter REST API:

Figure 117: Sample Output From a pRouter REST API


```
[
  - {
        href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-mx80-1?flat",
        name: "a7-mx80-1"
    },
  - {
        href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-mx80-2?flat",
        name: "a7-mx80-2"
    },
  - {
        href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-ex3?flat",
        name: "a7-ex3"
    },
  - {
        href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-ex2?flat",
        name: "a7-ex2"
    }
]
```


s042104

Details of a pRouter UVE can be obtained from your system, using a URL of the following form:

```
http://<host ip>:8081/analytics/uves/prouter/a7-ex3?flat
```

The following is sample output of a pRouter UVE.

Figure 118: Sample Output From a pRouter UVE


```
{
  - PRouterFlowEntry: {
        flow_export_source_ip: "10.84.63.114"
    },
  - PRouterLinkEntry: {
      - link_table: [
          - {
                remote_interface_name: "ge-1/0/1",
                local_interface_name: "ge-0/0/0.0",
                remote_interface_index: 517,
                local_interface_index: 503,
                type: 1,
                remote_system_name: "a7-mx80-1"
            },
          - {
                remote_interface_name: "ge-1/0/1",
                local_interface_name: "ge-0/0/1.0",
                remote_interface_index: 517,
                local_interface_index: 505,
                type: 1,
                remote_system_name: "a7-mx80-2"
            },
          - {
                remote_interface_name: "eth1",
                local_interface_name: "ge-0/0/2.0",
                remote_interface_index: 1,
                local_interface_index: 507,
                type: 2,
                remote_system_name: "a7s35"
            },
          - {
                remote_interface_name: "eth1",
                local_interface_name: "ge-0/0/3.0",
                remote_interface_index: 1,
                local_interface_index: 509,
                type: 2,
                remote_system_name: "a7s36"
            }
        ]
    },
  - PRouterEntry: {
      + ipMib: […],
      + ifTable: […],
      + ifXTable: […],
      + arpTable: […],
      + lldpTable: {…},
      + ifStats: […]
    }
}
```


s042435

Contrail User Interface for Underlay Overlay Analytics

The topology view and related functionality is accessed from the Contrail Web user interface, **Monitor > Physical Topology**.

Enabling Physical Topology on the Web UI

To enable the **Physical Topology** section in the Contrail Web UI:

1. Add the following lines to the `/etc/contrail/config.global.js` file of all the `contrail-webui` nodes:

```
config.optFeatureList = {};
config.optFeatureList.mon_infra_underlay = true;
```

2. Restart webui supervisor.

   `service supervisor-webui restart`

   The **Physical Topology** section is now available on the Contrail Web UI.

Viewing Topology to the Virtual Machine Level

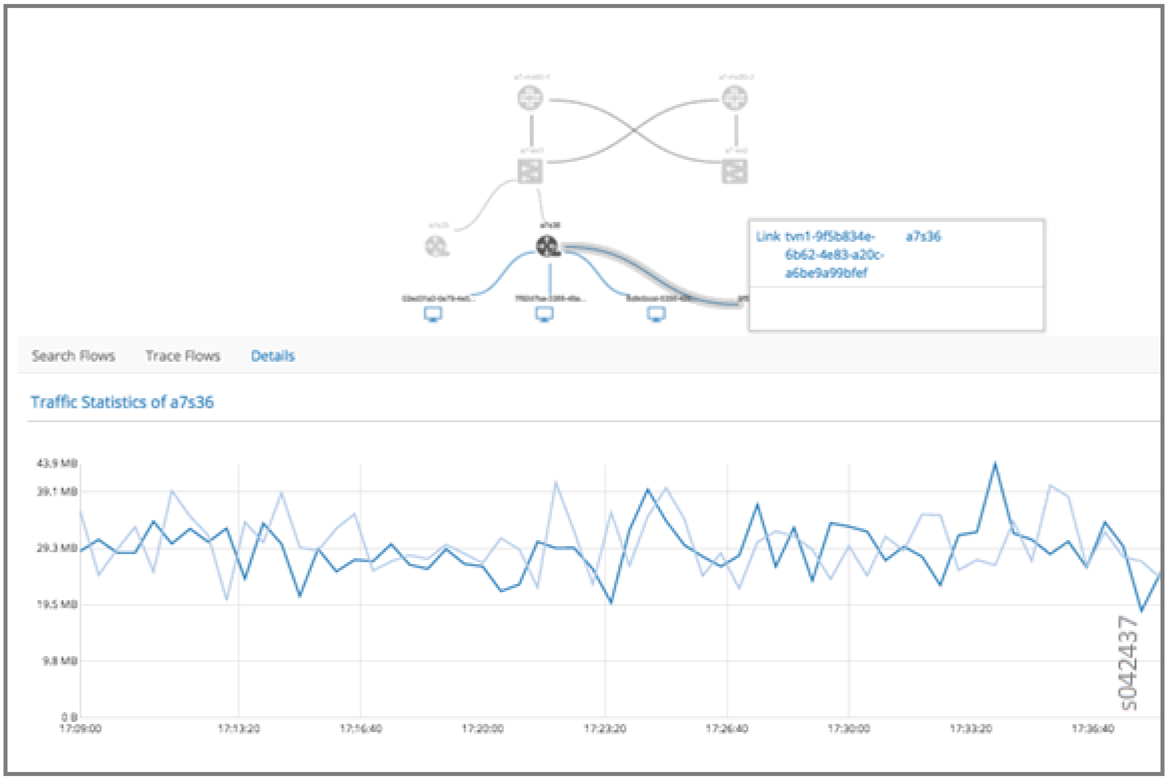
In the Contrail user interface, it is possible to drill down through displayed topology to the virtual machine level. The following diagram shows the virtual machines instantiated on a7s36 vRouter and the full physical topology related to each.

Figure 119: Physical Topology Related to a vRouter

Viewing the Traffic of any Link

At **Monitor > Physical Topology**, double click any link on the topology to display the traffic statistics graph for that link. The following is an example.

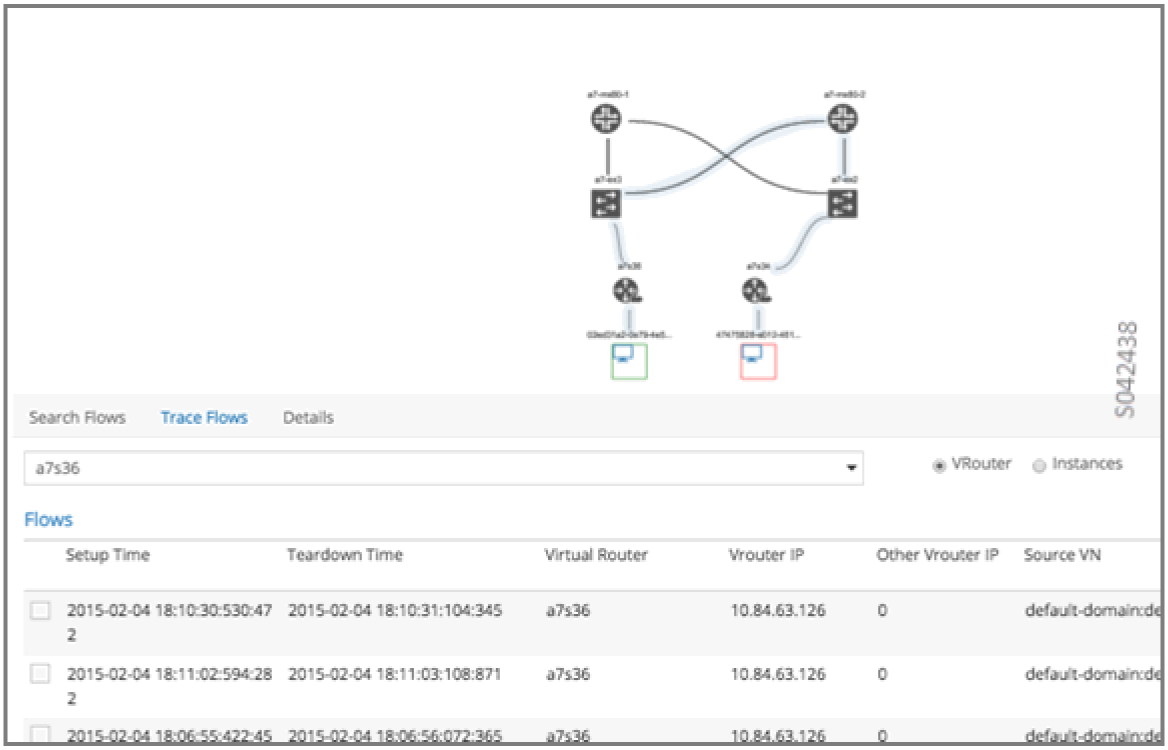
Figure 120: Traffic Statistics Graph



Trace Flows

Click the **Trace Flows** tab to see a list of active flows. To see the path of a flow, click a flow in the active flows list, then click the **Trace Flow** button. The path taken in the underlay by the selected flow displays. The following is an example.

Figure 121: List of Active Flows



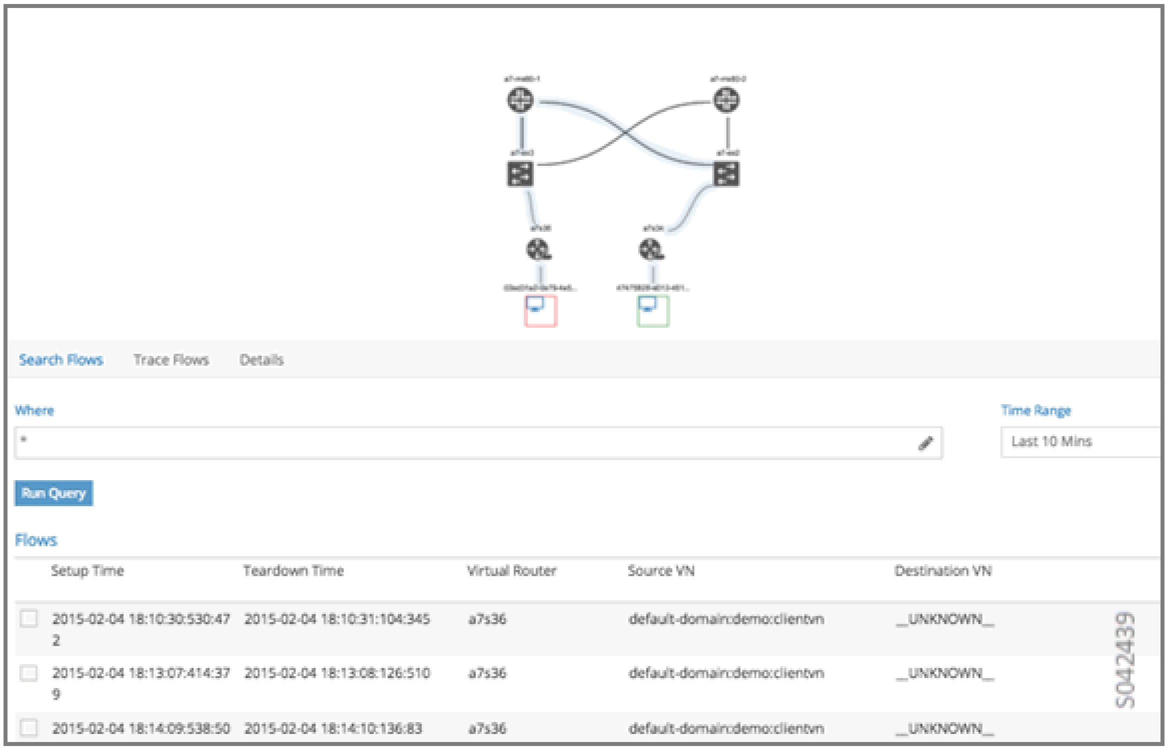
Limitations of Trace Flow Feature

Because the Trace Flow feature uses ip traceroute to determine the path between the two vRouters involved in the flow, it has the same limitations as the ip traceroute, including that Layer 2 routers in the path are not listed, and therefore do not appear in the topology.

Search Flows and Map Flows

Click the **Search Flows** tab to open a search dialog, then click the **Search** button to list the flows that match the search criteria. You can select a flow from the list and click **Map Flow** to display the underlay path taken by the selected flow in the topology. The following is an example.

Figure 122: Underlay Path



Overlay to Underlay Flow Map Schemas

The schema to query the underlay mapping information for an overlay flow is obtained from a REST API, which can be accessed on your system using a URL of the following form:

`http://<host ip>:8081/analytics/table/OverlayToUnderlayFlowMap/schema`

Example: Overlay to Underlay Flow Map Schema

```
{"type": "FLOW",

"columns": [

{"datatype": "string", "index": true, "name": "o_svn", "select": false, "suffixes": ["o_sip"]},

{"datatype": "string", "index": false, "name": "o_sip", "select": false, "suffixes": null},

{"datatype": "string", "index": true, "name": "o_dvn", "select": false, "suffixes": ["o_dip"]},

{"datatype": "string", "index": false, "name": "o_dip", "select": false, "suffixes": null},
```

```
{"datatype": "int", "index": false, "name": "o_sport", "select": false, "suffixes": null},

{"datatype": "int", "index": false, "name": "o_dport", "select": false, "suffixes": null},

{"datatype": "int", "index": true, "name": "o_protocol", "select": false, "suffixes":
["o_sport", "o_dport"]},

{"datatype": "string", "index": true, "name": "o_vrouter", "select": false, "suffixes": null},

{"datatype": "string", "index": false, "name": "u_prouter", "select": null, "suffixes": null},

{"datatype": "int", "index": false, "name": "u_pifindex", "select": null, "suffixes": null},

{"datatype": "int", "index": false, "name": "u_vlan", "select": null, "suffixes": null},

{"datatype": "string", "index": false, "name": "u_sip", "select": null, "suffixes": null},

{"datatype": "string", "index": false, "name": "u_dip", "select": null, "suffixes": null},

{"datatype": "int", "index": false, "name": "u_sport", "select": null, "suffixes": null},

{"datatype": "int", "index": false, "name": "u_dport", "select": null, "suffixes": null},

{"datatype": "int", "index": false, "name": "u_protocol", "select": null, "suffixes": null},

{"datatype": "string", "index": false, "name": "u_flowtype", "select": null, "suffixes": null},

{"datatype": "string", "index": false, "name": "u_otherinfo", "select": null, "suffixes": null}]}
```

The schema for underlay data across pRouters is defined in the Contrail installation at:

`http://<host ip>:8081/analytics/table/StatTable.UFlowData.flow/schema`

Example: Flow Data Schema for Underlay

```
{"type": "STAT",

"columns": [

{"datatype": "string", "index": true, "name": "Source", "suffixes": null},

{"datatype": "int", "index": false, "name": "T", "suffixes": null},
```

```
{"datatype": "int", "index": false, "name": "CLASS(T)", "suffixes": null},

{"datatype": "int", "index": false, "name": "T=", "suffixes": null},

{"datatype": "int", "index": false, "name": "CLASS(T=)", "suffixes": null},

{"datatype": "uuid", "index": false, "name": "UUID", "suffixes": null},

{"datatype": "int", "index": false, "name": "COUNT(flow)", "suffixes": null},

{"datatype": "string", "index": true, "name": "name", "suffixes": ["flow.pifindex"]},

{"datatype": "int", "index": false, "name": "flow.pifindex", "suffixes": null},

{"datatype": "int", "index": false, "name": "SUM(flow.pifindex)", "suffixes": null},

{"datatype": "int", "index": false, "name": "CLASS(flow.pifindex)", "suffixes": null},

{"datatype": "int", "index": false, "name": "flow.sport", "suffixes": null},

{"datatype": "int", "index": false, "name": "SUM(flow.sport)", "suffixes": null},

{"datatype": "int", "index": false, "name": "CLASS(flow.sport)", "suffixes": null},

{"datatype": "int", "index": false, "name": "flow.dport", "suffixes": null},

{"datatype": "int", "index": false, "name": "SUM(flow.dport)", "suffixes": null},

{"datatype": "int", "index": false, "name": "CLASS(flow.dport)", "suffixes": null},

{"datatype": "int", "index": true, "name": "flow.protocol", "suffixes": ["flow.sport",
"flow.dport"]},

{"datatype": "int", "index": false, "name": "SUM(flow.protocol)", "suffixes": null},

{"datatype": "int", "index": false, "name": "CLASS(flow.protocol)", "suffixes": null},

{"datatype": "string", "index": true, "name": "flow.sip", "suffixes": null},

{"datatype": "string", "index": true, "name": "flow.dip", "suffixes": null},

{"datatype": "string", "index": true, "name": "flow.vlan", "suffixes": null},
```

```
{"datatype": "string", "index": false, "name": "flow.flowtype", "suffixes": null},

{"datatype": "string", "index": false, "name": "flow.otherinfo", "suffixes": null}]}
```

Example: Typical Query for Flow Map

The following is a typical query. Internally, the `analytics-api` performs a query into the `FlowRecordTable`, then into the `StatTable.UFlowData.flow`, to return list of `(prouter, pifindex)` pairs that give the underlay path taken for the given overlay flow.

```
FROM

OverlayToUnderlayFlowMap

SELECT

prouter, pifindex

WHERE

o_svn, o_sip, o_dvn, o_dip, o_sport, o_dport, o_protocol = <overlay flow>
```

Module Operations for Overlay Underlay Mapping

SNMP Collector Operation

The Contrail SNMP collector uses a Net-SNMP library to talk to a physical router or any SNMP agent. Upon receiving SNMP packets, the data is translated to the Python dictionary, and corresponding UVE objects are created. The UVE objects are then posted to the SNMP collector.

The SNMP module sleeps for some configurable period, then forks a collector process and waits for the process to complete. The collector process goes through a list of devices to be queried. For each device, it forks a greenlet task (Python coroutine), accumulates SNMP data, writes the summary to a JSON file, and exits. The parent process then reads the JSON file, creates UVEs, sends the UVEs to the collector, then goes to sleep again.

The pRouter UVE sent by the SNMP collector carries only the raw MIB information.

Example: pRouter Entry Carried in pRouter UVE

The definition below shows the `pRouterEntry` carried in the `pRouterUVE`. Additionally, an example `LldpTable` definition is shown.

The following create a virtual table as defined by:

```
http://<host ip>:8081/analytics/table/StatTable.UFlowData.flow/schema

struct LldpTable {

  1: LldpLocalSystemData lldpLocalSystemData

  2: optional list<LldpRemoteSystemsData> lldpRemoteSystemsData

}

struct PRouterEntry {

  1: string name (key="ObjectPRouter")

  2: optional bool deleted

  3: optional LldpTable lldpTable

  4: optional list<ArpTable> arpTable

  5: optional list<IfTable> ifTable

  6: optional list<IfXTable> ifXTable

  7: optional list<IfStats> ifStats (tags="name:.ifIndex")

  8: optional list<IpMib> ipMib

}

uve sandesh PRouterUVE {

  1: PRouterEntry data

}
```

Topology Module Operation

The topology module reads UVEs posted by the SNMP collector and computes the neighbor table, populating the table with remote system name, local and remote interface names, the remote type (pRouter or vRouter) and local and remote ifindices. The topology module sleeps for a while, reads UVEs, then computes the neighbor table and posts the UVE to the collector.

The pRouter UVE sent by the topology module carries the neighbor list, so the clients can put together all of the pRouter neighbor lists to compute the full topology.

The corresponding pRouter UVE definition is the following.

```
struct LinkEntry {

  1: string remote_system_name

  2: string local_interface_name

  3: string remote_interface_name

  4: RemoteType type

  5: i32 local_interface_index

  6: i32 remote_interface_index

}

struct PRouterLinkEntry {

  1: string name (key="ObjectPRouter")

  2: optional bool deleted

  3: optional list<LinkEntry> link_table

}

uve sandesh PRouterLinkUVE {

  1: PRouterLinkEntry data

}
```

IPFIX and sFlow Collector Operation

An IPFIX and sFlow collector has been implemented in the Contrail collector. The collector receives the IPFIX and sFlow samples and stores them as statistics samples in the analytics database.

Example: IPFIX sFlow Collector Data

The following definition shows the data stored for the statistics samples and the indices that can be used to perform queries.

```
struct UFlowSample {

  1: u64 pifindex

  2: string sip

  3: string dip

  4: u16 sport

  5: u16 dport

  6: u16 protocol

  7: u16 vlan

  8: string flowtype

  9: string otherinfo

}

struct UFlowData {

  1: string name (key="ObjectPRouterIP")

  2: optional bool deleted

  3: optional list<UFlowSample> flow
```

```
(tags="name:.pifindex, .sip, .dip, .protocol:.sport, .protocol:.dport, .vlan")

}
```

Troubleshooting Underlay Overlay Mapping

This section provides a variety of links where you can research errors that may occur with underlay overlay mapping.

System Logs

Logs for `contrail-snmp-collector` and `contrail-topology` are in the following locations on an installed Contrail system:

```
/var/log/contrail/contrail-snmp-collector-stdout.log
```

```
/var/log/contrail/contrail-topology.log
```

Introspect Utility

Use URLs of the following forms on your Contrail system to access the introspect utilities for SNMP data and for topology data.

- SNMP data introspect

  ```
  http://<host ip>:5920/Snh_SandeshUVECacheReq?x=PRouterEntry
  ```

- Topology data introspect

  ```
  http://<host ip>:5921/Snh_SandeshUVECacheReq?x=PRouterLinkEntry
  ```

Script to add pRouter Objects

The usual mechanism for adding pRouter objects to `contrail-config` is through Contrail UI. But you also have the ability to add these objects using the Contrail `vnc-api`. To add one pRouter, save the file with the name `cfg-snmp.py`, and then execute the command as shown:

```
python cfg-snmp.py
```

Example: Content for cfg-snmp.py

```
#!python

from vnc_api import vnc_api

from vnc_api.gen.resource_xsd import SNMPCredentials



vnc = vnc_api.VncApi('admin', 'abcde123', 'admin')

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-mx80-1')

apr.set_physical_router_management_ip('ip_address')

apr.set_physical_router_dataplane_ip(''ip_address')

apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2, v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-mx80-2')

apr.set_physical_router_management_ip('ip_address')

apr.set_physical_router_dataplane_ip('ip_address')

apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2, v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123'

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-ex3')

apr.set_physical_router_management_ip('source_ip')

apr.set_physical_router_dataplane_ip('source_ip'')
```

```
apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2, v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123'

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-ex2')

apr.set_physical_router_management_ip('ip_address')

apr.set_physical_router_dataplane_ip('ip_address')

apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2, v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123'
```

RELATED DOCUMENTATION

Understanding Contrail Analytics | **325**

Contrail Alerts | **326**

CHAPTER 14

Configuring Contrail Analytics

IN THIS CHAPTER

- Analytics Scalability | 353
- High Availability for Analytics | 354
- Role-Based Access Control for Analytics | 355
- System Log Receiver in Contrail Analytics | 356
- Sending Flow Messages to the Contrail System Log | 357
- More Efficient Flow Queries | 358
- Ceilometer Support in a Contrail Cloud | 359

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.

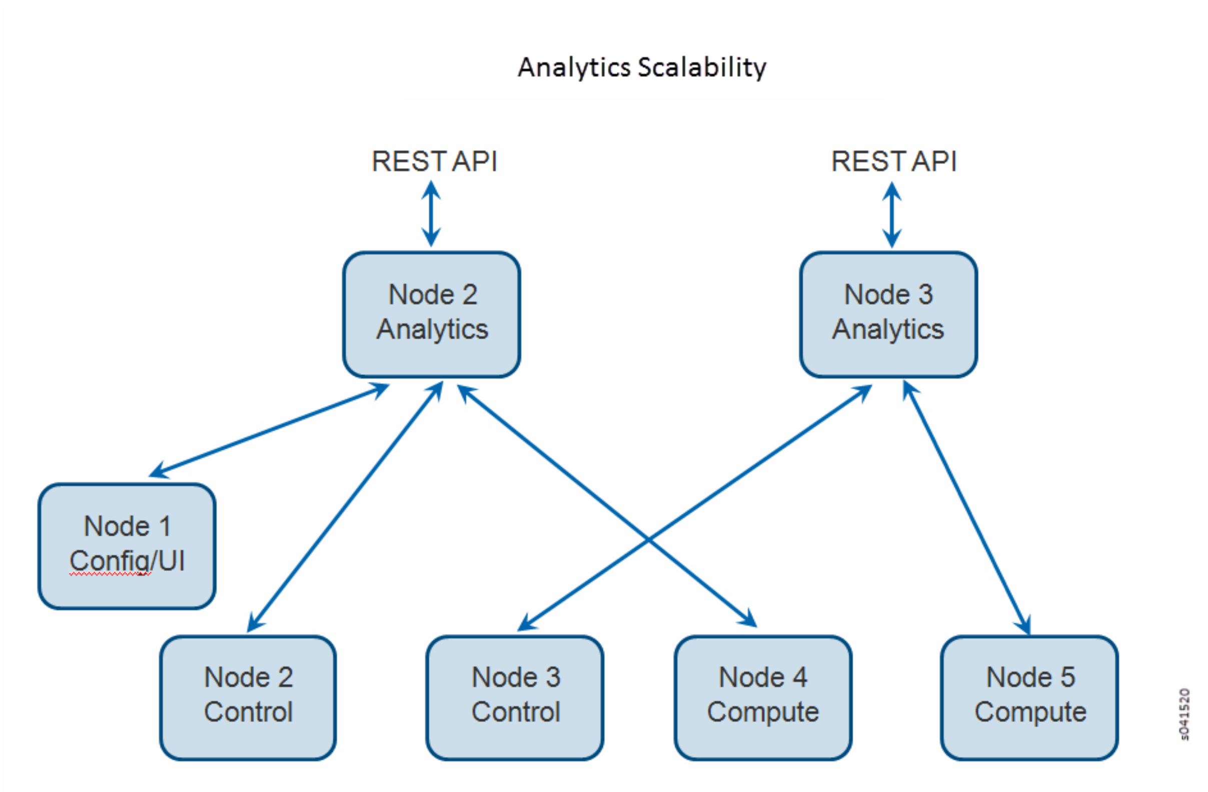
The following is the recommended locations for the various component roles of the Contrail system for a 5-node configuration.

- 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 123 on page 354 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 123: Analytics Scalability



The analytics nodes collect and store data and provide this data through various REST API queries. Scalability is provided for the control nodes, the compute nodes, and the REST API users, with the API output displayed in the Contrail user interface. As the number of control and compute nodes increase in the system, the analytics nodes can also be increased.

High Availability for Analytics

Contrail supports multiple instances of analytics for high availability and load balancing.

Contrail analytics provides two broad areas of functionality:

- **contrail-collector** —Receives status, logs, and flow information from all Contrail processing elements (for example, generators) and records them.

  Every generator is connected to one of the **contrail-collector** instances at any given time. If an instance fails (or is shut down), all the generators that are connected to it are automatically moved to another functioning instance, typically in a few seconds or less. Some messages may be lost during this movement. UVEs are resilient to message loss, so the state shown in a UVE is kept consistent to the state in the generator.

- **contrail-opserver** —Provides an external API to report UVEs and to query logs and flows.

  Each analytics component exposes a northbound REST API represented by the **contrail-opserver** service (port 8081) so that the failure of one analytics component or one **contrail-opserver** service should not impact the operation of other instances.

  These are the ways to manage connectivity to the **contrail-opserver** endpoints:

  - Periodically poll the **contrail-opserver** service on a set of analytics nodes to determine the list of functioning endpoints, then make API requests from one or more of the functioning endpoints.

  - The Contrail user interface makes use of the same northbound REST API to present dashboards, and reacts to any **contrail-opserver** high availability event automatically.

Role-Based Access Control for Analytics

The analytics API uses role-based access control (RBAC) to provide the ability to access UVE and query information based on the permissions of the user for the UVE or queried object.

Contrail Release 4.1 extends authenticated access so that tenants can view network monitoring information about the networks for which they have read permissions. RBAC for anaytics is a Beta feature in Contrail Release 4.1.

The analytics API can map query and UVE objects to configuration objects on which RBAC rules are applied, so that read permissions can be verified using the VNC API.

RBAC is applied to analytics in the following ways:

- For statistics queries, annotations are added to the Sandesh file so that indices and tags on statistics queries can be associated with objects and UVEs. These are used by the contrail-analytics-api to determine the object level read permissions.

- For flow and log queries, the object read permissions are evaluated for each AND term in the where query.

- For UVEs list queries (e.g. analytics/uve/virtual-networks/), the contrail-analytics-api gets a list of UVEs that have read permissions for a given token. For a UVE query for a specific resource (e.g. analytics/uves/virtual-network/vn1), contrail-analytics-api checks the object level read permissions using VNC API.

Tenants cannot view system logs and flow logs, those logs are displayed for cloud-admin roles only.

A non-admin user can see only non-global UVEs, including:

- virtual_network
- virtual_machine
- virtual_machine_interface
- service_instance
- service_chain
- tag
- firewall_policy
- firewall_rule
- address_group
- service_group
- aaplication_policy_set

In `/etc/contrail/contrail-analytics-api.conf`, in the section `DEFAULTS`, the parameter `aaa_mode` now supports `rbac` as one of the values.

System Log Receiver in Contrail Analytics

IN THIS SECTION

- Overview | 357
- Redirecting System Logs to Contrail Collector | 357
- Exporting Logs from Contrail Analytics | 357

Overview

The contrail-collector process on the Contrail Analytics node can act as a system log receiver.

Redirecting System Logs to Contrail Collector

You can enable the contrail-collector to receive system logs by giving a valid `syslog_port` as a command line option:

`--DEFAULT.syslog_port <arg>`

or by adding `syslog_port` in the DEFAULT section of the configuration file at `/etc/contrail/contrail-collector.conf` .

For nodes to send system logs to the contrail-collector, the system log configuration for the node should be set up to direct the system logs to contrail-collector.

Example

Add the following line in `/etc/rsyslog.d/50-default.conf` on an Ubuntu system to redirect the system logs to contrail-collector.

```
*.* @<collector_ip>:<collector_syslog_port> :: @ for udp, @@ for tcp
```

The logs can be retrieved by using Contrail tool, either by using the contrail-logs utility on the analytics node or by using the Contrail user interface on the system log query page.

Exporting Logs from Contrail Analytics

You can also export logs stored in Contrail analytics to another system log receiver by using the `contrail-logs` utility.

The contrail-logs utility can take these options: `--send-syslog`, `--syslog-server`, `--syslog-port`, to query Contrail analytics, then send the results as system logs to a system log server. This is an on-demand command, one can write a cron job or a job that continuously invokes `contrail-logs` to achieve continuous sending of logs to another system log server.

Sending Flow Messages to the Contrail System Log

The `contrail-vrouter-agent` can be configured to send flow messages and other messages to the system log (syslog). To send flow messages to syslog, configure the following parameters in `/etc/contrail/contrail-vrouter-agent.conf`.

The following parameters are under the section `DEFAULT`:

- `log_flow=1`—Enables logging of all flow messages.
- `use_syslog=1`—Enables sending of all messages, including flow messages, to syslog.
- `syslog_facility=LOG_LOCAL0`—Enables sending messages from the `contrail-vrouter-agent` to the syslog, using the facility `LOCAL0`. You can configure `LOCAL0` to your required facility.
- `log_level=SYS_INFO`—Changes the logging level of `contrail-vrouter-agent` to INFO.

If syslog is enabled, flow messages are *not* sent to Contrail Analytics because the two destinations are mutually exclusive.

Flow log sampling settings apply regardless of the flow log destination specified. If sampling is enabled, the syslog messages will be sampled using the same rules that would apply to Contrail Analytics. If non-sampled flow data is required, sampling must be disabled by means of configuration settings.

Flow events for termination will include both the appropriate tear-down fields and the appropriate setup fields.

The flow messages will be sent to the syslog with a severity of INFO.

The user can configure the remote system log (`rsyslog`) on the compute node to send syslog messages with facility LOCAL0, severity of INFO (and lower), to the remote syslog server. Messages with a higher severity than INFO can be logged to a local file to allow for debugging.

Flow messages appear in the syslog in a format similar to the following log example:

```
May 24 14:40:13 a7s10 contrail-vrouter-agent[29930]: 2016-05-24 Tue 14:40:13:921.098 PDT a7s10 [Thread
139724471654144, Pid 29930]: [SYS_INFO]: FlowLogDataObject: flowdata= [ [ [ flowuuid = 7ea8bf8f-b827-496e-
b93e-7622a0c8eeea direction_ing = 1 sourcevn = default-domain:mock-gen-test:vn8 sourceip = 1.0.0.9 destvn =
default-domain:mock-gen-test:vn58 destip = 1.0.0.59 protocol = 1 sport = -29520 dport = 20315 setup_time =
1464125225556930 bytes = 1035611592 packets = 2024830 diff_bytes = 27240 diff_packets = 40 ], ] ]
```

NOTE: Several individual flow messages might be packed into a single syslog message for improved efficiency.

More Efficient Flow Queries

Flow queries are now analyzed on a 7-tuple basis, enabling more efficient flow queries by focusing on elements more important for analysis, and de-emphasizing lesser elements. More efficient queries enable load reduction and allow application of security policy.

An enhanced security framework is implemented to manage connectivity between workloads, or VMIs. Each VMI is tagged with the attributes of Deployment, App, Tier, and Site, and the user specifies security policies for VMIs using the values of these tags. Contrail can analyze the traffic flow between groups of VMI, where groups are categorized according to one or more values of the tags.

The existing FlowLogData is replaced by SessionEndpointData, which is a combination of the local VMI tags and VNs, the security policy and security rule, and route attributes for the remote endpoint. A SessionAggregate map and counts both enable traffic analysis within and across security policies by means of session sampling and session aggregate counts.

The flow export feature is disabled by default. Until the session_export_rate is set explicitly, flow queries will not return any results regardless of the traffic. To use this feature, set the session export rate in the Contrail WebUI at **Config->Global Config->Forwarding Options**.

Ceilometer Support in a Contrail Cloud

IN THIS SECTION

- Overview | 359
- Ceilometer Details | 360
- Verification of Ceilometer Operation | 360
- Contrail Ceilometer Plugin | 363
- Ceilometer Installation and Provisioning | 366

Ceilometer is an OpenStack feature that provides an infrastructure for collecting SDN metrics from OpenStack projects. The metrics can be used by various rating engines to transform events into billable items. The Ceilometer collection process is sometimes referred to as “metering”. The Ceilometer service provides data that can be used by platforms that provide metering, tracking, billing, and similar services. This topic describes how to configure the Ceilometer service for Contrail.

Overview

Contrail Release 2.20 and later supports the OpenStack Ceilometer service, on the OpenStack Juno release on Ubuntu 14.04.1 LTS.

The prerequisites for installing Ceilometer are:

- Contrail Cloud installation

- Provisioned using **enable_ceilometer = True** in the **provisioning** file.

> **NOTE**: Ceilometer services are only installed on the first OpenStack controller node and do not support high availability in Contrail Release 2.20.

Ceilometer Details

Ceilometer is used to reliably collect measurements of the utilization of the physical and virtual resources comprising deployed clouds, persist these data for subsequent retrieval and analysis, and trigger actions when defined criteria are met.

The Ceilometer architecture consists of:

Polling agent	Agent designed to poll OpenStack services and build meters. The polling agents are also run on the compute nodes in addition to the OpenStack controller.
Notification agent	Agent designed to listen to notifications on message queue and convert them to events and samples.
Collector	Gathers and records event and metering data created by the notification and polling agents.
API server	Provides a REST API to query and view data recorded by the collector service.
Alarms	Daemons to evaluate and notify based on defined alarming rules.
Database	Stores the metering data, notifications, and alarms. The supported databases are MongoDB, SQL-based databases compatible with SQLAlchemy, and HBase. The recommended database is MongoDB, which has been thoroughly tested with Contrail and deployed on a production scale.

Verification of Ceilometer Operation

The Ceilometer services are named slightly differently on the Ubuntu and RHEL Server 7.0.

On Ubuntu, the service names are:

Polling agent	`ceilometer-agent-central` and `ceilometer-agent-compute`
Notification agent	`ceilometer-agent-notification`
Collector	`ceilometer-collector`

API Server `ceilometer-api`

Alarms `ceilometer-alarm-evaluator` and `ceilometer-alarm-notifier`

On RHEL Server 7.0, the service names are:

Polling agent `openstack-ceilometer-central` and `openstack-ceilometer-compute`

Notification agent `openstack-ceilometer-notification`

Collector `openstack-ceilometer-collector`

API server `openstack-ceilometer-api`

Alarms `openstack-ceilometer-alarm-evaluator` and `openstack-ceilometer-alarm-notifier`

To verify the Ceilometer installation, users can verify that the Ceilometer services are up and running by using the `openstack-status` command.

For example, using the **openstack-status** command on an all-in-one node running Ubuntu 14.04.1 LTS with release 2.2 of Contrail installed shows the following Ceilometer services as active:

```
== Ceilometer services ==
ceilometer-api:              active
ceilometer-agent-central:    active
ceilometer-agent-compute:    active
ceilometer-collector:        active
ceilometer-alarm-notifier:   active
ceilometer-alarm-evaluator:  active
ceilometer-agent-notification:active
```

You can issue the `ceilometer meter-list` command on the OpenStack controller node to verify that meters are being collected, stored, and reported via the REST API. The following is an example of the output:

```
user@host:~# (source /etc/contrail/openstackrc; ceilometer meter-list)
+------------------------------+------------+---------+--------------------------------------
+----------------------------------+----------------------------------+
| Name                         | Type       | Unit    | Resource ID                          |
User ID                          | Project ID                       |
+------------------------------+------------+---------+--------------------------------------
+----------------------------------+----------------------------------+
| ip.floating.receive.bytes    | cumulative | B       | a726f93a-65fa-4cad-828b-54dbfcf4a119 |
None                             | None                             |
```

```
| ip.floating.receive.packets  | cumulative | packet  | a726f93a-65fa-4cad-828b-54dbfcf4a119 |
None                              | None                              |
| ip.floating.transmit.bytes   | cumulative | B       | a726f93a-65fa-4cad-828b-54dbfcf4a119 |
None                              | None                              |
| ip.floating.transmit.packets | cumulative | packet  | a726f93a-65fa-4cad-828b-54dbfcf4a119 |
None                              | None                              |
| network                      | gauge      | network | 7fa6796b-756e-4320-9e73-87d4c52ecc83 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network                      | gauge      | network | 9408e287-d3e7-41e2-89f0-5c691c9ca450 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network                      | gauge      | network | b3b72b98-f61e-4e1f-9a9b-84f4f3ddec0b |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network                      | gauge      | network | cb829abd-e6a3-42e9-a82f-0742db55d329 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create               | delta      | network | 7fa6796b-756e-4320-9e73-87d4c52ecc83 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create               | delta      | network | 9408e287-d3e7-41e2-89f0-5c691c9ca450 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create               | delta      | network | b3b72b98-f61e-4e1f-9a9b-84f4f3ddec0b |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create               | delta      | network | cb829abd-e6a3-42e9-a82f-0742db55d329 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| port                         | gauge      | port    | 0d401d96-c2bf-4672-abf2-880eecf25ceb |
01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port                         | gauge      | port    | 211b94a4-581d-45d0-8710-c6c69df15709 |
01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port                         | gauge      | port    | 2287ce25-4eef-4212-b77f-3cf590943d36 |
01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port.create                  | delta      | port    | f62f3732-222e-4c40-8783-5bcbc1fd6a1c |
01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port.create                  | delta      | port    | f8c89218-3cad-48e2-8bd8-46c1bc33e752 |
01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port.update                  | delta      | port    | 43ed422d-b073-489f-877f-515a3cc0b8c4 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet                       | gauge      | subnet  | 09105ed1-1654-4b5f-8c12-f0f2666fa304 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet                       | gauge      | subnet  | 4bf00aac-407c-4266-a048-6ff52721ad82 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet.create                | delta      | subnet  | 09105ed1-1654-4b5f-8c12-f0f2666fa304 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet.create                | delta      | subnet  | 4bf00aac-407c-4266-a048-6ff52721ad82 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
```

```
+------------------------------+------------+---------+-------------------------------------
+----------------------------------+----------------------------------+
```

NOTE: The `ceilometer meter-list` command lists the meters only if images have been created, or instances have been launched, or if subnet, port, floating IP addresses have been created, otherwise the meter list is empty. You also need to source the **/etc/contrail/openstackrc** file when executing the command.

Contrail Ceilometer Plugin

The Contrail Ceilometer plugin adds the capability to meter the traffic statistics of floating IP addresses in Ceilometer. The following meters for each floating IP resource are added by the plugin in Ceilometer.

```
ip.floating.receive.bytes
ip.floating.receive.packets
ip.floating.transmit.bytes
ip.floating.transmit.packets
```

The Contrail Ceilometer plugin configuration is done in the **/etc/ceilometer/pipeline.yaml** file when Contrail is installed by the Fabric provisioning scripts.

The following example shows the configuration that is added to the file:

```
sources:
    - name: contrail_source
      interval: 600
      meters:
        - "ip.floating.receive.packets"
        - "ip.floating.transmit.packets"
        - "ip.floating.receive.bytes"
        - "ip.floating.transmit.bytes"
      resources:
          - contrail://<IP-address-of-Contrail-Analytics-Node>:8081
      sinks:
        - contrail_sink
sinks:
    - name: contrail_sink
      publishers:
```

```
      - rpc://
    transformers:
```

The following example shows the Ceilometer meter list output for the floating IP meters:

```
+-------------------------------+------------+-----------
+----------------------------------------------------------------------
+----------------------------------+----------------------------------+
| Name                          | Type       | Unit       | Resource
ID                                                   | User ID
| Project ID                       |
+-------------------------------+------------+-----------
+----------------------------------------------------------------------
+----------------------------------+----------------------------------+
| ip.floating.receive.bytes     | cumulative | B          | 451c93eb-
e728-4ba1-8665-6e7c7a8b49e2                                  | None
| None                             |
| ip.floating.receive.bytes     | cumulative | B          | 9cf76844-8f09-4518-a09e-
e2b8832bf894                              | None                             |
None                             |
| ip.floating.receive.packets   | cumulative | packet     | 451c93eb-
e728-4ba1-8665-6e7c7a8b49e2                                  | None
| None                             |
| ip.floating.receive.packets   | cumulative | packet     | 9cf76844-8f09-4518-a09e-
e2b8832bf894                              | None                             |
None                             |
| ip.floating.transmit.bytes    | cumulative | B          | 451c93eb-
e728-4ba1-8665-6e7c7a8b49e2                                  | None
| None                             |
| ip.floating.transmit.bytes    | cumulative | B          | 9cf76844-8f09-4518-a09e-
e2b8832bf894                              | None                             |
None                             |
| ip.floating.transmit.packets  | cumulative | packet     | 451c93eb-
e728-4ba1-8665-6e7c7a8b49e2                                  | None
| None                             |
| ip.floating.transmit.packets  | cumulative | packet     | 9cf76844-8f09-4518-a09e-
e2b8832bf894                              | None                             |
None                             |
```

In the meter -list output, the Resource ID refers to the floating IP.

The following example shows the output from the `ceilometer resource-show -r 451c93eb-e728-4ba1-8665-6e7c7a8b49e2` command:

```
+-------------+----------------------------------------------------------------------+
| Property    | Value                                                                |
+-------------+----------------------------------------------------------------------+
| metadata    | {u'router_id': u'None', u'status': u'ACTIVE', u'tenant_id':          |
|             | u'ceed483222f9453ab1d7bcdd353971bc', u'floating_network_id':         |
|             | u'6d0cca50-4be4-4b49-856a-6848133eb970', u'fixed_ip_address':        |
|             | u'2.2.2.4', u'floating_ip_address': u'3.3.3.4', u'port_id': u'c6ce2abf- |
|             | ad98-4e56-ae65-ab7c62a67355', u'id':                                 |
|             | u'451c93eb-e728-4ba1-8665-6e7c7a8b49e2', u'device_id':               |
|             | u'00953f62-df11-4b05-97ca-30c3f6735ffd'}                             |
| project_id  | None                                                                 |
| resource_id | 451c93eb-e728-4ba1-8665-6e7c7a8b49e2                                 |
| source      | openstack                                                            |
| user_id     | None                                                                 |
+-------------+----------------------------------------------------------------------+
```

The following example shows the output from the `ceilometer statistics` command and the `ceilometer sample-list` command for the **ip.floating.receive.packets** meter:

```
+--------+----------------------------+----------------------------+-------+-----+-------
+--------+----------------+------------+----------------------------+----------------------------
+
| Period | Period Start               | Period End                 | Count | Min | Max   |
Sum    | Avg            | Duration   | Duration Start             | Duration End               |
+--------+----------------------------+----------------------------+-------+-----+-------
+--------+----------------+------------+----------------------------+----------------------------
+
| 0      | 2015-02-13T19:50:40.795000 | 2015-02-13T19:50:40.795000 | 2892  | 0.0 | 325.0 |
1066.0 | 0.368603042877 | 439069.674 | 2015-02-13T19:50:40.795000 | 2015-02-18T21:48:30.469000 |
+--------+----------------------------+----------------------------+-------+-----+-------
+--------+----------------+------------+----------------------------+----------------------------
+

+--------------------------------------+------------------------------+------------+--------
+--------+----------------------------+
| Resource ID                          | Name                         | Type       | Volume |
Unit   | Timestamp                  |
+--------------------------------------+------------------------------+------------+--------
```

```
+--------+----------------------------+
| 9cf76844-8f09-4518-a09e-e2b8832bf894 | ip.floating.receive.packets | cumulative | 208.0  |
packet | 2015-02-18T21:48:30.469000 |
| 451c93eb-e728-4ba1-8665-6e7c7a8b49e2 | ip.floating.receive.packets | cumulative | 325.0  |
packet | 2015-02-18T21:48:28.354000 |
| 9cf76844-8f09-4518-a09e-e2b8832bf894 | ip.floating.receive.packets | cumulative | 0.0    |
packet | 2015-02-18T21:38:30.350000 |
```

Ceilometer Installation and Provisioning

There are two scenarios possible for Contrail Ceilometer plugin installation.

1. If you install your own OpenStack distribution, you can install the Contrail Ceilometer plugin on the OpenStack controller node.

2. When using Contrail Cloud services, the Ceilometer controller services are installed and provisioned as part of the OpenStack controller node and the compute agent service is installed as part of the compute node when `enable_ceilometer` is set as `True` in the cluster **config** or **testbed** files.

CHAPTER 15

Using Contrail Analytics to Monitor and Troubleshoot the Network

IN THIS CHAPTER

- Monitoring the System | 367
- Debugging Processes Using the Contrail Introspect Feature | 371
- Monitor > Infrastructure > Dashboard | 376
- Monitor > Infrastructure > Control Nodes | 380
- Monitor > Infrastructure > Virtual Routers | 391
- Monitor > Infrastructure > Analytics Nodes | 405
- Monitor > Infrastructure > Config Nodes | 413
- Monitor > Networking | 417
- Query > Flows | 429
- Query > Logs | 439
- Example: Debugging Connectivity Using Monitoring for Troubleshooting | 446

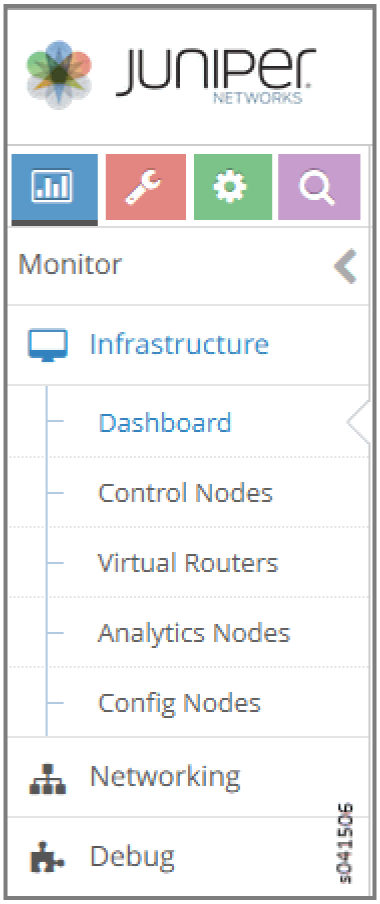
Monitoring the System

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 124 on page 368

Figure 124: Monitor Menu



See Table 44 on page 368 for descriptions of the items available under each of the menu options from the **Monitor** icon.

Table 44: Monitor Menu Options

Option	Description
Infrastructure > Dashboard	Shows “at-a-glance” status view of the infrastructure components, including the numbers of virtual routers, control nodes, analytics nodes, and config nodes currently operational, and a bubble chart of virtual routers showing the CPU and memory utilization, log messages, system information, and alerts. See "Monitor > Infrastructure > Dashboard" on page 376.

Table 44: Monitor Menu Options *(Continued)*

Option	Description
Infrastructure > Control Nodes	View a summary for all control nodes in the system, and for each control node, view: • Graphical reports of memory usage and average CPU load. • Console information for a specified time period. • A list of all peers with details about type, ASN, and the like. • A list of all routes, including next hop, source, local preference, and the like. See "Monitor > Infrastructure > Control Nodes" on page 380.
Infrastructure > Virtual Routers	View a summary of all vRouters in the system, and for each vRouter, view: • Graphical reports of memory usage and average CPU load. • Console information for a specified time period. • A list of all interfaces with details such as label, status, associated network, IP address, and the like. • A list of all associated networks with their ACLs and VRFs. • A list of all active flows with source and destination details, size, and time. See "Monitor > Infrastructure > Virtual Routers" on page 391.
Infrastructure > Analytics Nodes	View activity for the analytics nodes, including memory and CPU usage, analytics host names, IP address, status, and more. See "Monitor > Infrastructure > Analytics Nodes" on page 405.
Infrastructure > Config Nodes	View activity for the config nodes, including memory and CPU usage, config host names, IP address, status, and more. See "Monitor > Infrastructure > Config Nodes" on page 413.

Table 44: Monitor Menu Options *(Continued)*

Option	Description
Networking > Networks	For all virtual networks for all projects in the system, view graphical traffic statistics, including: • Total traffic in and out. • Inter VN traffic in and out. • The most active ports, peers, and flows for a specified duration. • All traffic ingress and egress from connected networks, including their attached policies. See "Monitor > Networking" on page 417.
Networking > Dashboard	For all virtual networks for all projects in the system, view graphical traffic statistics, including: • Total traffic in and out. • Inter VN traffic in and out. You can view the statistics in varying levels of granularity, for example, for a whole project, or for a single network. See "Monitor > Networking" on page 417.
Networking > Projects	View essential information about projects in the system including name, associated networks, and traffic in and out.
Networking > Networks	View essential information about networks in the system including name and traffic in and out.
Networking > Instances	View essential information about instances in the system including name, associated networks, interfaces, vRouters, and traffic in and out.

Table 44: Monitor Menu Options *(Continued)*

Option	Description
Debug > Packet Capture	• Add and manage packet analyzers. • Attach packet captures and configure their details. • View a list of all packet analyzers in the system and the details of their configurations, including source and destination networks, ports, and IP addresses.

RELATED DOCUMENTATION

Monitor > Infrastructure > Dashboard | 376
Monitor > Infrastructure > Control Nodes | 380
Monitor > Infrastructure > Virtual Routers | 391
Monitor > Networking | 417
Query > Logs | 439
Query > Flows | 429

Debugging Processes Using the Contrail Introspect Feature

This topic describes how to use the Sandesh infrastructure and the Contrail Introspect feature to debug processes.

Introspect is a mechanism for taking a program object and querying information about it.

Sandesh is the name of a unified infrastructure in the Contrail Virtual Networking solution.

Sandesh is a way for the Contrail daemons to provide a request-response mechanism. Requests and responses are defined in Sandesh format and the Sandesh compiler generates code to process the requests and send responses.

Sandesh also provides a way to use a Web browser to send Sandesh requests to a Contrail daemon and get the Sandesh responses. This feature is used to debug processes by looking into the operational status of the daemons.

Each Contrail daemon starts an HTTP server, with the following page types:

- The main index.html listing all Sandesh modules and the links to them.
- Sandesh module pages that present HTML forms for each Sandesh request.
- XML-based dynamically-generated pages that display Sandesh responses.
- An automatically generated page that shows all code needed for rendering and all HTTP server-client interactions.

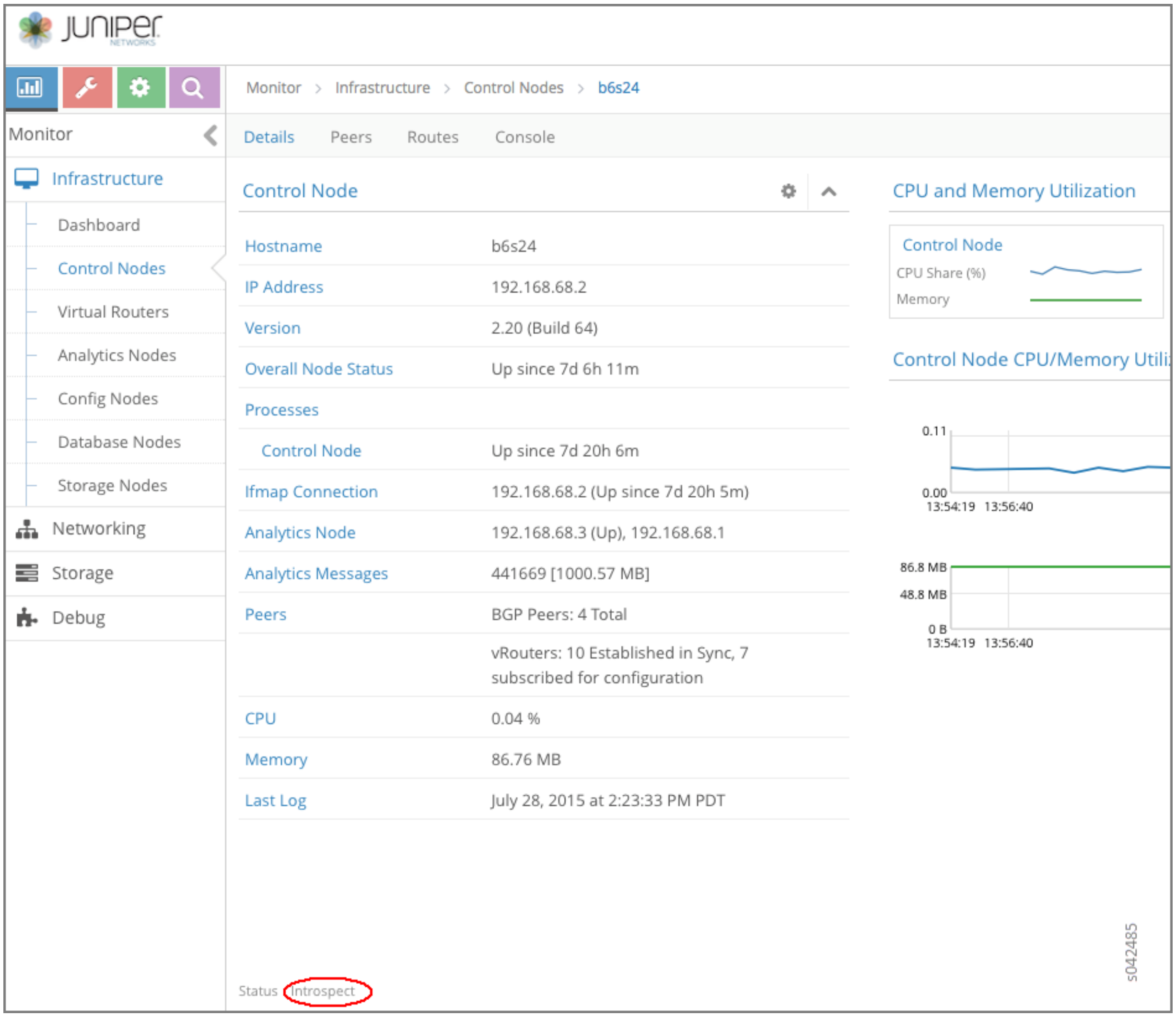
You can display the HTTP introspect of a Contrail daemon directly by accessing the following Introspect ports:

- *<controller-ip>*:8083. This port displays the *contrail-control* introspect port.
- *<compute-ip>*:8085 This port displays the *contrail-vrouter-agent* introspect port.

Another way to launch the Introspect page is by browsing to a particular node page using the Contrail Web user interface.

Figure 125 on page 373 shows the contrail-control infrastructure page. Notice the Introspect link at the bottom of the Control Nodes Details tab window.

Figure 125: Control Nodes Details Tab Window



The following are the Sandesh modules for the Contrail control process (contrail-control) Introspect port.

- bgp_peer.xml
- control_node.xml
- cpuinfo.xml
- discovery_client_stats.xml
- ifmap_log.xml
- ifmap_server_show.xml
- rtarget_group.xml

- sandesh_trace.xml
- sandesh_uve.xml
- service_chaining.xml
- static_route.xml
- task.xml
- xmpp_server.xml

Figure 126 on page 374 shows the Controller Introspect window.

Figure 126: Controller Introspect Window

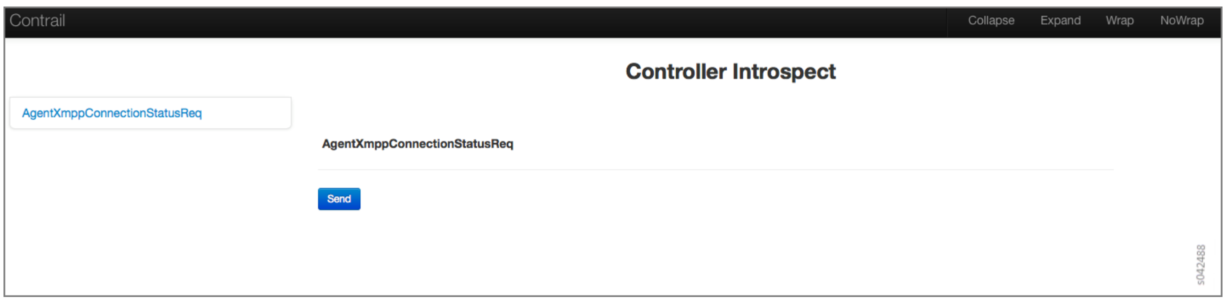


Figure 127 on page 374 shows an example of the BGP Peer (bgp_peer.xml) Introspect page.

Figure 127: BGP Peer Introspect Page

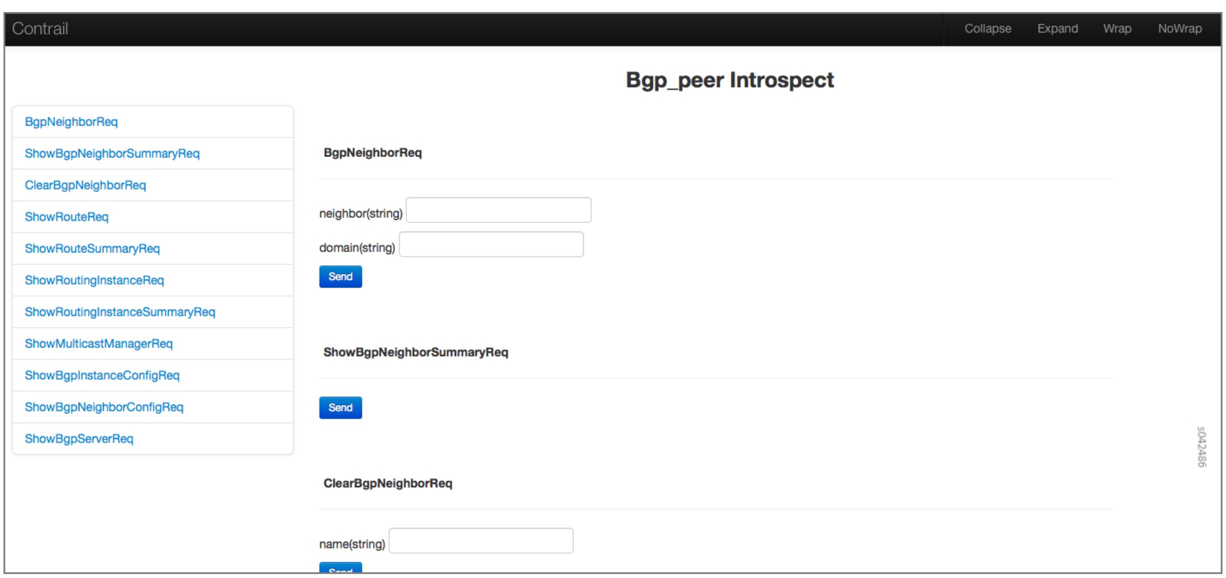
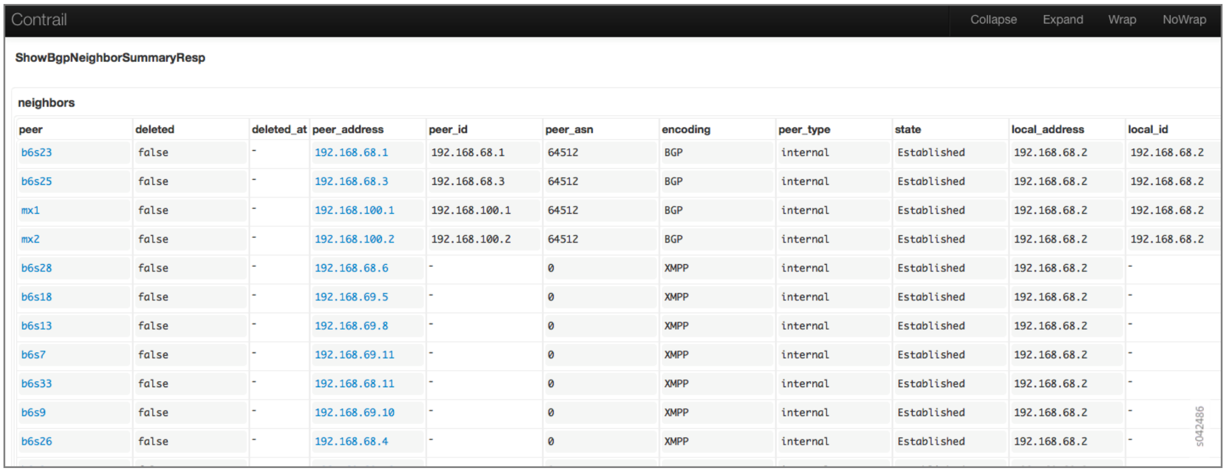


Figure 128 on page 375 shows an example of the BGP Neighbor Summary Introspect page.

Figure 128: BGP Neighbor Summary Introspect Page


Contrail Collapse Expand Wrap NoWrap

ShowBgpNeighborSummaryResp

neighbors

peer	deleted	deleted_at	peer_address	peer_id	peer_asn	encoding	peer_type	state	local_address	local_id
b6s23	false	-	192.168.68.1	192.168.68.1	64512	BGP	internal	Established	192.168.68.2	192.168.68.2
b6s25	false	-	192.168.68.3	192.168.68.3	64512	BGP	internal	Established	192.168.68.2	192.168.68.2
mx1	false	-	192.168.100.1	192.168.100.1	64512	BGP	internal	Established	192.168.68.2	192.168.68.2
mx2	false	-	192.168.100.2	192.168.100.2	64512	BGP	internal	Established	192.168.68.2	192.168.68.2
b6s28	false	-	192.168.68.6	-	0	XMPP	internal	Established	192.168.68.2	-
b6s18	false	-	192.168.69.5	-	0	XMPP	internal	Established	192.168.68.2	-
b6s13	false	-	192.168.69.8	-	0	XMPP	internal	Established	192.168.68.2	-
b6s7	false	-	192.168.69.11	-	0	XMPP	internal	Established	192.168.68.2	-
b6s33	false	-	192.168.68.11	-	0	XMPP	internal	Established	192.168.68.2	-
b6s9	false	-	192.168.69.10	-	0	XMPP	internal	Established	192.168.68.2	-
b6s26	false	-	192.168.68.4	-	0	XMPP	internal	Established	192.168.68.2	-

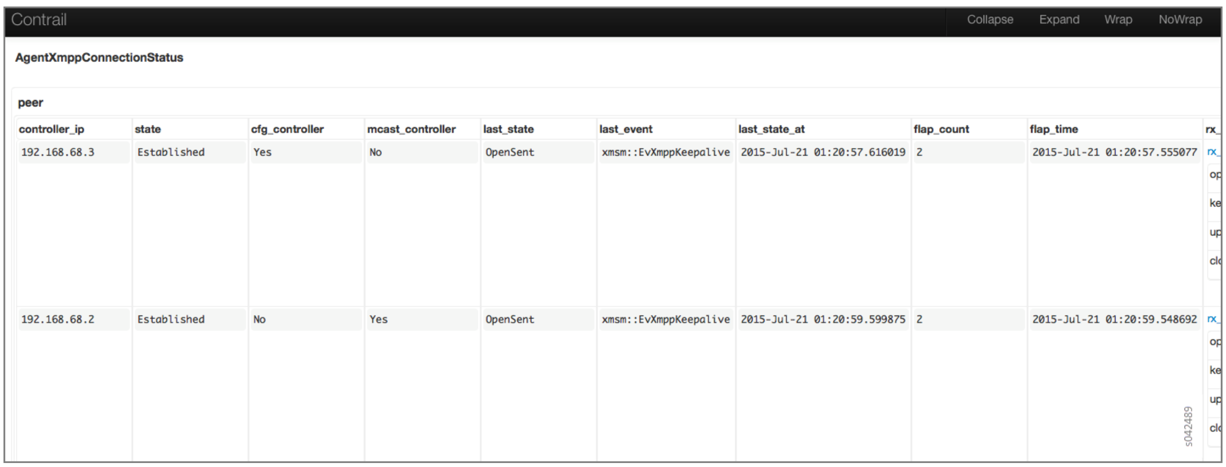
The following are the Sandesh modules for the Contrail vRouter agent (**contrail-vrouter-agent**) Introspect port.

- agent.xml
- agent_stats_interval.xml
- cfg.xml
- controller.xml
- cpuinfo.xml
- diag.xml
- discovery_client_stats.xml
- flow_stats_interval.xml
- ifmap_agent.xml
- kstate.xml
- multicast.xml
- pkt.xml
- port_ipc.xml
- sandesh_trace.xml

- sandesh_uve.xml
- services.xml
- stats_interval.xml
- task.xml
- xmpp_server.xml

Figure 129 on page 376 shows an example of the Agent (agent.xml) Introspect page.

Figure 129: Agent Introspect Page



Monitor > Infrastructure > Dashboard

IN THIS SECTION

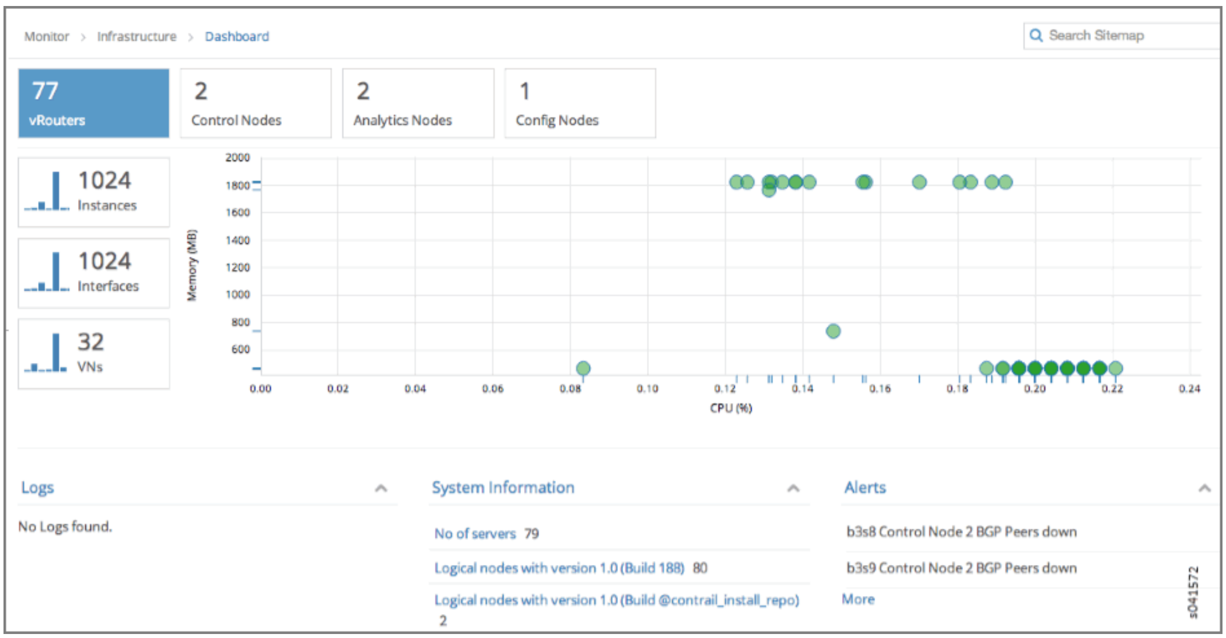
- Monitor Dashboard | 377
- Monitor Individual Details from the Dashboard | 377
- Using Bubble Charts | 378
- Color-Coding of Bubble Charts | 379

Use **Monitor > Infrastructure > Dashboard** to get an "at-a-glance" view of the system infrastructure components, including the numbers of virtual routers, control nodes, analytics nodes, and config nodes currently operational, a bubble chart of virtual routers showing the CPU and memory utilization, log messages, system information, and alerts.

Monitor Dashboard

Click **Monitor > Infrastructure > Dashboard** on the left to view the **Dashboard.** See Figure 130 on page 377.

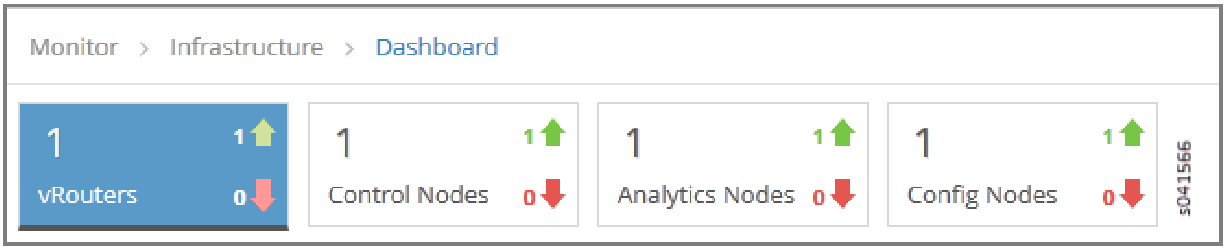
Figure 130: Monitor > Infrastructure > Dashboard



Monitor Individual Details from the Dashboard

Across the top of the **Dashboard** screen are summary boxes representing the components of the system that are shown in the statistics. See Figure 131 on page 378. 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 131: Dashboard Summary Boxes



Detailed information about monitoring each of the areas represented by the boxes is provided in the links in Table 45 on page 378.

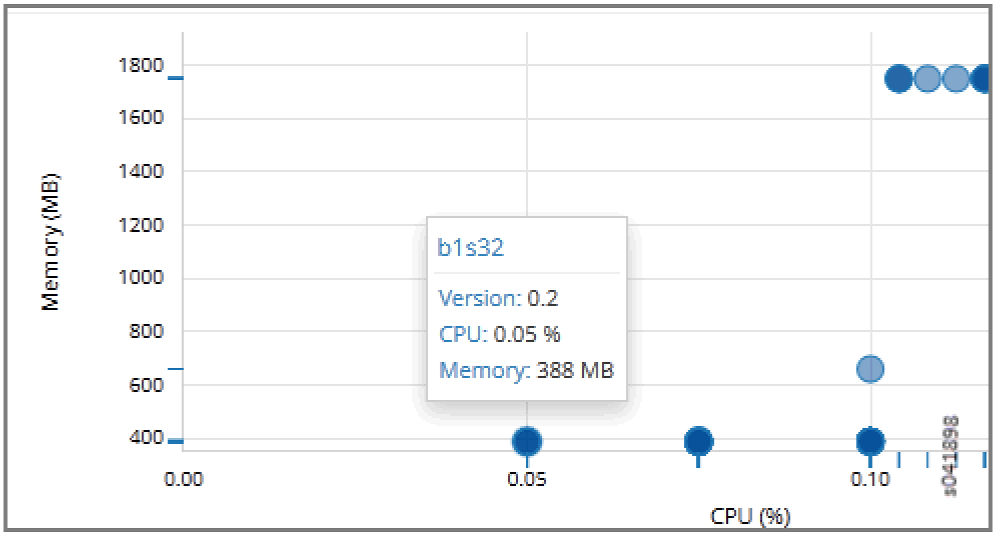
Table 45: Dashboard Summary Boxes

Box	For More Information
vRouters	"Monitor > Infrastructure > Virtual Routers" on page 391
Control Nodes	"Monitor > Infrastructure > Control Nodes" on page 380
Analytics Nodes	"Monitor > Infrastructure > Analytics Nodes" on page 405
Config Nodes	"Monitor > Infrastructure > Config Nodes" on page 413

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 132 on page 379. You can click through the summary information to get more details about the component.

Figure 132: 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 | 391
Monitor > Infrastructure > Control Nodes | 380
Monitor > Infrastructure > Analytics Nodes | 405
Monitor > Infrastructure > Config Nodes | 413

Monitor > Infrastructure > Control Nodes

IN THIS SECTION

- Monitor Control Nodes Summary | 380
- Monitor Individual Control Node Details | 381
- Monitor Individual Control Node Console | 383
- Monitor Individual Control Node Peers | 386
- Monitor Individual Control Node Routes | 388

Use **Monitor > Infrastructure > Control Nodes** to gain insight into usage statistics for control nodes.

Monitor Control Nodes Summary

Select **Monitor > Infrastructure > Control Nodes** to see a graphical chart of average memory usage versus average CPU percentage usage for all control nodes in the system. Also on this screen is a list of all control nodes in the system. See Figure 133 on page 380. See Table 46 on page 381 for descriptions of the fields on this screen.

Figure 133: Control Nodes Summary

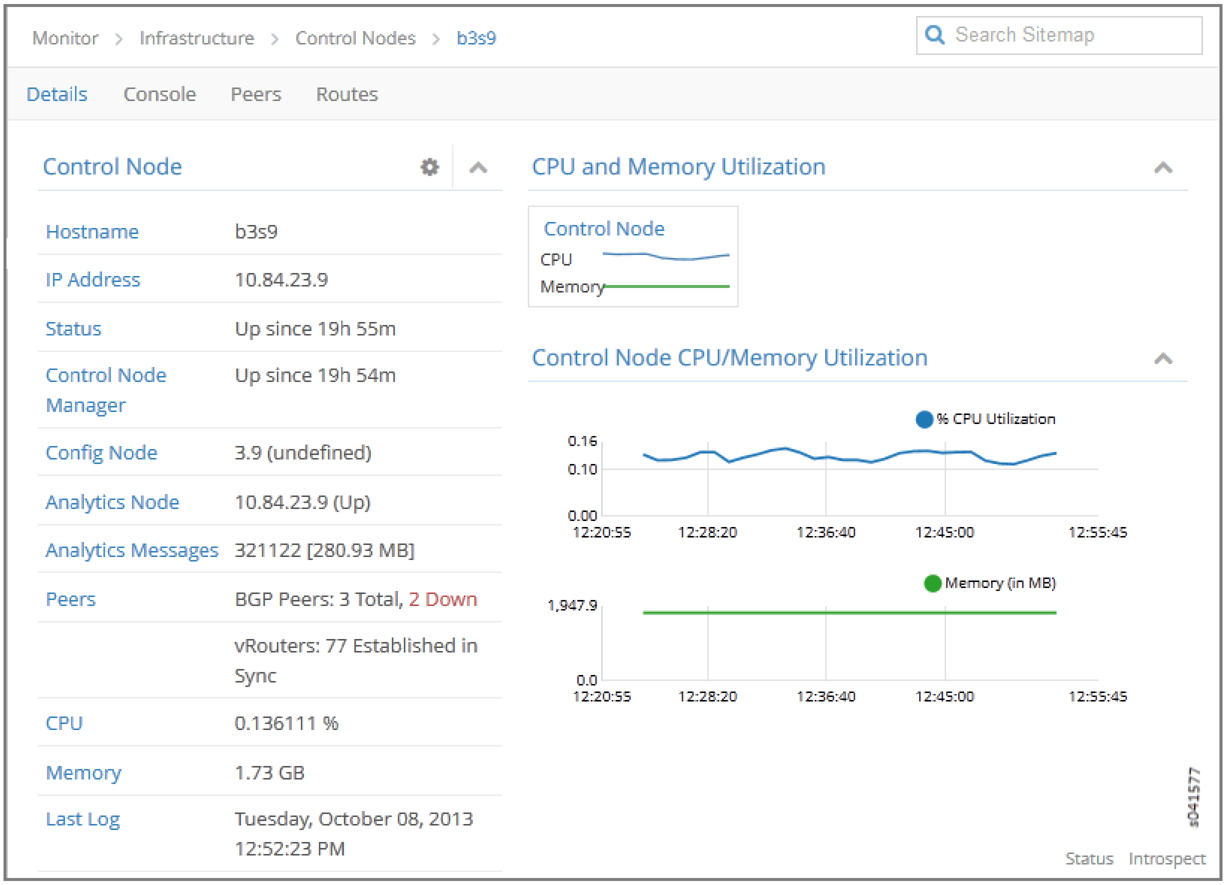
Table 46: Control Nodes Summary Fields

Field	Description
Host name	The name of the control node.
IP Address	The IP address of the control node.
Version	The software version number that is installed on the control node.
Status	The current operational status of the control node — Up or Down.
CPU (%)	The CPU percentage currently in use by the selected control node.
Memory	The memory in MB currently in use and the total memory available for this control node.
Total Peers	The total number of peers for this control node.
Established in Sync Peers	The total number of peers in sync for this control node.
Established in Sync vRouters	The total number of vRouters in sync for this control node.

Monitor Individual Control Node Details

Click the name of any control nodes listed under the **Control Nodes** title to view an array of graphical reports of usage and numerous details about that node. There are several tabs available to help you probe into more details about the selected control node. The first tab is the **Details** tab; see Figure 134 on page 382.

Figure 134: 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 47 on page 382 for descriptions of the fields on this tab.

Table 47: Individual Control Node—Details Tab Fields

Field	Description
Hostname	The host name defined for this control node.
IP Address	The IP address of the selected node.
Status	The operational status of the control node.
Control Node Manager	The operational status of the control node manager.

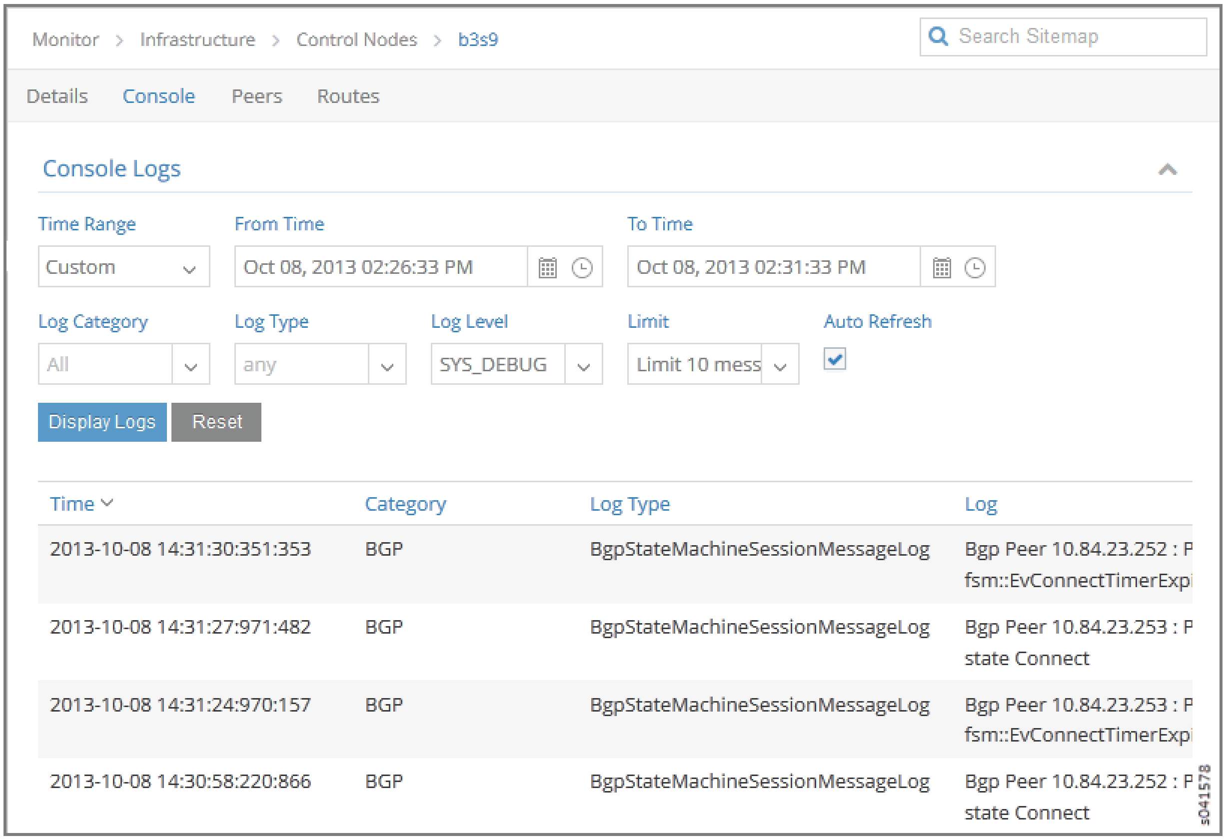
Table 47: Individual Control Node—Details Tab Fields *(Continued)*

Field	Description
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 135 on page 384.

Figure 135: Individual Control Node—Console Tab



See Table 48 on page 384 for descriptions of the fields on the **Console** tab screen.

Table 48: Control Node: Console Tab Fields

Field	Description
Time Range	Select a timeframe for which to review logging information as sent to the console. There are 11 options, ranging from the **Last 5 mins** through to the **Last 24 hrs**. The default display is for the **Last 5 mins**.
Log Category	Select a log category to display: 1. All 2. _default_ 3. XMPP 4. TCP

Table 48: Control Node: Console Tab Fields *(Continued)*

Field	Description
Log Type	Select a log type to display.
Log Level	Select a log severity level to display: 1. SYS_EMERG 2. SYS_ALERT 3. SYS_CRIT 4. SYS_ERR 5. SYS_WARN 6. SYS_NOTICE 7. SYS_INFO 8. SYS_DEBUG
Search	Enter any text string to search and display logs containing that string.
Limit	Select from a list an amount to limit the number of messages displayed: 1. No Limit 2. Limit 10 messages 3. Limit 50 messages 4. Limit 100 messages 5. Limit 200 messages 6. Limit 500 messages
Auto Refresh	Click the check box to automatically refresh the display if more messages occur.
Display Logs	Click this button to refresh the display if you change the display criteria.

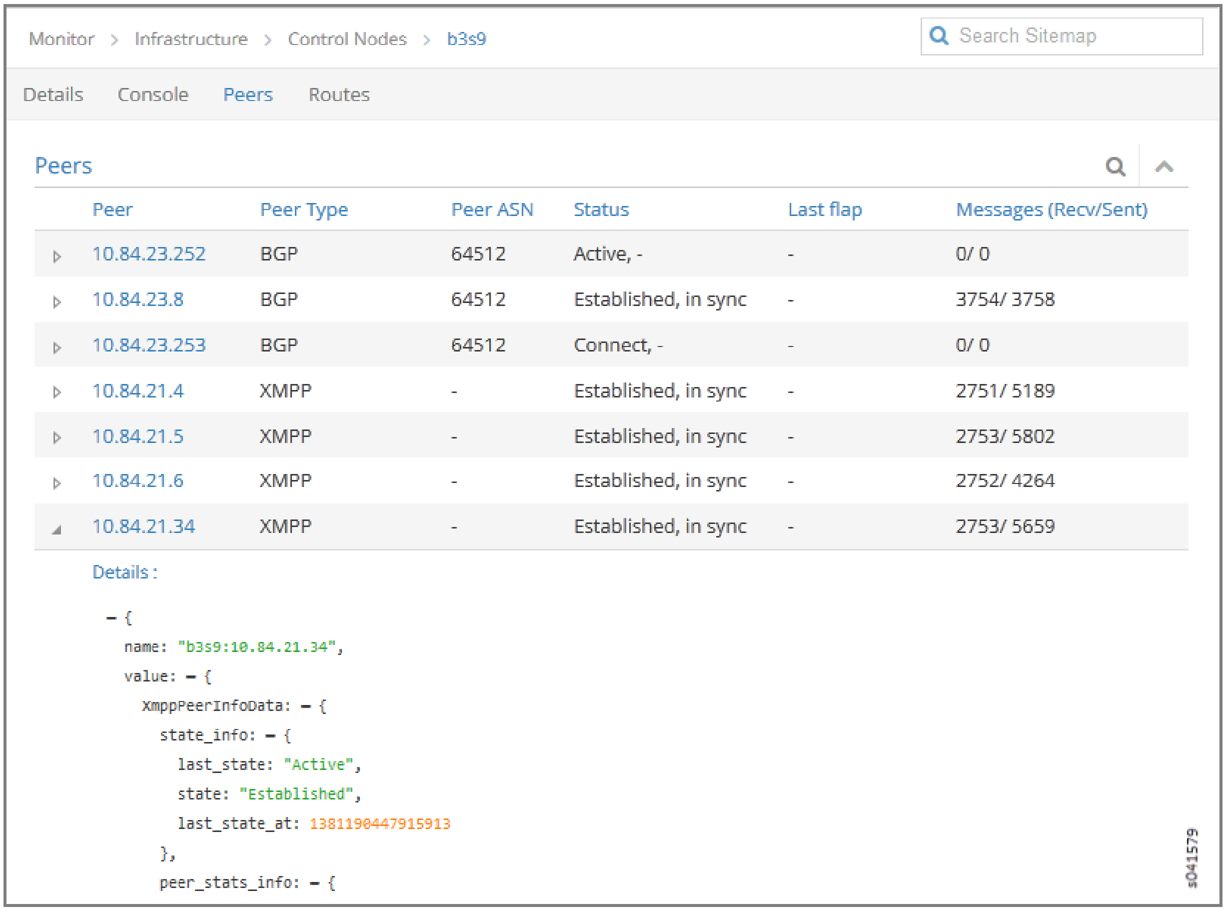
Table 48: Control Node: Console Tab Fields *(Continued)*

Field	Description
Reset	Click this button to clear any selected display criteria and reset all criteria to their default settings.
Time	This column lists the time received for each log message displayed.
Category	This column lists the log category for each log message displayed.
Log Type	This column lists the log type for each log message displayed.
Log	This column lists the log message for each log displayed.

Monitor Individual Control Node Peers

The **Peers** tab displays the peers for an individual control node and their peering state. Click the expansion arrow next to the address of any peer to reveal more details. See Figure 136 on page 387.

Figure 136: Individual Control Node—Peers Tab



See Table 49 on page 387 for descriptions of the fields on the **Peers** tab screen.

Table 49: 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.

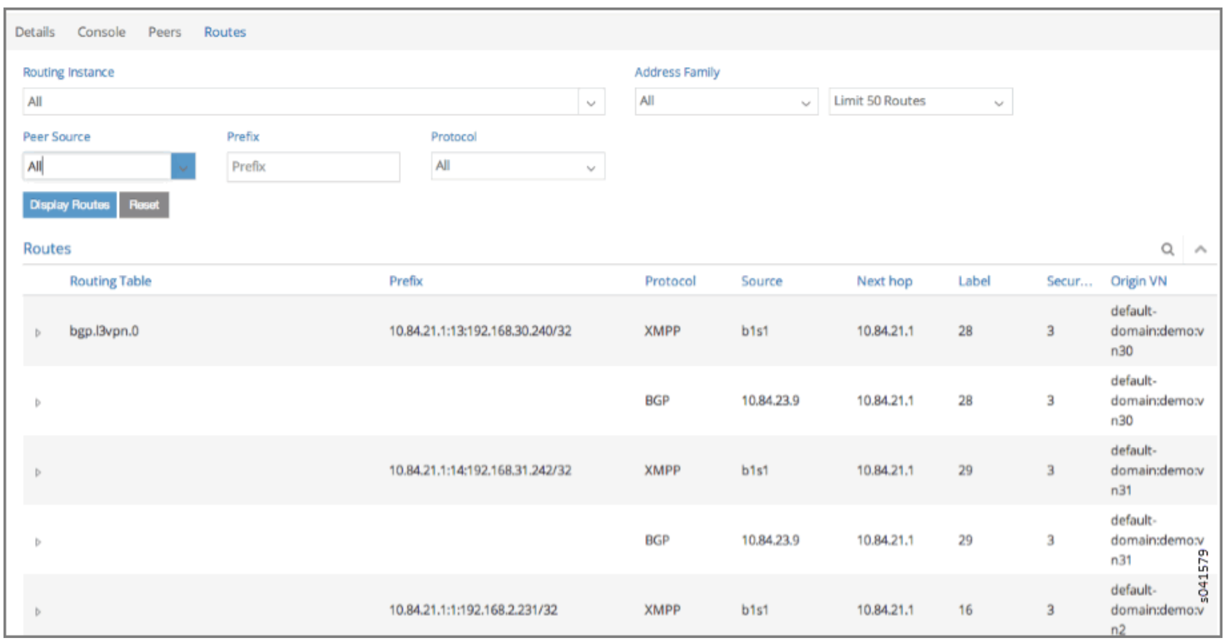
Table 49: Control Node: Peers Tab Fields *(Continued)*

Field	Description
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 137 on page 388.

Figure 137: Individual Control Node—Routes Tab



See Table 50 on page 389 for descriptions of the fields on the **Routes** tab screen.

Table 50: Control Node: Routes Tab Fields

Field	Description
Routing Instance	You can select a single routing instance from a list of all instances for which to display the active routes.
Address Family	Select an address family for which to display the active routes: 1. All (default) 2. l3vpn 3. inet 4. inetmcast
(Limit Field)	Select to limit the display of active routes: 1. Limit 10 Routes 2. Limit 50 Routes 3. Limit 100 Routes 4. Limit 200 Routes
Peer Source	Select from a list of available peers the peer for which to display the active routes, or select All.
Prefix	Enter a route prefix to limit the display of active routes to only those with the designated prefix.
Protocol	Select a protocol for which to display the active routes: 1. All (default) 2. XMPP 3. BGP 4. ServiceChain 5. Static

Table 50: Control Node: Routes Tab Fields *(Continued)*

Field	Description
Display Routes	Click this button to refresh the display of routes after selecting different display criteria.
Reset	Click this button to clear any selected criteria and return the display to default values.
Column	*Description*
Routing Table	The name of the routing table that stores this route.
Prefix	The route prefix for each active route displayed.
Protocol	The protocol used by the route.
Source	The host source for each active route displayed.
Next hop	The IP address of the next hop for each active route displayed.
Label	The label for each active route displayed.
Security	The security value for each active route displayed.
Origin VN	The virtual network from which the route originates.
AS Path	The AS path for each active route displayed.

Monitor > Infrastructure > Virtual Routers

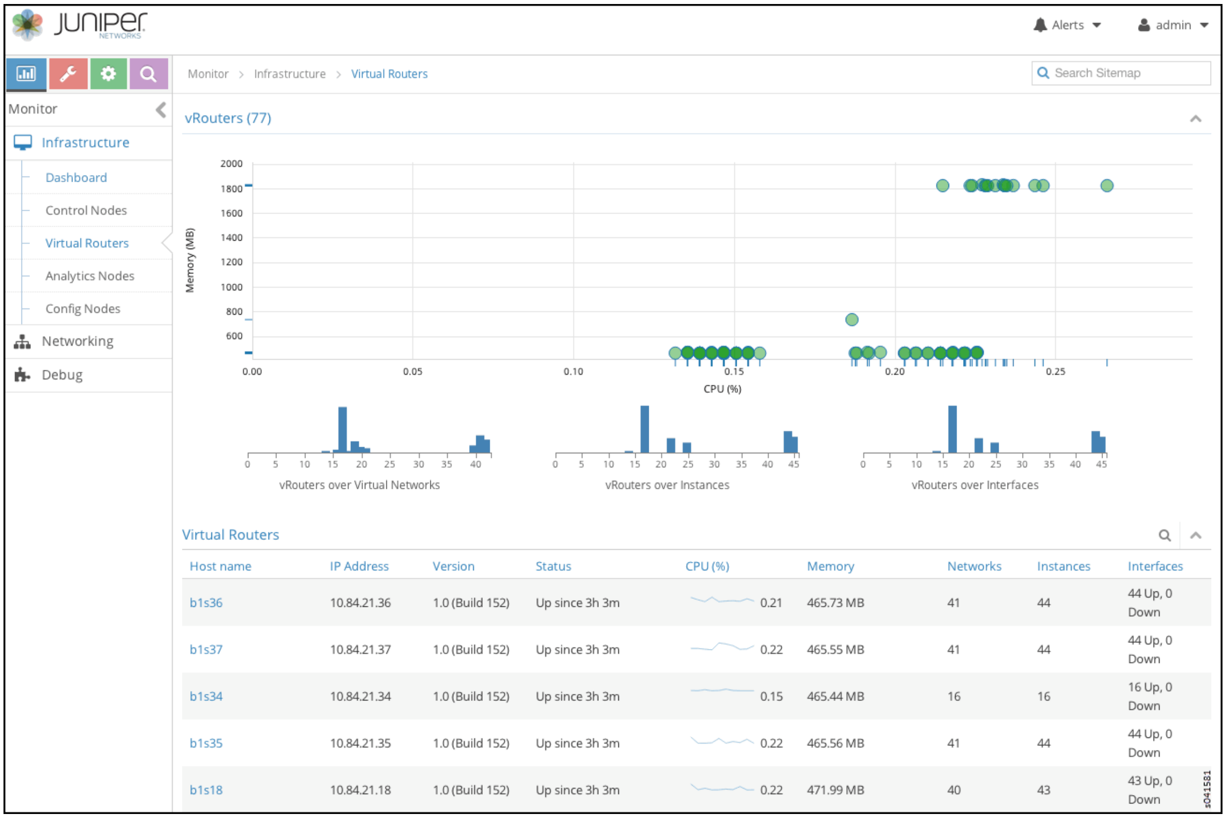
IN THIS SECTION

- Monitor vRouters Summary | 391
- Monitor Individual vRouters Tabs | 393
- Monitor Individual vRouter Details Tab | 393
- Monitor Individual vRouters Interfaces Tab | 395
- Monitor Individual vRouters Networks Tab | 397
- Monitor Individual vRouters ACL Tab | 398
- Monitor Individual vRouters Flows Tab | 400
- Monitor Individual vRouters Routes Tab | 401
- Monitor Individual vRouter Console Tab | 402

Monitor vRouters Summary

Click **Monitor > Infrastructure > Virtual Routers** to view the **vRouters** summary screen. See Figure 138 on page 392.

Figure 138: vRouters Summary



See Table 51 on page 392 for descriptions of the fields on the **vRouters Summary** screen.

Table 51: vRouters Summary Fields

Field	Description
Host name	The name of the vRouter. Click the name of any vRouter to reveal more details.
IP Address	The IP address of the vRouter.
Version	The version of software installed on the system.
Status	The current operational status of the vRouter — Up or Down.
CPU (%)	The CPU percentage currently in use by the selected vRouter.

Table 51: vRouters Summary Fields *(Continued)*

Field	Description
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.
Interfaces	The total number of interfaces for this vRouter.

Monitor Individual vRouters Tabs

Click the name of any vRouter to view details about performance and activities for that vRouter. Each individual vRouters screen has the following tabs.

- **Details**—similar display of information as on individual control nodes **Details** tab. See Figure 139 on page 394.
- **Console**—similar display of information as on individual control nodes **Console** tab. See Figure 145 on page 403.
- **Interfaces**—details about associated interfaces. See Figure 140 on page 396.
- **Networks**—details about associated networks. See Figure 141 on page 397.
- **ACL**—details about access control lists. See Figure 142 on page 399.
- **Flows**—details about associated traffic flows. See Figure 143 on page 400.
- **Routes**—details about associated routes. See Figure 144 on page 402.

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 139 on page 394. SeeTable 52 on page 394 for descriptions of the fields on this tab.

Figure 139: Individual vRouters—Details Tab

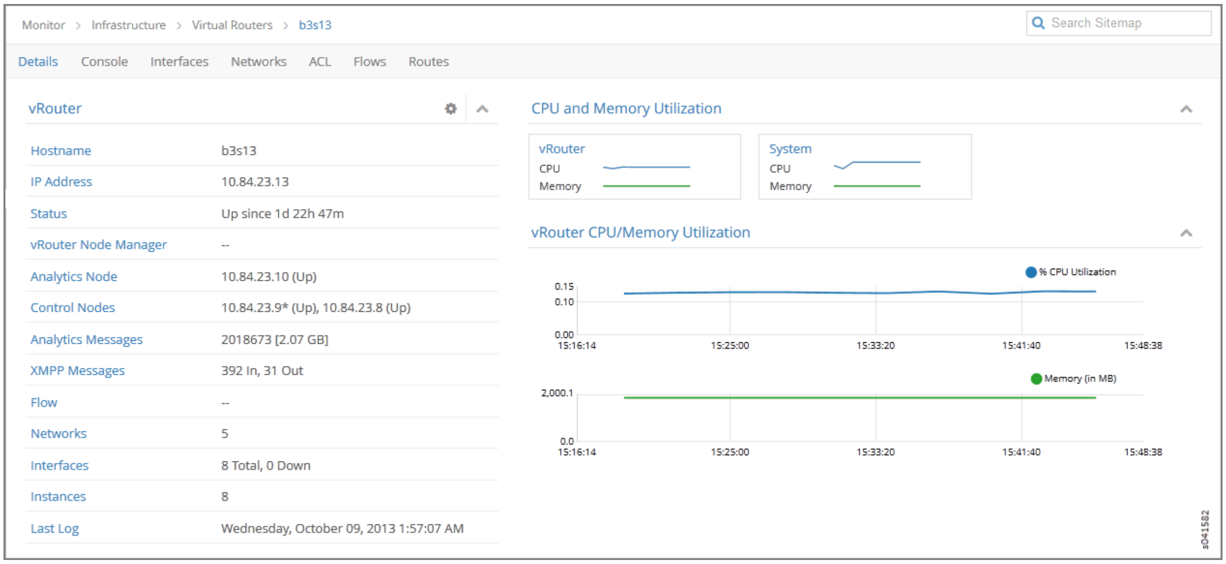


Table 52: vRouters Details Tab Fields

Field	Description
Hostname	The hostname of the vRouter.
IP Address	The IP address of the selected vRouter.
Status	The operational status of the vRouter.
vRouter Node Manager	The operational status of the vRouter node manager.
Analytics Node	The IP address of the node from which analytics (monitor) information is derived.
Control Nodes	The IP address of the configuration node associated with this vRouter.
Analytics Messages	The total number of analytics messages in and out from this node.
XMPP Messages	The total number of XMPP messages that have gone in and out of this vRouter.
Flow	The number of active flows and the total flows for this vRouter.

Table 52: vRouters Details Tab Fields *(Continued)*

Field	Description
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 140 on page 396. See Table 53 on page 396 for descriptions of the fields on the **Interfaces** tab screen.

Figure 140: Individual vRouters—Interfaces Tab

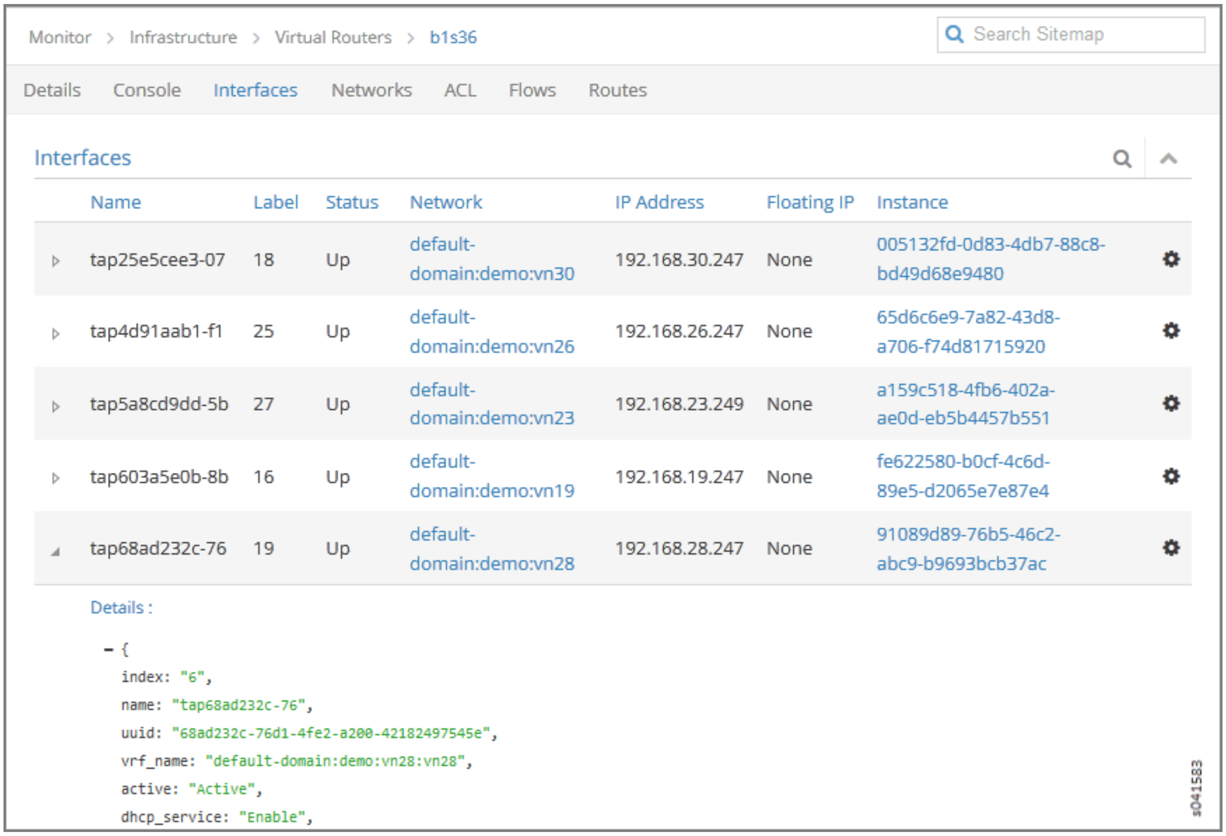


Table 53: vRouters: Interfaces Tab Fields

Field	Description
Name	The name of the interface.
Label	The label for the interface.
Status	The current status of the interface.
Network	The network associated with the interface.
IP Address	The IP address of the interface.
Floating IP	Displays any floating IP addresses associated with the interface.

Table 53: vRouters: Interfaces Tab Fields *(Continued)*

Field	Description
Instance	The name of any instance associated with the interface.

Monitor Individual vRouters Networks Tab

The **Networks** tab displays details about the networks associated with an individual vRouter. Click the expansion arrow at the name of any network to reveal more details. See Figure 141 on page 397. See Table 54 on page 398 for descriptions of the fields on the **Networks** tab screen.

Figure 141: Individual vRouters—Networks Tab

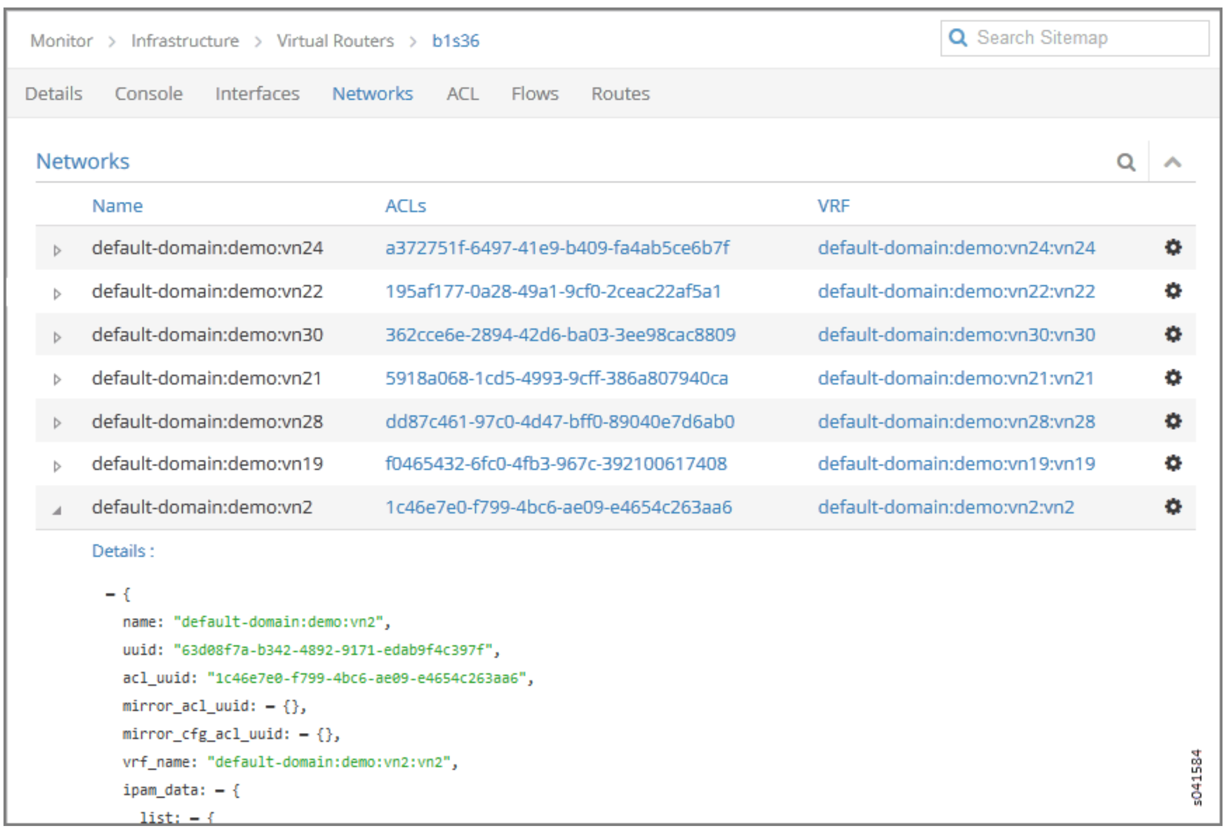


Table 54: vRouters: Networks Tab Fields

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 142 on page 399. See Table 55 on page 399 for descriptions of the fields on the **ACL** tab screen.

Figure 142: Individual vRouters—ACL Tab

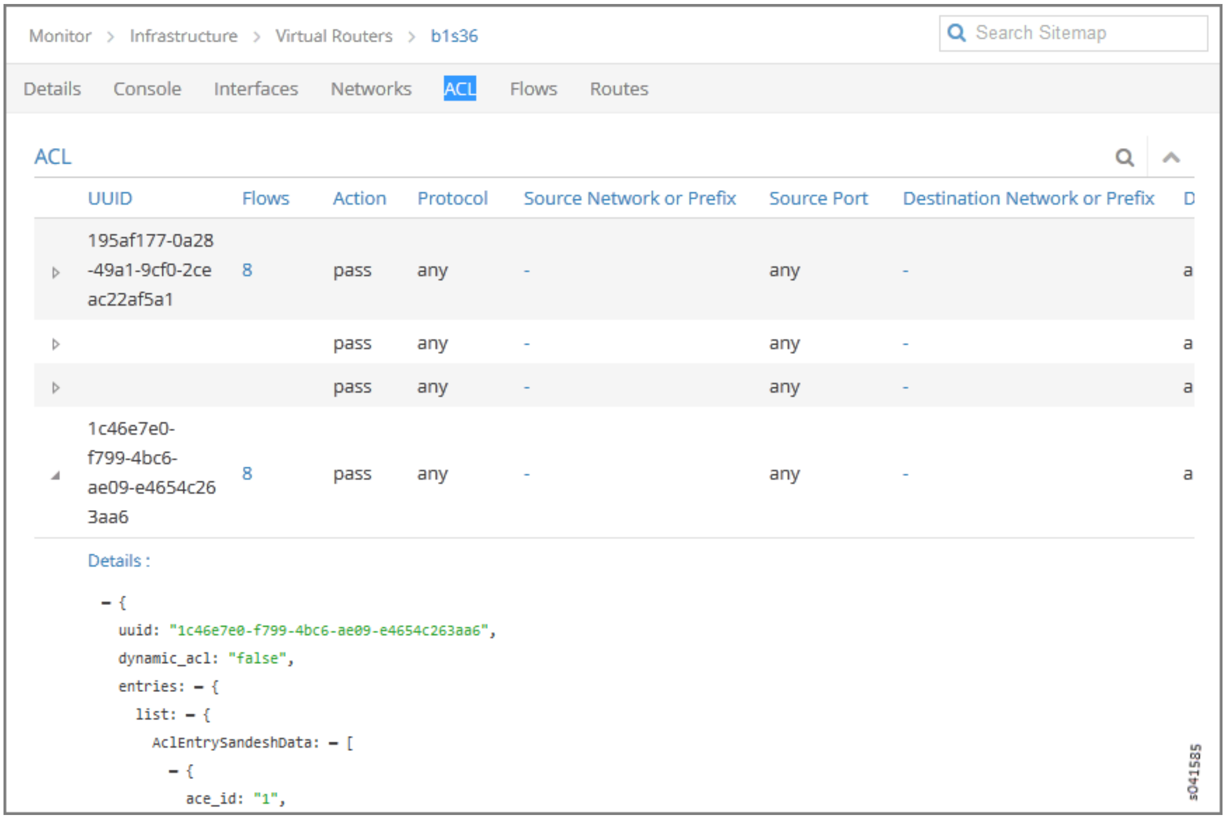


Table 55: vRouters: ACL Tab Fields

Field	Description
UUID	The universal unique identifier (UUID) associated with the listed ACL.
Flows	The flows associated with the listed ACL.
Action	The traffic action defined by the listed ACL.
Protocol	The protocol associated with the listed ACL.
Source Network or Prefix	The name or prefix of the source network associated with the listed ACL.
Source Port	The source port associated with the listed ACL.

Table 55: vRouters: ACL Tab Fields *(Continued)*

Field	Description
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 arrrow 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 143 on page 400. See Table 56 on page 401 for descriptions of the fields on the **Flows** tab screen.

Figure 143: Individual vRouters—Flows Tab

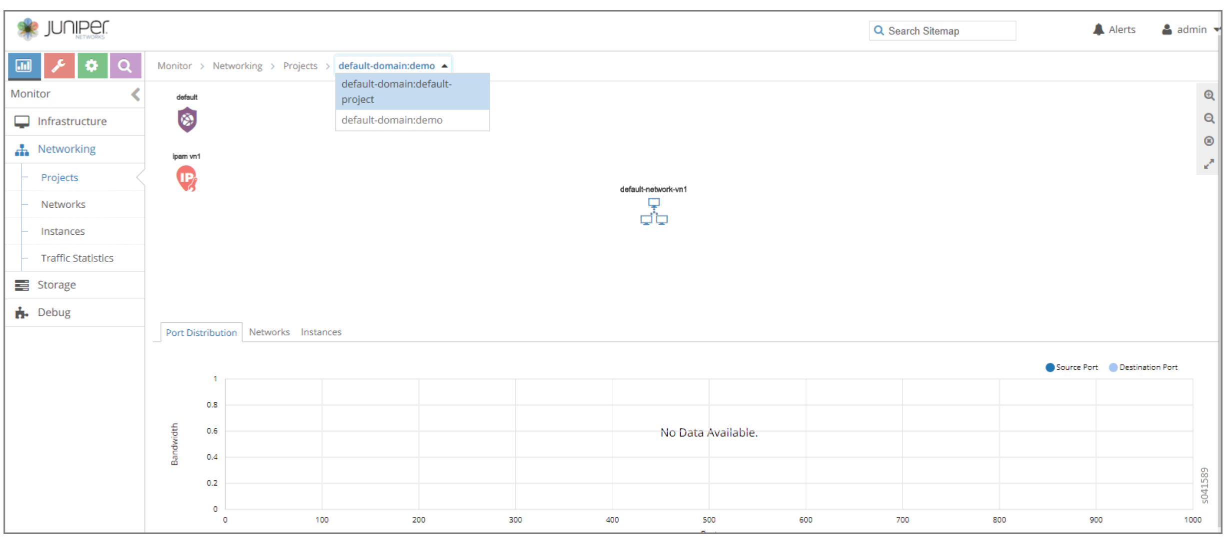


Table 56: vRouters: Flows Tab Fields

Field	Description
ACL UUID	The default is to show **All** flows, however, you can select from a drop down list any single flow to view its details.
ACL / SG UUID	The universal unique identifier (UUID) associated with the listed ACL or SG.
Protocol	The protocol associated with the listed flow.
Src Network	The name of the source network associated with the listed flow.
Src IP	The source IP address associated with the listed flow.
Src Port	The source port of the listed flow.
Dest Network	The name of the destination network associated with the listed flow.
Dest IP	The destination IP address associated with the listed flow.
Dest Port	The destination port associated with the listed flow.
Bytes/Pkts	The number of bytes and packets associated with the listed flow.
Setup Time	The setup time associated with the listed flow.

Monitor Individual vRouters Routes Tab

The **Routes** tab displays details about unicast and multicast routes in specific VRFs for an individual vRouter. Click the expansion arrow next to the route prefix to reveal more details. See Figure 144 on page 402. See Table 57 on page 402 for descriptions of the fields on the **Routes** tab screen.

Figure 144: Individual vRouters—Routes Tab

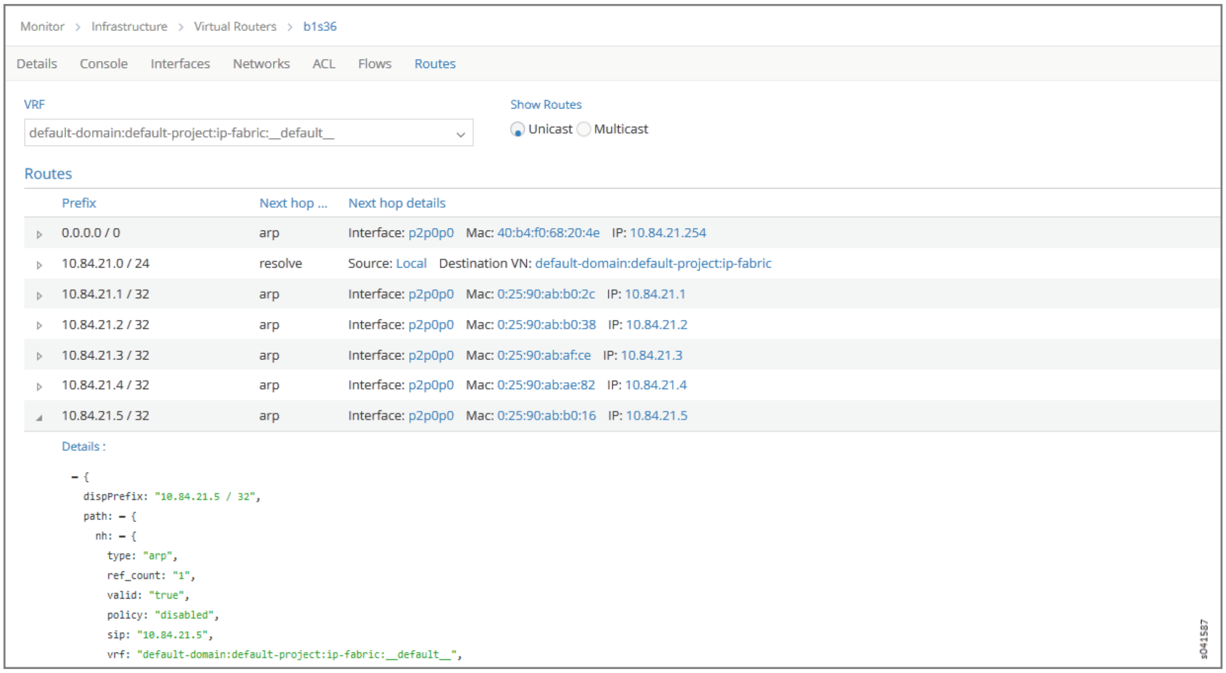


Table 57: vRouters: Routes Tab Fields

Field	Description
VRF	Select from a drop down list the virtual routing and forwarding (VRF) to view.
Show Routes	Select to show the route type: **Unicast** or **Multicast**.
Prefix	The IP address prefix of a route.
Next hop	The next hop method for this route.
Next hop details	The next hop details for this route.

Monitor Individual vRouter Console Tab

Click the **Console** tab for an individual vRouter to display system logging information for a defined time period, with the last 5 minutes of information as the default display. See Figure 145 on page 403. See Table 58 on page 403 for descriptions of the fields on the **Console** tab screen.

Figure 145: Individual vRouter—Console Tab

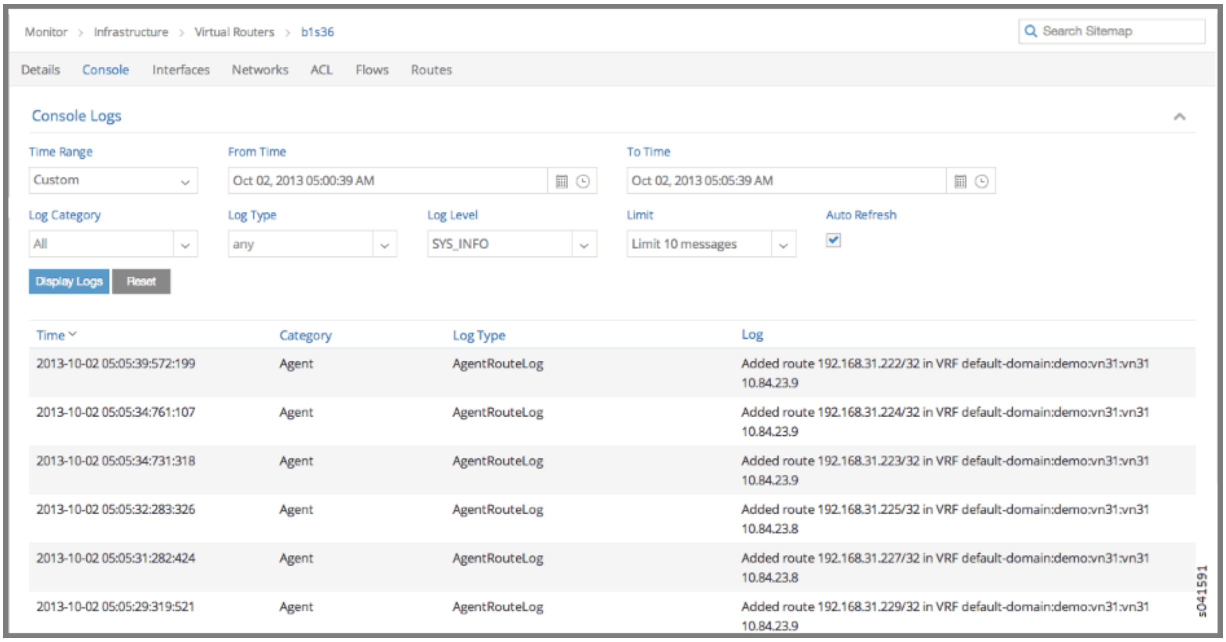


Table 58: Control Node: Console Tab Fields

Field	Description
Time Range	Select a timeframe for which to review logging information as sent to the console. There are several options, ranging from **Last 5 mins** through to the **Last 24 hrs**, plus a **Custom** time range.
From Time	If you select **Custom** in **Time Range**, enter the start time.
To Time	If you select **Custom** in **Time Range**, enter the end time.
Log Category	Select a log category to display: • All • _default_ • XMPP • TCP
Log Type	Select a log type to display.

Table 58: Control Node: Console Tab Fields *(Continued)*

Field	Description
Log Level	Select a log severity level to display: • SYS_EMERG • SYS_ALERT • SYS_CRIT • SYS_ERR • SYS_WARN • SYS_NOTICE • SYS_INFO • SYS_DEBUG
Limit	Select from a list an amount to limit the number of messages displayed: • No Limit • Limit 10 messages • Limit 50 messages • Limit 100 messages • Limit 200 messages • Limit 500 messages
Auto Refresh	Click the check box to automatically refresh the display if more messages occur.
Display Logs	Click this button to refresh the display if you change the display criteria.
Reset	Click this button to clear any selected display criteria and reset all criteria to their default settings.
Columns	

Table 58: Control Node: Console Tab Fields *(Continued)*

Field	Description
Time	This column lists the time received for each log message displayed.
Category	This column lists the log category for each log message displayed.
Log Type	This column lists the log type for each log message displayed.
Log	This column lists the log message for each log displayed.

Monitor > Infrastructure > Analytics Nodes

IN THIS SECTION

- Monitor Analytics Nodes | **405**
- Monitor Analytics Individual Node Details Tab | **407**
- Monitor Analytics Individual Node Generators Tab | **408**
- Monitor Analytics Individual Node QE Queries Tab | **409**
- Monitor Analytics Individual Node Console Tab | **410**

Select **Monitor > Infrastructure > Analytics Nodes** to view the console logs, generators, and query expansion (QE) queries of the analytics nodes.

Monitor Analytics Nodes

Select **Monitor > Infrastructure > Analytics Nodes** to view a summary of activities for the analytics nodes; see Figure 146 on page 406. See Table 59 on page 406 for descriptions of the fields on the analytics summary.

Figure 146: Analytics Nodes Summary

Table 59: 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 147 on page 407.

See Table 60 on page 407 for descriptions of the fields on this screen.

Figure 147: Monitor Analytics Individual Node Details Tab

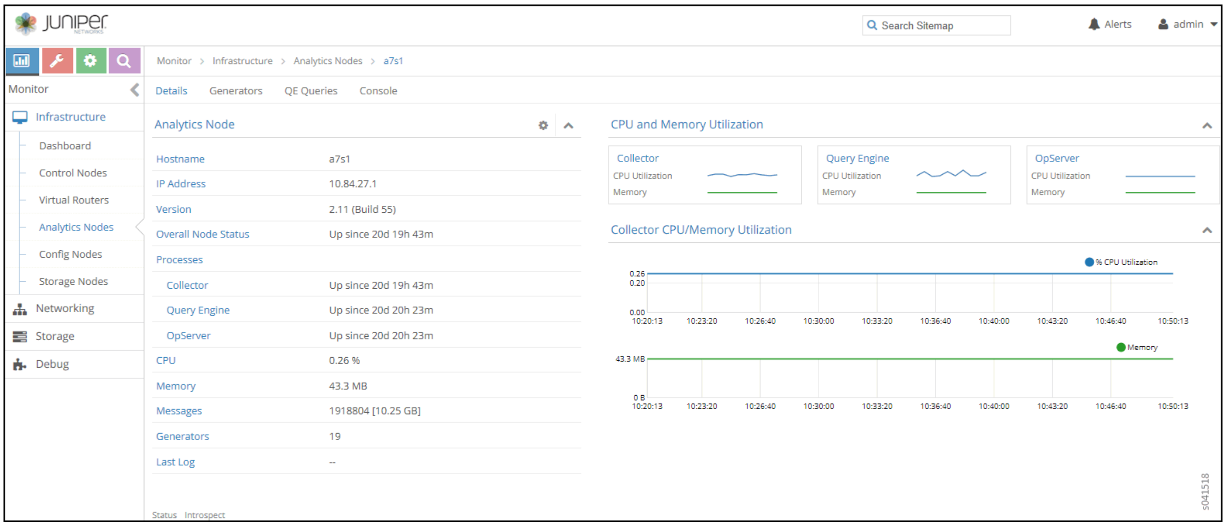


Table 60: Monitor Analytics Individual Node Details Tab Fields

Field	Description
Hostname	The name of this node.
IP Address	The IP address of this node.
Version	The installed version of the software.
Overall Node Status	The current operational status of the node — Up or Down — and the length of time in this state.
Processes	The current status of each analytics process, including Collector, Query Engine, and OpServer.

Table 60: Monitor Analytics Individual Node Details Tab Fields *(Continued)*

Field	Description
CPU (%)	The average CPU percentage usage for this node.
Memory	The average memory usage of this node.
Messages	The total number of messages for this node.
Generators	The total number of generators associated with this node.
Last Log	The date and time of the last log message issued about this node.

Monitor Analytics Individual Node Generators Tab

The **Generators** tab displays information about the generators for an individual analytics node; see Figure 148 on page 408. Click the expansion arrow next to any generator name to reveal more details. See Table 61 on page 409 for descriptions of the fields on the **Peers** tab screen.

Figure 148: Individual Analytics Node—Generators Tab

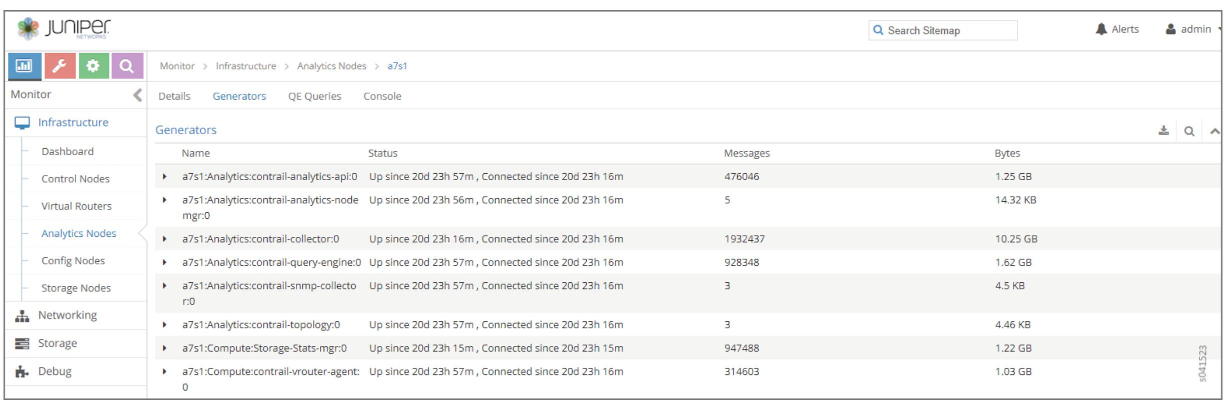


Table 61: Monitor Analytics Individual Node Generators Tab Fields

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 149 on page 409.

See Table 62 on page 409 for descriptions of the fields on the **QE Queries** tab screen.

Figure 149: Individual Analytics Node—QE QueriesTab

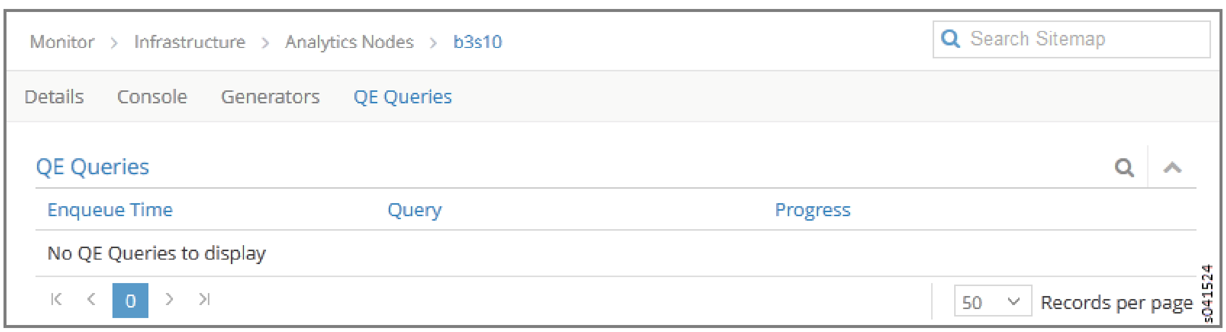


Table 62: Analytics Node QE Queries Tab Fields

Field	Description
Enqueue Time	The length of time this message has been in the queue waiting to be delivered.
Query	The query message.

Table 62: Analytics Node QE Queries Tab Fields *(Continued)*

Field	Description
Progress (%)	The percentage progress for the message delivery.

Monitor Analytics Individual Node Console Tab

Click the **Console** tab for an individual analytics node to display system logging information for a defined time period. See Figure 150 on page 410. See Table 63 on page 410 for descriptions of the fields on the **Console** tab screen.

Figure 150: Analytics Individual Node—Console Tab

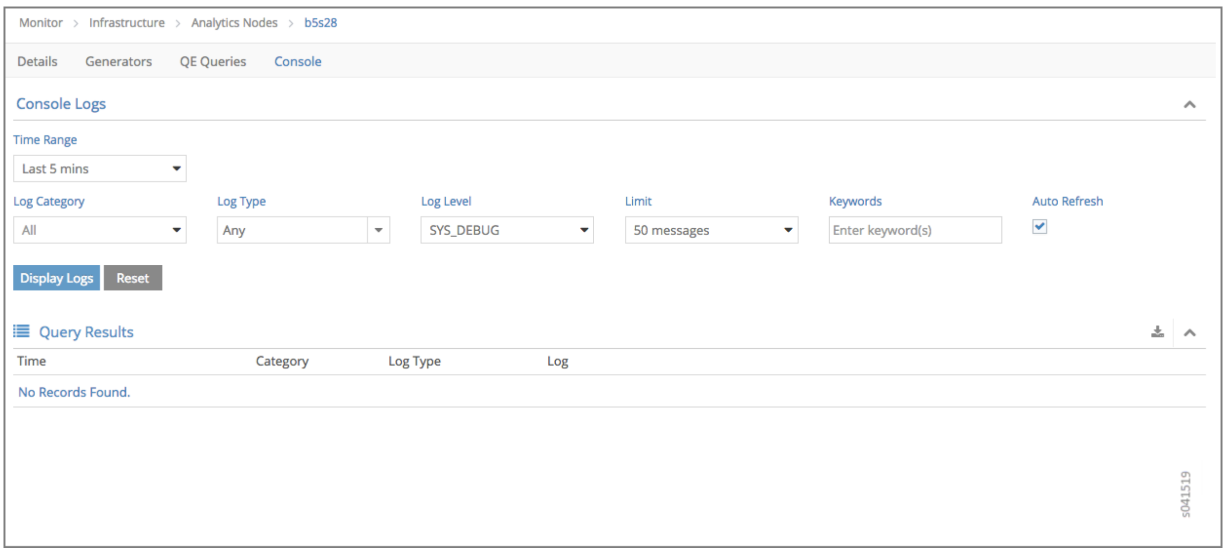


Table 63: Monitor Analytics Individual Node Console Tab Fields

Field	Description
Time Range	Select a timeframe for which to review logging information as sent to the console. There are 11 options, ranging from the **Last 5 mins** through to the **Last 24 hrs**. The default display is for the **Last 5 mins**.

Table 63: Monitor Analytics Individual Node Console Tab Fields *(Continued)*

Field	Description
Log Category	Select a log category to display: 1. All 2. _default_ 3. XMPP 4. TCP
Log Type	Select a log type to display.
Log Level	Select a log severity level to display: 1. SYS_EMERG 2. SYS_ALERT 3. SYS_CRIT 4. SYS_ERR 5. SYS_WARN 6. SYS_NOTICE 7. SYS_INFO 8. SYS_DEBUG
Keywords	Enter any text string to search for and display logs containing that string.

Table 63: Monitor Analytics Individual Node Console Tab Fields *(Continued)*

Field	Description
(Limit field)	Select the number of messages to display: 1. No Limit 2. Limit 10 messages 3. Limit 50 messages 4. Limit 100 messages 5. Limit 200 messages 6. Limit 500 messages
Auto Refresh	Click the check box to automatically refresh the display if more messages occur.
Display Logs	Click this button to refresh the display if you change the display criteria.
Reset	Click this button to clear any selected display criteria and reset all criteria to their default settings.
Time	This column lists the time received for each log message displayed.
Category	This column lists the log category for each log message displayed.
Log Type	This column lists the log type for each log message displayed.
Log	This column lists the log message for each log displayed.

Monitor > Infrastructure > Config Nodes

IN THIS SECTION

- Monitor Config Nodes | 413
- Monitor Individual Config Node Details | 414
- Monitor Individual Config Node Console | 415

Select **Monitor > Infrastructure > Config Nodes** to view the information about the system config nodes.

Monitor Config Nodes

Select **Monitor > Infrastructure > Config Nodes** to view a summary of activities for the analytics nodes. See Figure 151 on page 413.

Figure 151: Config Nodes Summary

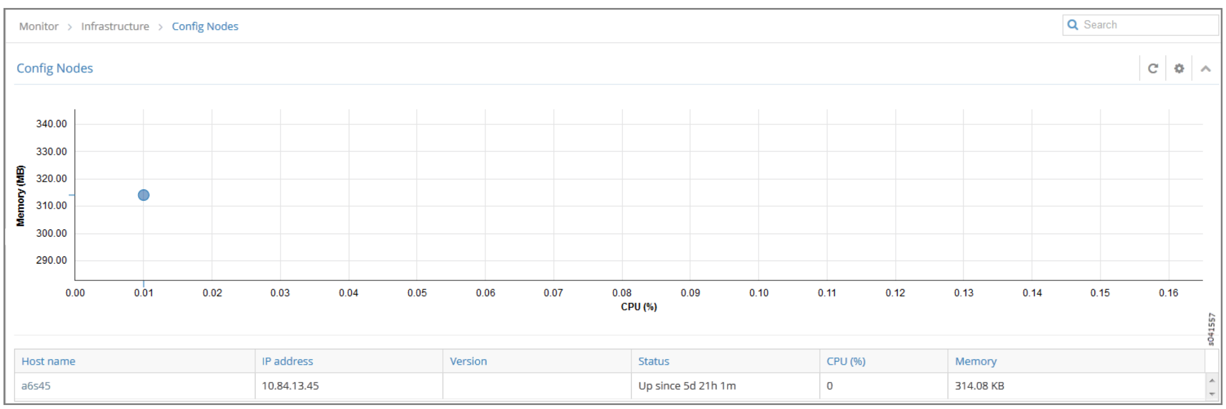


Table 64 on page 413 describes the fields in the Config Nodes summary.

Table 64: Config Nodes Summary Fields

Field	Description
Host name	The name of this node.

Table 64: Config Nodes Summary Fields *(Continued)*

Field	Description
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 152 on page 414.

Figure 152: Individual Config Nodes— Details Tab

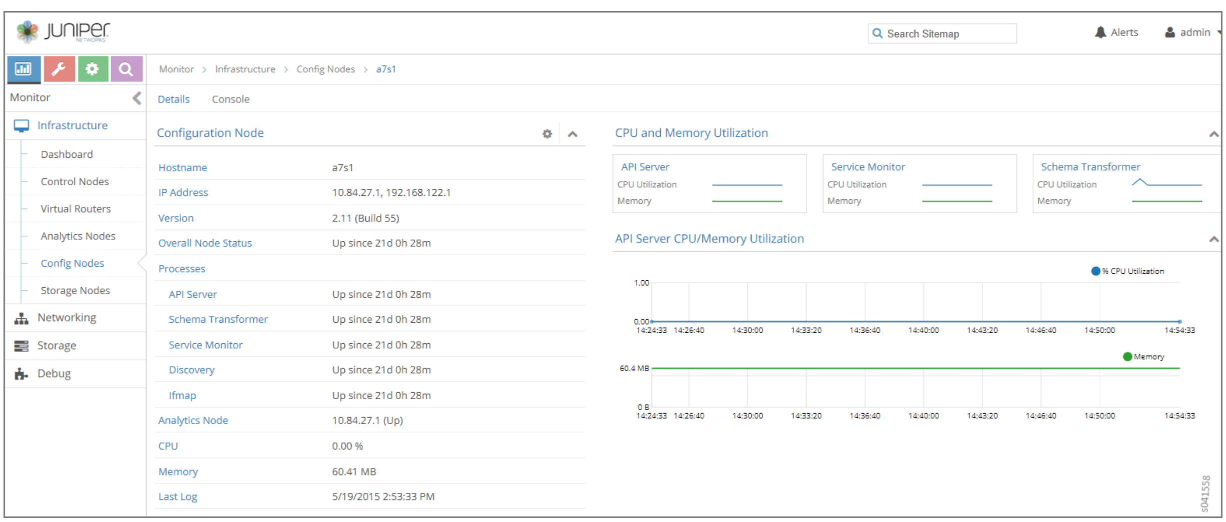


Table 65 on page 415 describes the fields on the Details screen.

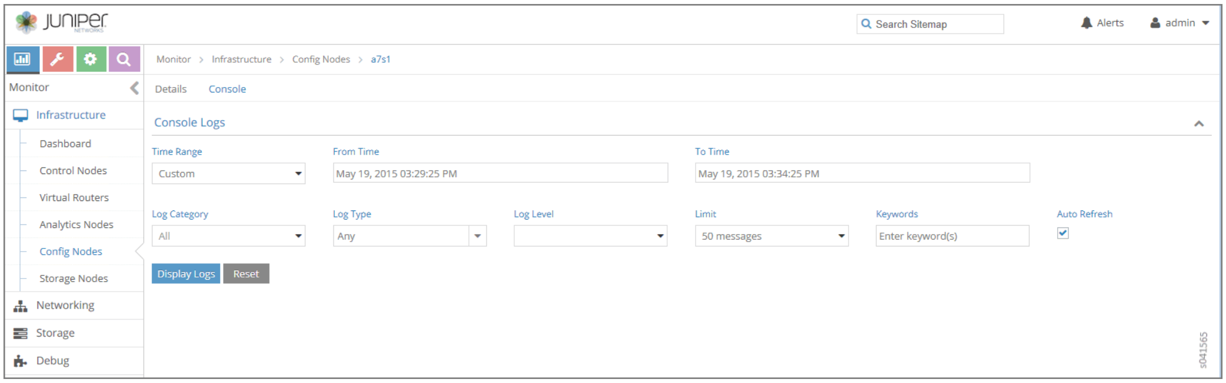
Table 65: Individual Config Nodes— Details Tab Fields

Field	Description
Hostname	The name of the config node.
IP Address	The IP address of this node.
Version	The installed version of the software.
Overall Node Status	The current operational status of the node — Up or Down — and the length of time it is in this state.
Processes	The current operational status of the processes associated with the config node, including AI Server, Schema Transformer, Service Monitor, and the like.
Analytics Node	The analytics node associated with this node.
CPU (%)	The average CPU percentage usage for this node.
Memory	The average memory usage by this node.

Monitor Individual Config Node Console

Click the **Console** tab for an individual config node to display system logging information for a defined time period. See Figure 153 on page 416.

Figure 153: Individual Config Node—Console Tab



See Table 66 on page 416 for descriptions of the fields on the **Console** tab screen.

Table 66: 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.
Log Level	Select a log severity level to display:
Limit	Select from a list an amount to limit the number of messages displayed: 1. All 2. Limit 10 messages 3. Limit 50 messages 4. Limit 100 messages 5. Limit 200 messages 6. Limit 500 messages

Table 66: Individual Config Node-Console Tab Fields ***(Continued)***

Field	Description
Keywords	Enter any key words by which to filter the log messages displayed.
Auto Refresh	Click the check box to automatically refresh the display if more messages occur.
Display Logs	Click this button to refresh the display if you change the display criteria.
Reset	Click this button to clear any selected display criteria and reset all criteria to their default settings.

Monitor > Networking

IN THIS SECTION

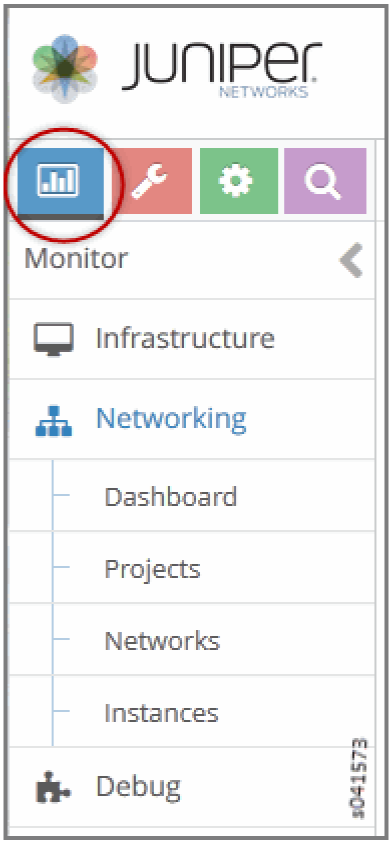
- Monitor > Networking Menu Options | **417**
- Monitor -> Networking -> Dashboard | **418**
- Monitor > Networking > Projects | **420**
- Monitor Projects Detail | **421**
- Monitor > Networking > Networks | **424**

The **Monitor -> Networking** pages give an overview of the networking traffic statistics and health of domains, projects within domains, virtual networks within projects, and virtual machines within virtual networks.

Monitor > Networking Menu Options

Figure 154 on page 418 shows the menu options available under **Monitor > Networking**.

Figure 154: 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 155 on page 419.

Figure 155: Traffic Statistics for Domain Window

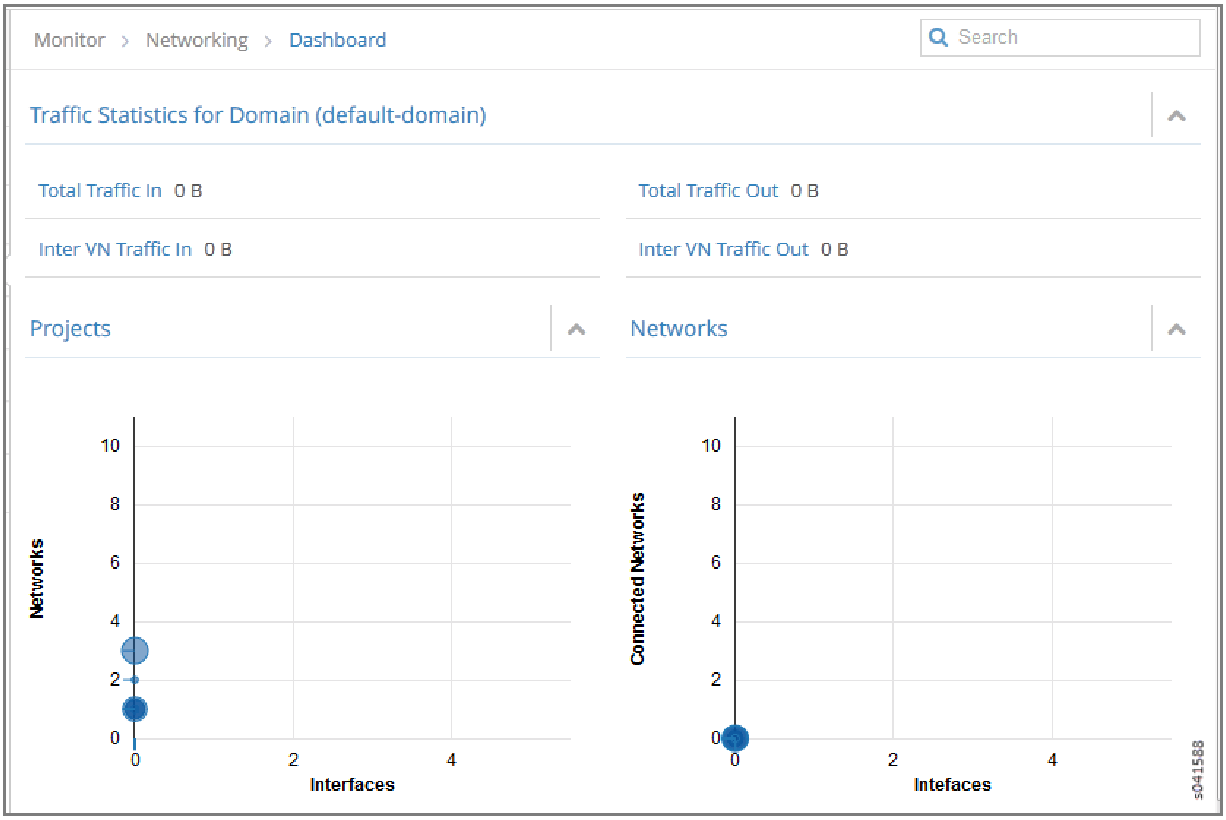


Table 67 on page 419 describes the fields in the Traffic Statistics for Domain window.

Table 67: Projects Summary Fields

Field	Description
Total Traffic In	The volume of traffic into this domain
Total Traffic Out	The volume of traffic out of this domain.
Inter VN Traffic In	The volume of inter-virtual network traffic into this domain.
Inter VN Traffic Out	The volume of inter-virtual network traffic out of this domain.

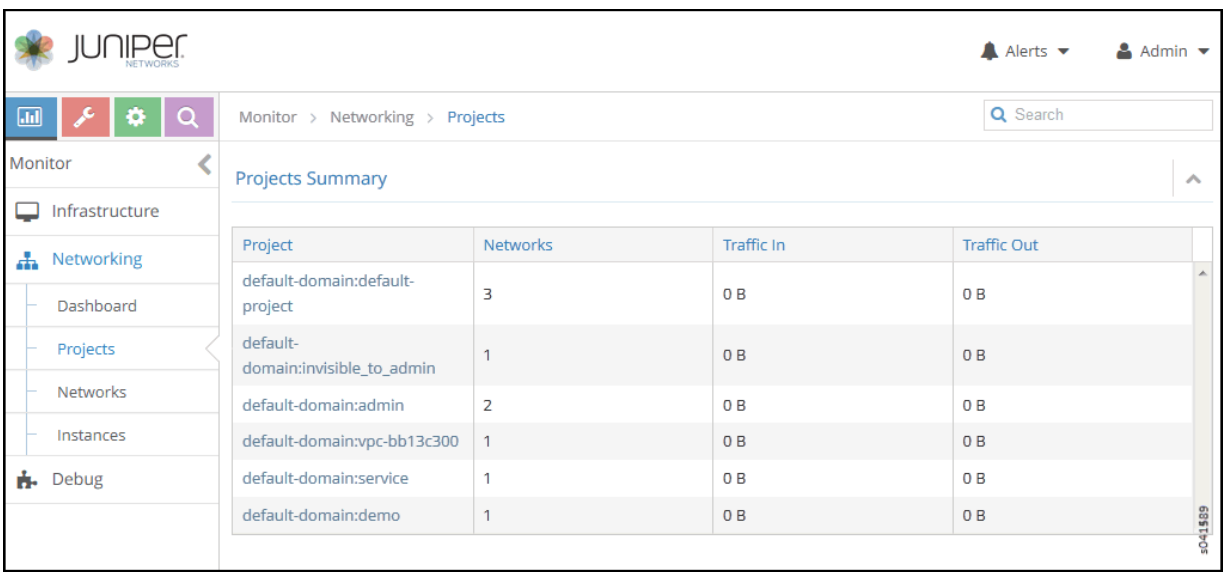
Table 67: Projects Summary Fields *(Continued)*

Field	Description
Projects	This chart displays the networks and interfaces for projects with the most throughput over the past 30 minutes. Click **Projects** then select **Monitor > Networking > Projects**, to display more detailed statistics.
Networks	This chart displays the networks for projects with the most throughput over the past 30 minutes. Click **Networks** then select **Monitor > Networking > Networks**, to display more detailed statistics.

Monitor > Networking > Projects

Select **Monitor > Networking > Projects** to see information about projects in the system. See Figure 156 on page 420.

Figure 156: Monitor > Networking > Projects



See Table 68 on page 421 for descriptions of the fields on this screen.

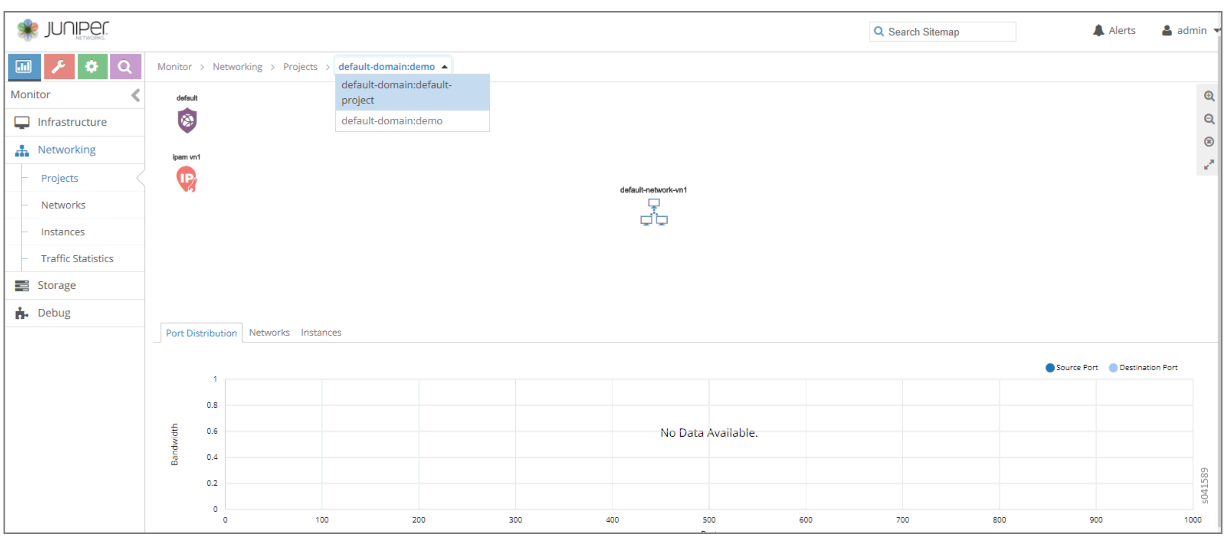
Table 68: Projects Summary Fields

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.
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 157 on page 421. Hover over any of the connections to get more details.

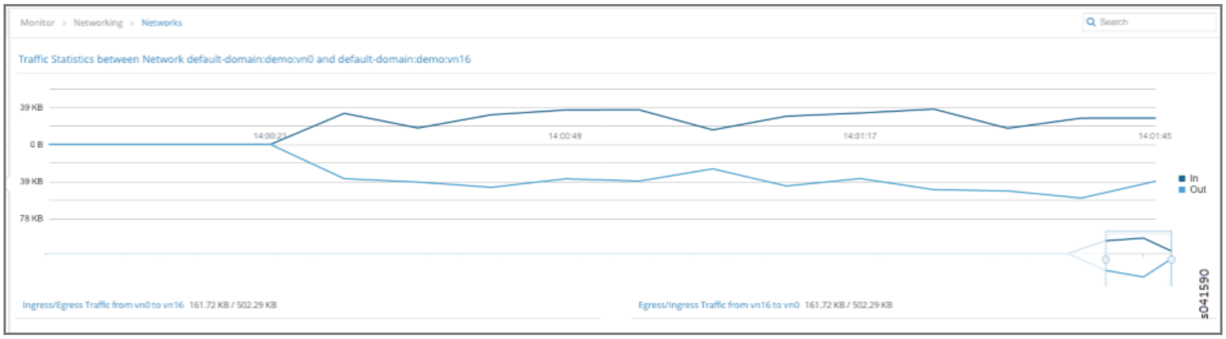
Figure 157: Monitor Projects Connectivity Details



In the Connectivity Details window you can click the links between the virtual networks to view the traffic statistics between the virtual networks.

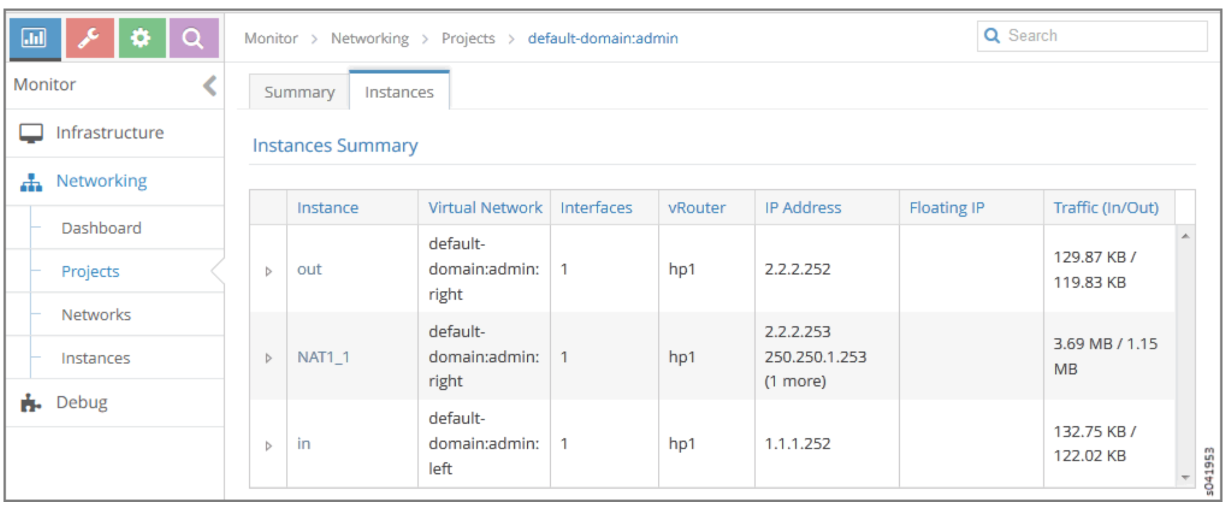
The Traffic Statistics information is also available when you select **Monitor > Networking > Networks** as shown in Figure 158 on page 422.

Figure 158: 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 159: Projects Instances Summary



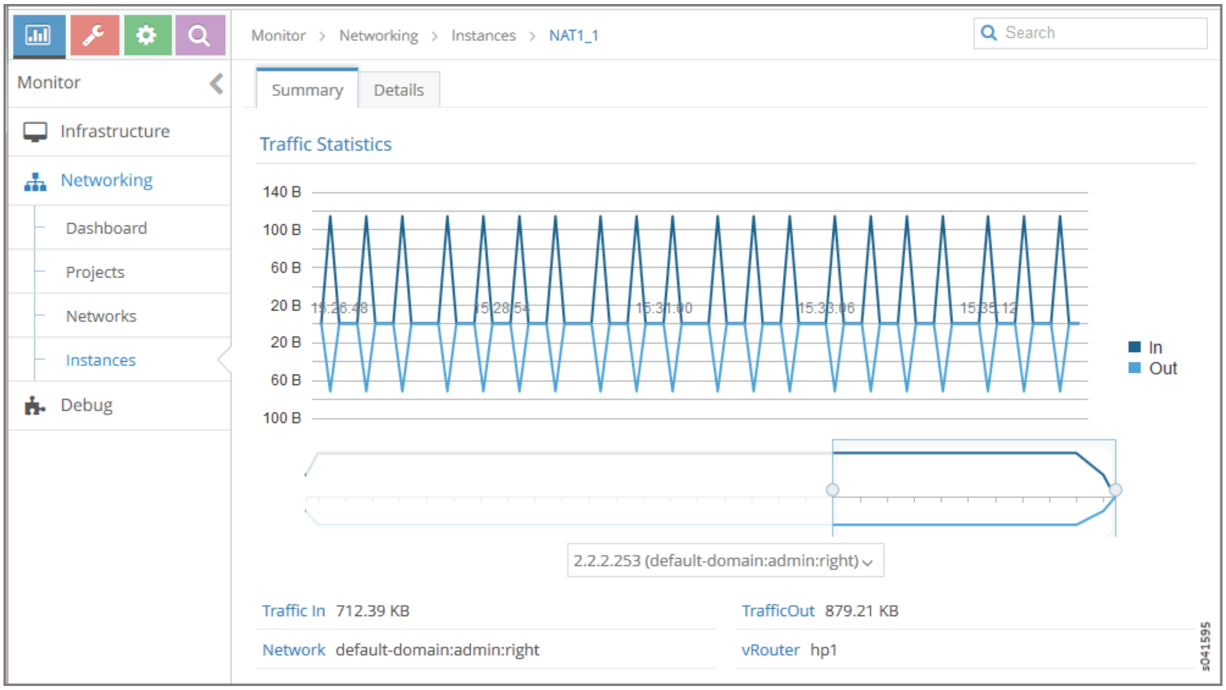
See Table 3 for a description of the fields on this screen.

Table 69: Projects Instances Summary Fields

Field	Description
Instance	The name of the instance. Click the name then select **Monitor > Networking > Instances** to display details about the traffic statistics for this instance.
Virtual Network	The virtual network associated with this instance.
Interfaces	The number of interfaces associated with this instance.
vRouter	The name of the vRouter associated with this instance.
IP Address	Any IP addresses associated with this instance.
Floating IP	Any floating IP addresses associated with this instance.
Traffic (In/Out)	The volume of traffic in KB or MB that is passing in and out of this instance.

Select **Monitor > Networking > Instances** to display instance traffic statistics as shown in Figure 160 on page 424.

Figure 160: Instance Traffic Statistics



Monitor > Networking > Networks

Select **Monitor > Networking > Networks** to view a summary of the virtual networks in your system. See Figure 161 on page 424.

Figure 161: Network Summary

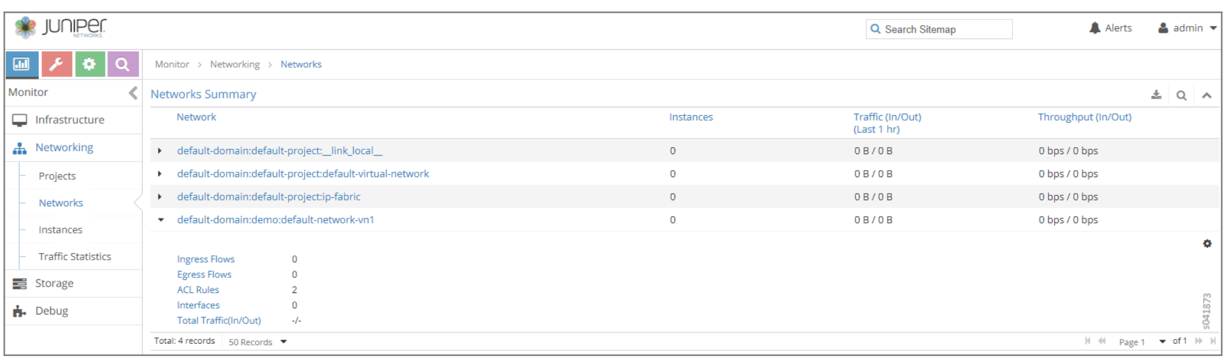


Table 70: Network Summary Fields

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 162 on page 425 shows the **Summary** tab for an individual network, which displays connectivity details and traffic statistics for the selected network.

Figure 162: Individual Network Connectivity Details—Summary Tab

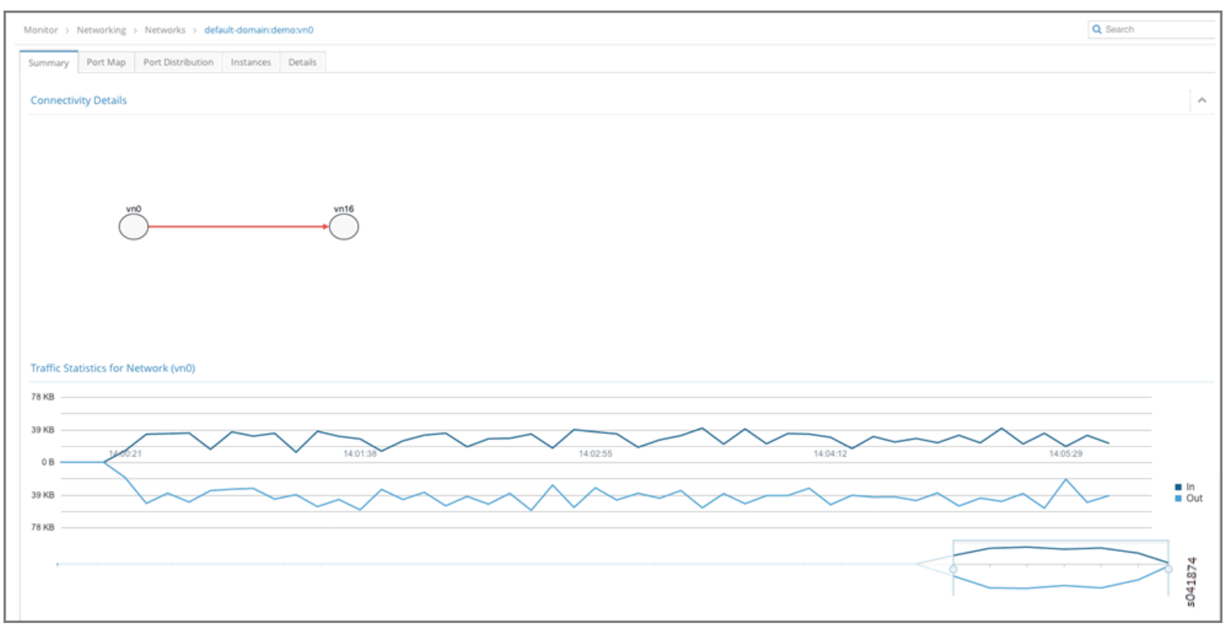


Figure 163 on page 426 shows the **Port Map** tab for an individual network, which displays the relative distribution of traffic for this network by protocol, by port.

Figure 163: Individual Network-– Port Map Tab

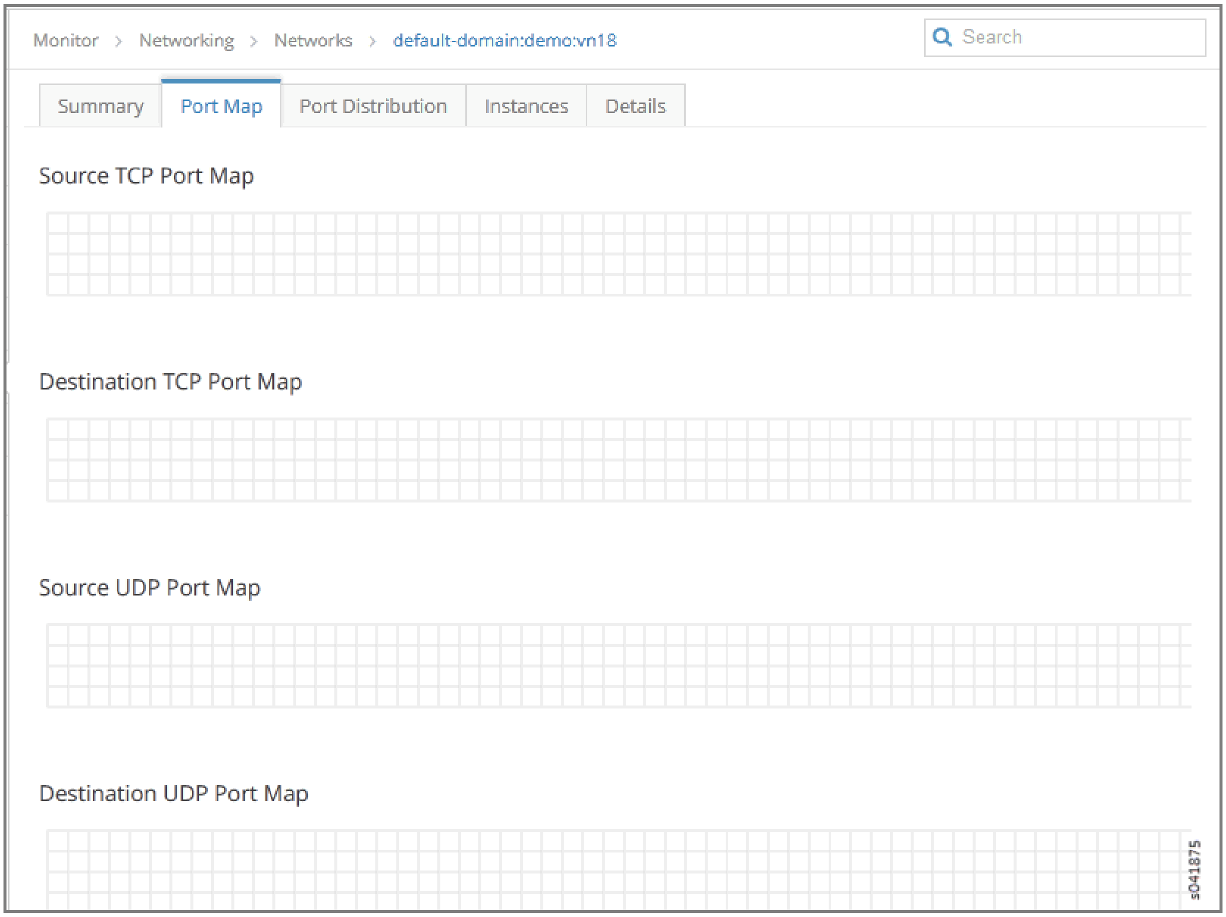


Figure 164 on page 427 shows the **Port Distribution** tab for an individual network, which displays the relative distribution of traffic in and out by source port and destination port.

Figure 164: Individual Network-– Port Distribution Tab

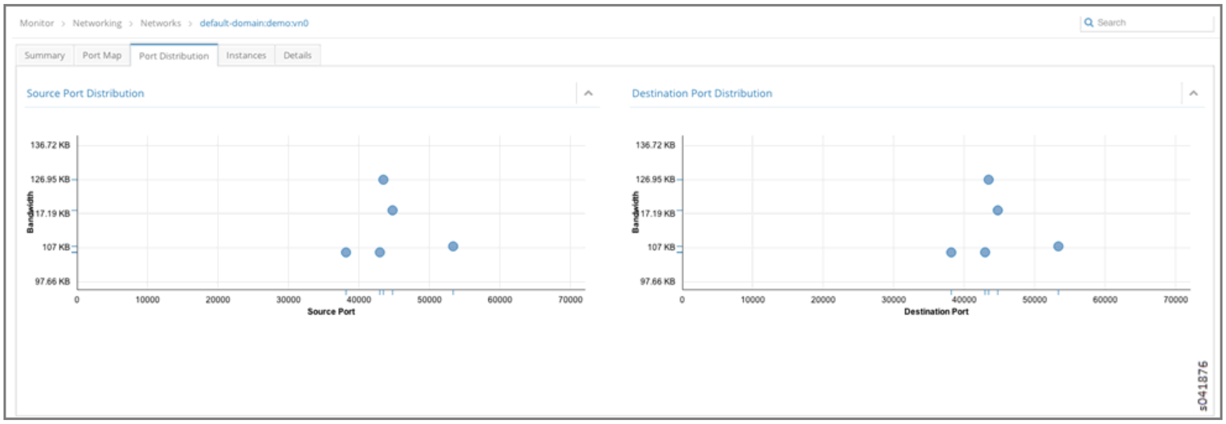


Figure 165 on page 428 shows the **Instances** tab for an individual network, which displays details for each instance associated with this network, including the number of interfaces, the associated vRouter, the instance IP address, and the volume of traffic in and out.

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 165: Individual Network Instances Tab

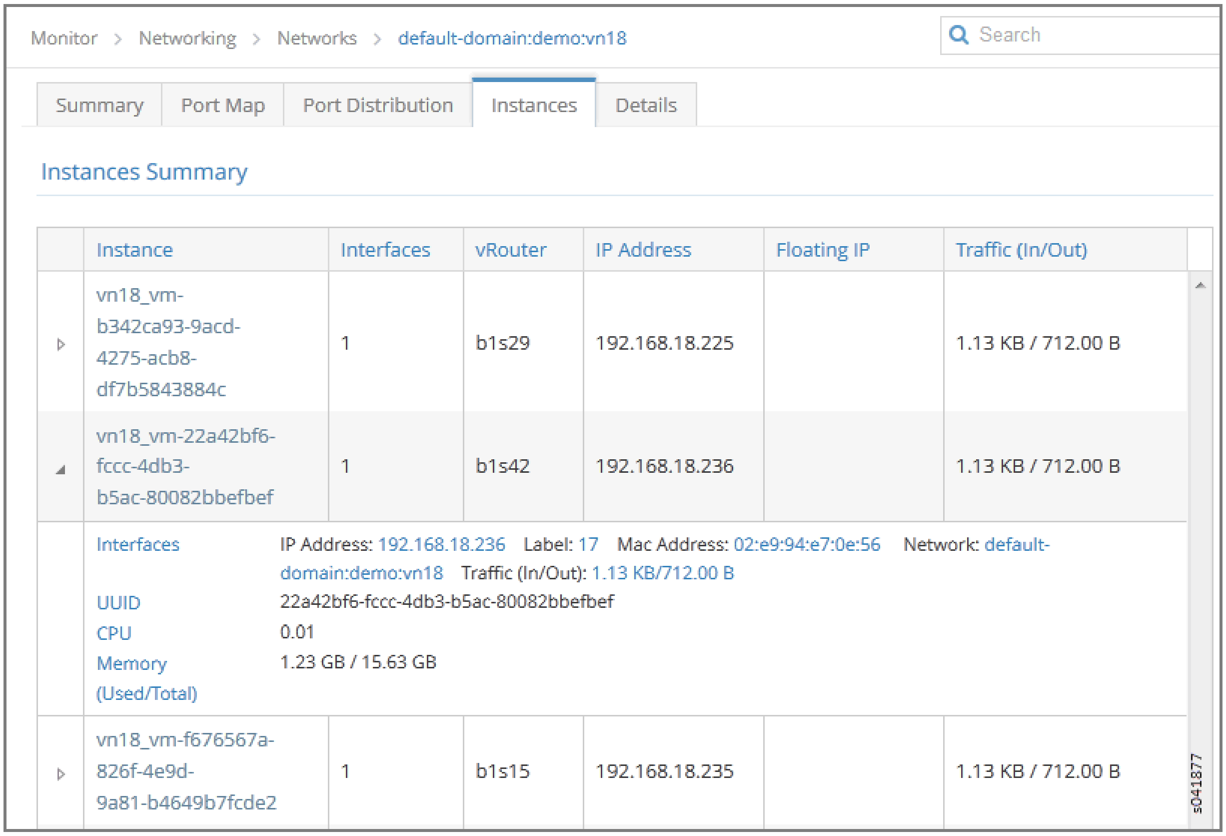
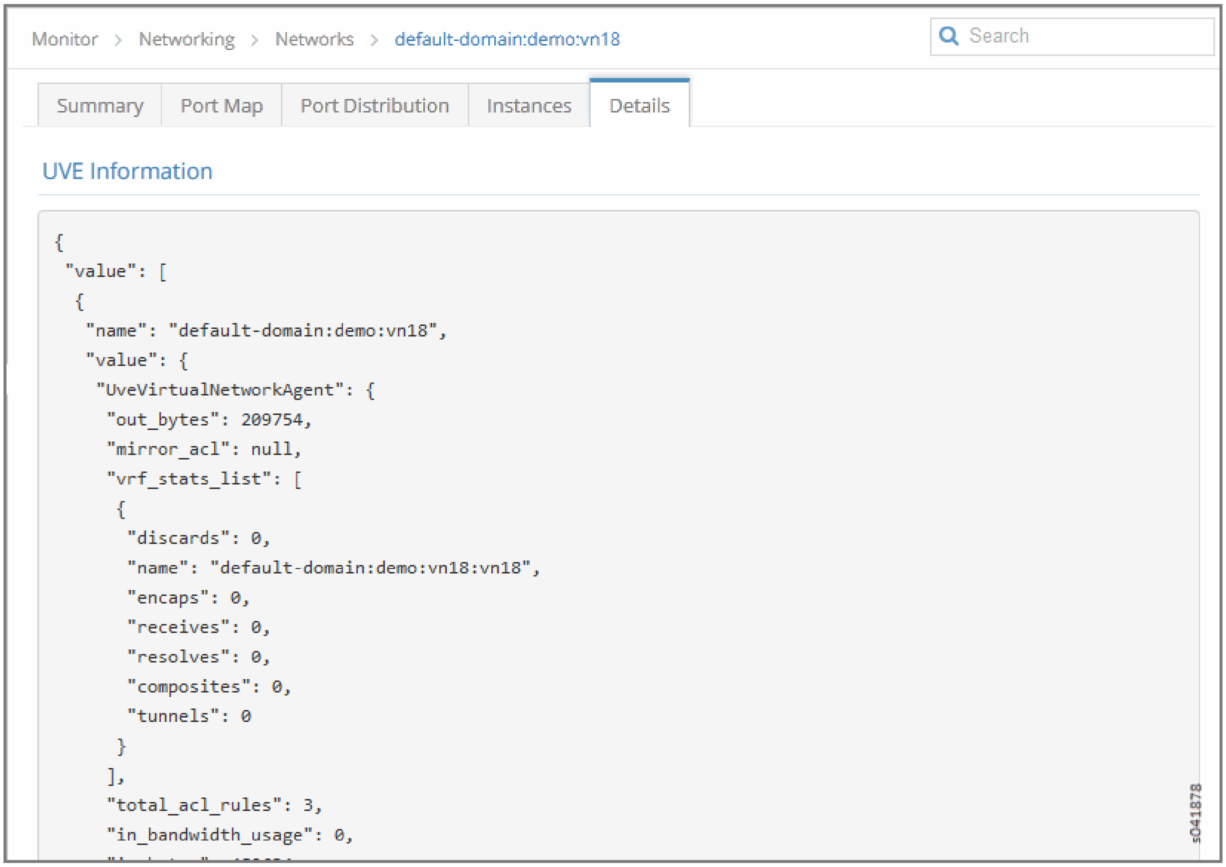


Figure 166 on page 429 shows the **Details** tab for an individual network, which displays the code used to define this network --the User Virtual Environment (UVE) code.

Figure 166: Individual Network Details Tab



Query > Flows

IN THIS SECTION

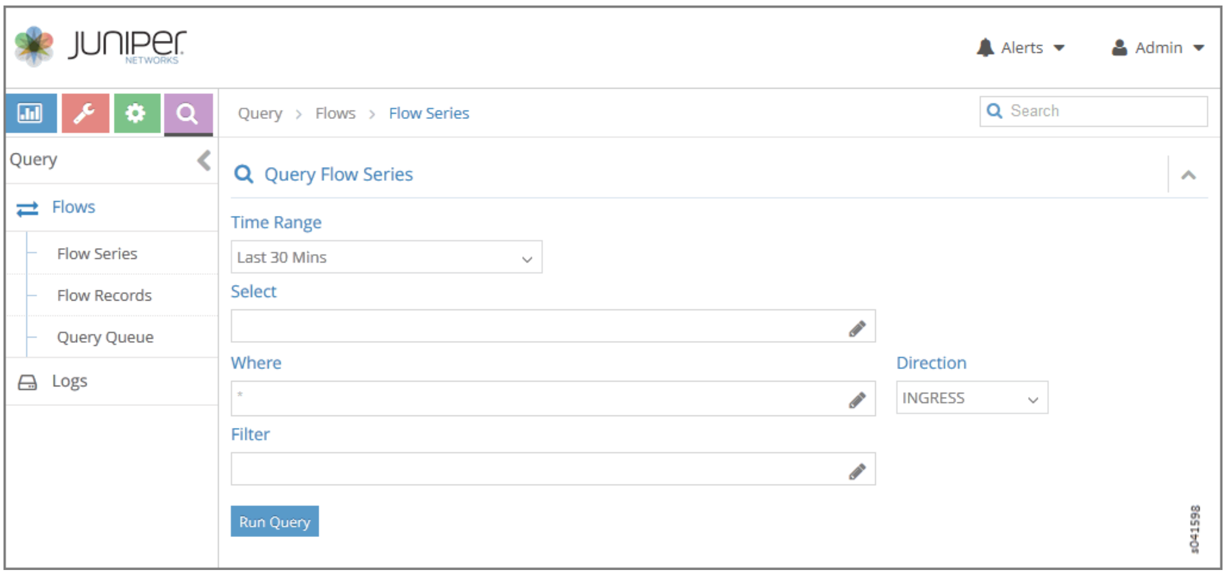
- Query > Flows > Flow Series | 430
- Example: Query Flow Series | 433
- Query > Flow Records | 435
- Query > Flows > Query Queue | 438

Select **Query > Flows** to perform rich and complex SQL-like queries on flows in the Contrail Controller. You can use the query results for such things as gaining insight into the operation of applications in a virtual network, performing historical analysis of flow issues, and pinpointing problem areas with flows.

Query > Flows > Flow Series

Select **Query > Flows > Flow Series** to create queries of the flow series table. The results are in the form of time series data for flow series. See Figure 167 on page 430

Figure 167: Query Flow Series Window



The query fields available on the screen for the **Flow Series** tab are described in Table 71 on page 431. Enter query data into the fields to create a SQL-like query to display and analyze flows.

Table 71: Query Flow Series Fields

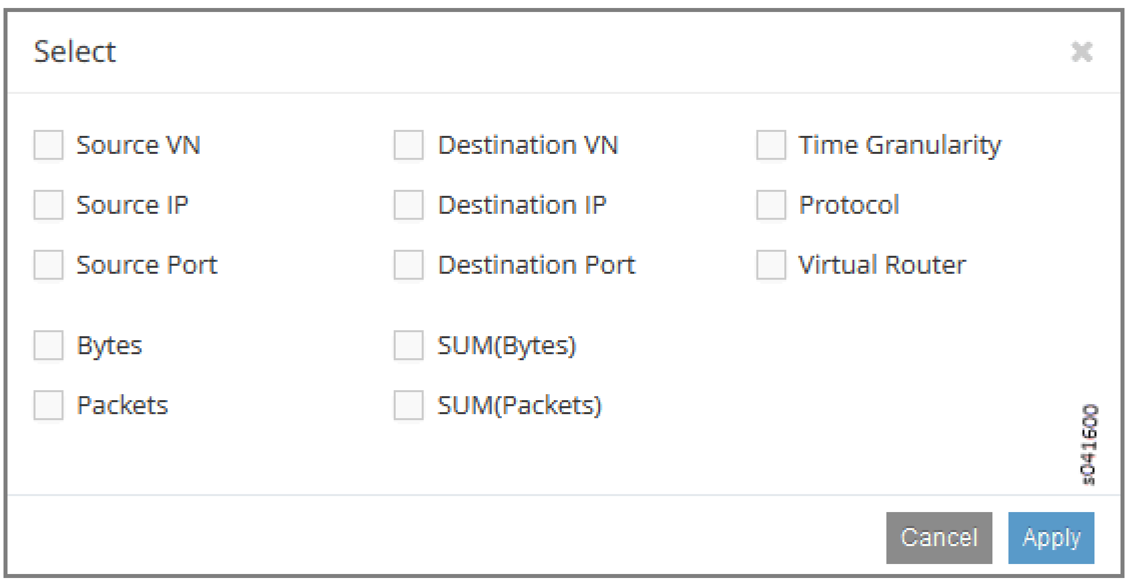
Field	Description
Time Range	Select a range of time to display the flow series: • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom Click **Custom** to enter a specific custom time range in two fields: **From Time** and **To Time**.
Select	Click the edit button (pencil icon) to open a **Select** window (Figure 168 on page 432), where you can click one or more boxes to select the fields to display from the flow series, such as **Source VN, Dest VN, Bytes, Packets**, and more.
Where	Click the edit button (pencil icon) to open a query-writing window, where you can specify query values for variables such as **sourcevn, sourceip, destvn, destip, protocol, sport, dport**.
Direction	Select the desired flow direction: **INGRESS** or **EGRESS**.
Filter	Click the edit button (pencil icon) to open a **Filter** window (Figure 169 on page 433), where you can select filter items to sort by, the sort order, and limits to the number of results returned.
Run Query	Click **Run Query** to retrieve the flows that match the query you created. The flows are listed on the lower portion of the screen in a box with columns identifying the selected fields for each flow.
(graph buttons)	When **Time Granularity** is selected, you have the option to view results in graph or flowchart form. Graph buttons appear on the screen above the **Export** button. Click a graph button to transform the tabular results into a graphical chart display.

Table 71: Query Flow Series Fields *(Continued)*

Field	Description
Export	The Export button is displayed after you click **Run Query**. This allows you to export the list of flows to a text `.csv` file.

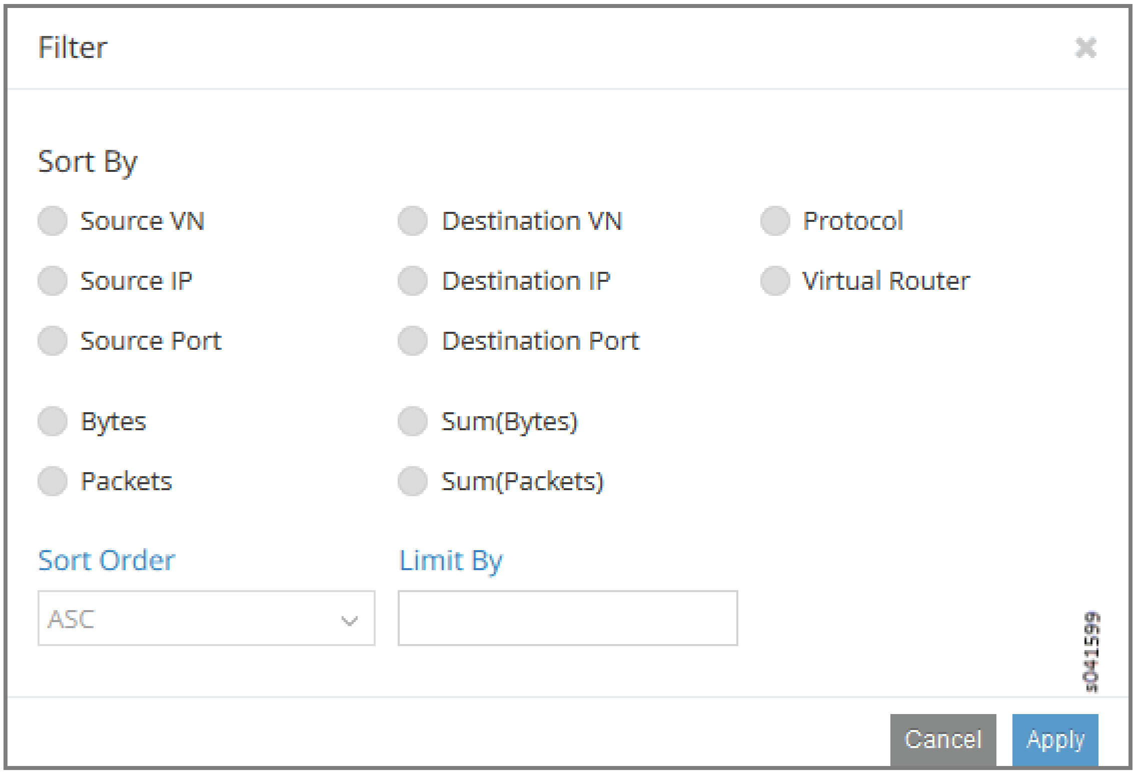
The **Select** window allows you to select one or more attributes of a flow series by clicking the check box for each attribute desired, see Figure 168 on page 432. The upper section of the **Select** window includes field names, and the lower portion lets you select units. Select **Time Granularity** and then select **SUM(Bytes)** or **SUM(Packets)** to aggregate bytes and packets in intervals.

Figure 168: Flow Series Select



Use the **Filter** window to refine the display of query results for flows, by defining an attribute by which to sort the results, the sort order of the results, and any limit needed to restrict the number of results. See Figure 169 on page 433.

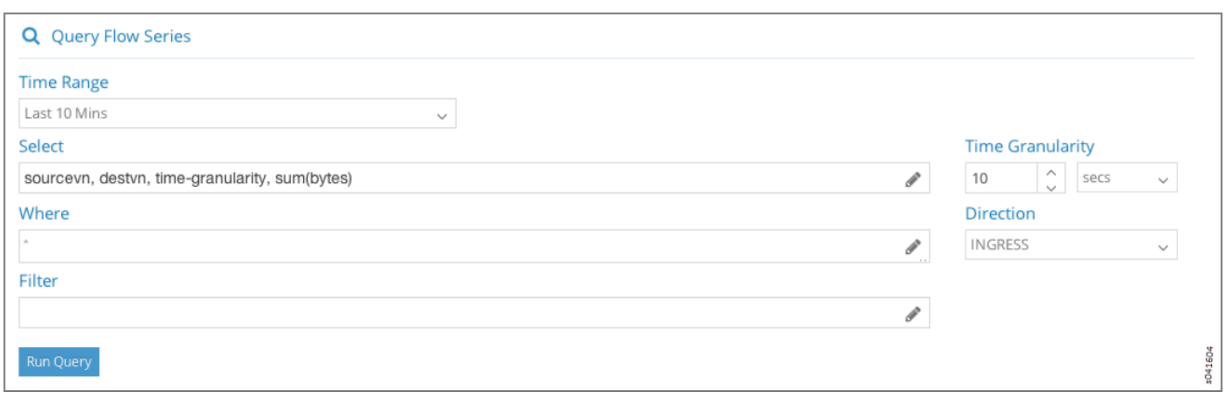
Figure 169: Flow Series Filter



Example: Query Flow Series

The following is an example flow series query that returns the time series of the summation traffic in bytes for all combinations of source VN and destination VN for the last 10 minutes, with the bytes aggregated in 10 second intervals. See Figure 170 on page 433.

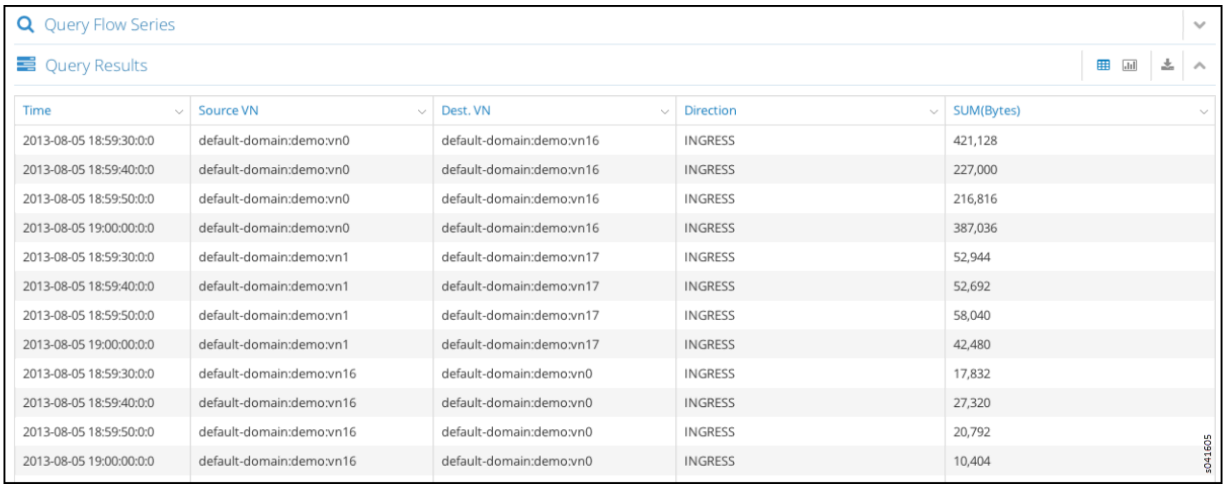
Figure 170: Example: Query Flow Series



The query returns tabular time series data, see Figure 171 on page 434, 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

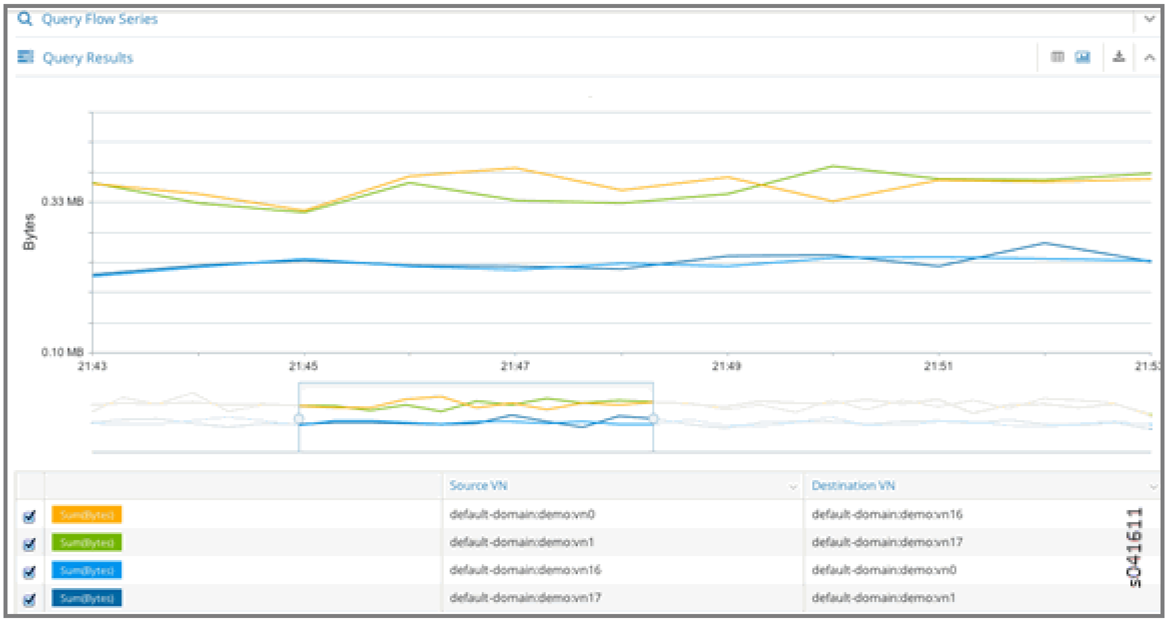
Figure 171: Query Flow Series Tabular Results



Time	Source VN	Dest. VN	Direction	SUM(Bytes)
2013-08-05 18:59:30:0:0	default-domain:demo:vn0	default-domain:demo:vn16	INGRESS	421,128
2013-08-05 18:59:40:0:0	default-domain:demo:vn0	default-domain:demo:vn16	INGRESS	227,000
2013-08-05 18:59:50:0:0	default-domain:demo:vn0	default-domain:demo:vn16	INGRESS	216,816
2013-08-05 19:00:00:0:0	default-domain:demo:vn0	default-domain:demo:vn16	INGRESS	387,036
2013-08-05 18:59:30:0:0	default-domain:demo:vn1	default-domain:demo:vn17	INGRESS	52,944
2013-08-05 18:59:40:0:0	default-domain:demo:vn1	default-domain:demo:vn17	INGRESS	52,692
2013-08-05 18:59:50:0:0	default-domain:demo:vn1	default-domain:demo:vn17	INGRESS	58,040
2013-08-05 19:00:00:0:0	default-domain:demo:vn1	default-domain:demo:vn17	INGRESS	42,480
2013-08-05 18:59:30:0:0	default-domain:demo:vn16	default-domain:demo:vn0	INGRESS	17,832
2013-08-05 18:59:40:0:0	default-domain:demo:vn16	default-domain:demo:vn0	INGRESS	27,320
2013-08-05 18:59:50:0:0	default-domain:demo:vn16	default-domain:demo:vn0	INGRESS	20,792
2013-08-05 19:00:00:0:0	default-domain:demo:vn16	default-domain:demo:vn0	INGRESS	10,404

Because **Time Granularity** is selected, the results can also be displayed as graphical charts. Click the graph button on the right side of the tabular results. The results are displayed in a graphical flow chart. See Figure 172 on page 435.

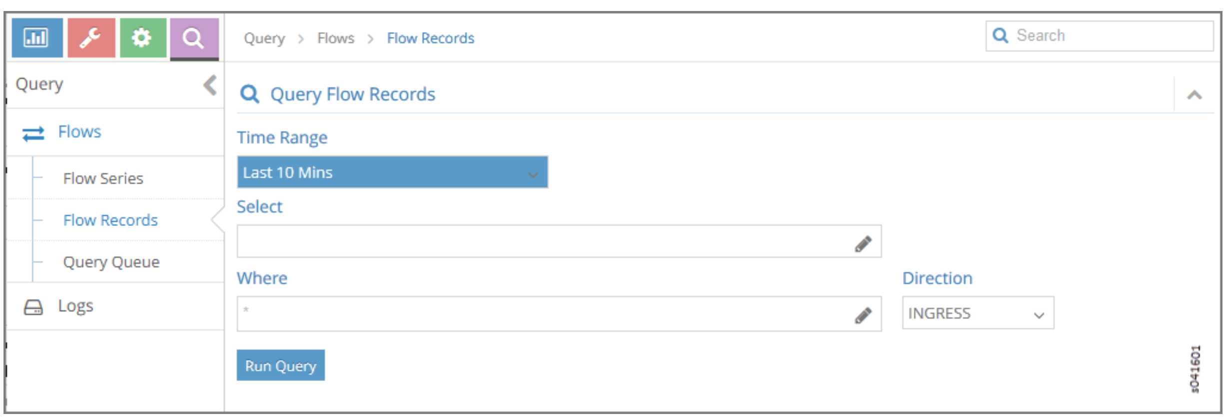
Figure 172: Query Flow Series Graphical Results



Query > Flow Records

Select **Query > Flow Records** to create queries of individual flow records for detailed debugging of connectivity issues between applications and virtual machines. Queries at this level return records of the active flows within a given time period.

Figure 173: Flow Records



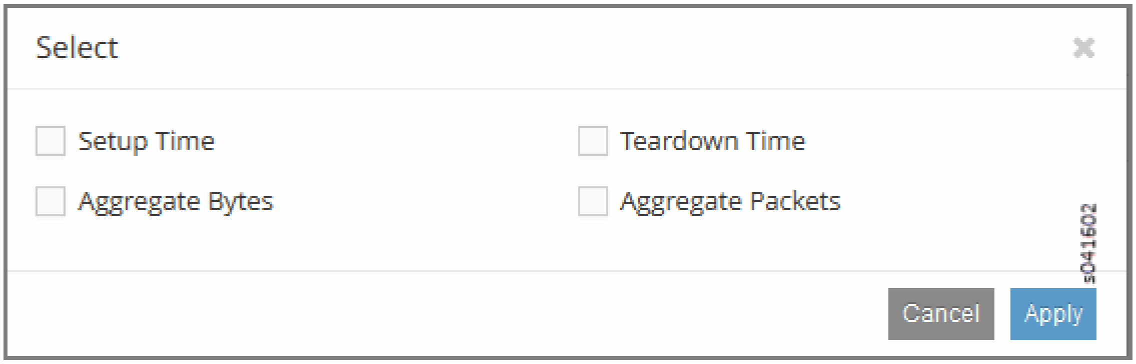
The query fields available on the screen for the **Flow Records** tab are described in Table 72 on page 436. Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 72: Query Flow Records Fields

Field	Description
Time Range	Select a range of time for the flow records: • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom Click **Custom** to enter a specified custom time range in two fields: **From Time** and **To Time**.
Select	Click the edit button (pencil icon) to open a **Select** window (Figure 174 on page 437), where you can click one or more boxes to select attributes to display for the flow records, including **Setup Time, Teardown Time, Aggregate Bytes,** and **Aggregate Packets**.
Where	Click the edit button (pencil icon) to open a query-writing window where you can specify query values for **sourcevn, sourceip, destvn, destip, protocol, sport, dport**. .
Direction	Select the desired flow direction: **INGRESS** or **EGRESS**.
Run Query	Click **Run Query** to retrieve the flow records that match the query you created. The records are listed on the lower portion of the screen in a box with columns identifying the fields for each flow.
Export	The **Export** button is displayed after you click **Run Query**, allowing you to export the list of flows to a text **.csv** file.

The **Select** window allows you to select one or more attributes to display for the flow records selected, see Figure 174 on page 437.

Figure 174: Flow Records Select Window

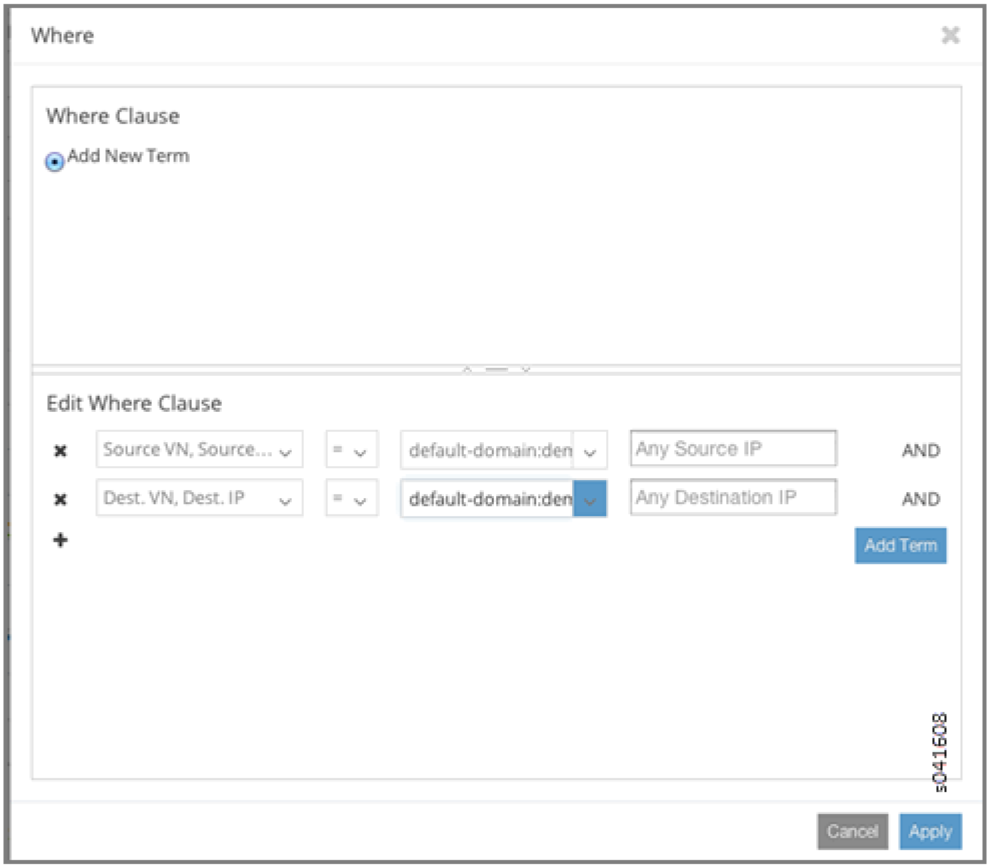


You can restrict the query to a particular source VN and destination VN combination using the **Where** section.

The **Where Clause** supports logical AND and logical OR operations, and is modeled as a logical OR of multiple AND terms. For example: ((term1 AND term2 AND term3..) OR (term4 AND term5) OR...).

Each term is a single variable expression such as **Source VN = VN1**.

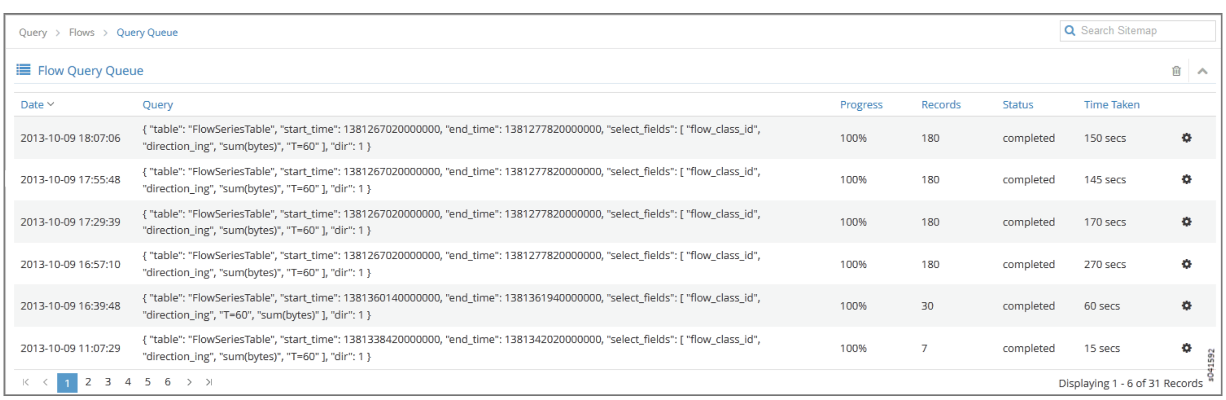
Figure 175: Where Clause Window



Query > Flows > Query Queue

Select **Query > Flows > Query Queue** to display queries that are in the queue waiting to be performed on the data. See Figure 176 on page 438.

Figure 176: Flows Query Queue



The query fields available on the screen for the **Flow Records** tab are described in Table 73 on page 439. Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 73: Query Flow Records Fields

Field	Description
Date	The date and time the query was started.
Query	A display of the parameters set for the query.
Progress	The percentage completion of the query to date.
Records	The number of records matching the query to date.
Status	The status of the query, such as **completed**.
Time Taken	The amount of time in seconds it has taken the query to return the matching records.
(Action icon)	Click the **Action** icon and select **View Results** to view a list of the records that match the query, or click **Delete** to remove the query from the queue.

RELATED DOCUMENTATION

Understanding Flow Sampling

Query > Logs

IN THIS SECTION

- Query > Logs Menu Options | **440**
- Query > Logs > System Logs | **440**
- Sample Query for System Logs | **442**

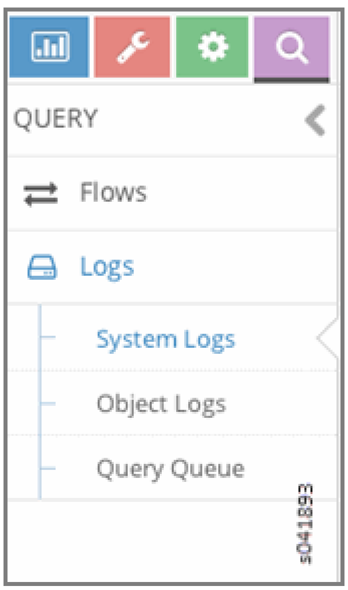
- Query > Logs > Object Logs | 444

The **Query > Logs** option allows you to access the system log and object log activity of any Contrail Controller component from one central location.

Query > Logs Menu Options

Click **Query > Logs** to access the **Query Logs** menu, where you can select **System Logs** to view system log activity, **Object Logs** to view object logs activity, and **Query Queue** to create custom queries of log activity; see Figure 177 on page 440.

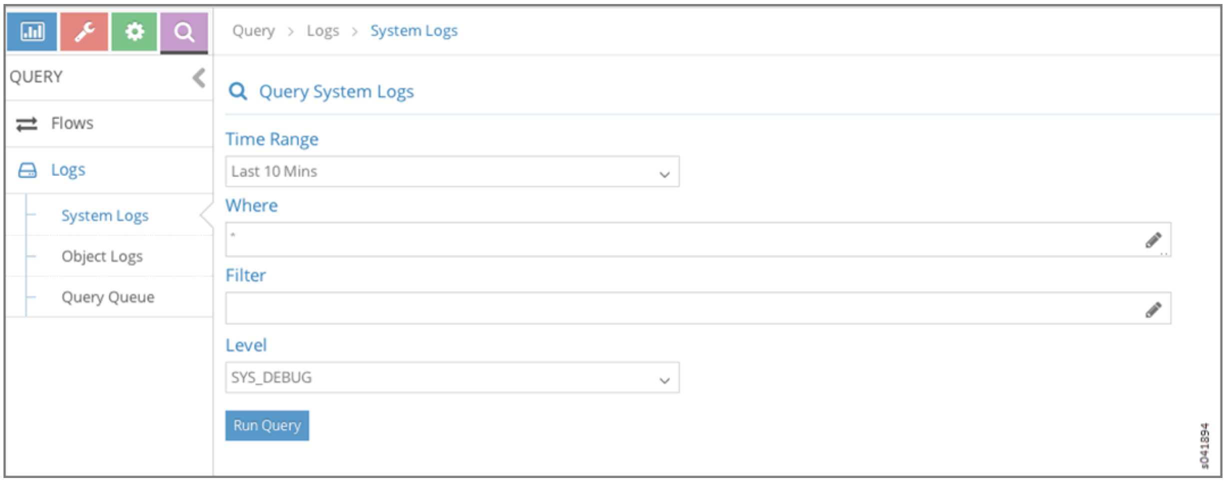
Figure 177: Query > Logs



Query > Logs > System Logs

Click **Query > Logs > System Logs** to access the **Query System Logs** menu, where you can view system logs according to criteria that you determine. See Figure 178 on page 441.

Figure 178: Query > Logs > System Logs



The query fields available on the **Query System Logs** screen are described in Table 74 on page 441.

Table 74: Query System Logs Fields

Field	Description
Time Range	Select a range of time for which to see the system logs: • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom If you click Custom, enter a desired time range in two new fields: **From Time** and **To Time**.
Where	Click the edit button (pencil icon) to open a query-writing window, where you can specify query values for variables such as Source, Module, MessageType, and the like, in order to retrieve specific information.

Table 74: Query System Logs Fields *(Continued)*

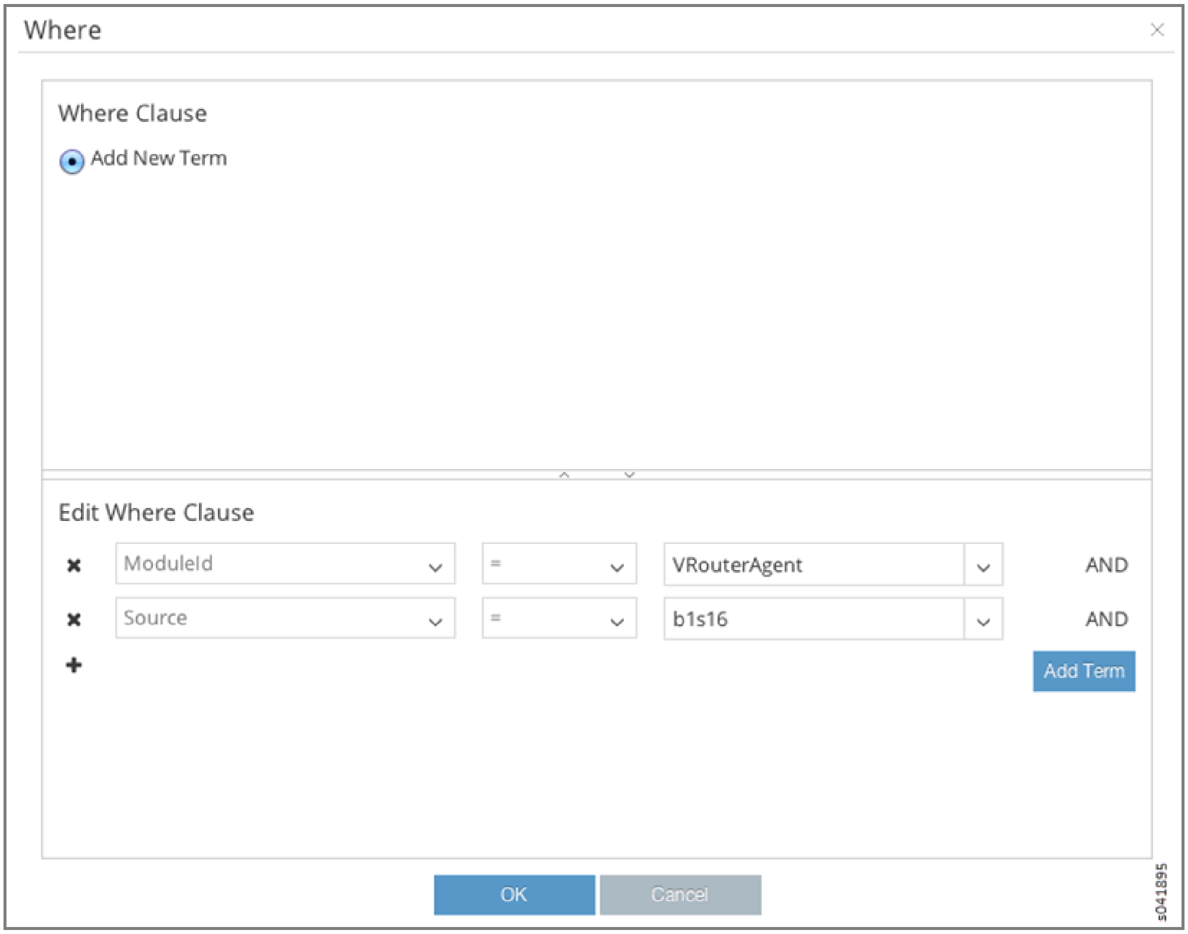
Field	Description
Level	Select the message severity level to view: • SYS_NOTICE • SYS_EMERG • SYS_ALERT • SYS_CRIT • SYS_ERR • SYS_WARN • SYS_INFO • SYS_DEBUG
Run Query	Click this button to retrieve the system logs that match the query. The logs are listed in a box with columns showing the **Time**, **Source**, **Module Id**, **Category**, **Log Type**, and **Log** message.
Export	This button appears after you click **Run Query**, allowing you to export the list of system messages to a text/csv file.

Sample Query for System Logs

This section shows a sample system logs query designed to show all **System Logs** from `ModuleId = VRouterAgent on Source = b1s16` and filtered by **Level** `= SYS_DEBUG`.

1. At the **Query System Logs** screen, click in the **Where** field to access the **Where** query screen and enter information defining the location to query in the **Edit Where Clause** section and click **OK**; see Figure 179 on page 443.

Figure 179: Edit Where Clause



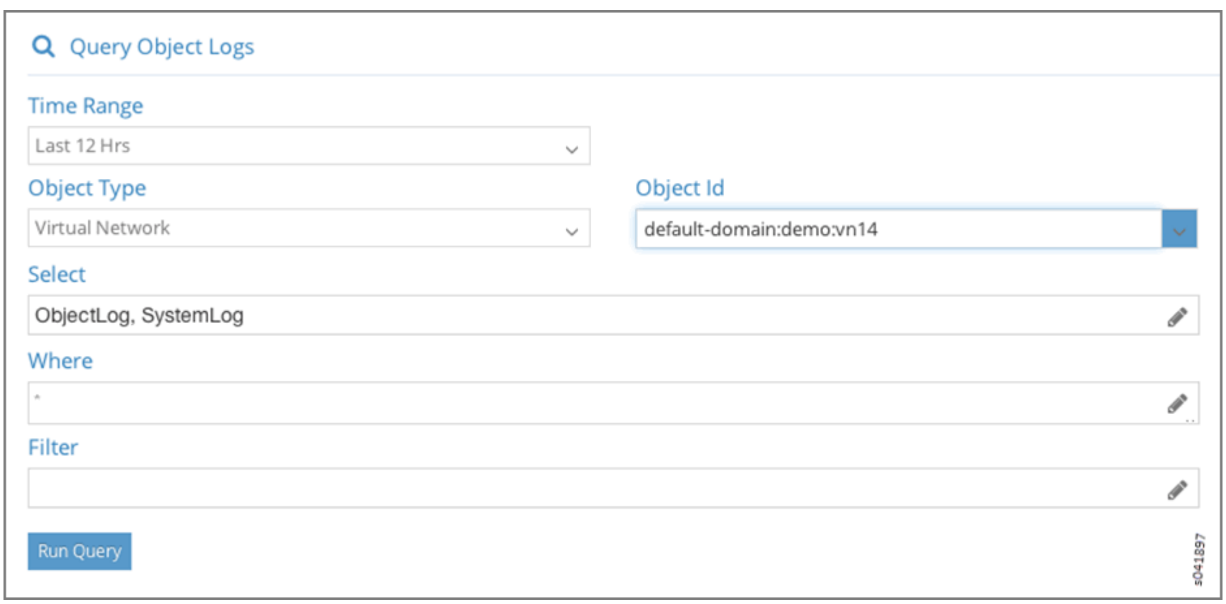
2. The information you defined at the Where screen displays on the **Query System Logs**. Enter any more defining information needed; see Figure 180 on page 444. When finished, click **Run Query** to display the results.

Figure 180: 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 181 on page 444.

Figure 181: Query > Logs > Object Logs



The query fields available on the **Object Logs** screen are described in Table 75 on page 445.

Table 75: Object Logs Query Fields

Field	Description
Time Range	Select a range of time for which to see the logs: • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom If you click Custom, enter a desired time range in two new fields: **From Time** and **To Time**.
Object Type	Select the object type for which to show logs: • Virtual Network • Virtual Machine • Virtual Router • BGP Peer • Routing Instance • XMPP Connection
Object Id	Select from a list of available identifiers the name of the object you wish to use.
Select	Click the edit button (pencil icon) to open a window where you can select searchable types by clicking a checkbox: • ObjectLog • SystemLog

Table 75: Object Logs Query Fields *(Continued)*

Field	Description
Where	Click the edit button (pencil icon) to open the query-writing window, where you can specify query values for variables such as **Source**, **ModuleId**, and **MessageType**, in order to retrieve information as specific as you wish.
Run Query	Click this button to retrieve the system logs that match the query. The logs are listed in a box with columns showing the **Time**, **Source**, **Module Id**, **Category**, **Log Type**, and **Log** message.
Export	This button appears after you click **Run Query**, allowing you to export the list of system messages to a text/csv file.

Example: Debugging Connectivity Using Monitoring for Troubleshooting

IN THIS SECTION

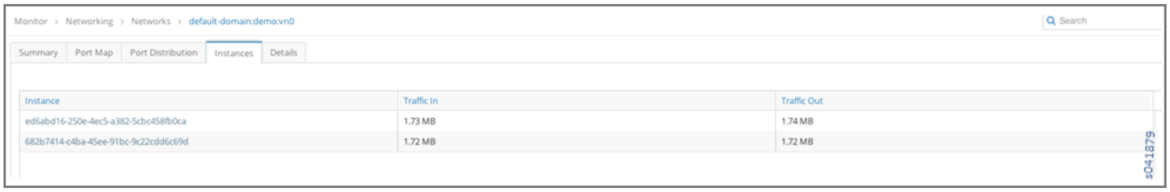
- Using Monitoring to Debug Connectivity | 446

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.

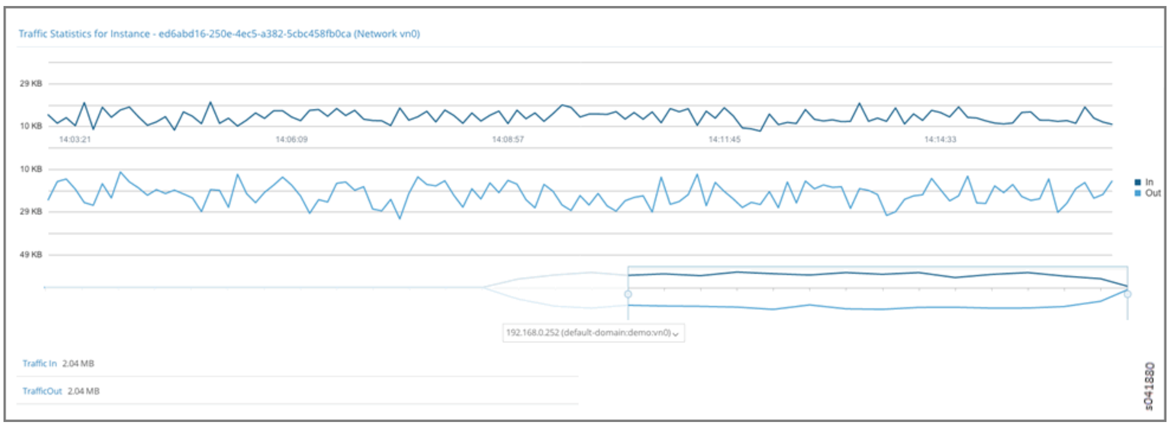
1. Navigate to **Monitor -> Networking -> Networks ->** `default-domain:demo:vn0`, **Instance** `ed6abd16-250e-4ec5-a382-5cbc458fb0ca` with **IP address** `192.168.0.252` in the virtual network `vn0`; see Figure 182 on page 447

Figure 182: Navigate to Instance



2. Click the instance to view **Traffic Statistics for Instance**. see Figure 183 on page 447.

Figure 183: 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 184 on page 447 and Figure 185 on page 448.

Figure 184: Navigate to Instance

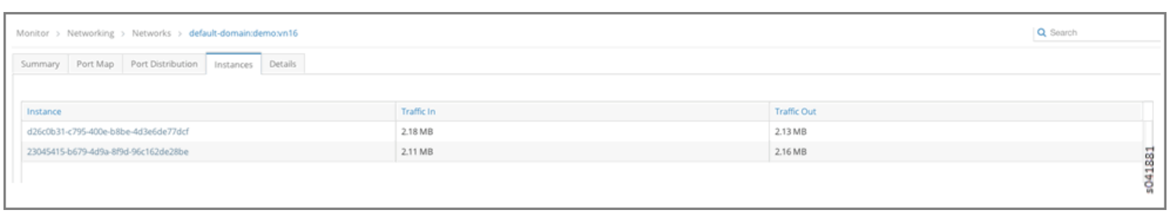
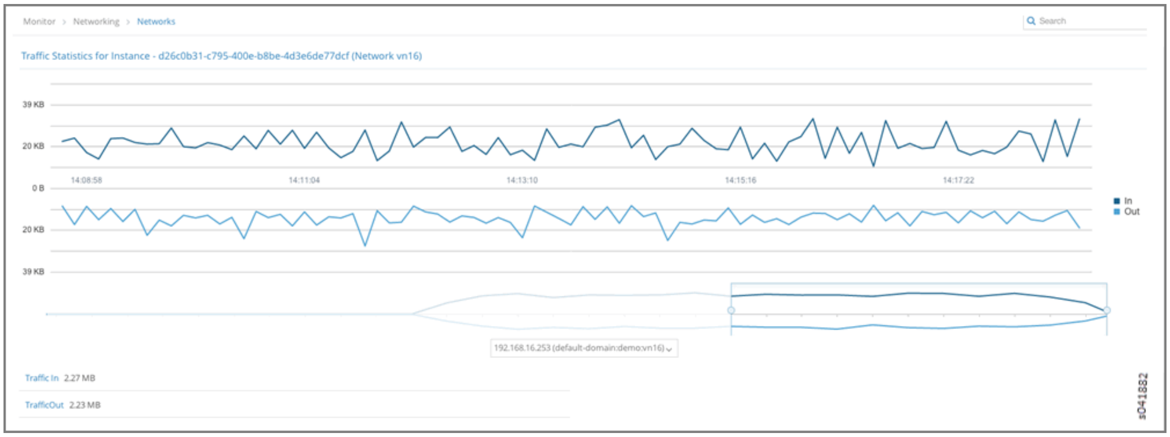
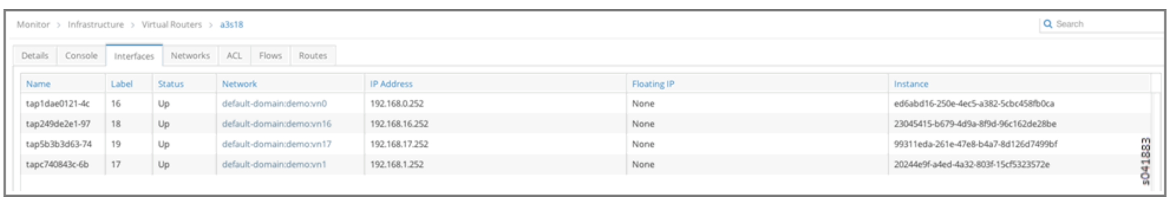


Figure 185: Traffic Statistics for Instance



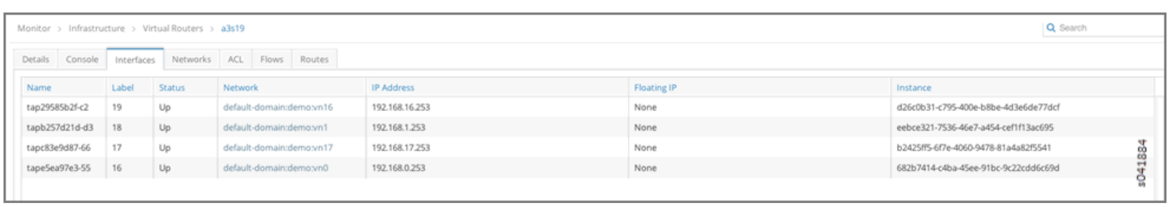
4. From **Monitor->Infrastructure->Virtual Routers->a3s18->Interfaces**, we can see that **Instance** `ed6abd16-250e-4ec5-a382-5cbc458fb0ca` is hosted on **Virtual Router** `a3s18`; see Figure 186 on page 448.

Figure 186: Navigate to a3s18 Interfaces



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 187 on page 448.

Figure 187: Navigate to a3s19 Interfaces



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 188 on page 449 and Figure 189 on page 449.

Figure 188: ACL Connectivity a3s18

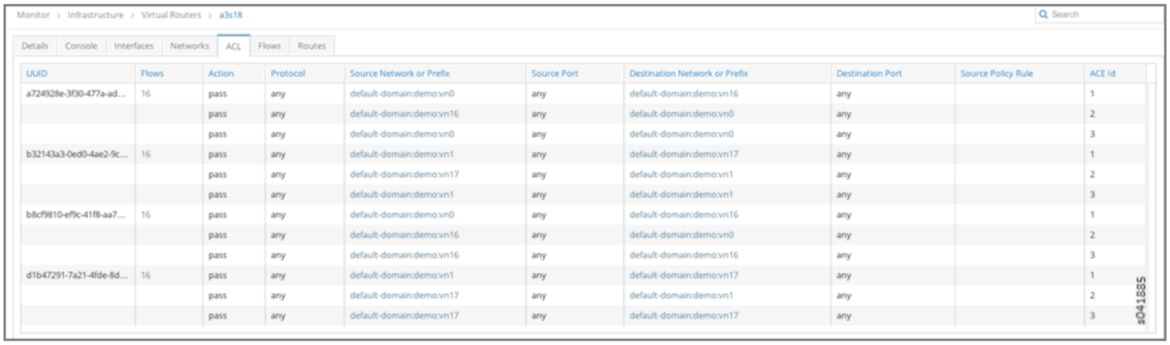
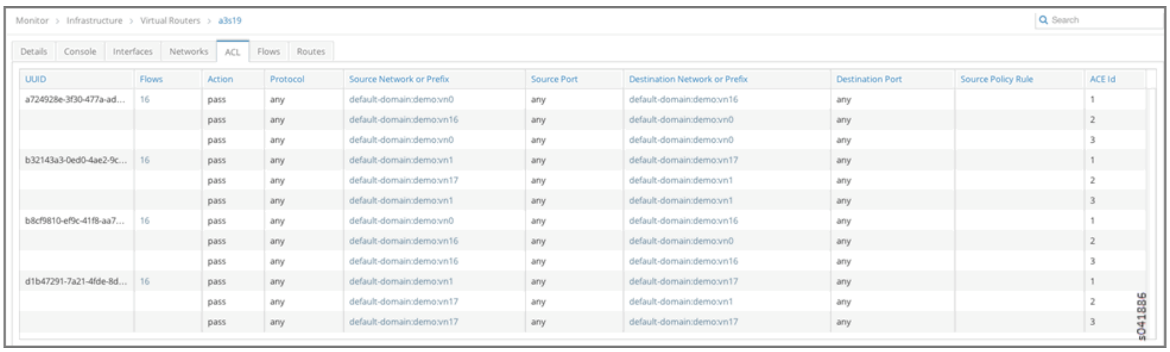
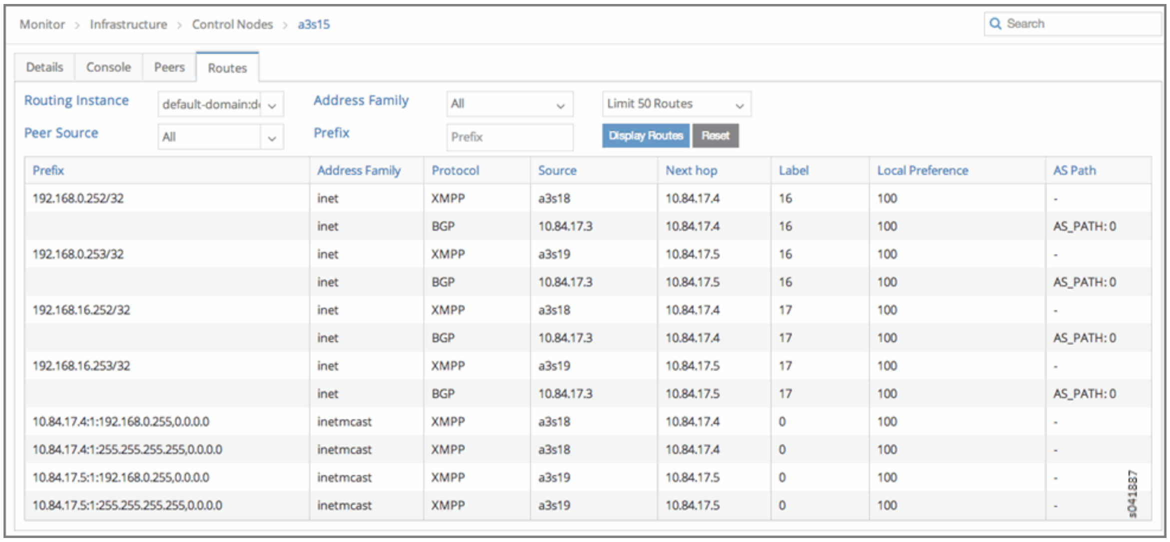


Figure 189: ACL Connectivity a3s19



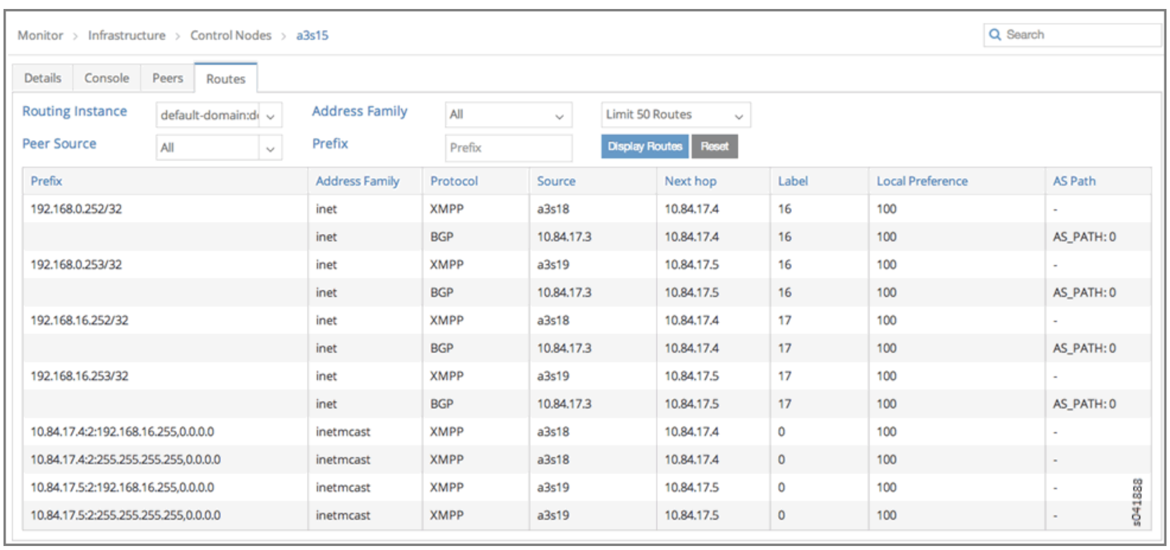
7. Next, verify the routes on the control node for routing instances `default-domain:demo:vn0:vn0` and `default-domain:demo:vn16:vn16`; see Figure 190 on page 450 and Figure 191 on page 450.

Figure 190: Routes default-domain:demo:vn0:vn0



Prefix	Address Family	Protocol	Source	Next hop	Label	Local Preference	AS Path
192.168.0.252/32	inet	XMPP	a3s18	10.84.17.4	16	100	-
	inet	BGP	10.84.17.3	10.84.17.4	16	100	AS_PATH: 0
192.168.0.253/32	inet	XMPP	a3s19	10.84.17.5	16	100	-
	inet	BGP	10.84.17.3	10.84.17.5	16	100	AS_PATH: 0
192.168.16.252/32	inet	XMPP	a3s18	10.84.17.4	17	100	-
	inet	BGP	10.84.17.3	10.84.17.4	17	100	AS_PATH: 0
192.168.16.253/32	inet	XMPP	a3s19	10.84.17.5	17	100	-
	inet	BGP	10.84.17.3	10.84.17.5	17	100	AS_PATH: 0
10.84.17.4:1:192.168.0.255,0.0.0.0	inetmcast	XMPP	a3s18	10.84.17.4	0	100	-
10.84.17.4:1:255.255.255.255,0.0.0.0	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	-

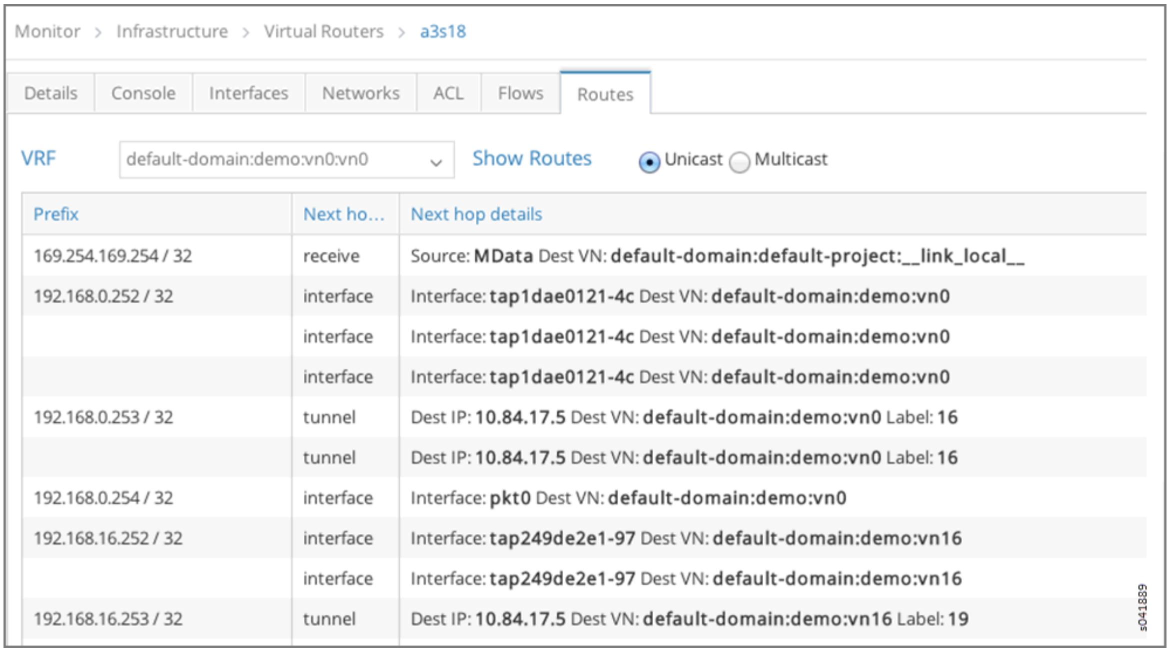
Figure 191: Routes default-domain:demo:vn16:vn16



Prefix	Address Family	Protocol	Source	Next hop	Label	Local Preference	AS Path
192.168.0.252/32	inet	XMPP	a3s18	10.84.17.4	16	100	-
	inet	BGP	10.84.17.3	10.84.17.4	16	100	AS_PATH: 0
192.168.0.253/32	inet	XMPP	a3s19	10.84.17.5	16	100	-
	inet	BGP	10.84.17.3	10.84.17.5	16	100	AS_PATH: 0
192.168.16.252/32	inet	XMPP	a3s18	10.84.17.4	17	100	-
	inet	BGP	10.84.17.3	10.84.17.4	17	100	AS_PATH: 0
192.168.16.253/32	inet	XMPP	a3s19	10.84.17.5	17	100	-
	inet	BGP	10.84.17.3	10.84.17.5	17	100	AS_PATH: 0
10.84.17.4:2:192.168.16.255,0.0.0.0	inetmcast	XMPP	a3s18	10.84.17.4	0	100	-
10.84.17.4:2:255.255.255.255,0.0.0.0	inetmcast	XMPP	a3s18	10.84.17.4	0	100	-
10.84.17.5:2:192.168.16.255,0.0.0.0	inetmcast	XMPP	a3s19	10.84.17.5	0	100	-
10.84.17.5:2:255.255.255.255,0.0.0.0	inetmcast	XMPP	a3s19	10.84.17.5	0	100	-

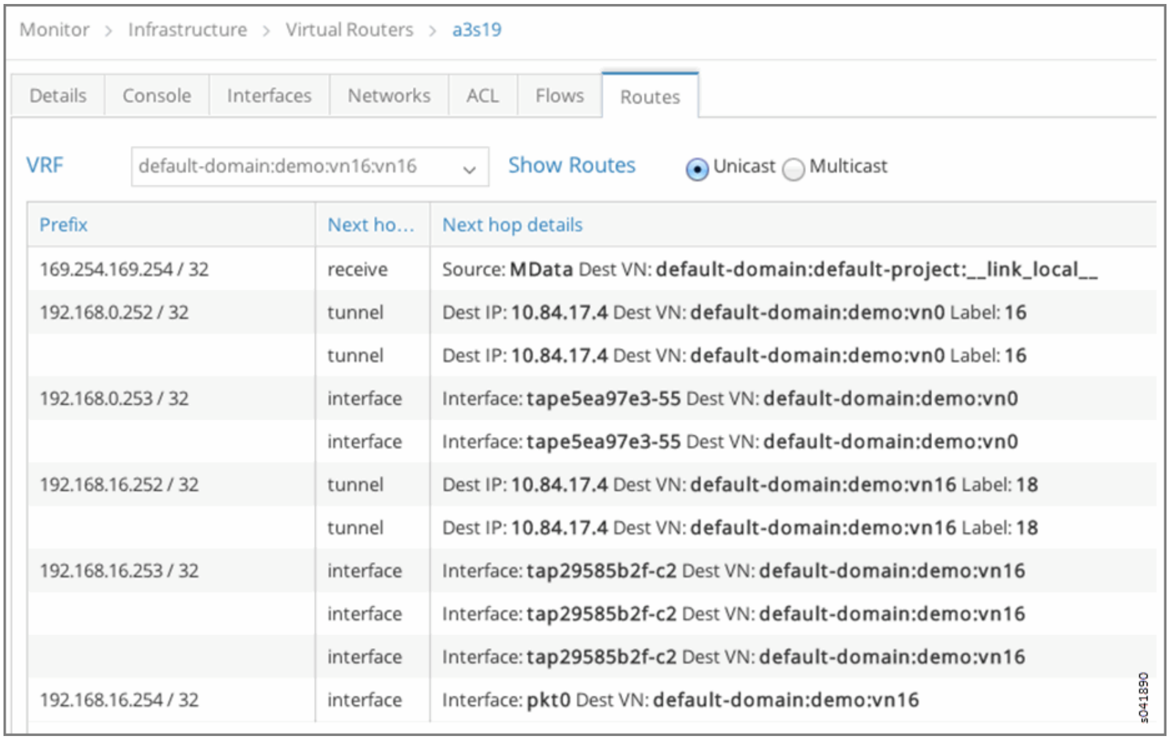
8. We can see that VRF `default-domain:demo:vn0:vn0` on Virtual Router `a3s18` has the appropriate route and next hop to reach VRF `default-domain:demo:front-end` on Virtual Router `a3s19`; see Figure 192 on page 451.

Figure 192: Verify Route and Next Hop a3s18



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 193 on page 452.

Figure 193: Verify Route and Next Hop a3s19



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 194 on page 452 and Figure 195 on page 453.

Figure 194: Flows for a3s18

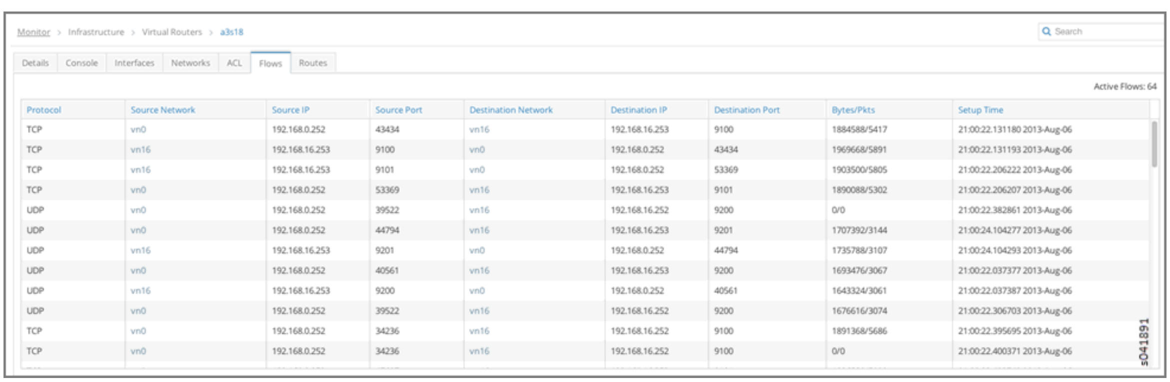


Figure 195: Flows for a3s19