

Contrail[™]

Contrail Getting Started Guide

Published
2025-08-18

RELEASE
5.0.2

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

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

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

Contrail™ Contrail Getting Started Guide

5.0.2

Copyright © 2025 Juniper Networks, Inc. All rights reserved.

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

YEAR 2000 NOTICE

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

END USER LICENSE AGREEMENT

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

Table of Contents

[About This Guide | xi](#)

1

Overview

[Understanding Contrail | 2](#)

[Contrail Overview | 2](#)

[Contrail Description | 3](#)

2

Installing and Upgrading Contrail

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

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

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

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

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

[Downloading Installation Software | 13](#)

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

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

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

[Contrail Global Controller | 39](#)

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

[Installation and Configuration Scenarios | 51](#)

[Simple Underlay Connectivity without Gateway | 51](#)

[Dynamic Kernel Module Support \(DKMS\) for vRouter | 55](#)

[Upgrading Contrail Software | 56](#)

[Contrail In-Service Software Upgrade from Releases 3.2 and 4.1 to 5.0.x using Ansible Deployer | 56](#)

[Contrail In-Service Software Upgrade from Releases 3.2 and 4.1 to 5.0.x using Helm Deployer | 66](#)

Backup and Restore Contrail Software | 78

Backing up Contrail Databases in JSON Format | 78

Contrail Command | 86

Configuring Contrail Command | 86

- Requirements | 86

- Overview | 87

- Configuration | 87

- Sample command_servers.yml File | 89

Deploying Contrail Cluster using the Contrail Command UI | 93

- Requirements | 94

- Overview | 94

- Configuration | 96

Deploying Contrail Cluster using Contrail-Command and instances.yml | 106

Importing Contrail Cluster Data using Contrail Command | 113

Multicloud Contrail | 118

Contrail Deployment on Microsoft Azure | 118

Deploying Contrail on Microsoft Azure | 119

- Deployment of Contrail on Azure | 119

- Deleting Contrail Deployment from Azure | 124

On-Premise and Azure Multicloud Deployment | 125

- Installing On-Premise Contrail | 125

- Extending On-Premise Contrail To Microsoft Azure | 128

Modifying Multicloud Topology | 137

Using Contrail with Red Hat | 138

Deploying Contrail with Red Hat OpenStack Platform Director 13 | 138

Provisioning Red Hat OpenShift Container Platform Clusters Using Ansible Deployer | 193

- Installing a Standalone OpenShift Cluster Using Ansible Deployer | 193

- Provisioning of Nested OpenShift Clusters Using Ansible Deployer—Beta | 203

- Configure network connectivity to Contrail configuration and data plane functions | 203

- Installing Nested OpenShift Cluster using Ansible Deployer | 207

Using Contrail with Juju Charms | 208

Deploying Contrail by Using Juju Charms | 208

Preparing to Deploy Contrail by Using Juju Charms | 209

Deploying Contrail Charms | 211

Deploying Contrail Charms in a Bundle | 211

Deploying Juju Charms Manually | 218

Options for Juju Charms | 223

Configuring Contrail

Configuring Virtual Networks | 232

Creating Projects in OpenStack for Configuring Tenants in Contrail | 232

Creating a Virtual Network with Juniper Networks Contrail | 234

Creating a Virtual Network with OpenStack Contrail | 238

Creating an Image for a Project in OpenStack Contrail | 240

Creating a Floating IP Address Pool | 244

Using Security Groups with Virtual Machines (Instances) | 246

Security Groups Overview | 246

Creating Security Groups and Adding Rules | 247

Support for IPv6 Networks in Contrail | 251

Configuring EVPN and VXLAN | 255

Configuring the VXLAN Identifier Mode | 257

Configuring Forwarding | 259

Configuring the VXLAN Identifier | 260

Configuring Encapsulation Methods | 261

Example of Deploying a Multi-Tier Web Application Using Contrail | 265

Example: Deploying a Multi-Tier Web Application | 265

Multi-Tier Web Application Overview | 265

Example: Setting Up Virtual Networks for a Simple Tiered Web Application | 266

Verifying the Multi-Tier Web Application | 269

Sample Addressing Scheme for Simple Tiered Web Application | 269

Sample Physical Topology for Simple Tiered Web Application | 270

Sample Physical Topology Addressing | 271

Sample Network Configuration for Devices for Simple Tiered Web Application | 273

Configuring Services | 280

Configuring DNS Servers | 280

- DNS Overview | 280

- Defining Multiple Virtual Domain Name Servers | 281

- IPAM and Virtual DNS | 282

- DNS Record Types | 282

- Configuring DNS Using the Interface | 283

- Configuring DNS Using Scripts | 291

Support for Multicast | 293

- Subnet Broadcast | 293

- All-Broadcast/Limited-Broadcast and Link-Local Multicast | 294

- Host Broadcast | 295

Using Static Routes with Services | 295

- Static Routes for Service Instances | 295

- Configuring Static Routes on a Service Instance | 296

- Configuring Static Routes on Service Instance Interfaces | 297

- Configuring Static Routes as Host Routes | 299

Configuring Metadata Service | 300

Configuring Service Chaining | 302

Service Chaining | 302

- Service Chaining Basics | 302

- Service Chaining Configuration Elements | 304

Service Chaining MX Series Configuration | 306

ECMP Load Balancing in the Service Chain | 308

Customized Hash Field Selection for ECMP Load Balancing | 309

Using the Contrail Heat Template | 314

Service Chain Route Reorigination | 319

Service Instance Health Checks | 341

- Health Check Object | 342

Bidirectional Forwarding and Detection Health Check over Virtual Machine Interfaces	346
Bidirectional Forwarding and Detection Health Check for BGPaaS	347
Health Check of Transparent Service Chain	347
Service Instance Fate Sharing	347

Examples: Configuring Service Chaining | 349

Example: Creating an In-Network Service Chain by Using Contrail Command | 349

Hardware and Software Requirements	349
Overview	350
Configuration	350

Example: Creating an In-Network-NAT Service Chain by Using Contrail Command | 357

Prerequisites	358
Overview	359
Configuration	359

Example: Creating a Transparent Service Chain by Using Contrail Command | 366

Prerequisites	366
Overview	367
Configuration	367

4

Monitoring and Troubleshooting the Network Using Contrail Analytics

Understanding Contrail Analytics | 376

Understanding Contrail Analytics | 376

Contrail Alerts | 377

Underlay Overlay Mapping in Contrail | 381

Overview: Underlay Overlay Mapping using Contrail Analytics	382
Underlay Overlay Analytics Available in Contrail	382
Architecture and Data Collection	383
New Processes/Services for Underlay Overlay Mapping	383
External Interfaces Configuration for Underlay Overlay Mapping	384
Physical Topology	384
SNMP Configuration	385
Link Layer Discovery Protocol (LLDP) Configuration	385
IPFIX and sFlow Configuration	385
Sending pRouter Information to the SNMP Collector in Contrail	388

- pRouter UVEs | 388
- Contrail User Interface for Underlay Overlay Analytics | 390
- Enabling Physical Topology on the Web UI | 391
- Viewing Topology to the Virtual Machine Level | 391
- Viewing the Traffic of any Link | 391
- Trace Flows | 392
- Search Flows and Map Flows | 393
- Overlay to Underlay Flow Map Schemas | 394
- Module Operations for Overlay Underlay Mapping | 397
- SNMP Collector Operation | 397
- Topology Module Operation | 399
- IPFIX and sFlow Collector Operation | 400
- Troubleshooting Underlay Overlay Mapping | 401
- Script to add pRouter Objects | 401

Configuring Contrail Analytics | 404

Analytics Scalability | 404

High Availability for Analytics | 405

Role-Based Access Control for Analytics | 406

System Log Receiver in Contrail Analytics | 407

- Overview | 408
- Redirecting System Logs to Contrail Collector | 408
- Exporting Logs from Contrail Analytics | 408

Sending Flow Messages to the Contrail System Log | 408

More Efficient Flow Queries | 409

Ceilometer Support in a Contrail Cloud | 410

- Overview | 410
- Ceilometer Details | 411
- Verification of Ceilometer Operation | 411
- Contrail Ceilometer Plugin | 414
- Ceilometer Installation and Provisioning | 417

Using Contrail Analytics to Monitor and Troubleshoot the Network | 418

Monitoring the System | 418

Debugging Processes Using the Contrail Introspect Feature | 422

Monitor > Infrastructure > Dashboard | 427

Monitor Dashboard | 428

Monitor Individual Details from the Dashboard | 428

Using Bubble Charts | 429

Color-Coding of Bubble Charts | 430

Monitor > Infrastructure > Control Nodes | 431

Monitor Control Nodes Summary | 431

Monitor Individual Control Node Details | 432

Monitor Individual Control Node Console | 434

Monitor Individual Control Node Peers | 437

Monitor Individual Control Node Routes | 439

Monitor > Infrastructure > Virtual Routers | 442

Monitor vRouters Summary | 442

Monitor Individual vRouters Tabs | 444

Monitor Individual vRouter Details Tab | 444

Monitor Individual vRouters Interfaces Tab | 446

Monitor Individual vRouters Networks Tab | 448

Monitor Individual vRouters ACL Tab | 449

Monitor Individual vRouters Flows Tab | 451

Monitor Individual vRouters Routes Tab | 452

Monitor Individual vRouter Console Tab | 453

Monitor > Infrastructure > Analytics Nodes | 456

Monitor Analytics Nodes | 456

Monitor Analytics Individual Node Details Tab | 458

Monitor Analytics Individual Node Generators Tab | 459

Monitor Analytics Individual Node QE Queries Tab | 460

Monitor Analytics Individual Node Console Tab | 461

Monitor > Infrastructure > Config Nodes | 464

Monitor Config Nodes | 464

Monitor Individual Config Node Details | 465

Monitor Individual Config Node Console | 466

Monitor > Networking | 468

Monitor > Networking Menu Options | 468

Monitor -> Networking -> Dashboard | 469

Monitor > Networking > Projects | 471

Monitor Projects Detail | 472

Monitor > Networking > Networks | 475

Query > Flows | 480

Query > Flows > Flow Series | 481

Example: Query Flow Series | 484

Query > Flow Records | 486

Query > Flows > Query Queue | 489

Query > Logs | 490

Query > Logs Menu Options | 491

Query > Logs > System Logs | 491

Sample Query for System Logs | 493

Query > Logs > Object Logs | 495

Example: Debugging Connectivity Using Monitoring for Troubleshooting | 497

Using Monitoring to Debug Connectivity | 497

About This Guide

Use this guide to install Contrail, perform initial configuration tasks, provision roles and storage, establish network connectivity, and monitor the network. This guide also provides information about the system requirements and configuration scenarios.

After completing the installation and basic configuration procedures covered in this guide, refer to the Contrail Feature Guide for information about further software configuration.

RELATED DOCUMENTATION

README Access to Contrail Registry 5.0
Contrail Release 5.0
Release Notes 5.0.1
Day One: Understanding OpenContrail Architecture
Juniper Contrail Configuration API Reference
Juniper Networks TechWiki: Contrail

1

PART

Overview

- [Understanding Contrail | 2](#)
-

Understanding Contrail

IN THIS CHAPTER

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

Contrail Overview

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

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

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

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

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

RELATED DOCUMENTATION

| [Contrail Description](#) | 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

2

PART

Installing and Upgrading Contrail

- [Server Requirements and Supported Platforms | 6](#)
 - [Installing Contrail and Provisioning Roles | 7](#)
 - [Installation and Configuration Scenarios | 51](#)
 - [Upgrading Contrail Software | 56](#)
 - [Backup and Restore Contrail Software | 78](#)
 - [Contrail Command | 86](#)
 - [Multicloud Contrail | 118](#)
 - [Using Contrail with Red Hat | 138](#)
 - [Using Contrail with Juju Charms | 208](#)
-

Server Requirements and Supported Platforms

IN THIS CHAPTER

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

Server Requirements and Supported Platforms

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

Each server must have a minimum of:

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

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

RELATED DOCUMENTATION

| [Downloading Installation Software | 13](#)

Installing Contrail and Provisioning Roles

IN THIS CHAPTER

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

Introduction to Containerized Contrail Modules

IN THIS SECTION

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

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

Why Use Containers?

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

These dependencies lead to complexities of deployment, including:

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

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

Overview of Contrail Containers

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

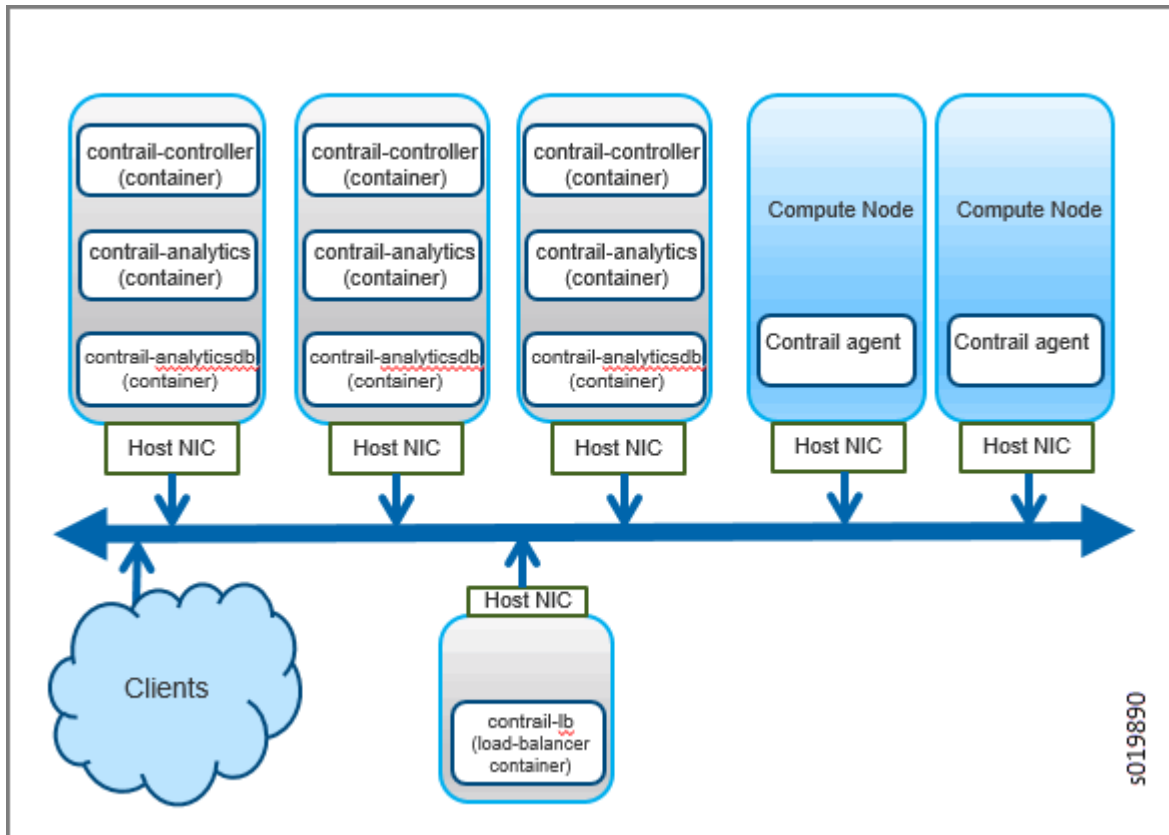
In Contrail 4.0, the containerized components include Contrail controller, analytics, and load-balancer applications. Contrail OpenStack components are not containerized at this time.

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

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

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

Figure 1: Sample Configuration Containerized Contrail



Contrail 4.0 Containers

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

contrail-controller

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

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

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

contrail-analytics

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

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

contrail-analyticsdb

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

contrail-lb

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

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

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

DPDK vRouter

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

Summary of Container Design, Configuration Management, and Orchestration

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

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

Introduction to Contrail Microservices Architecture

IN THIS SECTION

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

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

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

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

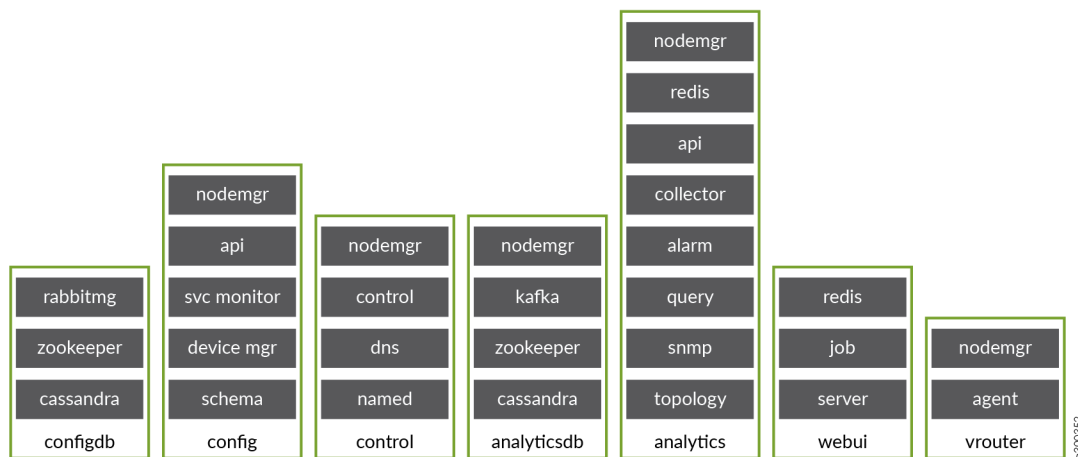
What is Contrail Microservices Architecture?

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

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

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

Figure 2: Contrail Containers, Pods, and Microservices



Installing Contrail with Microservices Architecture

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

- ["Understanding contrail-ansible-deployer used in Contrail Command" on page 13](#)
- [Installing and Managing Contrail Microservices Architecture Using Helm Charts](#)

Downloading Installation Software

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

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

The Contrail image includes the following software:

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

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

IN THIS SECTION

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

Contrail Release 5.0 introduces microservices architecture. This section provides an overview of using contrail-ansible-deployer used by The Contrail Command tool for installing Contrail Networking with a microservices architecture. For an introduction to Contrail microservices, refer to "[Introduction to Contrail Microservices Architecture](#)" on page 11. For step by step procedure on how to install contrail using contrail command deployer, refer to "[Installing Contrail Cluster using Contrail Command and instances.yml](#)" on page 106.

What is the contrail-ansible-deployer?

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

The contrail-ansible-deployer contains three plays:

playbooks/provision_instances.yml

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

- kvm
- gce
- aws

playbooks/configure_instances.yml

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

playbooks/install_contrail.yml

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

Preparing to Install with Contrail

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

Prerequisites

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

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

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

Supported Providers

The playbooks support installing Contrail on the following providers:

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

Configure a Yaml File for Your Environment

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

`config/instances.yaml`

The configuration has multiple main sections, including:

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

Provider Configuration

The section `provider_config` configures provider-specific settings.

KVM Provider Example

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

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

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

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

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

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

```

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

```

GCE Provider Example

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

```

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

```

Global Services Configuration

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

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

Contrail Services Configuration

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

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

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

Kolla Services Configuration

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

```
kolla_config:
```

Instances Configuration

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

GCE Default All-in-One Instance

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

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

AWS Default All-in-One Instance

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

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

KVM Contrail Plane Instance

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

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

More Examples

Refer to the following for more configuration examples for instances.

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

Installing a Contrail System

To perform a full installation of a Contrail system, refer to the installation instructions in: "[Installing Contrail Cluster using Contrail Command and instances.yml](#)" on page 106.

RELATED DOCUMENTATION

| [Deploying Contrail Cluster using Contrail-Command and instances.yml](#) | 106

Installing Contrail with OpenStack and Kolla Ansible

IN THIS SECTION

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

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

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

To deploy a Contrail Cluster using Contrail Command, see "[Installing Contrail Cluster using Contrail Command and instances.yml](#)" on page 106.

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

Set Up the Base Host

This procedure assumes you are installing with CentOS 7.5 kernel 3.10.0-862.11.6.el7.x86_64. The vRouter has a [dependency](#) with the host kernel. Install this kernel version on the target nodes before provisioning.

To set up the base host:

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

2. Install Ansible.

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

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

3. Untar the `tgz` file.

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

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

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

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

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

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

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

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

Example: instances.yaml

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

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

```

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

```

Example: instances.yaml

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

```

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

```

```

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

```

5. Run the following Commands:

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

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

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

Run OpenStack Commands

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

1. Manually install the python-openstackclient.

```

yum install -y gcc python-devel

pip install python-openstackclient

```

```
pip install python-ironicclient
```

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

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

Multiple Interface Configuration Sample for Multinode OpenStack HA and Contrail

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

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

Configuration Sample—Multiple Interface



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

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

```

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

```

```

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

```

Single Interface Configuration Sample for Multinode OpenStack HA and Contrail

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

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

Configuration Sample—Single Interface

```

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

```

```

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

```




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

Frequently Asked Questions

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

Using Host-Specific Parameters

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

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

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

Containers from Private Registry Not Accessible

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

Error: Failed to insert vrouter kernel module

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

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

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

```

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

```

2. In this release, the vrouter module requires the host kernel version to be 3.10.0-862.11.6.el7.x86_64. To get this kernel version, before running provision, install the kernel version on the target nodes.

```

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

yum update
reboot

```

Fatal Error When Vrouter Doesn't Specify OpenStack

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

```

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

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

```

```

config
  directories exist\n  ^ here\n"}

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

```

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

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

Need for HAProxy and Virtual IP on a Single OpenStack Cluster

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

Using the kolla_toolbox Container to Run OpenStack Commands

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

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

```

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

docker exec -it kolla_toolbox bash

```

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

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

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

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

RELATED DOCUMENTATION

[Deploying Contrail Cluster using Contrail-Command and instances.yml](#) | 106

Configuring the Control Node with BGP

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

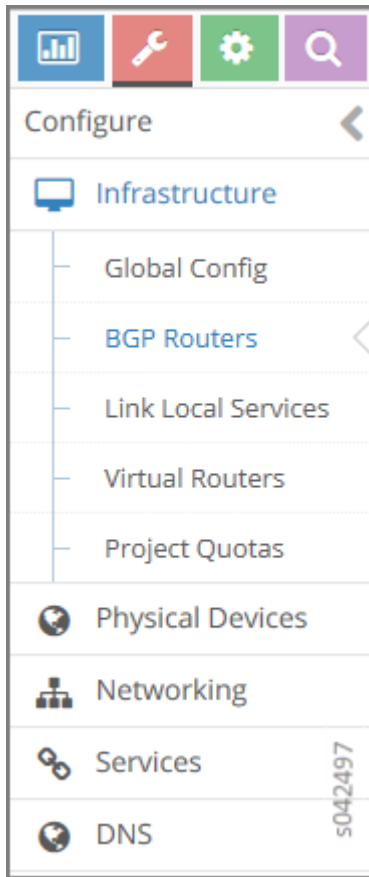
Before you begin, ensure that the following tasks are completed:

- The Contrail Controller base system image has been installed on all servers.
- The role-based services have been assigned and provisioned.
- IP connectivity has been verified between all nodes of the Contrail Controller.
- You can access the Contrail user interface at <http://nn.nn.nn.nn:8143>, where *nn.nn.nn.nn* is the IP address of the configuration node server that is running the **contrail-webui** service.

To configure BGP peering in the control node:

1. From the Contrail Controller module control node (<http://nn.nn.nn.nn:8143>), select **Configure > Infrastructure > BGP Routers**; see [Figure 3 on page 35](#).

Figure 3: Configure > Infrastructure > BGP Routers



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

Figure 4: BGP Routers Summary

Configure > Infrastructure > BGP Routers

BGP Routers					+	🗑️	📄	🔍	^
<input type="checkbox"/> IP Address	Type	Vendor	HostName						
▶ <input type="checkbox"/> 10.84.25.31	Control Node	contrail	b5s31						⚙️
▶ <input type="checkbox"/> 10.84.11.252	BGP Router	mx	a3-mx80-1						⚙️
▶ <input type="checkbox"/> 10.84.25.30	Control Node	contrail	b5s30						⚙️
▶ <input type="checkbox"/> 10.84.25.29	Control Node	contrail	b5s29						⚙️
▶ <input type="checkbox"/> 10.84.25.28	Control Node	contrail	b5s28						⚙️
▶ <input type="checkbox"/> 10.84.25.27	Control Node	contrail	b5s27						⚙️
▶ <input type="checkbox"/> 10.84.11.253	BGP Router	mx	mx1						⚙️

Total: 7 records | 50 Records ▼ | Page 1 ▼ of 1 | ⏪ ⏩

s042498


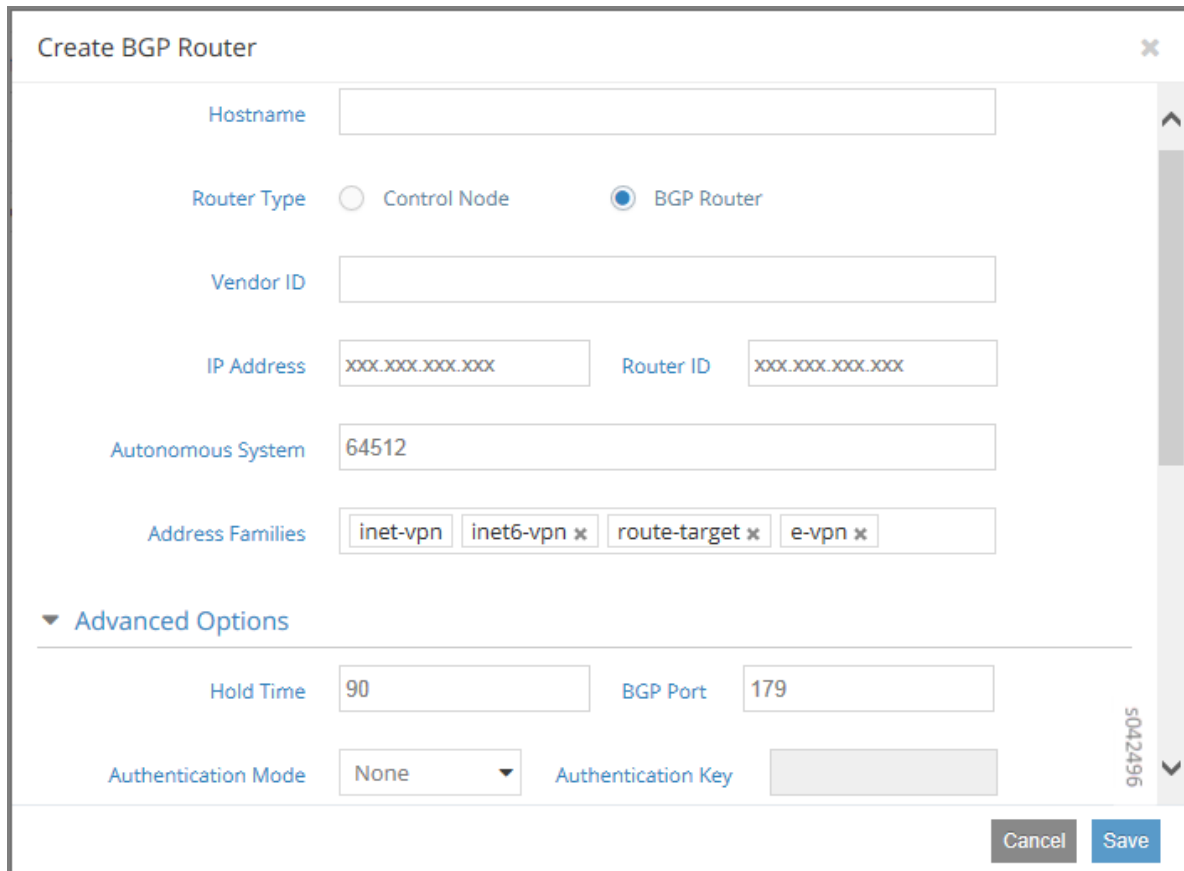
2. (Optional) The global AS number is 64512 by default. To change the AS number, on the **BGP Router** summary screen click the gear wheel and select **Edit**. In the Edit BGP Router window enter the new number.
3. To create control nodes and BGP routers, on the **BGP Routers** summary screen, click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

Figure 5: Create BGP Router



Create BGP Router

Hostname

Router Type ☐ Control Node ☒ BGP Router

Vendor ID

IP Address Router ID

Autonomous System

Address Families

▼ **Advanced Options**

Hold Time BGP Port

Authentication Mode Authentication Key

Cancel Save

4. In the **Create BGP Router** window, click **BGP Router** to add a new BGP router or click **Control Node** to add control nodes.
- For each node you want to add, populate the fields with values for your system. See [Table 1 on page 37](#).

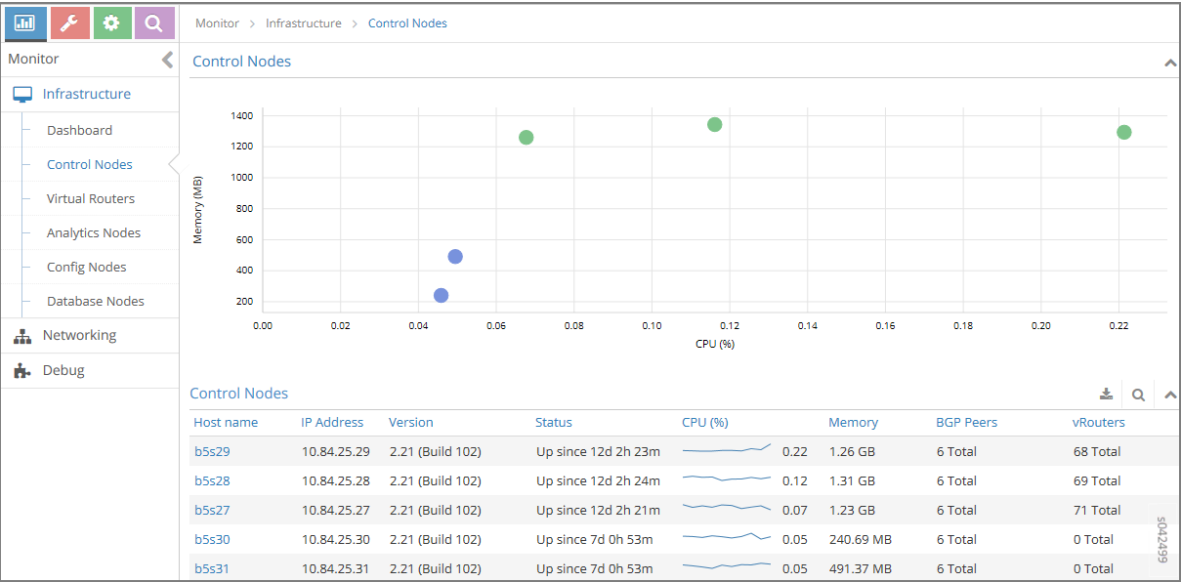
Table 1: Create BGP Router Fields

Field	Description
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.

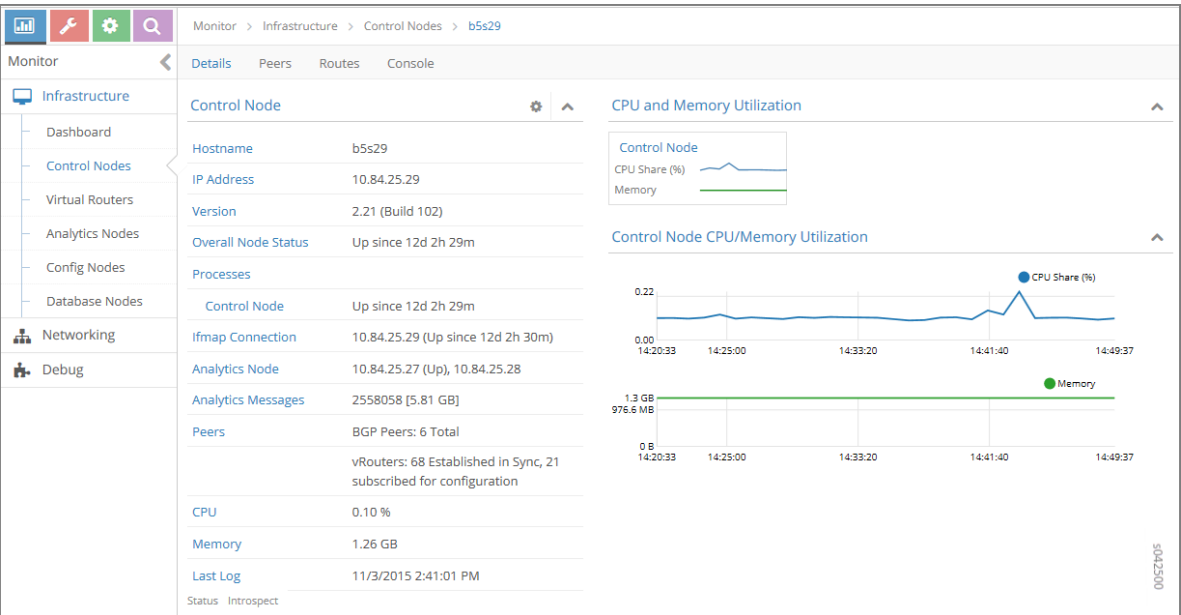
7. You can check for peers by selecting **Monitor > Infrastructure > Control Nodes**; see [Figure 6 on page 38](#).

Figure 6: Control Nodes



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

Figure 7: Control Node Details



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

Figure 8: Control Node Peers Tab

Monitor > Infrastructure > Control Nodes > b5s29

DetailsPeersRoutesConsole

Peers

DownloadSearchExpand

Peer	Peer Type	Peer ASN	Status	Last flap	Messages (Recv/Sent)
▶ 10.84.21.1	XMPP	-	Established, in sync	-	35497 / 138229
▶ 10.84.21.10	XMPP	-	Established, in sync	-	35511 / 137011
▶ 10.84.21.11	XMPP	-	Established, in sync	-	37045 / 141735
▶ 10.84.21.12	XMPP	-	Established, in sync	-	37493 / 140054
▶ 10.84.21.13	XMPP	-	Established, in sync	-	35540 / 137864
▶ 10.84.21.14	XMPP	-	Established, in sync	-	40098 / 112770
▼ 10.84.21.15	XMPP	-	Established, in sync	-	35450 / 137599

Details:

```
- {
  name: "b5s29:10.84.21.15",
  value: - {
    xmppPeerInfoData: - {
      state_info: - {
        last_state: "Active",
        state: "Established",

```

5042501

RELATED DOCUMENTATION

- Creating a Virtual Network with Juniper Networks Contrail | 234
- Creating a Virtual Network with OpenStack Contrail | 238

Contrail Global Controller

IN THIS SECTION

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

Starting with Release 3.1, Contrail provides support for a global controller. The global controller feature provides a seamless controller experience across multiple regions in a cloud environment by helping

manage multiple OpenStack installations, each having its own Keystone, Neutron, Nova and so on. High availability is provided by using separate failure domains by region.

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

Resource Identifier Management

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

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

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

Multiple Location Resource Provisioning

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

Requirements, Assumptions, and Constraints

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

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

Platform Support

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

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

Installation

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

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

Role and Resource-Based Access Control

IN THIS SECTION

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

Contrail Role and Resource-Based Access (RBAC) Overview

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

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

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

API-Level Access Control

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

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

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

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

Where:

object An API resource such as network or subnet.

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

role The Keystone role name.

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

Example: ACL RBAC Object

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

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

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

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

Rule Sets and ACL Objects

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

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

- User project
- User domain
- Default domain

It is possible for the project or domain access object to be empty.

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

Object Level Access Control

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

The `perms2` property has the following fields:

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

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

The `permission` field has the following options:

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

Access is allowed as follows:

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

Configuration

This section describes the parameters used in Contrail RBAC.

Parameter: `aaa-mode`

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

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

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



NOTE: The `multi_tenancy` parameter is deprecated, starting with Contrail 3.0. The parameter should be removed from the configuration. Instead, use the `aaa_mode` parameter for RBAC to take effect.

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

Parameter: `cloud_admin_role`

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

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

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

Configuration Files with Cloud Admin Credentials

The following configuration files contain `cloud_admin_role` credentials:

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

Changing Cloud Admin Configuration Files

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

1. Change the configuration files with the new information.

2. Restart the following:

- API server

```
service supervisor-config restart
```

- Neutron server

```
service neutron-server restart
```

- WebUI

```
service supervisor-webui restart
```

Global Read-Only Role

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

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

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

Setting the Global Read-Only Role

To set the global read-only role:

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

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

```
global_read_only_role = <new-admin-read-role>
```

2. Restart the `contrail-api` service:

```
service contrail-api restart
```


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

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

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

Upgrading from Previous Releases

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

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

Configuring RBAC Using the Contrail User Interface

To use the Contrail UI with RBAC:

1. Set the `aaa_mode` to `no_auth`.

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

aaa_mode = no-auth
```

2. Restart the `analytics-api` service.

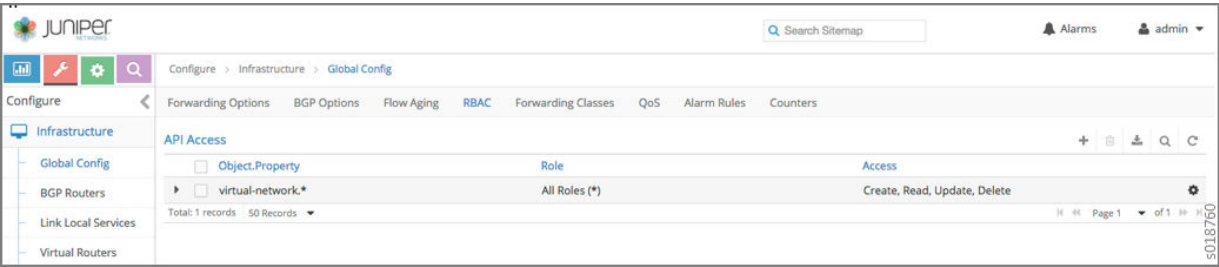
```
service contrail-analytics-api restart
```

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

Configuring RBAC at the Global Level

To configure RBAC at the global level, navigate to **Configure > Infrastructure > Global Config > RBAC**, see [Figure 9 on page 47](#).

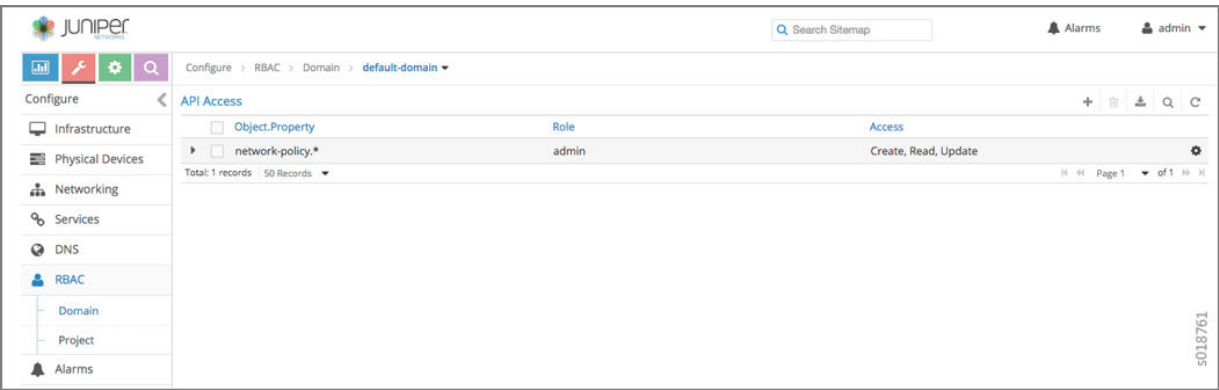
Figure 9: RBAC Global Level



Configuring RBAC at the Domain Level

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

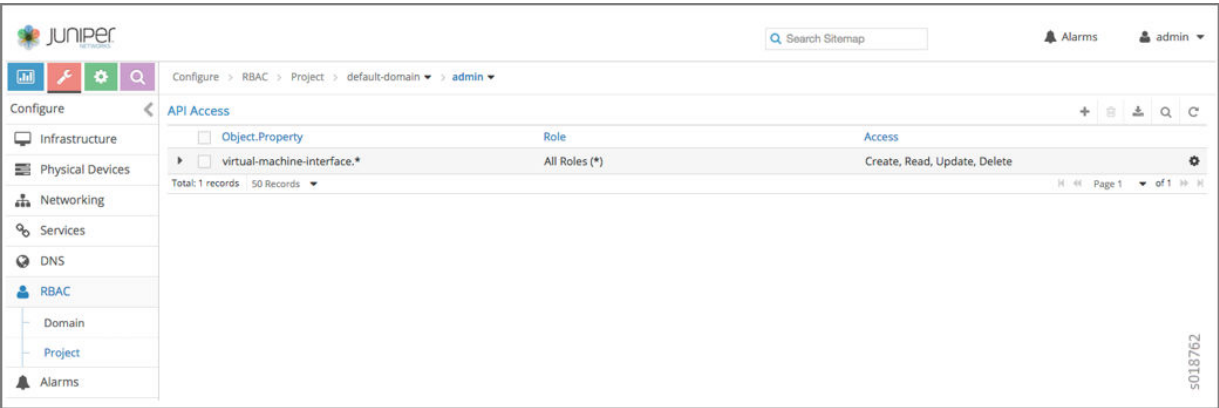
Figure 10: RBAC Domain Level



Configuring RBAC at the Project Level

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

Figure 11: RBAC Project Level



Configuring RBAC Details

Configuring RBAC is similar at all of the levels. To add or edit an API access list, navigate to the global, domain, or project page, then click the plus (+) icon to add a list, or click the gear icon to select from Edit, Insert After, or Delete, see [Figure 12 on page 48](#).

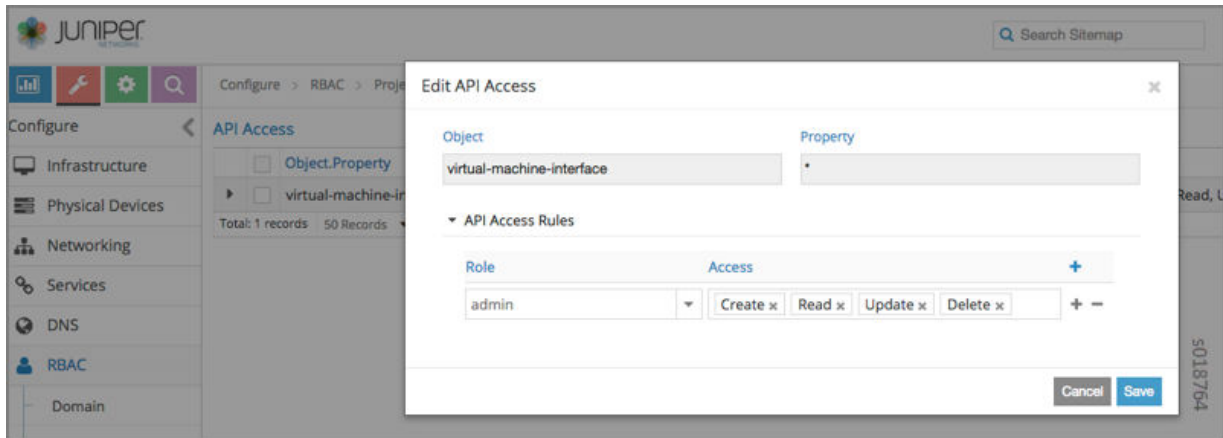
Figure 12: RBAC Details API Access



Creating or Editing API Level Access

Clicking create, edit, or insert after activates the Edit API Access popup window, where you enter the details for the API Access Rules. Enter the user type in the Role field, and use the + icon in the Access field to enter the types of access allowed for the role, including, Create, Read, Update, Delete, and so on, see [Figure 13 on page 49](#).

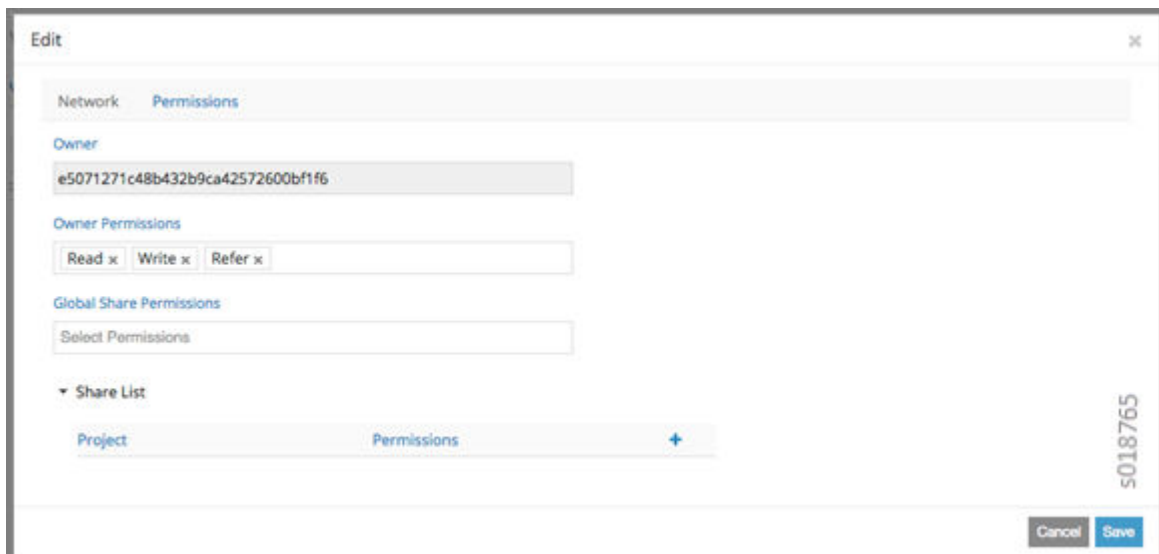
Figure 13: Edit API Access



Creating or Editing Object Level Access

You can configure fine-grained access control by resource. A **Permissions** tab is available on all create or edit popups for resources. Use the **Permissions** popup to configure owner permissions and global share permissions. You can also share the resource to other tenants by configuring it in the **Share List**, see [Figure 14 on page 49](#).

Figure 14: Edit Object Level Access



RBAC Resources

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

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

Installation and Configuration Scenarios

IN THIS CHAPTER

- [Simple Underlay Connectivity without Gateway | 51](#)
- [Dynamic Kernel Module Support \(DKMS\) for vRouter | 55](#)

Simple Underlay Connectivity without Gateway

IN THIS SECTION

- [Simple Routing of Packets Without a Gateway | 51](#)
- [Supported Use Cases | 52](#)
- [Implementation: Routing Instances | 52](#)
- [Implementation | 54](#)

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.

Dynamic Kernel Module Support (DKMS) for vRouter

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

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

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

For more information about DKMS, refer to:

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

Upgrading Contrail Software

IN THIS CHAPTER

- [Contrail In-Service Software Upgrade from Releases 3.2 and 4.1 to 5.0.x using Ansible Deployer | 56](#)
- [Contrail In-Service Software Upgrade from Releases 3.2 and 4.1 to 5.0.x using Helm Deployer | 66](#)

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

IN THIS SECTION

- [Contrail In-Service Software Upgrade \(ISSU\) Overview | 56](#)
- [Prerequisites | 57](#)
- [Preparing the Contrail System for the Ansible Deployer ISSU Procedure | 57](#)
- [Provisioning Control Nodes and Performing Synchronization Steps | 59](#)
- [Transferring the Compute Nodes into the New Cluster | 62](#)
- [Finalizing the Contrail Ansible Deployer ISSU Process | 65](#)

Contrail In-Service Software Upgrade (ISSU) Overview

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



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

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



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

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

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

Prerequisites

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

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



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

Preparing the Contrail System for the Ansible Deployer ISSU Procedure

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

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

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



NOTE: You must provide keystone authorization information for setup.

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

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

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

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

```

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

```

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

```

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

```

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

Provisioning Control Nodes and Performing Synchronization Steps

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

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

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

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

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

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

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

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

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

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

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

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

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



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

The configuration appears as follows and can be stored locally:

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

1. Detect the config-api image ID.

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

2. Run the pre-synchronization.

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


3. Run the run-synchronization.

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

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

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

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

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

Transferring the Compute Nodes into the New Cluster

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

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

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

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

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

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



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

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

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

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

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

2. Run the ansible playbooks.

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

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

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

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

```
docker restart control_control_1
```

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

Finalizing the Contrail Ansible Deployer ISSU Process

Finalize the Contrail ansible deployer ISSU as follows:

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

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

2. Run the post synchronization commands.

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

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

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

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

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

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

5. Run the following commands from any controller node.



NOTE: All **host_info* parameters should contain the list of new hosts.

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

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

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

IN THIS SECTION

- [Contrail In-Service Software Upgrade \(ISSU\) Overview | 67](#)
- [Prerequisites | 67](#)
- [Preparing the Contrail System for the Helm Deployer ISSU Procedure | 68](#)
- [Provisioning Control Nodes and Performing Synchronization Steps | 68](#)
- [Transferring the Compute Nodes into the New Cluster | 71](#)
- [Finalizing the Contrail Helm Deployer ISSU Process | 75](#)

Contrail In-Service Software Upgrade (ISSU) Overview

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



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

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



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

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

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

Prerequisites

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

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



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

Preparing the Contrail System for the Helm Deployer ISSU Procedure

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

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



NOTE: You must provide keystone authorization information for setup.

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

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

Provisioning Control Nodes and Performing Synchronization Steps

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

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

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

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

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

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

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

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

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

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

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

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

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



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

The configuration appears as follows and can be stored locally:

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

1. Detect the config-api image ID:

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

2. Run the pre-synchronization.

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

3. Run the run-synchronization.

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

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

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

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

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

Transferring the Compute Nodes into the New Cluster

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

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

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

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

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

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



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

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

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

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

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

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



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

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

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

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

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

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

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

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



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

7. Add vrouter as follows:

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

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

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



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

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

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

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

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

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

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

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

Finalizing the Contrail Helm Deployer ISSU Process

Finalize the Contrail Helm deployer ISSU as follows:

1. Stop the issu-run-sync container.

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

2. Run the post synchronization commands.

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

3. Start all previously stopped containers.

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

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

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

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

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

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

6. Run the following commands from any controller node:



NOTE: All **host_info* parameters should contain the list of new hosts.

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

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

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

Backup and Restore Contrail Software

IN THIS CHAPTER

- [Backing up Contrail Databases in JSON Format | 78](#)

Backing up Contrail Databases in JSON Format

IN THIS SECTION

- [Preliminary Caution | 78](#)
- [Simple Database Backup in JSON Format | 79](#)
- [Restore Database from the Backup | 79](#)
- [Example Backup and Restore in JSON | 81](#)

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

Preliminary Caution



CAUTION: Database backups must be consistent across all systems because the state of the Contrail database is associated with other system databases, such as OpenStack databases. Database changes associated with northbound APIs must be stopped on all systems before performing any backup operation. For example, you might block the external VIP for northbound APIs at the load balancer level, such as HAproxy.

Simple Database Backup in JSON Format

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



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

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

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

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

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

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

3. See a cleaner version of the dump.

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

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

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

Restore Database from the Backup

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

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

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

2. Stop config services on all the controllers.

```
docker stop config_svcmonitor_1
```

```
docker stop config_devicemgr_1
```

```
docker stop config_schema_1
```

```
docker stop config_api_1
```

```
docker stop config_nodemgr_1
```

```
docker stop config_database_nodemgr_1
```

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

```
docker stop config_database_cassandra_1
```

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

```
docker stop config_database_zookeeper_1
```

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

```
docker stop analytics_database_kafka_1
```

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

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

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

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

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

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

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

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

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

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

10. Start Zookeeper on all the controllers.

```
docker start config_database_zookeeper_1
```

11. Start Cassandra on all the controllers.

```
docker start config_database_cassandra_1
```

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

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

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

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

14. Restore the data in new running docker.

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

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

```
docker start analytics_database_kafka_1
```

16. Start config services on all the controllers.

```
docker start config_svcmonitor_1
```

```
docker start config_devicemgr_1
```

```
docker start config_schema_1
```

```
docker start config_api_1
```

```
docker start config_nodemgr
```

```
docker start config_database_nodemgr
```

Example Backup and Restore in JSON

This section provides an example of a simple database backup and restore of a system that has three controllers with config-db and separate IPs with the following host IDs:

- nodec53
- nodec54
- nodec55

Example: Perform Simple Database Backup in JSON Format

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

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

Example: Restore Database from the Backup

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

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

2. Stop config services on all the controllers.

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

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

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

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

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

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

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

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

```
[root@nodec53 ~]# docker stop config_database_zookeeper_1
[root@nodec54 ~]# docker stop config_database_zookeeper_1
[root@nodec55 ~]# docker stop config_database_zookeeper_1
```

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

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

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

```
[root@nodec53 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra
[root@nodec53 config_database_config_cassandra]# rm -rf _data/*

[root@nodec54 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra
[root@nodec54 config_database_config_cassandra]# rm -rf _data/*

[root@nodec55 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra
[root@nodec55 config_database_config_cassandra]# rm -rf _data/*
```

7. Delete config Cassandra.

```
[root@nodec53 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra
[root@nodec53 config_database_config_cassandra]# rm -rf _data/*

[root@nodec54 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra
[root@nodec54 config_database_config_cassandra]# rm -rf _data/*

[root@nodec55 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra
[root@nodec55 config_database_config_cassandra]# rm -rf _data/*
```

8. Delete config Zookeeper.

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

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

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

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

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

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

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

10. Run db_json_exim.py to restore the data from json dump.

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

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

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

12. Start config services on all the controllers.

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

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

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


Contrail Command

IN THIS CHAPTER

- [Configuring Contrail Command | 86](#)
- [Deploying Contrail Cluster using the Contrail Command UI | 93](#)
- [Deploying Contrail Cluster using Contrail-Command and instances.yml | 106](#)
- [Importing Contrail Cluster Data using Contrail Command | 113](#)

Configuring Contrail Command

IN THIS SECTION

- [Requirements | 86](#)
- [Overview | 87](#)
- [Configuration | 87](#)
- [Sample command_servers.yml File | 89](#)

The Contrail Command user interface (UI) is supported starting with Contrail Release 5.0.1. Contrail Command is an intuitive, wizard-based UI which provides automated work flows such as the following:

- Contrail cluster deployment (Kolla-based OpenStack cluster)
- Automating the data center IP fabric
- Orchestrating virtual machines and bare metal servers

Requirements

The system requirements to install the Contrail Command server are:

- A VM or physical server with:
 - 8 vCPUs
 - 64 GB RAM
 - 300 GB disk out of which 256 GB is allocated to **/root** directory.
- Internet access to and from the physical server, hereafter referred to as the Contrail Command server
- (Recommended) x86 server with CentOS 7.5 as the base OS to install Contrail Command

Overview

Contrail Command is an intuitive, wizard-based user interface (UI) to manage private and public clouds, physical and virtual workloads and devices.

Configuration

IN THIS SECTION

- [Procedure | 87](#)

Prerequisite

`docker-py` is obsolete in Contrail Release 5.0.2. You must remove `docker-py` and `docker` Python packages from all the nodes where you want to install the Contrail Command UI.

```
pip uninstall docker-py docker
```

Procedure

Step-by-Step Procedure

Perform the following steps to configure and install Contrail Command.

1. Install Docker on the Contrail Command server. These packages are necessary to automate the deployment of Contrail Command software.

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

```
yum install -y docker-ce
```

```
systemctl start docker
```

2. Download the contrail-command-deployer Docker container image to deploy contrail-command (contrail_command, contrail_mysql containers) from hub.juniper.net. Allow Docker to connect to the private secure registry.

```
docker login hub.juniper.net --username <container_registry_username> --password <container_registry_password>
```

Pull Contrail-Command-Deployer Container from the private secure registry.

```
docker pull hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
```

Example, for container_tag: 5.0.1-0.214, use the following command:

```
docker pull hub.juniper.net/contrail/contrail-command-deployer:5.0.1-0.214
```

3. Create the input configuration **command_servers.yml** file.

Use the ["Sample command_servers.yml File" on page 89](#) to create the **command_servers.yml** file.

4. Start the Contrail_Command_Deployer container to deploy the Contrail-Command server.

```
docker run -t --net host -v ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE:/command_servers.yml -d --privileged --name  
contrail_command_deployer hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
```

ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE—path to the **command_servers.yml** file that you created in step ["3" on page 88](#).

Example, for container_tag: 5.0.1-0.214, use the following command:

```
docker run -t --net host -v /root/command_servers.yml:/command_servers.yml -d --privileged --name  
contrail_command_deployer hub.juniper.net/contrail/contrail-command-deployer:5.0.1-0.214
```

The contrail_command and contrail_mysql Contrail Command containers are deployed.

5. (Optional) You can also upgrade Contrail-Command UI without deleting existing database information. To update contrail_command container and not make changes to the database container, use the following command.

```
docker run -t --net host -e delete_db=no -v <ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE>:/command_servers.yml -d --  
privileged --name contrail_command_deployer hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
```



NOTE: Code changes that involve schema modifications require updating the database container as well. Step ["5" on page 88](#) is recommended only if the UI application requires an update.

6. (Optional) Track the progress of Step "4" on page 88.

```
docker logs -f contrail_command_deployer
```

7. Once the playbook execution completes, log in to Contrail Command using [https:// Contrail-Command-Server-IP-Address:9091](https://Contrail-Command-Server-IP-Address:9091). Use the same user name and password that was entered in "3" on page 88. Default username is admin and password is contrail123.

Sample command_servers.yml File

```
---
command_servers:
  server1:
    ip: <IP Address>
    connection: ssh
    ssh_user: root
    ssh_pass: <contrail command server password>
    sudo_pass: <contrail command server root password>
    ntpserver: <NTP Server address>

    # Specify either container_path
    #container_path: /root/contrail-command-051618.tar
    # or registry details and container_name
    # registry_insecure: true
    # container_registry: ci-repo.englab.juniper.net:5010
    registry_insecure: false
    container_registry: hub.juniper.net/contrail
    container_name: contrail-command
    container_tag: 5.0.2-0.349
    container_registry_username: <registry username>
    container_registry_password: <registry password>
    config_dir: /etc/contrail

    # contrail command container configurations given here go to /etc/contrail/contrail.yml
    contrail_config:
      # Database configuration. MySQL/PostgreSQL supported
      database:
        # MySQL example
        type: mysql
        dialect: mysql
        host: localhost
        user: root
        password: contrail123
```

```

    name: contrail_test
    # Postgres example
    #connection: "user=root dbname=contrail_test sslmode=disable"
    #type: postgres
    #dialect: postgres

    # Max Open Connections for DB Server
    max_open_conn: 100
    connection_retries: 10
    retry_period: 3s

# Log Level
log_level: debug

# Server configuration
server:
    enabled: true
    read_timeout: 10
    write_timeout: 5
    log_api: true
    address: ":9091"

# TLS Configuration
tls:
    enabled: true
    key_file: /usr/share/contrail/ssl/cs-key.pem
    cert_file: /usr/share/contrail/ssl/cs-cert.pem

# Enable GRPC or not
enable_grpc: false

# Static file config
# key: URL path
# value: file path. (absolute path recommended in production)
static_files:
    /: /usr/share/contrail/public

# API Proxy configuration
# key: URL path
# value: String list of backend host
#proxy:
#    /contrail:
#        - http://localhost:8082

```

```

notify_etcd: false

# Keystone configuration
keystone:
  local: true
  assignment:
    type: static
  data:
    domains:
      default: &default
      id: default
      name: default
    projects:
      admin: &admin
      id: admin
      name: admin
      domain: *default
      demo: &demo
      id: demo
      name: demo
      domain: *default
    users:
      admin:
        id: admin
        name: Admin
        domain: *default
        password: contrail123
        email: admin@juniper.nets
        roles:
          - id: admin
            name: Admin
            project: *admin
      bob:
        id: bob
        name: Bob
        domain: *default
        password: bob_password
        email: bob@juniper.net
        roles:
          - id: Member
            name: Member
            project: *demo

```

```

    store:
        type: memory
        expire: 36000
        insecure: true
        authurl: https://localhost:9091/keystone/v3

# disable authentication with no_auth true and comment out keystone configuraion.
#no_auth: true
insecure: true

etcd:
    endpoints:
        - localhost:2379
    username: ""
    password: ""
    path: contrail

watcher:
    enabled: false
    storage: json

client:
    id: admin
    password: contrail123
    project_name: admin
    domain_id: default
    schema_root: /
    endpoint: https://localhost:9091

compilation:
    enabled: false
    # Global configuration
    plugin_directory: 'etc/plugins/'
    number_of_workers: 4
    max_job_queue_len: 5
    msg_queue_lock_time: 30
    msg_index_string: 'MsgIndex'
    read_lock_string: "MsgReadLock"
    master_election: true

# Plugin configuration
plugin:
    handlers:

```

```

        create_handler: 'HandleCreate'
        update_handler: 'HandleUpdate'
        delete_handler: 'HandleDelete'

    agent:
        enabled: true
        backend: file
        watcher: polling
        log_level: debug

    # The following are optional parameters used to patch/cherrypick
    # revisions into the contrail-ansible-deployer sandbox. These configs
    # go into the /etc/contrail/contrail-cluster.tpl file
    #   cluster_config:
    #       ansible_fetch_url: "https://review.opencontrail.org/Juniper/contrail-ansible-
    #       deployer refs/changes/80/40780/20"
    #       ansible_cherry_pick_revision: FETCH_HEAD
    #       ansible_revision: GIT_COMMIT_HASH

```

RELATED DOCUMENTATION

[Deploying Contrail Cluster using the Contrail Command UI | 93](#)

[Deploying Contrail Cluster using Contrail-Command and instances.yml | 106](#)

[Importing Contrail Cluster Data using Contrail Command | 113](#)

Deploying Contrail Cluster using the Contrail Command UI

IN THIS SECTION

- [Requirements | 94](#)
- [Overview | 94](#)
- [Configuration | 96](#)

This example topic describes how to use the Contrail Command User interface (UI) to deploy a Contrail Cluster starting with Contrail Release 5.0.1.

Requirements

- Contrail Controller – 8 vCPU, 64G memory, 300G storage
- OpenStack Controller – 4 vCPU , 32G memory, 100G storage
- Contrail Server Node (CSN) – 4 vCPU, 16G memory, 100G storage
- Compute nodes– Dependent on the workloads

Overview

IN THIS SECTION

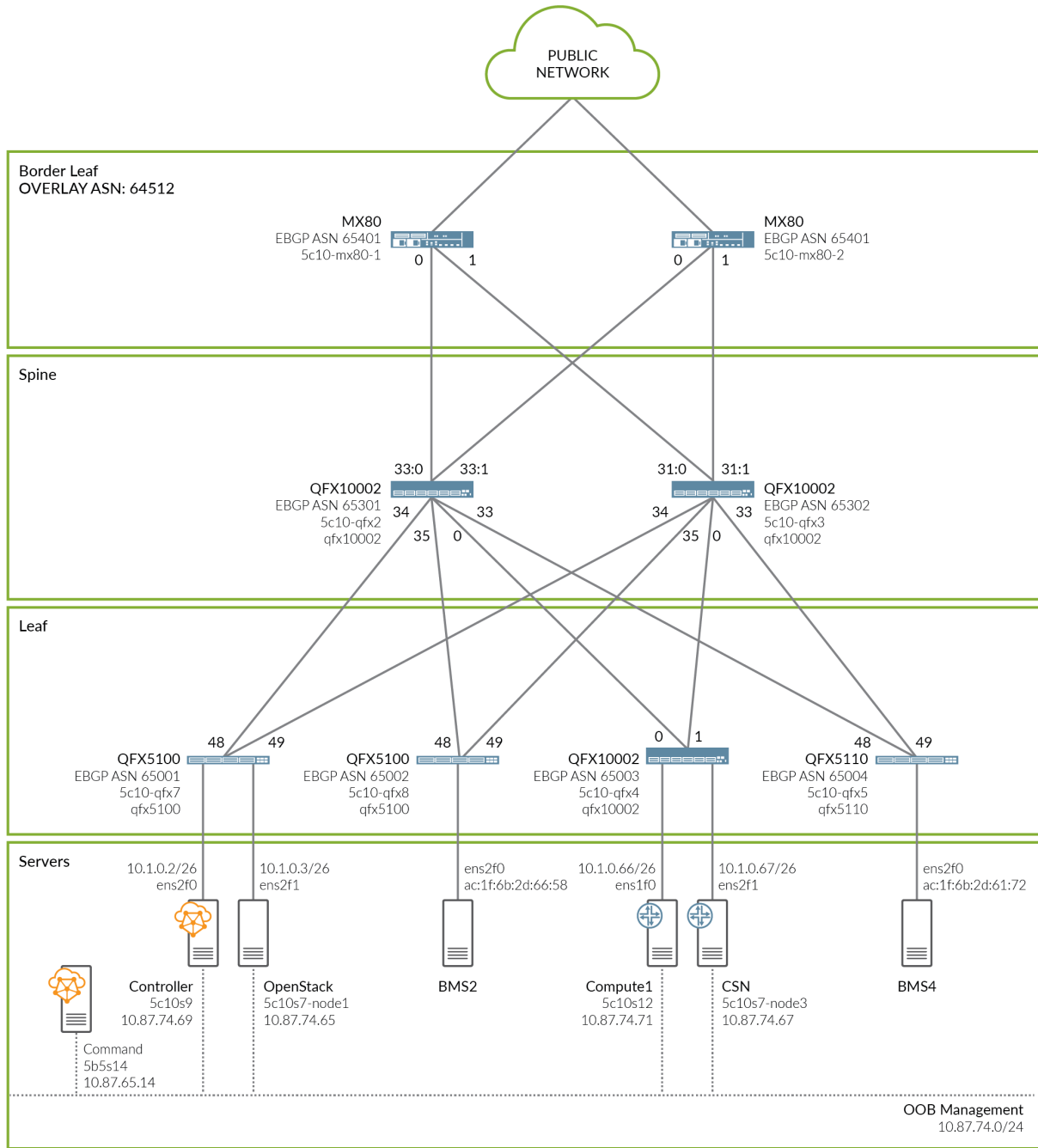
- [Topology | 94](#)

Contrail Cluster is an OpenStack orchestration coupled with the Contrail Networking plugin.

Topology

Consider a sample cluster topology, with a non-HA environment of one Contrail Controller and one OpenStack Controller, one compute node and one CSN, as displayed in [Figure 15 on page 95](#).

Figure 15: Sample Contrail Cluster Topology



g200484

Configuration

IN THIS SECTION

- [Deploying a Contrail Cluster | 96](#)

Deploying a Contrail Cluster

Step-by-Step Procedure

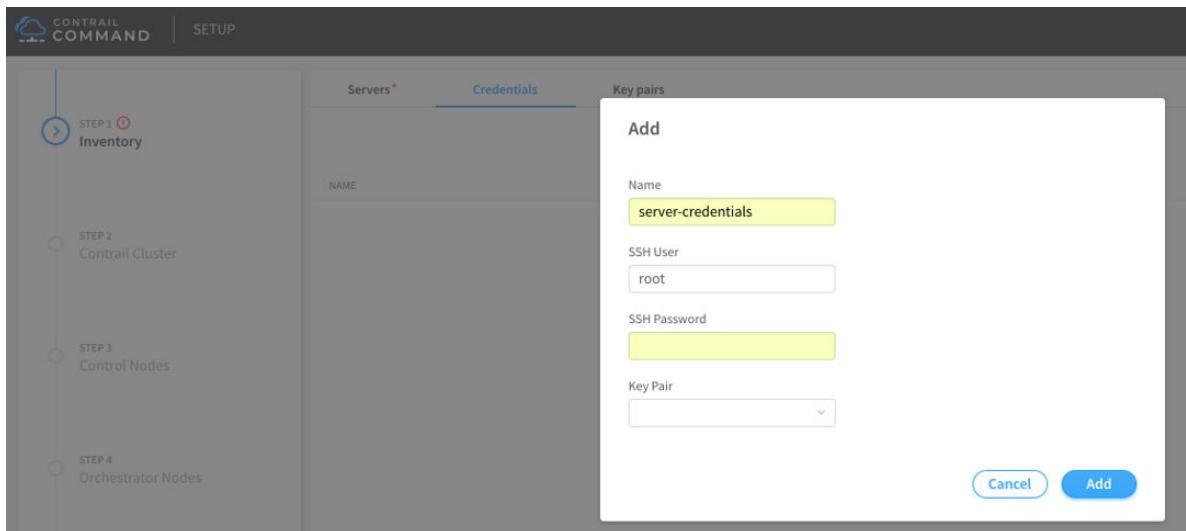
To deploy a Contrail Cluster using Contrail Command, perform the following steps.

1. Add physical servers. You can add a server in the following two ways:

- One by one
- CSV file bulk import



NOTE: Create server login credentials before adding the servers.



- **One by one:** You can add each server one by one. To add servers one by one, the following mandatory parameters should be entered.

Hostname, Management IP address and the Management Interface.

- **CSV file bulk import:** You can upload a CSV file which contains the details of each server. This CSV file must conform to the Contrail Command format. You can download the CSV template from Contrail Command and reuse the template as per your requirements. To download a sample CSV file, navigate to **Infrastructure > Servers > Add Servers** in the Contrail Command UI.

Use the Bulk Import option for large deployments. Bulk Import option requires a CSV file input. For large deployments, use the **Bulk Import (csv)** option. Click **Bulk Import (csv)**, to download the template. A sample CSV file is shown here:

```
Workload Type,HostName,Management IP,Disk Partition,Network Interface,MAC address,IPMI
Driver,IPMI Address,IPMI UserName,IPMI Password,Memory mb,CPU's,CPU Arch,Local
gb,Capabilities,Number of Network Interfaces,Interface Name,Interface MAC
Address,Interface IP,Enable PXE,Interface Name,Interface MAC Address,Interface IP,Enable
PXE

physical,5c10s9,10.87.74.69,,enp4s0f0,,,,,,,,,2,enp4s0f0,,10.87.74.69,,ens2f0,,10.1.0.2,

physical,5c10s7-node1,10.87.74.65,,eno1,,,,,,,,,2,eno1,,10.87.74.65,,ens2f1,,10.1.0.3,

physical,5c10s7-node3,10.87.74.67,,eno1,,,,,,,,,2,eno1,,10.87.74.67,,ens2f1,,10.1.0.67,

physical,5c10s12,10.87.74.71,,eno1,,,,,,,,,2,eno1,,10.87.74.71,,ens1f0,,10.1.0.66,
```



NOTE: The demo topology above has only one compute node. If you are deploying additional compute nodes, you must include them in the CSV file.

Figure 16: Add Server

The screenshot shows the 'Contrail Command' setup interface. On the left, a vertical sidebar lists steps: STEP 1 Inventory, STEP 2 Contrail Cluster, STEP 3 Control Nodes, STEP 4 Orchestrator Nodes, STEP 5 (optional) Compute Nodes, STEP 6 (optional) Contrail Service Nodes, and STEP 7 Summary. The main area is titled 'SETUP' and has tabs for 'Servers', 'Credentials', and 'Key pairs'. The 'Servers' tab is active, showing a table with one server entry: '5c10s9'. A 'Create Server' dialog box is open, allowing configuration for this server. The dialog includes fields for 'Choose Mode' (with 'Detailed' selected), 'Select workload type' (with 'Physical/Virtual Node' selected), 'Hostname' (5c10s9), 'Management IP' (10.87.74.65), 'Management Interface' (eno1), 'Credentials' (server-credentials), 'MAC Address' (a placeholder 'Enter valid MAC'), 'Disk Partition(s)' (vda, vdb), and a 'Network Interfaces' section with one entry: 'ens2f1' with IP '10.1.0.3'. At the bottom right of the dialog are 'Cancel' and 'Create' buttons.

2. Create a cluster.

If **Container registry** = `hub.juniper.net/contrail`. This registry is secure. Unselect the **Insecure** box. Also, **Contrail version** = `contrail_container_tag` for your release of Contrail as listed in [README Access for Contrail](#).

Default vRouter Gateway = Default gateway for the compute nodes. If any one of the compute nodes has a different default gateway than the one provided here, enter that gateway in "5" on page 102 and "6" on page 103 for service nodes.

Set the order of **Encapsulation Priority** for the EVPN supported methods - MPLS over UDP, MPLS over GRE And VxLAN.

VXLAN, MPLSoUDP, MPLSoGRE

Select **Show Advanced Options**. Add the following configuration parameters:

- **CONTROLLER_NODES** = List of comma separated mgmt interface IP addresses of the Contrail controller

- CONTROL_NODES = List of comma separated data interface IP addresses of the Contrail controller
- TSN_NODES = List of comma separated data interface IP addresses of the Contrail service nodes
- CONTRAIL_CONTAINER_TAG = *contrail_container_tag-ocata* or *container_tag-queens*

Figure 17: Create Cluster

Contrail Command SETUP

STEP 1 Inventory

STEP 2 **Contrail Cluster**

STEP 3 Control Nodes

STEP 4 Orchestrator Nodes

STEP 5 (optional) Compute Nodes

STEP 6 (optional) Contrail Service Nodes

STEP 7 Summary

STEP 8 Provisioning

Cluster Name*
Demo-cluster

Container Registry*
hub.juniper.net/contrail

Container Registry Username*
username

Container Registry Password*
password

Contrail Version*
5.0.1-0.214

Provisioner Type
Ansible

Domain Suffix
local

NTP Server
10.84.5.100

Default Vrouter Gateway
10.1.0.254

Encapsulation Priority
VXLAN,MPLS,UDP,MPLS...

☐ Enable ZTP

☒ Show Advanced Options

Contrail Configuration

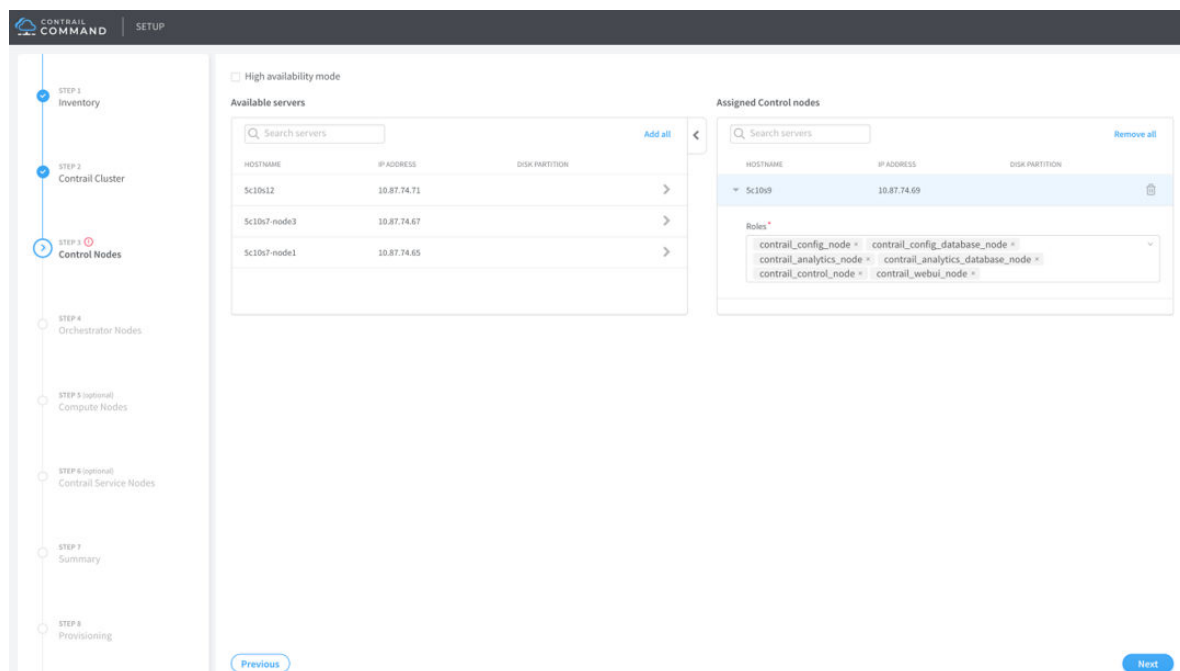
Key	Value
CONTROLLER_NODES	10.87.74.69
CONTROL_NODES	10.1.0.2
TSN_NODES	10.1.0.67
CONTRAIL_CONTAINER_TAG	5.0.1-0.214-ocata

+ Add

Previous Next

3. Select the contrail-control node.

Figure 18: Select Control Nodes



4. Select the orchestration node.

Select **Show Advanced** option to customize your deployment and then select **Orchestration Nodes**.

In order to run Openstack services on the control data network, set the following parameters:

- **Control & Data Network Virtual IP address:** It is an internal VIP. e.g. - 10.87.74.100
- **Management Network Virtual IP address:** It is an external VIP. e.g. - 10.1.0.100
- **keepalived_virtual_router_id:** It is to be added as a key value pair. It can be set to any value between 0-255

Add the following under custom configuration for VM based setup:

```
nova.conf: |
    [libvirt]
    virt_type=qemu
    cpu_mode=none
```



NOTE: Minimum 8 indent spaces are required for lines following the nova.conf.

In Contrail Command, key-value pairs handle parameters that be enabled or disabled in Kolla Global.

Set the following in Kolla Globals:

```
enable_ironic = yes
enable_swift = yes
upgrade_kernel = yes
```

If Ironic is not needed and if you are not going to use Life Cycle Management in Contrail Command then you need not deploy Ironic.

Swift can be enabled for Image management uses case, this parameter is disabled by default.

Compute node kernel version = kernel-3.10.0-862.3.2.el7.x86_64; Else kernel upgrade is required.

To configure kolla password add keystone_admin_password <password> key-value pair in the kolla passwords section. This password will be used for logging onto the contrail command UI after the provisioning completes.

Figure 19: Select Orchestrator Nodes

The screenshot displays the 'SETUP' page for Contrail Command, specifically the 'STEP 4: Orchestrator Nodes' configuration screen. On the left, a vertical sidebar lists steps from 'STEP 1: Inventory' to 'STEP 8: Provisioning', with 'STEP 4' currently selected. The main area contains several configuration sections: 'Orchestrator type' is a dropdown menu set to 'Openstack' with a 'Show Advanced' checkbox; 'Container Registry' is a text field with 'default'; 'Openstack Release' is a text field with 'ocata'; there are two text fields for 'Control & Data Network Virtual IP address' and 'Management Network Virtual IP address', both with 'Enter valid IPv4' placeholder text; a 'Customize configuration' section with a text area; a 'Kolla Globals' section with two rows of 'Key' and 'Value' pairs, where 'enable_ironic' and 'enable_swift' are both set to 'yes'; and a 'Kolla Passwords' section with a 'Key' and 'Value' pair where 'keystone_admin_password' is set to 'Juniper123'. At the bottom, there are 'Previous' and 'Next' buttons, along with links for 'Available services' and 'Available Openstack nodes'.

Select the Openstack (orchestration) node.

The screenshot shows the CONTRAIL COMMAND SETUP interface. On the left, a vertical sidebar lists the steps: STEP 1: Inventory, STEP 2: Contrail Cluster, STEP 3: Control Nodes, STEP 4: Orchestrator Nodes (highlighted with a blue circle), STEP 5 (optional): Compute Nodes, STEP 6 (optional): Contrail Service Nodes, STEP 7: Summary, and STEP 8: Provisioning.

The main content area is divided into several sections:

- Kolla Globals:** A table with two rows:

Key	Value
enable_ironic	yes
enable_swift	yes

 Below the table is a "+ Add" button.
- Kolla Passwords:** A table with one row:

Key	Value
keystone_admin_password	Juniper123

 Below the table is a "+ Add" button.
- Available servers:** A table with three columns: HOSTNAME, IP ADDRESS, and DISK PARTITION. It lists three servers:

HOSTNAME	IP ADDRESS	DISK PARTITION
Sc10n12	10.87.74.71	>
Sc10n7-node3	10.87.74.67	>
Sc10n8	10.87.74.69	>

 Below the table is a "Previous" button.
- Assigned Openstack nodes:** A table with three columns: HOSTNAME, IP ADDRESS, and DISK PARTITION. It lists one server:

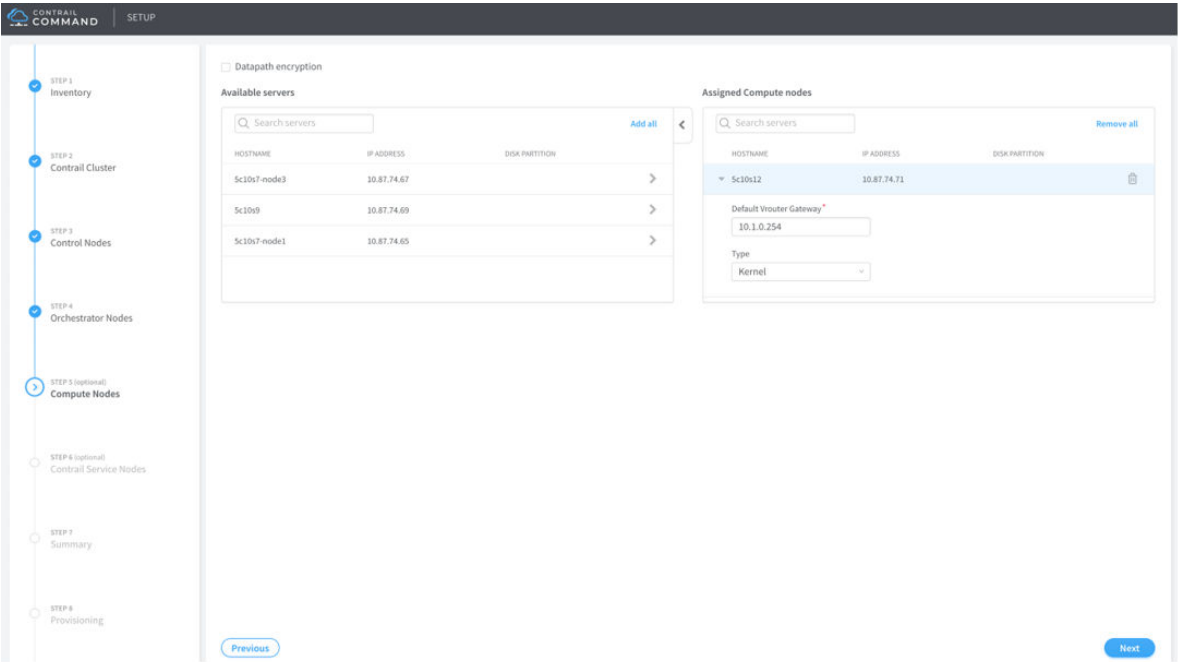
HOSTNAME	IP ADDRESS	DISK PARTITION
Sc10n7-node1	10.87.74.65	>

 Below the table, there is a "Roles" section with a dropdown menu showing selected roles:
 - openstack_control_node
 - openstack_network_node
 - openstack_storage_node
 - openstack_monitoring_node
 Below the roles is a "Next" button.

5. Select the Compute Nodes. For each compute node, enter the gateway, if it is different from what was added in the global parameters in "2" on page 98. Then set the mode.

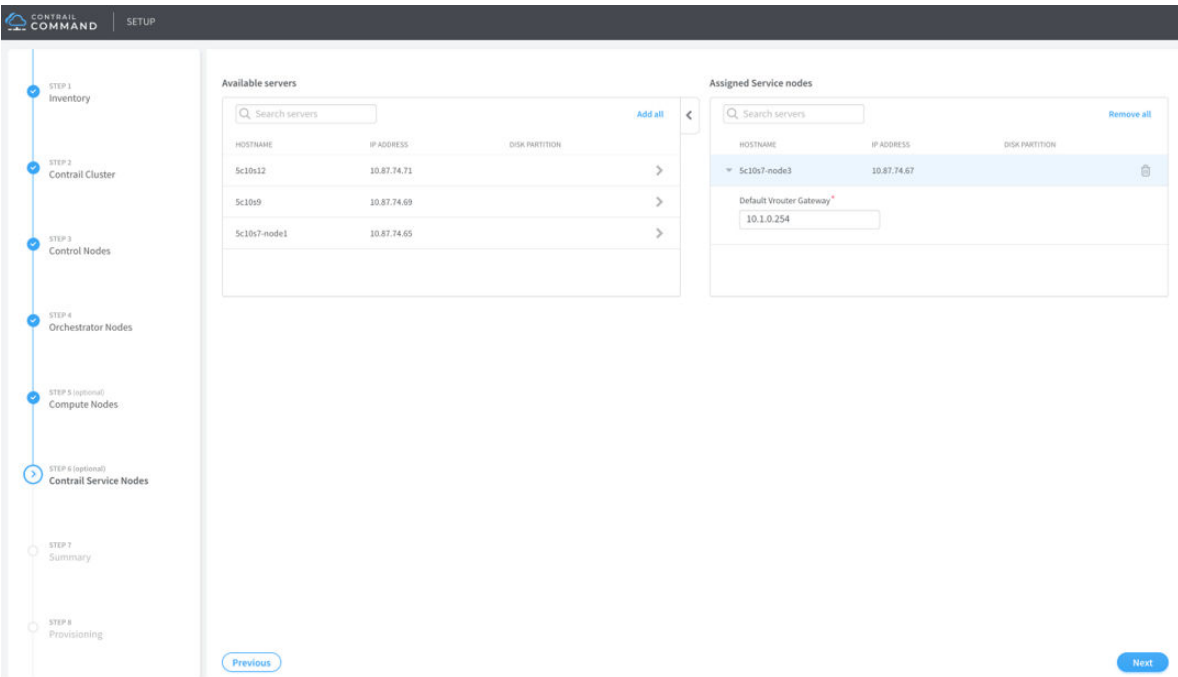
As of now, only *Kernel* mode is supported

Figure 20: Select Compute Nodes



6. Select the Contrail Service Node.

Figure 21: Select Contrail Service Nodes



7. Verify the summary of your cluster configuration and click **Provision**.

Figure 22: Verify Summary

Cluster overview

Display name

Container registry

Container registry username

Container registry password

Contrail version

Provisioner type

Domain Suffix

NTP server

Default Vrouter Gateway

Encapsulation priority

Enable ZTP

Contrail configuration

CONTROLLER_NODES

CONTROL_NODES

TSN_NODES

CONTRAIL_CONTAINER_TAG

High availability mode

Orchestrator

Openstack release

Openstack internal virtual IP

Openstack external virtual IP

Kolla globals

enable_ironic

enable_swift

Kolla passwords

keystone_admin_password

Demo-cluster

hub.juniper.net/contrail

username

password

5.0.1-0.214

ansible

local

10.84.5.100

10.1.0.254

VXLAN,MPLSoUDP,MPLSoGRE

false

10.87.74.69

10.1.02

10.1.0.67

5.0.1-0.214-ocata

false

openstack

ocata

-

-

yes

yes

Juniper123

CONTRAIL
COMMAND

SETUP

STEP 1
Inventory

STEP 2
Contrail Cluster

STEP 3
Control Nodes

STEP 4
Orchestrator Nodes

STEP 5 (optional)
Compute Nodes

STEP 6 (optional)
Contrail Service Nodes

STEP 7
Summary

STEP 8
Provisioning

Cluster overview

Display name

Container registry

Container registry username

Container registry password

Contrail version

Provisioner type

Domain Suffix

NTP server

Default Vrouter Gateway

Encapsulation priority

Enable ZTP

Contrail configuration

High availability mode

Orchestrator

Openstack release

Openstack internal virtual IP

Openstack external virtual IP

Kolla globals

Kolla passwords

Demo-cluster

hub.juniper.net/contrail

username

password

5.0.1-0.214

ansible

local

10.84.5.100

10.1.0.254

VXLAN,MPLSoUDP,MPLSoGRE

false

10.87.74.69

10.1.02

10.1.0.67

5.0.1-0.214-ocata

false

openstack

ocata

-

-

yes

yes

Juniper123

Nodes overview

All cluster nodes

Control nodes

Compute nodes

Openstack nodes

Service nodes

NAME	TYPE	IP ADDRESS	ROLES
Sc10a12	physical/virtual node	10.87.74.71	Compute node
Sc10a7-node1	physical/virtual node	10.87.74.65	Openstack node
Sc10a7-node3	physical/virtual node	10.87.74.67	Service node
Sc10a9	physical/virtual node	10.87.74.69	Control node

Previous

Provision

RELATED DOCUMENTATION

[Configuring Contrail Command | 86](#)

[Deploying Contrail Cluster using Contrail-Command and instances.yml | 106](#)

[Importing Contrail Cluster Data using Contrail Command | 113](#)

Deploying Contrail Cluster using Contrail-Command and instances.yml

Contrail Release 5.0.1 supports deploying a Contrail cluster using Contrail Command and the **instances.yml** file.

System Requirements

- A VM or physical server with:
 - 8 vCPUs
 - 64 GB RAM
 - 300 GB disk out of which 256 GB is allocated to **/root** directory.
- Internet access to and from the physical server, hereafter referred to as the Contrail Command server
- (Recommended) x86 server with CentOS 7.5 (Minimal ISO) as the base OS to install Contrail Command

Prerequisite

`docker-py` is obsolete in Contrail Release 5.0.2. You must remove `docker-py` and `docker` Python packages from all the nodes where you want to install the Contrail Command UI.

```
pip uninstall docker-py docker
```

Configuration

Perform the following steps to deploy a Contrail cluster using Contrail Command and the **instances.yml** file.

1. Install Docker on the Contrail Command server. These packages are necessary to automate the deployment of Contrail Command software.

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

```
yum install -y docker-ce
```

```
systemctl start docker
```

2. Download Contrail-Command-Deployer docker container image from hub.juniper.net. To download these containers and for access to hub.juniper.net, refer to the *Access to Contrail Registry* topic on the [Contrail software download](#) page. Allow Docker to connect to the private secure registry.

```
docker login hub.juniper.net --username < container_registry_username > --password <
container_registry_password >
```

Pull Contrail-Command-Deployer Container from the private secure registry.

```
docker pull hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
```

Example, for container_tag: 5.0.1-0.214, use the following command:

```
docker pull hub.juniper.net/contrail/contrail-command-deployer:5.0.1-0.214
```

3. Edit the input configuration **instances.yml** file. See "[No Link Title](#)" on page 107 for a sample **instances.yml** file.
4. Start the Contrail_Command_Deployer container to deploy the Contrail-Command (UI) server and provision Contrail-Cluster using the **instances.yml** file provided.

```
docker run -t --net host -e action=provision_cluster -v <ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE>:/
command_servers.yml -v <ABSOLUTE_PATH_TO_INSTANCES_FILE>:/instances.yml -d --privileged --name
contrail_command_deployer hub.juniper.net/contrail/contrail-command-deployer: <container_tag>
```

The contrail_command and contrail_mysql Contrail Command containers are deployed. Contrail cluster is also provisioned using the given **instances.yml** file.

5. (Optional) Track the progress of 4.

```
docker logs -f contrail_command_deployer
```

6. Once the playbook execution completes, log in to Contrail Command using [https:// Contrail-Command-Server-IP-Address:9091](https://Contrail-Command-Server-IP-Address:9091). Use the same user name and password that was entered in 3. Default username is admin and password is contrail123.

Sample instances.yml File

```
global_configuration:
  CONTAINER_REGISTRY: hub.juniper.net/contrail
  CONTAINER_REGISTRY_USERNAME: < container_registry_username >
  CONTAINER_REGISTRY_PASSWORD: < container_registry_password >
provider_config:
  bms:
    ssh_pwd: <Pwd>
    ssh_user: root
    ntpserver: <NTP Server>
```

```

    domainsuffix: local
instances:
  bms1:
    provider: bms
    ip: <BMS IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      vrouter:
      openstack:
      openstack_compute:
  bms2:
    provider: bms
    ip: <BMS2 IP>
    roles:
      openstack:
  bms3:
    provider: bms
    ip: <BMS3 IP>
    roles:
      openstack:
  bms4:
    provider: bms
    ip: <BMS4 IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
  bms5:
    provider: bms
    ip: <BMS5 IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:

```

```

    analytics:
    webui:
bms6:
  provider: bms
  ip: <BMS6 IP>
  roles:
    config_database:
    config:
    control:
    analytics_database:
    analytics:
    webui:
bms7:
  provider: bms
  ip: <BMS7 IP>
  roles:
    vrouter:
      PHYSICAL_INTERFACE: <Interface name>
      VROUTER_GATEWAY: <Gateway IP>
    openstack_compute:
bms8:
  provider: bms
  ip: <BMS8 IP>
  roles:
    vrouter:
      # Add following line for TSN Compute Node
      TSN_EVPN_MODE: True
    openstack_compute:
contrail_configuration:
  CLOUD_ORCHESTRATOR: openstack
  CONTRAIL_VERSION: latest or <contrail_container_tag>
  CONTRAIL_CONTAINER_TAG: <contrail_container_tag>-queens
  RABBITMQ_NODE_PORT: 5673
  VROUTER_GATEWAY: <Gateway IP>
  ENCAP_PRIORITY: VXLAN,MPLSoUDP,MPLSoGRE
  AUTH_MODE: keystone
  KEYSTONE_AUTH_HOST: <Internal VIP>
  KEYSTONE_AUTH_URL_VERSION: /v3
  CONTROLLER_NODES: < list of mgmt. ip of control nodes >
  CONTROL_NODES: <list of control-data ip of control nodes>
  OPENSTACK_VERSION: queens
kolla_config:
  kolla_globals:

```



```
openstack_release: queens
kolla_internal_vip_address: <Internal VIP>
kolla_external_vip_address: <External VIP>
openstack_release: queens
enable_haproxy: "no"      ("no" by default, set "yes" to enable)
enable_ironic: "no"       ("no" by default, set "yes" to enable)
enable_swift: "no"        ("no" by default, set "yes" to enable)
keepalived_virtual_router_id: <Value between 0-255>
kolla_passwords:
  keystone_admin_password: <Keystone Admin Password>
```

The following [Table 2 on page 110](#) defines the function of each variable present in the instances.yml file.

Table 2: Description of Variables in Instances.yml file

Instances	Description
CONTROLLER_NODES	<p>Enter the management interface IP address of the controller node.</p> <p>The default value is either populated from the IP address in the instances.yml file or the IP address provided while adding the server from contrail command UI.</p> <p>CONTROLLER_NODES is a meta variable that is used to populate other unpopulated nodes like CONFIG_NODES, CONTROL_NODES, ANALYTICS_NODES, and others.</p>
CONTROL_NODES	<p>Enter the data interface IP address of the controller node.</p> <p>The default value is populated from the CONTROLLER_NODES variable.</p> <p>The CONTROL_NODES variable also indicates the interfaces on which the control services can function.</p>
kolla_internal_vip_address	<p>Enter the ctrl-data-network IP address.</p> <p>kolla_internal_vip_address is a virtual IP address that separates internal communication requests.</p>

Table 2: Description of Variables in Instances.yml file (*Continued*)

Instances	Description
kolla_external_vip_address	<p>Enter the management network IP address.</p> <p>kolla_external_vip_address is a virtual IP address that separates external communication requests.</p>
KEYSTONE_AUTH_HOST	<p>Enter the IP address of the host, which has AUTH_MODE set to keystone.</p> <p>The default value is populated automatically.</p>
CLOUD_ORCHESTRATOR	<p>Enter the name of orchestrator you are using.</p> <p>Default value: none.</p> <p>A CLOUD_ORCHESTRATOR is the platform that automates provisioning of cloud services.</p>
CONTRAIL_VERSION	<p>Enter the Contrail version number.</p> <p>Default value: latest.</p>
RABBITMQ_NODE_PORT	<p>Enter the port number assigned to RabbitMQ node.</p> <p>Default value: 5672.</p> <p>RabbitMQ is a software that provides message queuing service, which is an efficient method of exchanging data between applications and servers.</p>
VROUTER_GATEWAY	<p>Enter the gateway IP address of the virtual router. The default gateway assigned to the virtual router is known as virtual router gateway.</p> <p>The default value is the default gateway IP address of management subnet in case of single interface setup.</p> <p>The VROUTER_GATEWAY is default gateway IP address for the compute nodes in the contrail cluster.</p>

Table 2: Description of Variables in Instances.yml file (Continued)

Instances	Description
ENCAP_PRIORITY	<p>ENCAP_PRIORITY is used to set the order of Encapsulation Priority for the EVPN supported methods - MPLS over UDP, MPLS over GRE And VxLAN.</p> <p>Default value: MPLSoUDP, MPLSoGRE, VXLAN.</p>

Table 2: Description of Variables in Instances.yml file (Continued)

AUTH_MODE	<p>Enter the desired keystone authentication mode.</p> <p>Default value: noauth.</p>
KEYSTONE_AUTH_URL_VERSION	<p>Enter the URL of the Keystone authentication server.</p> <p>Default value: /v2.0.</p> <p>Keystone is a Opentack identity service. All Openstack operations are authenticated via the keystone server.</p>
OPENSTACK_VERSION	<p>Enter the software version number of the Openstack platform.</p> <p>Default value: queens.</p>
enable_haproxy	<p>Enter yes to enable haproxy.</p> <p>Enter no to disable haproxy.</p> <p>Default value: no.</p> <p>High Availability Proxy (Haproxy) provides high availability load balancing and proxy servers in cloud networking.</p>
enable_irony	<p>Enter yes to deploy Ironic notification manager service container.</p> <p>Default value: no.</p> <p>Deploy Ironic if you want to perform Life Cycle Management in Contrail Command.</p>

enable_swift	<p>Enter yes to deploy Swift.</p> <p>Default value: no.</p> <p>Deploy Swift if you want to perform Image management in Contrail Command.</p>
keepalived_virtual_router_id	<p>Enter value between 0-255.</p> <p>Default value: 51.</p> <p>keepalived_virtual_router_id is used to configure keepalives. In Openstack HA, the keepalives are used to run and elect a master based on VRRP protocol.</p>
keystone_admin_password	<p>Enter keystone admin password.</p> <p>Default value: contrail123.</p> <p>keystone_admin_password is used to set a password for the admin to access keystone. Keystone is a Openstack identity service.</p>

RELATED DOCUMENTATION

[Configuring Contrail Command | 86](#)

[Deploying Contrail Cluster using the Contrail Command UI | 93](#)

[Importing Contrail Cluster Data using Contrail Command | 113](#)

Importing Contrail Cluster Data using Contrail Command

Contrail Release 5.0.1 supports importing of Contrail cluster data using Contrail Command.

Before you begin

docker-py is obsolete in Contrail Release 5.0.2. You must remove docker-py and docker Python packages from all the nodes where you want to install the Contrail Command UI.

```
pip uninstall docker-py docker
```

System Requirements

- A VM or physical server with:
 - 8 vCPUs
 - 64 GB RAM
 - 300 GB disk out of which 256 GB is allocated to **/root** directory.
- Internet access to and from the physical server, hereafter referred to as the Contrail Command server
- (Recommended) x86 server with CentOS 7.5 as the base OS to install Contrail Command

Configuration

Perform the following steps to import Contrail cluster data.

1. Install Docker on the Contrail Command server. These packages are necessary to automate the deployment of Contrail Command software.

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

```
yum install -y docker-ce
```

```
systemctl start docker
```

2. Download the contrail-command-deployer Docker container image to deploy contrail-command (contrail_command, contrail_mysql containers) from hub.juniper.net. Allow Docker to connect to the private secure registry.

```
docker login hub.juniper.net --username <container_registry_username> --password <container_registry_password>
```

Pull Contrail-Command-Deployer Container from the private secure registry.

```
docker pull hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
```

Example, for container_tag: 5.0.1-0.214, use the following command:

```
docker pull hub.juniper.net/contrail/contrail-command-deployer:5.0.1-0.214
```

3. Get the **instances.yml** file that was used to provision the Contrail cluster.
4. Start the Contrail-Command-Deployer container to deploy the Contrail Command (UI) server and import Contrail cluster data to Contrail Command (UI) server using the **instances.yml** file provided.

- To import a Contrail cluster using OpenStack:

```
docker run -t --net host -e orchestrator=openstack -e action=import_cluster -v <
ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE>:/command_servers.yml -v < ABSOLUTE_PATH_TO_INSTANCES_FILE>:/
instances.yml -d --privileged --name contrail_command_deployer hub.juniper.net/contrail/contrail-command-
deployer: <container_tag>
```

- To import a Contrail cluster using Kubernetes:

```
docker run -t --net host -e orchestrator=kubernetes -e action=import_cluster -v <
ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE>:/command_servers.yml -v < ABSOLUTE_PATH_TO_INSTANCES_FILE>:/
instances.yml -d --privileged --name contrail_command_deployer hub.juniper.net/contrail/contrail-command-
deployer: <container_tag>
```



NOTE: These steps explain how to import Contrail cluster data provisioned using OpenStack and Kubernetes. If your orchestrator is different, use `-e orchestrator=<YOUR_ORCHESTRATOR>` in the above command. The following orchestrators are supported:

- OpenStack—Use `orchestrator=openstack`
- Kubernetes—Use `orchestrator=kubernetes`
- Red Hat OpenShift—Use `orchestrator=openshift`
- VMware vCenter—Use `orchestrator=vcenter`

Sample instances.yml File

```
global_configuration:
  CONTAINER_REGISTRY: hub.juniper.net/contrail
  CONTAINER_REGISTRY_USERNAME: < container_registry_username >
  CONTAINER_REGISTRY_PASSWORD: < container_registry_password >
provider_config:
  bms:
    ssh_pwd: <Pwd>
    ssh_user: root
    ntpserver: <NTP Server>
    domainsuffix: local
instances:
  bms1:
    provider: bms
    ip: <BMS1 IP>
    roles:
      openstack:
  bms2:
    provider: bms
    ip: <BMS2 IP>
    roles:
      openstack:
  bms3:
    provider: bms
```

```

ip: <BMS3 IP>
roles:
  openstack:
bms4:
  provider: bms
  ip: <BMS4 IP>
  roles:
    config_database:
    config:
    control:
    analytics_database:
    analytics:
    webui:
bms5:
  provider: bms
  ip: <BMS5 IP>
  roles:
    config_database:
    config:
    control:
    analytics_database:
    analytics:
    webui:
bms6:
  provider: bms
  ip: <BMS6 IP>
  roles:
    config_database:
    config:
    control:
    analytics_database:
    analytics:
    webui:
bms7:
  provider: bms
  ip: <BMS7 IP>
  roles:
    vrouter:
      PHYSICAL_INTERFACE: <Interface name>
      VROUTER_GATEWAY: <Gateway IP>
    openstack_compute:
bms8:
  provider: bms

```

```

ip: <BMS8 IP>
roles:
  vrouter:
    # Add following line for TSN Compute Node
    TSN_EVPN_MODE: True
  openstack_compute:
contrail_configuration:
  CLOUD_ORCHESTRATOR: openstack
  CONTRAIL_VERSION: latest or <contrail_container_tag>
  CONTRAIL_CONTAINER_TAG: <contrail_container_tag>-queens
  RABBITMQ_NODE_PORT: 5673
  VROUTER_GATEWAY: <Gateway IP>
  ENCAP_PRIORITY: VXLAN,MPLSoUDP,MPLSoGRE
  AUTH_MODE: keystone
  KEYSTONE_AUTH_HOST: <Internal VIP>
  KEYSTONE_AUTH_URL_VERSION: /v3
  CONTROLLER_NODES: < list of mgmt. ip of control nodes >
  CONTROL_NODES: <list of control-data ip of control nodes>
  OPENSTACK_VERSION: queens
kolla_config:
  kolla_globals:
    openstack_release: queens
    kolla_internal_vip_address: <Internal VIP>
    kolla_external_vip_address: <External VIP>
    openstack_release: queens
    enable_haproxy: "no"      ("no" by default, set "yes" to enable)
    enable_ironic: "no"      ("no" by default, set "yes" to enable)
    enable_swift: "no"       ("no" by default, set "yes" to enable)
    keepalived_virtual_router_id: <Value between 0-255>
  kolla_passwords:
    keystone_admin_password: <Keystone Admin Password>

```

RELATED DOCUMENTATION

[Configuring Contrail Command | 86](#)

[Deploying Contrail Cluster using the Contrail Command UI | 93](#)

[Deploying Contrail Cluster using Contrail-Command and instances.yml | 106](#)

Multicloud Contrail

IN THIS CHAPTER

- [Contrail Deployment on Microsoft Azure | 118](#)
- [Deploying Contrail on Microsoft Azure | 119](#)
- [On-Premise and Azure Multicloud Deployment | 125](#)
- [Modifying Multicloud Topology | 137](#)

Contrail Deployment on Microsoft Azure

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

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

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

This workflow of spinning up Contrail SDN in a multicloud environment for Azure is automated. See ["Deploying Contrail on Microsoft Azure" on page 119](#) for information on deploying Contrail on Azure.

RELATED DOCUMENTATION

[Deploying Contrail on Microsoft Azure | 119](#)

[On-Premise and Azure Multicloud Deployment | 125](#)

[Modifying Multicloud Topology | 137](#)

Deploying Contrail on Microsoft Azure

IN THIS SECTION

- [Deployment of Contrail on Azure | 119](#)
- [Deleting Contrail Deployment from Azure | 124](#)

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

Deployment of Contrail on Azure

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

Perform the following detailed steps for deploying Contrail on Azure.

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

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

You are now redirected to the **Software Download** page.
 - c. Log in to the download page.

A **End User License Agreement** is displayed. Select **I Agree** and click on **Proceed**.
 - d. Download the file on your localhost or on your device.

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

2. Extract the contents of the .tgz file.

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

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

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

4. Define the topology. The **topology.yml** file comprises the resource group, virtual networks or clouds, and instances. Instances can have roles such as the gateway role for the multicloud gateway, controller and k8s_master roles for the controller nodes, and compute role for the compute nodes. The instance type as defined in standard Azure documentation and you must be aware of what is available in Azure for populating the topology.

```
# vi topology.yml
```

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

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

```

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

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

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

- name: contrail-az-2
  cidr_block: 10.0.0.0/16
  subnets:
    - name: subnet_contrail_az_2
      cidr_block: 10.0.100.0/24
      security_group: allow_all_protocols-contrail-az-2
  security_groups:
    - name: allow_all_protocols-contrail-az-2
      rules:
        - name: all_in-contrail-az-2

```

```

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

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

```

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

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

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

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

For example:

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

6. Set up the deployer.

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

For example:

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

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

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

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

8. Navigate to the multicloud directory.

```
# cd multicloud
```

9. Log in to Azure and authenticate your session.

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

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

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

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

10. (Optional) View your subscription details.

```
# az account list
```

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

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

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

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

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

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

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

Delete kube-dns pod using the following command.

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

Deleting Contrail Deployment from Azure

To delete Contrail from Azure, perform the following steps.

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

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

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

```
./teardown.sh
```

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

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

RELATED DOCUMENTATION

[Contrail Deployment on Microsoft Azure | 118](#)

[On-Premise and Azure Multicloud Deployment | 125](#)

[Modifying Multicloud Topology | 137](#)

On-Premise and Azure Multicloud Deployment

IN THIS SECTION

- [Installing On-Premise Contrail | 125](#)
- [Extending On-Premise Contrail To Microsoft Azure | 128](#)

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

Installing On-Premise Contrail

Before you begin

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

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

Configuration

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

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

```
>> cat /etc/apt/sources.list
# # ##### Ubuntu Main Repos
```



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

2. Run an update after changing the sources files.

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

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

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

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

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

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

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

c. Log in to the download page.

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

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

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

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

```
provider_config:
  bms:
    ssh_pwd: c0ntrail123
    ssh_user: root
    ssh_public_key:
    ssh_private_key:
    ntpserver: 10.204.217.158
    domainsuffix: device.example.net

instances:
  bms1:
    provider: bms
    ip: 19x.xxx.1.1
    private_ip: 19x.xxx.1.1
```

```

roles:
  config_database:
  config:
  control:
  analytics_database:
  analytics:
  webui:
  openstack:

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

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

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

contrail_configuration:
  CLOUD_ORCHESTRATOR: openstack
  OPENSTACK_VERSION: queens
  CONTRAIL_CONTAINER_TAG: 5.0.2-0.360-queens
  CONTROLLER_NODES: 19x.xxx.1.1
  CONTROL_NODES: 19x.xxx.1.1
  VROUTER_GATEWAY: 19x.xxx.1.254
  RABBITMQ_NODE_PORT: 5673
  KEYSTONE_AUTH_HOST: 19x.xxx.1.1
  KEYSTONE_AUTH_ADMIN_PASSWORD: c0ntrail123
  KEYSTONE_AUTH_URL_VERSION: /v3

kolla_config:

```

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

kolla_passwords:
  metadata_secret: c0ntrail123
  keystone_admin_password: c0ntrail123
```

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

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

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

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

8. Provision Openstack.

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

9. Provision Contrail.

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

Extending On-Premise Contrail To Microsoft Azure

Before you begin

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

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

Configuration

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

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

- Untar the deployer package.

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

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

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

2. Navigate to the deployer directory.

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

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

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

The agent has no identities.

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

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

```
- provider: OnPrem
  organization: Juniper
  project: multicloud
  instances:
    - name: nodec33
      roles:
        - gateway
      provision: true
      username: root
      password: c0ntrail123
      public_ip: 1x.xxx.217.168
      private_ip: 19x.xxx.2.1
      private_subnet:
        - 19x.xxx.2.0/24
        - 19x.xxx.1.0/24
      protocols_mode:
        - ssl_client
      interface: enp1s0f1
      gateway: 19x.xxx.2.254
    - name: nodec28
      roles:
        - controller: false
        - k8s_master
      provision: true
      username: root
      password: c0ntrail123
      public_ip: 1x.xxx.217.13
      private_ip: 19x.xxx.1.1
      private_subnet: 19x.xxx.1.0/24
```

```

    interface: enp1s0f1
  - name: nodec10
    roles:
      - compute_node: false
      - k8s_node
    provision: true
    username: root
    password: c0ntrail123
    public_ip: 1x.xxx.217.176
    private_ip: 19x.xxx.1.2
    private_subnet: 19x.xxx.1.0/24
    interface: enp1s0f1
    gateway: 19x.xxx.1.254
  - name: nodec50
    roles:
      - compute_node: false
      - k8s_node
    provision: true
    username: root
    password: c0ntrail123
    public_ip: 1x.xxx.217.153
    private_ip: 19x.xxx.1.3
    private_subnet: 19x.xxx.1.0/24
    interface: enp1s0f1
    gateway: 19x.xxx.1.254
- provider: azure
  organization: Juniper
  project: multicloud
  regions:
    - name: WestUS2
      resource_group: contrail-test-west-us-2
      vnet:
        - name: rg-vpc-1
          cidr_block: 17x.xx.1.0/24
          subnets:
            - name: rg-subnet-1
              cidr_block: 17x.xx.1.0/25
              security_group: rg-sg-1
          security_groups:
            - name: rg-sg-1
              rules:
                - name: rg-all_in_1
                  direction: inbound

```

```

      - name: rg-all_out_1
        direction: outbound
instances:
- name: rg-gw-1
  availability_zone: 1
  provision: true
  username: ubuntu
  os: ubuntu16
  os_version: 16.04.201705080
  instance_type: Standard_F2
  subnets: rg-subnet-1
  interface: eth1
  roles:
    - gateway
  protocols_mode:
    - ssl_server
    - ipsec_server
    - ipsec_client
- name: rg-gw-2
  availability_zone: 1
  provision: true
  username: ubuntu
  os: ubuntu16
  os_version: 16.04.201705080
  instance_type: Standard_F2
  subnets: rg-subnet-1
  interface: eth1
  roles:
    - gateway
  protocols_mode:
    - ssl_server
    - ipsec_server
    - ipsec_client
- name: rg-compute-1
  availability_zone: 1
  provision: true
  username: ubuntu
  os: ubuntu16
  os_version: 16.04.201705080
  instance_type: Standard_F2
  subnets: rg-subnet-1
  interface: eth0
  roles:

```

```

      - compute_node
- name: rg-compute-2
  availability_zone: 1
  provision: true
  username: ubuntu
  os: ubuntu16
  os_version: 16.04.201705080
  instance_type: Standard_F2
  subnets: rg-subnet-1
  interface: eth0
  roles:
    - compute_node
- name: rg-vpc-2
  cidr_block: 17x.xx.3.0/24
  subnets:
    - name: rg-subnet-2
      cidr_block: 17x.xx.3.0/25
      security_group: rg-sg-2
  security_groups:
    - name: rg-sg-2
      rules:
        - name: rg-all_in_2
          direction: inbound
        - name: rg-all_out_2
          direction: outbound
  instances:
    - name: rg-gw-21
      availability_zone: 2
      provision: true
      username: ubuntu
      os: ubuntu16
      os_version: 16.04.201705080
      instance_type: Standard_F2
      subnets: rg-subnet-2
      interface: eth1
      roles:
        - gateway
  protocols_mode:
    - ssl_server
    - ipsec_server
    - ipsec_client
- name: rg-gw-22
  availability_zone: 2

```

```

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

```

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

- contrail common yaml file

```

# vi contrail-multi-cloud/ansible/contrail/common.yml
ANSIBLE_DEPLOYER_BRANCH: R5.0
global_configuration:

```



```

#REGISTRY_PRIVATE_INSECURE: TRUE
CONTAINER_REGISTRY: "hub.juniper.net/contrail"
CONTAINER_REGISTRY_USERNAME: XXXX
CONTAINER_REGISTRY_PASSWORD: XXXX
contrail_user: admin
contrail_password: c0ntrail123
contrail_tenant: admin
contrail_port: 8082
contrail_tenant: default-project
provider_config:
  bms:
    ssh_pwd: c0ntrail123
    ssh_user: root
contrail_configuration:
  CONTRAIL_VERSION: 5.0.2-0.360
  ENCAP_PRIORITY: "MPLSoUDP,MPLSoGRE,VXLAN"
  CLOUD_ORCHESTRATOR: kubernetes
# VROUTER_ENCRYPTION: True

```

- gateway common yaml file

```

# vi contrail-multi-cloud/ansible/gateway/common.yml
PATH_CONFIG: "/etc/multicloud"
PATH_SSL_CONFIG_LOCAL: "~/.multicloud/ssl"
PATH_SSL_CONFIG: "{{ PATH_CONFIG }}/ssl"
PATH_OPENVPN_CONFIG: "{{ PATH_CONFIG }}/openvpn"
PATH_BIRD_CONFIG: "{{ PATH_CONFIG }}/bird"
PATH_STRONGSWAN_CONFIG: "{{ PATH_CONFIG }}/strongswan"
PATH_VRRP_CONFIG: "{{ PATH_CONFIG }}/vrrp"
PATH_AWS_CONFIG: "{{ PATH_CONFIG }}/aws"
PATH_INTERFACE_CONFIG: "/etc/network/interfaces.d"
PATH_FW_CONFIG: "{{ PATH_CONFIG }}/firewall"
PATH_GCP_CONFIG: "{{ PATH_CONFIG }}/gcp"
PATH_SECRET_CONFIG: "{{ PATH_CONFIG }}/secret"
CONTAINER_REGISTRY: "hub.juniper.net/contrail"
CONTRAIL_MULTICLOUD_VERSION: 5.0.2-0.360
CONTAINER_REGISTRY_USERNAME: XXXX
CONTAINER_REGISTRY_PASSWORD: XXXX
UPGRADE_KERNEL: False
AS: 65000
vpn_lo_network: 1xx.x5.0.0/16
vpn_network: 1xx.x4.0.0/16

```

```

required_bgp_rrs: 1
openvpn_port: 443
local_interface: eth1
bfd_interval: 200ms
bfd_multiplier: 5
bfd_interval_multihop: 500ms
bfd_multiplier_multihop: 5
core_bgp_secret: bgp_

```

6. Log in to the Docker repository to check out the **contrail-multi-cloud** deployer container.
 >> docker login hub.juniper.net/contrail-nightly -u XXX -p xxx
7. Start the deployer container and mount the deployer files to the container.
 >> ./deployer.sh -r hub.juniper.net/contrail-nightly -t 5.0.2-0.349 -v \$PWD:/root/multicloud -k
8. Check if the generated key file is added to the ssh agent. If the key file has not been added, add it manually.
 >> ssh-add -l

 >> ssh-add keys/<keyfile>
9. Log in to the container with default multicloud password.
 >> ssh -o PreferredAuthentications=password -o PubkeyAuthentication=no -A root@127.0.0.1 -p 2222
10. Ensure that the ssh agent is running and contains the correct key file. Else, start the agent and add the keyfile.
 >> ssh-add -l
11. Check if the ssh-keys are added to all the on-premise nodes for Ansible playbooks to succeed.
 >> sshpass -p \"c0ntrail123\" ssh-copy-id -o StrictHostKeyChecking=no -i /root/multicloud/keys/contrail-multicloud-key-11513.pub root@<OnPrem nodes mgmt IP>
12. (Optional) If step 11 fails, use the following command to add keys to all four on-premise nodes.
 >> ssh-copy-id root@<OnPrem nodes mgmt IP>
13. Register your device and log in to Azure.
 Using the `az login` command displays a secure link to the Azure portal and a code for device authentication.

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

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

- Enter your Azure account login credentials.

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

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

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

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

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

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

- `ssh -i contrail-multicloud-key-11513 ubuntu@17x.xx.1.5`
- `ssh -i contrail-multicloud-key-11513 ubuntu@<public-ip-of-Azure-GW-VM>`

RELATED DOCUMENTATION

[Contrail Deployment on Microsoft Azure | 118](#)

[Deploying Contrail on Microsoft Azure | 119](#)

[Modifying Multicloud Topology | 137](#)

Modifying Multicloud Topology

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

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

```
# vi topology.yml
```

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

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

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

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



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

RELATED DOCUMENTATION

[Contrail Deployment on Microsoft Azure | 118](#)

[Deploying Contrail on Microsoft Azure | 119](#)

[On-Premise and Azure Multicloud Deployment | 125](#)

Using Contrail with Red Hat

IN THIS CHAPTER

- [Deploying Contrail with Red Hat OpenStack Platform Director 13 | 138](#)
- [Provisioning Red Hat OpenShift Container Platform Clusters Using Ansible Deployer | 193](#)

Deploying Contrail with Red Hat OpenStack Platform Director 13

IN THIS SECTION

- [Overview | 138](#)
- [OSPd Features | 139](#)
- [Composable Roles | 139](#)
- [Preparing the Environment for Deployment | 140](#)
- [Creating Infrastructure | 143](#)
- [undercloud Configuration | 145](#)
- [undercloud Post Configuration | 150](#)
- [overcloud Configuration | 151](#)

This document explains how to integrate a Contrail 5.0.1 installation with Red Hat OpenStack Platform Director 13.

Overview

Red Hat OpenStack Platform provides an installer named Director (RHOSPD). The Red Hat Director installer is based on the OpenStack project TripleO (OOO, OpenStack on OpenStack). TripleO is an open source project that uses features of OpenStack to deploy a fully functional, tenant-facing OpenStack environment.

TripleO can be used to deploy a RDO based OpenStack environment integrated with Tungsten Fabric. Red Hat OpenStack Platform Director (RHOSPd) can be used to deploy a RHOSP based OpenStack environment integrated with Contrail.

OSPd Features

OSPd uses the concepts of undercloud and overcloud. OSPd sets up an undercloud, an operator-facing deployment cloud that contains the OpenStack components needed to deploy and manage an overcloud, a tenant-facing cloud that hosts user workloads.

The overcloud is the deployed solution that can represent a cloud for any purpose, such as production, staging, test, and so on. The operator can select to deploy to their environment any of the available overcloud roles, such as controller, compute, and the like.

OSPd leverages existing core components of OpenStack including Nova, Ironi, Neutron, Heat, Glance, and Ceilometer to deploy OpenStack on bare metal hardware.

- Nova and Ironi are used in the undercloud to manage the bare metal instances that comprise the infrastructure for the overcloud.
- Neutron is used to provide a networking environment in which to deploy the overcloud.
- Glance stores machine images.
- Ceilometer collects metrics about the overcloud.

For more information about OSPd architecture, see [OSPd documentation](#)

Composable Roles

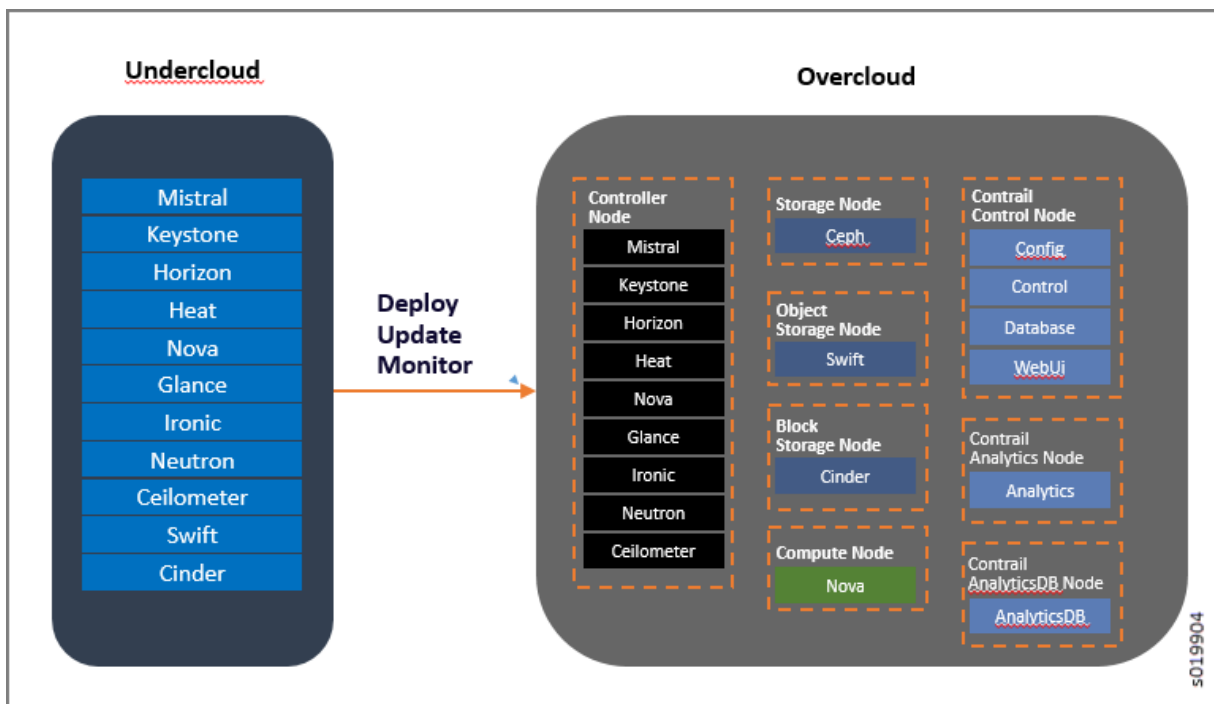
OSPd enables composable roles. Each role is a group of services that are defined in Heat templates. Composable roles gives the operator the flexibility to add and modify roles as needed.

The following are the Contrail roles used for integrating Contrail to the overcloud environment:

- Contrail Controller
- Contrail Analytics
- Contrail Analytics Database
- Contrail-TSN
- Contrail-DPDK

Figure 23 on page 140 shows the relationship and components of an undercloud and overcloud architecture for Contrail.

Figure 23: undercloud and overcloud with Roles



Preparing the Environment for Deployment

The overcloud roles can be deployed to bare metal servers or to virtual machines (VMs). The compute nodes must be deployed to bare metal systems.

Ensure your environment is prepared for the Red Hat deployment. Refer to [Red Hat documentation](#).

Preparing for the Contrail Roles

Ensure the following requirements are met for the Contrail nodes per role.

- Non-high availability: A minimum of 4 overcloud nodes are needed for control plane roles for a non-high availability deployment:
 - 1x contrail-config (includes Contrail control)
 - 1x contrail-analytics
 - 1x contrail-analytics-database

- 1x OpenStack controller
- High availability: A minimum of 12 overcloud nodes are needed for control plane roles for a high availability deployment:
 - 3x contrail-config (includes Contrail control)
 - 3x contrail-analytics
 - 3x contrail-analytics-database
 - 3x OpenStack controller
- If the control plane roles will be deployed to VMs, use 3 separate physical servers and deploy one role of each kind to each physical server.

RHOSP Director expects the nodes to be provided by the administrator, for example, if you are deploying to VMs, the administrator must create the VMs before starting with deployment.

Preparing for the Underlay Network

Refer to Red Hat documentation for planning and implementing underlay networking, including the kinds of networks used and the purpose of each:

- [Planning Networks](#)
- [Networking Requirements](#)

At a high level, every overcloud node must support IPMI.

Preparing for the Provisioning Network

Ensure the following requirements are met for the provisioning network.

- One NIC from every machine must be in the same broadcast domain of the provisioning network, and it should be the same NIC on each of the overcloud machines. For example, if you use the second NIC on the first overcloud machine, you should use the second NIC on each additional overcloud machine.

During installation, these NICs will be referenced by a single name across all overcloud machines.

- The provisioning network NIC should not be the same NIC that you are using for remote connectivity to the undercloud machine. During the undercloud installation, an Open vSwitch bridge will be created for Neutron and the provisioning NIC will be bridged to the Open vSwitch bridge. Consequently, connectivity would be lost if the provisioning NIC was also used for remote connectivity to the undercloud machine.

- The provisioning NIC on the overcloud nodes must be untagged.
- You must have the MAC address of the NIC that will PXE boot the IPMI information for the machine on the provisioning network. The IPMI information will include such things as the IP address of the IPMI NIC and the IPMI username and password.
- All of the networks must be available to all of the Contrail roles and computes.

Network Isolation

OSPd enables configuration of isolated overcloud networks. Using this approach, it is possible to host traffic in isolated networks for specific types of network traffic, such as tenants, storage, API, and the like. This enables assigning network traffic to specific network interfaces or bonds.

When isolated networks are configured, the OpenStack services are configured to use the isolated networks. If no isolated networks are configured, all services run on the provisioning network.

The following networks are typically used when using network isolation topology:

- Provisioning- for the undercloud control plane
- Internal API- for OpenStack internal APIs
- Tenant
- Storage
- Storage Management
- External
 - Floating IP- Can either be merged with external or can be a separate network.
- Management

Supported Combinations

The following combinations of Operating System/OpenStack/Deployer/Contrail are supported:

Table 3: Compatibility Matrix

Operating System	OpenStack	Deployer	Contrail
RHEL 7.5	OSP13	OSPd13	Contrail 5.0.1

Table 3: Compatibility Matrix (*Continued*)

Operating System	OpenStack	Deployer	Contrail
CentOS 7.5	RDO queens/stable	tripleo queens/stable	Tungsten Fabric latest

Creating Infrastructure

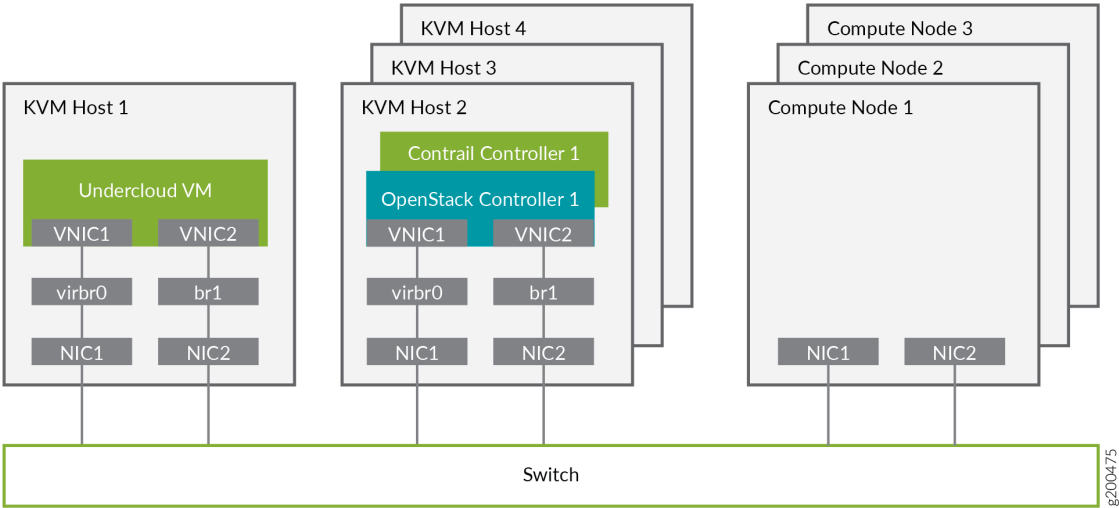
There are many different ways on how to create the infrastructure providing the control plane elements. The following example illustrates all control plane functions as Virtual Machines hosted on KVM hosts.

Table 4: Control Plane Functions

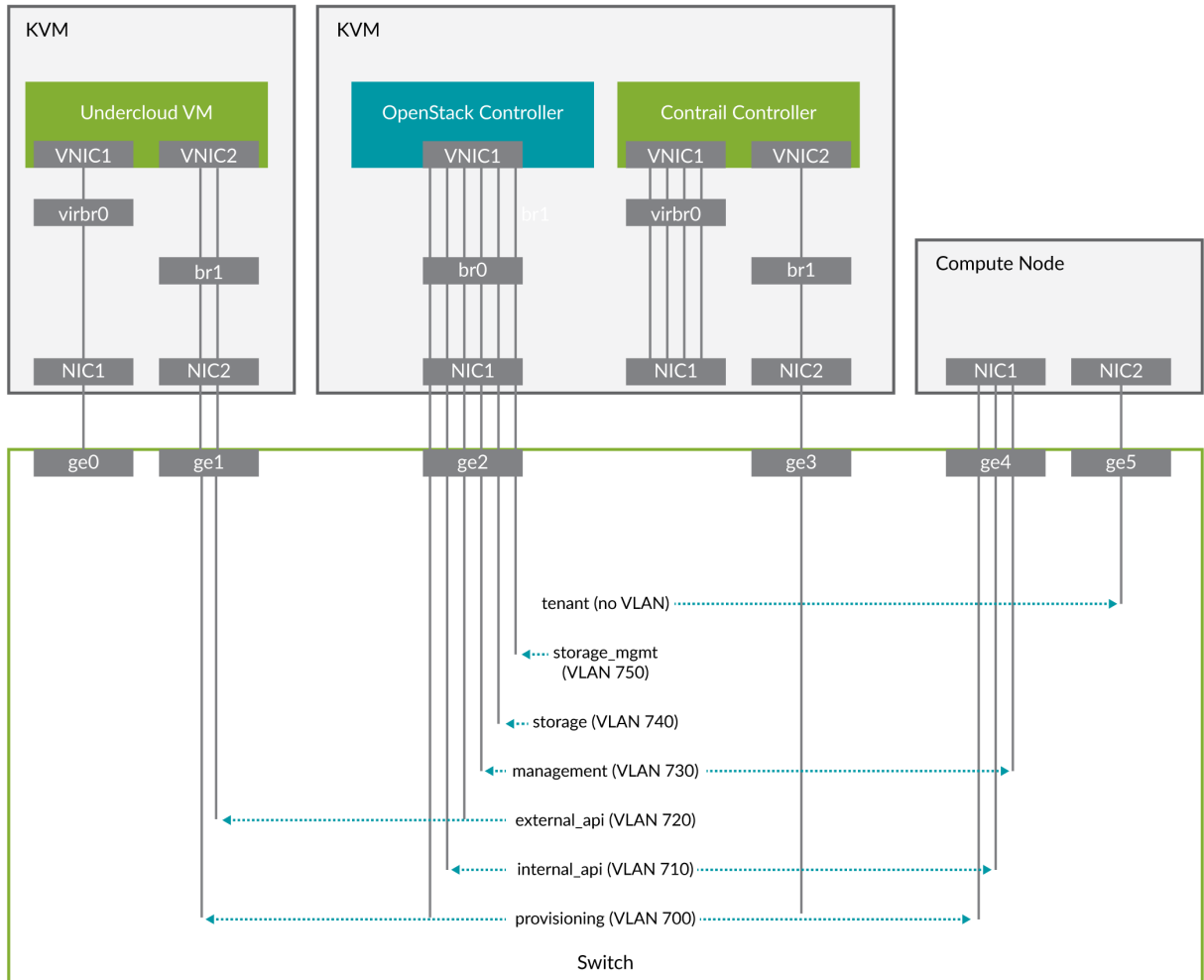
KVM Host	Virtual Machines
KVM1	undercloud
KVM2	OpenStack Controller 1, Contrail Controller 1
KVM3	OpenStack Controller 2, Contrail Controller 2
KVM4	OpenStack Controller 2, Contrail Controller 2

Sample Topology

Layer 1: Physical Layer



Layer 2: Logical Layer



g200476

undercloud Configuration

Physical Switch

Use the following information to create ports and Trunked VLANs

Table 5: Physical Switch

Port	Trunked VLAN	Native VLAN
ge0	-	-
ge1	700, 720	-
ge2	700, 710, 720, 730, 740, 750	-
ge3	-	-
ge4	710, 730	700
ge5	-	-

undercloud and overcloud KVM Host Configuration

undercloud and overcloud KVM hosts will need virtual switches and virtual machine definitions configured. You can deploy any KVM host operating system version which supports KVM and OVS. The following example shows a RHEL/CentOS based system. If you are using RHEL, the system must be subscribed.

- *Install Basic Packages*

```
yum install -y libguestfs \ libguestfs-tools \ openvswitch \ virt-install \ kvm libvirt \ libvirt-python \
python-virtualbmc \ python-virtinst
```

- *Start libvirtd and ovs*

```
systemctl start libvirtd systemctl start openvswitch
```

- *Configure vSwitch*

Table 6: Configure vSwitch

Bridge	Trunked VLAN	Native VLAN
br0	710, 720, 730 740, 750	700
br1	-	-

Create bridges

```

ovs-vsctl add-br br0 ovs-vsctl add-br br1 ovs-vsctl add-port br0 NIC1 ovs-vsctl add-port br1 NIC2 cat << EOF >
br0.xml <network> <name>br0</name> <forward mode='bridge'/> <bridge name='br0'/> <virtualport
type='openvswitch'/> <portgroup name='overcloud'/> <vlan trunk='yes'> <tag id='700' nativeMode='untagged'/>
<tag id='710'/> <tag id='720'/> <tag id='730'/> <tag id='740'/> <tag id='750'/> </vlan> </portgroup> </network>
EOF cat << EOF > br1.xml <network> <name>br1</name> <forward mode='bridge'/> <bridge name='br1'/> <virtualport
type='openvswitch'/> </network> EOF virsh net-define br0.xml virsh net-start br0 virsh net-autostart br0 virsh
net-define br1.xml virsh net-start br1 virsh net-autostart br1

```

- *Create overcloud VM Definitions on the overcloud KVM Hosts (KVM2-KVM4)*



NOTE: overcloud VM definition is required to create on each overcloud KVM host.



NOTE: Use the following formula to create the number of roles per overcloud KVM host:

ROLES=compute:2,contrail-controller:1,control:1

The following example defines:

2x compute nodes 1x cotrail controller node 1x openstack controller node

```

num=0 ipmi_user=<user> ipmi_password=<password> libvirt_path=/var/lib/libvirt/images port_group=overcloud
prov_switch=br0 /bin/rm ironic_list IFS=',' read -ra role_list <<< "${ROLES}" for role in ${role_list[@]}; do
role_name=`echo $role|cut -d ":" -f 1` role_count=`echo $role|cut -d ":" -f 2` for count in `seq 1 $
{role_count}`; do echo $role_name $count qemu-img create -f qcow2 ${libvirt_path}/${role_name}_${count}.qcow2
99G virsh define /dev/stdin <<EOF $(virt-install --name ${role_name}_${count} \ --disk ${libvirt_path}/${
role_name}_${count}.qcow2 \ --vcpus=4 \ --ram=16348 \ --network network=br0,model=virtio,portgroup=$
{port_group} \ --network network=br1,model=virtio \ --virt-type kvm \ --cpu host \ --import \ --os-variant
rhel7 \ --serial pty \ --console pty,target_type=virtio \ --graphics vnc \ --print-xml) EOF vbmc add $
{role_name}_${count} --port 1623${num} --username ${ipmi_user} --password ${ipmi_password} / vbmc start $
{role_name}_${count} prov_mac=`virsh domiflist ${role_name}_${count}|grep ${prov_switch}|awk '{print $5}'`

```

```
vm_name=${role_name}-${count}-${hostname} -s` kvm_ip=`ip route get 1 |grep src |awk '{print $7}'` echo $
{prov_mac} ${vm_name} ${kvm_ip} ${role_name} 1623${num}>> ironic_list num=$(expr $num + 1) done done
```



CAUTION: One *ironic_list* file per KVM host will be created. You need to combine all the *ironic_list* files from each KVM host on the undercloud.

The following output shows combined list from all the three Overvcloud KVM hosts:

```
52:54:00:e7:ca:9a compute-1-5b3s31 10.87.64.32 compute 16230 52:54:00:30:6c:3f compute-2-5b3s31
10.87.64.32 compute 16231 52:54:00:9a:0c:d5 contrail-controller-1-5b3s31 10.87.64.32 contrail-
controller 16232 52:54:00:cc:93:d4 control-1-5b3s31 10.87.64.32 control 16233 52:54:00:28:10:d4
compute-1-5b3s30 10.87.64.31 compute 16230 52:54:00:7f:36:e7 compute-2-5b3s30 10.87.64.31
compute 16231 52:54:00:32:e5:3e contrail-controller-1-5b3s30 10.87.64.31 contrail-controller
16232 52:54:00:d4:31:aa control-1-5b3s30 10.87.64.31 control 16233 52:54:00:d1:d2:ab
compute-1-5b3s32 10.87.64.33 compute 16230 52:54:00:ad:a7:cc compute-2-5b3s32 10.87.64.33
compute 16231 52:54:00:55:56:50 contrail-controller-1-5b3s32 10.87.64.33 contrail-controller
16232 52:54:00:91:51:35 control-1-5b3s32 10.87.64.33 control 16233
```

- *Create undercloud VM Definitions on the undercloud KVM host (KVM1)*



NOTE: undercloud VM definitions is required to create only on undercloud KVM.

1. Create images directory

```
mkdir ~/images cd images
```

2. Retrieve the image



NOTE: The image must be retrieved based on the operating system:

- CentOS

```
curl https://cloud.centos.org/centos/7/images/CentOS-7-x86_64-GenericCloud-1802.qcow2.xz
\ -o CentOS-7-x86_64-GenericCloud-1802.qcow2.xz zx -d images/CentOS-7-x86_64-
GenericCloud-1802.qcow2.xz cloud_image=~/.images/CentOS-7-x86_64-
GenericCloud-1804_02.qcow2
```

- RHEL

```
Download rhel-server-7.5-update-1-x86_64-kvm.qcow2 from Red Hat portal to ~/.images
cloud_image=~/.images/rhel-server-7.5-update-1-x86_64-kvm.qcow2
```

3. Customize the undercloud image

```
undercloud_name=queensa undercloud_suffix=local root_password=<password>
stack_password=<password> export LIBGUESTFS_BACKEND=direct qemu-img create -f
qcow2 /var/lib/libvirt/images/${undercloud_name}.qcow2 100G virt-resize --expand /dev/sda1 $
{cloud_image} /var/lib/libvirt/images/${undercloud_name}.qcow2 virt-customize -a /var/lib/
libvirt/images/${undercloud_name}.qcow2 \ --run-command 'xfs_growfs /' \ --root-password
password:${root_password} \ --hostname ${undercloud_name}.${undercloud_suffix} \ --run-
command 'useradd stack' \ --password stack:password:${stack_password} \ --run-command 'echo
"stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack' \ --chmod 0440:/etc/
sudoers.d/stack \ --run-command 'sed -i "s/PasswordAuthentication no/PasswordAuthentication
yes/g" /etc/ssh/sshd_config' \ --run-command 'systemctl enable sshd' \ --run-command 'yum
remove -y cloud-init' \ --selinux-relabel
```

4. Define the undercloud virsh template

```
vcpus=8 vram=32000 virt-install --name ${undercloud_name} \ --disk /var/lib/libvirt/images/$
{undercloud_name}.qcow2 \ --vcpus=${vcpus} \ --ram=${vram} \ --network
network=default,model=virtio \ --network network=br0,model=virtio,portgroup=overcloud \ --
virt-type kvm \ --import \ --os-variant rhel7 \ --graphics vnc \ --serial pty \ --
noautoconsole \ --console pty,target_type=virtio
```

5. Start the undercloud VM

```
virsh start ${undercloud_name}
```

6. Retrieve the undercloud IP

It might take several seconds before the IP is available.

```
undercloud_ip=`virsh domifaddr ${undercloud_name} |grep ipv4 |awk '{print $4}' |awk -F"/"
'${print $1}'` ssh-copy-id ${undercloud_ip}
```

undercloud Configuration

1. Login to the undercloud VM from the undercloud KVM host

```
ssh ${undercloud_ip}
```

2. Configure Hostname

```
undercloud_name='hostname -s' undercloud_suffix='hostname -d' hostnamectl set-hostname ${undercloud_name}.${
undercloud_suffix} hostnamectl set-hostname --transient ${undercloud_name}.${undercloud_suffix}
```




NOTE: Make sure to set undercloud IP in the host file located at **etc\hosts**.

The commands will be as follows assuming the mgmt NIC is eth0:

```
undercloud_ip=`ip addr sh dev eth0 |grep "inet " |awk '{print $2}' |awk -F"/" '{print $1}'` echo
${undercloud_ip} ${undercloud_name}.${undercloud_suffix} ${undercloud_name} >> /etc/hosts`
```

3. Setup Repositories



NOTE: The repository must be setup based on the operating system:

- CentOS

```
tripleo_repos=`python -c 'import requests;r = requests.get("https://trunk.rdoproject.org/centos7-queens/
current"); print r.text ' |grep python2-tripleo-repos|awk -F"href=\"" '{print $2}'|awk -F"\"" '{print $1}'`
yum install -y https://trunk.rdoproject.org/centos7-queens/current/${tripleo_repos} tripleo-repos -b queens
current
```

- RHEL

```
#Register with Satellite (can be done with CDN as well) satellite_fqdn=device.example.net act_key=xxx
org=example yum localinstall -y http://${satellite_fqdn}/pub/katello-ca-consumer-latest.noarch.rpm
subscription-manager register --activationkey=${act_key} --org=${org}
```

4. Install Tripleo Client

```
yum install -y python-tripleoclient tmux
```

5. Copy *undercloud.conf*

```
su - stack cp /usr/share/instack-undercloud/undercloud.conf.sample ~/undercloud.conf
```

undercloud Installation

Run the following command to install the undercloud:

```
openstack undercloud install source stackrc
```

undercloud Post Configuration

Complete the following configurations post undercloud installation:

- Configure forwarding:

```
sudo iptables -A FORWARD -i br-ctlplane -o eth0 -j ACCEPT sudo iptables -A FORWARD -i eth0 -o br-ctlplane -m
state --state RELATED,ESTABLISHED -j ACCEPT sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

- Add external API interface:

```
sudo ip link add name vlan720 link br-ctlplane type vlan id 720 sudo ip addr add 10.2.0.254/24 dev vlan720
sudo ip link set dev vlan720 up
```

- Add stack user to the docker group:

```
newgrp docker exit su - stack source stackrc
```

overcloud Configuration

Configuration

- Configure nameserver for overcloud nodes

```
undercloud_nameserver=8.8.8.8 openstack subnet set 'openstack subnet show ctlplane-subnet -c id -f value' --
dns-nameserver ${undercloud_nameserver}
```

- overcloud images

1. Create image directory

```
mkdir images cd images
```

2. Get overcloud images

- TripleO

```
curl -O https://images.rdoproject.org/queens/rdo_trunk/current-tripleo-rdo/ironic-python-agent.tar curl
-O https://images.rdoproject.org/queens/rdo_trunk/current-tripleo-rdo/overcloud-full.tar tar xvf
ironic-python-agent.tar tar xvf overcloud-full.tar
```

- OSP13

```
sudo yum install -y rhosp-director-images rhosp-director-images-ipa for i in /usr/share/rhosp-director-
images/overcloud-full-latest-13.0.tar /usr/share/rhosp-director-images/ironic-python-agent-
latest-13.0.tar ; do tar -xvf $i; done
```

3. Upload overcloud images

```
cd openstack overcloud image upload --image-path /home/stack/images/
```

- Prepare Ironic

OpenStack bare metal provisioning a.k.a Ironic is an integrated OpenStack program which aims to provision bare metal machines instead of virtual machines, forked from the Nova baremetal driver. It

is best thought of as a bare metal hypervisor API and a set of plugins which interact with the bare metal hypervisors



NOTE: Make sure to combine *ironic_list* files from the three overcloud KVM hosts.

1. Add the overcloud VMs to Ironic

```
ipmi_password=<password>
ipmi_user=<user>
while IFS= read -r line; do
    mac=`echo $line|awk '{print $1}'`
    name=`echo $line|awk '{print $2}'`
    kvm_ip=`echo $line|awk '{print $3}'`
    profile=`echo $line|awk '{print $4}'`
    ipmi_port=`echo $line|awk '{print $5}'`
    uuid=`openstack baremetal node create --driver ipmi \
        --property cpus=4 \
        --property memory_mb=16348 \
        --property local_gb=100 \
        --property cpu_arch=x86_64 \
        --driver-info ipmi_username=${ipmi_user} \
        --driver-info ipmi_address=${kvm_ip} \
        --driver-info ipmi_password=${ipmi_password} \
        --driver-info ipmi_port=${ipmi_port} \
        --name=${name} \
        --property capabilities=profile:$
{profile},boot_option:local \
        -c uuid -f value`
    openstack baremetal port create --node ${uuid} ${mac}
done < <(cat ironic_list)

DEPLOY_KERNEL=$(openstack image show bm-deploy-kernel -f value -c id)
DEPLOY_RAMDISK=$(openstack image show bm-deploy-ramdisk -f value -c id)

for i in `openstack baremetal node list -c UUID -f value`; do
    openstack baremetal node set $i --driver-info deploy_kernel=$DEPLOY_KERNEL --driver-info
    deploy_ramdisk=$DEPLOY_RAMDISK
done

for i in `openstack baremetal node list -c UUID -f value`; do
```

```
openstack baremetal node show $i -c properties -f value
done
```

2. Introspect overcloud node

```
for node in $(openstack baremetal node list -c UUID -f value) ; do
    openstack baremetal node manage $node
done
openstack overcloud node introspect --all-manageable --provide
```

3. Add Baremetal Server (BMS) to Ironic

- Automated profiling

Evaluate the attributes of the physical server. The server will be automatically profiled based on the rules.

The following example shows how to create a rule for system manufacturer as “Supermicro” and memory greater or equal to 128GByte

```
cat << EOF > ~/rule_compute.json
[
{
    "description": "set physical compute",
    "conditions": [
        {"op": "eq", "field": "data://auto_discovered", "value": true},
        {"op": "eq", "field": "data://inventory.system_vendor.manufacturer",
            "value": "Supermicro"},
        {"op": "ge", "field": "memory_mb", "value": 128000}
    ],
    "actions": [
        {"action": "set-attribute", "path": "driver_info/ipmi_username",
            "value": "<user>"},
        {"action": "set-attribute", "path": "driver_info/ipmi_password",
            "value": "<password>"},
        {"action": "set-capability", "name": "profile", "value": "compute"},
        {"action": "set-attribute", "path": "driver_info/ipmi_address", "value":
            "{data[inventory][bmc_address]}" }
    ]
}
```

```
]
EOF
```

You can import the rule by:

```
openstack baremetal introspection rule import ~/rule_compute.json
```

- Scanning of BMC ranges

Scan the BMC IP range and automatically add new servers matching the above rule by:

```
ipmi_range=10.87.122.25/32
ipmi_password=<password>
ipmi_user=<user>
openstack overcloud node discover --range ${ipmi_range} \
  --credentials ${ipmi_user}:${ipmi_password} \
  --introspect --provide
```

- Create Flavor

```
for i in compute-dpdk \
compute-sriov \
contrail-controller \
contrail-analytics \
contrail-database \
contrail-analytics-database; do
  openstack flavor create $i --ram 4096 --vcpus 1 --disk 40
  openstack flavor set --property "capabilities:boot_option"="local" \
    --property "capabilities:profile"="${i}" ${i}
done
```

- Create TripleO-Heat-Template Copy

```
cp -r /usr/share/openstack-tripleo-heat-templates/ tripleo-heat-templates
git clone https://github.com/juniper/contrail-tripleo-heat-templates -b stable/queens
cp -r contrail-tripleo-heat-templates/* tripleo-heat-templates/
```

- Create and Upload Containers

- OpenStack Contrainers

1. Create OpenStack container file



NOTE: The container must be created based on the OpenStack program:

- TripleO

```
openstack overcloud container image prepare \
  --namespace docker.io/tripleoqueens \
  --tag current-tripleo \
  --tag-from-label rdo_version \
  --output-env-file=~/.overcloud_images.yaml

tag=`grep "docker.io/tripleoqueens" docker_registry.yaml |tail -1 |awk -F":"
'{print $3}'`

openstack overcloud container image prepare \
  --namespace docker.io/tripleoqueens \
  --tag ${tag} \
  --push-destination 192.168.24.1:8787 \
  --output-env-file=~/.overcloud_images.yaml \
  --output-images-file=~/.local_registry_images.yaml
```

- OSP13

```
openstack overcloud container image prepare \
  --push-destination=192.168.24.1:8787 \
  --tag-from-label {version}-{release} \
  --output-images-file ~/.local_registry_images.yaml \
  --namespace=registry.access.Red Hat.com/rhosp13 \
  --prefix=openstack- \
  --tag-from-label {version}-{release} \
  --output-env-file ~/.overcloud_images.yaml
```

2. Upload OpenStack Containers

```
openstack overcloud container image upload --config-file ~/.local_registry_images.yaml
```

- Contrail Containers

1. Create Contrail container file



NOTE: This step is optional. The Contrail containers can be downloaded from external registries later.

```
cd ~/tripleo-heat-templates/tools/contrail
./import_contrail_container.sh -f container_outputfile -r registry -t tag [-i
insecure] [-u username] [-p password] [-c certificate pat
```

Here are few examples of importing Contrail containers from different sources:

- Import from password protected public registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r hub.juniper.net/
contrail -u USERNAME -p PASSWORD -t 1234
```

- Import from Dockerhub:

```
./import_contrail_container.sh -f /tmp/contrail_container -r docker.io/
opencontrailnightly -t 1234
```

- Import from private secure registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r
device.example.net:5443 -c http://device.example.net/pub/device.example.net.crt -t
1234
```

- Import from private insecure registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r 10.0.0.1:5443 -i 1 -t
1234
```

2. Upload Contrail containers to undercloud registry

```
openstack overcloud container image upload --config-file /tmp/contrail_container
```

Templates

Different YAML templates can be used to customize the overcloud

- Contrail Services customization

```
vi ~/tripleo-heat-templates/environments/contrail-services.yaml
parameter_defaults:
  ContrailSettings:
    VROUTER_GATEWAY: 10.0.0.1
    # KEY1: value1
    # KEY2: value2
```

- Contrail registry settings

```
vi ~/tripleo-heat-templates/environments/contrail-services.yaml
```

Here are few examples of default values for various registries:

- Public Juniper registry

```
parameter_defaults:
  ContrailRegistry: hub.juniper.net/contrail
  ContrailRegistryUser: <USER>
  ContrailRegistryPassword: <PASSWORD>
```

- Insecure registry

```
parameter_defaults:
  ContrailRegistryInsecure: true
  DockerInsecureRegistryAddress: 10.87.64.32:5000,192.168.24.1:8787
  ContrailRegistry: 10.87.64.32:5000
```


- Private secure registry

```
parameter_defaults:
  ContrailRegistryCertUrl: http://device.example.net/pub/device.example.net.crt
  ContrailRegistry: device.example.net:5443
```

- Contrail Container image settings

```
parameter_defaults:
  ContrailImageTag: queens-5.0-104-rhel-queens
```

- Network customization

In order to customize the network, define different networks and configure the overcloud nodes NIC layout. TripleO supports a flexible way of customizing the network.

The following networking customization example uses network as:

Table 7: Network Customization

Network	VLAN	overcloud Nodes
provisioning	-	All
internal_api	710	All
external_api	720	OpenStack CTRL
storage	740	OpenStack CTRL, Computes
storage_mgmt	750	OpenStack CTRL
tenant	-	Contrail CTRL, Computes

- Network activation in roles_data

The networks must be activated per role in the roles_data file:

```
vi ~/tripleo-heat-templates/roles_data_contrail_aio.yaml
```

- OpenStack Controller

```
#####
# Role: Controller                                     #
#####
- name: Controller
  description: |
    Controller role that has all the controller services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  tags:
    - primary
    - controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
```

- Compute Node

```
#####
# Role: Compute                                       #
#####
- name: Compute
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi
    - Tenant
    - Storage
```

- Contrail Controller

```
#####
# Role: ContrailController                                     #
#####
- name: ContrailController
  description: |
    ContrailController role that has all the Contrail controller services loaded
    and handles config, control and webui functions
  CountDefault: 1
  tags:
    - primary
    - contrailcontroller
  networks:
    - InternalApi
    - Tenant
```

- Compute DPDK

```
#####
# Role: ContrailDpdk                                         #
#####
- name: ContrailDpdk
  description: |
    Contrail Dpdk Node role
  CountDefault: 0
  tags:
    - contraildpdk
  networks:
    - InternalApi
    - Tenant
    - Storage
```

- Compute SRIOV

```
#####
# Role: ContrailSriov
#####
- name: ContrailSriov
  description: |
```

```

    Contrail Sriov Node role
CountDefault: 0
tags:
  - contrailsriov
networks:
  - InternalApi
  - Tenant
  - Storage

```

- Compute CSN

```

#####
# Role: ContrailTsn
#####
- name: ContrailTsn
  description: |
    Contrail Tsn Node role
  CountDefault: 0
  tags:
    - contrailtsn
  networks:
    - InternalApi
    - Tenant
    - Storage

```

- Network parameter configuration

```

cat ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml
resource_registry:
  OS::TripleO::Controller::Net::SoftwareConfig: ../../network/config/contrail/controller-
nic-config.yaml
  OS::TripleO::ContrailController::Net::SoftwareConfig: ../../network/config/contrail/
contrail-controller-nic-config.yaml
  OS::TripleO::ContrailControlOnly::Net::SoftwareConfig: ../../network/config/contrail/
contrail-controller-nic-config.yaml
  OS::TripleO::Compute::Net::SoftwareConfig: ../../network/config/contrail/compute-nic-
config.yaml
  OS::TripleO::ContrailDpdk::Net::SoftwareConfig: ../../network/config/contrail/contrail-
dpdk-nic-config.yaml
  OS::TripleO::ContrailSriov::Net::SoftwareConfig: ../../network/config/contrail/contrail-
sriov-nic-config.yaml

```

```
OS::TripleO::ContrailTsn::Net::SoftwareConfig: ../../network/config/contrail/contrail-
tsn-nic-config.yaml
```

```
parameter_defaults:
  # Customize all these values to match the local environment
  TenantNetCidr: 10.0.0.0/24
  InternalApiNetCidr: 10.1.0.0/24
  ExternalNetCidr: 10.2.0.0/24
  StorageNetCidr: 10.3.0.0/24
  StorageMgmtNetCidr: 10.4.0.0/24
  # CIDR subnet mask length for provisioning network
  ControlPlaneSubnetCidr: '24'
  # Allocation pools
  TenantAllocationPools: [{'start': '10.0.0.10', 'end': '10.0.0.200'}]
  InternalApiAllocationPools: [{'start': '10.1.0.10', 'end': '10.1.0.200'}]
  ExternalAllocationPools: [{'start': '10.2.0.10', 'end': '10.2.0.200'}]
  StorageAllocationPools: [{'start': '10.3.0.10', 'end': '10.3.0.200'}]
  StorageMgmtAllocationPools: [{'start': '10.4.0.10', 'end': '10.4.0.200'}]
  # Routes
  ControlPlaneDefaultRoute: 192.168.24.1
  InternalApiDefaultRoute: 10.1.0.1
  ExternalInterfaceDefaultRoute: 10.2.0.1
  # Vlans
  InternalApiNetworkVlanID: 710
  ExternalNetworkVlanID: 720
  StorageNetworkVlanID: 730
  StorageMgmtNetworkVlanID: 740
  TenantNetworkVlanID: 3211
  # Services
  EC2MetadataIp: 192.168.24.1 # Generally the IP of the undercloud
  DnsServers: ["172.x.x.x"]
  NtpServer: 10.0.0.1
```

- Network interface configuration

There are NIC configuration files per role.

```
cd ~/tripleo-heat-templates/network/config/contrail
```

- OpenStack Controller

```

heat_template_version: queens

description: >
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role. This is an example for a Nova compute node using
  Contrail vrouter and the vhost0 interface.
parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal_api network
    type: string
  InternalApiDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the internal api network.
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage_mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-
management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string

```

```

ExternalNetworkVlanID:
  default: 10
  description: Vlan ID for the external network traffic.
  type: number
InternalApiNetworkVlanID:
  default: 20
  description: Vlan ID for the internal_api network traffic.
  type: number
StorageNetworkVlanID:
  default: 30
  description: Vlan ID for the storage network traffic.
  type: number
StorageMgmtNetworkVlanID:
  default: 40
  description: Vlan ID for the storage mgmt network traffic.
  type: number
TenantNetworkVlanID:
  default: 50
  description: Vlan ID for the tenant network traffic.
  type: number
ManagementNetworkVlanID:
  default: 60
  description: Vlan ID for the management network traffic.
  type: number
ControlPlaneSubnetCidr: # Override this via parameter_defaults
  default: '24'
  description: The subnet CIDR of the control plane network.
  type: string
ControlPlaneDefaultRoute: # Override this via parameter_defaults
  description: The default route of the control plane network.
  type: string
ExternalInterfaceDefaultRoute: # Not used by default in this template
  default: '10.0.0.1'
  description: The default route of the external network.
  type: string
ManagementInterfaceDefaultRoute: # Commented out by default in this template
  default: unset
  description: The default route of the management network.
  type: string
DnsServers: # Override this via parameter_defaults
  default: []
  description: A list of DNS servers (2 max for some implementations) that will be
added to resolv.conf.

```

```

    type: comma_delimited_list
EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
                - type: interface
                  name: nic1
                  use_dhcp: false
                  dns_servers:
                    get_param: DnsServers
                  addresses:
                - ip_netmask:
                    list_join:
                      - '/'
                      - - get_param: ControlPlaneIp
                        - get_param: ControlPlaneSubnetCidr
                routes:
                - ip_netmask: 169.x.x.x/32
                  next_hop:
                    get_param: EC2MetadataIp
                - default: true
                  next_hop:
                    get_param: ControlPlaneDefaultRoute
            - type: vlan
              vlan_id:
                get_param: InternalApiNetworkVlanID
              device: nic1
              addresses:
                - ip_netmask:
                    get_param: InternalApiIpSubnet
            - type: vlan

```



```

        vlan_id:
            get_param: ExternalNetworkVlanID
        device: nic1
        addresses:
            - ip_netmask:
                get_param: ExternalIpSubnet
            - type: vlan
              vlan_id:
                  get_param: StorageNetworkVlanID
              device: nic1
              addresses:
                  - ip_netmask:
                      get_param: StorageIpSubnet
            - type: vlan
              vlan_id:
                  get_param: StorageMgmtNetworkVlanID
              device: nic1
              addresses:
                  - ip_netmask:
                      get_param: StorageMgmtIpSubnet
    outputs:
        OS::stack_id:
            description: The OsNetConfigImpl resource.
            value:
                get_resource: OsNetConfigImpl

```

- Contrail Controller

```

heat_template_version: queens
description: >
    Software Config to drive os-net-config to configure multiple interfaces
    for the compute role. This is an example for a Nova compute node using
    Contrail vrouter and the vhost0 interface.

parameters:
    ControlPlaneIp:
        default: ''
        description: IP address/subnet on the ctlplane network
        type: string
    ExternalIpSubnet:
        default: ''
        description: IP address/subnet on the external network

```

```

    type: string
InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal_api network
    type: string
InternalApiDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the internal api network.
    type: string
StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage_mgmt network
    type: string
TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
ManagementIpSubnet: # Only populated when including environments/network-
management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
TenantNetworkVlanID:

```

```

    default: 50
    description: Vlan ID for the tenant network traffic.
    type: number
ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
ExternalInterfaceDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the external network.
    type: string
ManagementInterfaceDefaultRoute: # Commented out by default in this template
    default: unset
    description: The default route of the management network.
    type: string
DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be
added to resolv.conf.
    type: comma_delimited_list
EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
                - type: interface

```

```

        name: nic1
        use_dhcp: false
        dns_servers:
            get_param: DnsServers
        addresses:
        - ip_netmask:
            list_join:
            - '/'
            - - get_param: ControlPlaneIp
              - get_param: ControlPlaneSubnetCidr
        routes:
        - ip_netmask: 169.x.x.x/32
          next_hop:
            get_param: EC2MetadataIp
        - default: true
          next_hop:
            get_param: ControlPlaneDefaultRoute
    - type: vlan
      vlan_id:
        get_param: InternalApiNetworkVlanID
      device: nic1
      addresses:
      - ip_netmask:
          get_param: InternalApiIpSubnet
    - type: interface
      name: nic2
      use_dhcp: false
      addresses:
      - ip_netmask:
          get_param: TenantIpSubnet

outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl

```

- Compute Node

```

heat_template_version: queens
description: >
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role. This is an example for a Nova compute node using

```

```

    Contrail vrouter and the vhost0 interface.
parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal_api network
    type: string
  InternalApiDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the internal api network.
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage_mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-
management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number

```

```

StorageNetworkVlanID:
  default: 30
  description: Vlan ID for the storage network traffic.
  type: number
StorageMgmtNetworkVlanID:
  default: 40
  description: Vlan ID for the storage mgmt network traffic.
  type: number
TenantNetworkVlanID:
  default: 50
  description: Vlan ID for the tenant network traffic.
  type: number
ManagementNetworkVlanID:
  default: 60
  description: Vlan ID for the management network traffic.
  type: number
ControlPlaneSubnetCidr: # Override this via parameter_defaults
  default: '24'
  description: The subnet CIDR of the control plane network.
  type: string
ControlPlaneDefaultRoute: # Override this via parameter_defaults
  description: The default route of the control plane network.
  type: string
ExternalInterfaceDefaultRoute: # Not used by default in this template
  default: '10.0.0.1'
  description: The default route of the external network.
  type: string
ManagementInterfaceDefaultRoute: # Commented out by default in this template
  default: unset
  description: The default route of the management network.
  type: string
DnsServers: # Override this via parameter_defaults
  default: []
  description: A list of DNS servers (2 max for some implementations) that will be
added to resolv.conf.
  type: comma_delimited_list
EC2MetadataIp: # Override this via parameter_defaults
  description: The IP address of the EC2 metadata server.
  type: string
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:

```

```

group: script
config:
  str_replace:
    template:
      get_file: ../../scripts/run-os-net-config.sh
  params:
    $network_config:
      network_config:
        - type: interface
          name: nic1
          use_dhcp: false
          dns_servers:
            get_param: DnsServers
          addresses:
            - ip_netmask:
                list_join:
                  - '/'
                  - - get_param: ControlPlaneIp
                    - get_param: ControlPlaneSubnetCidr
            routes:
              - ip_netmask: 169.x.x.x/32
                next_hop:
                  get_param: EC2MetadataIp
              - default: true
                next_hop:
                  get_param: ControlPlaneDefaultRoute
        - type: vlan
          vlan_id:
            get_param: InternalApiNetworkVlanID
          device: nic1
          addresses:
            - ip_netmask:
                get_param: InternalApiIpSubnet
        - type: vlan
          vlan_id:
            get_param: StorageNetworkVlanID
          device: nic1
          addresses:
            - ip_netmask:
                get_param: StorageIpSubnet
        - type: contrail_vrouter
          name: vhost0
          use_dhcp: false

```

```

        members:
          -
            type: interface
            name: nic2
            use_dhcp: false
        addresses:
          - ip_netmask:
              get_param: TenantIpSubnet

    outputs:
      OS::stack_id:
        description: The OsNetConfigImpl resource.
        value:
          get_resource: OsNetConfigImpl

```

- Advanced Network Configuration
- Advanced vRouter Kernel Mode Configurations

In addition to the standard NIC configuration, the vRouter kernel mode supports the following modes:

- VLAN
- Bond
- Bond + VLAN

NIC Template Configurations

The snippets below only shows the relevant section of the NIC configuration for each mode.

- VLAN

```

- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: nic2
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:

```



```

      str_replace:
        template: vlanVLANID
        params:
          VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
    addresses:
      - ip_netmask:
          get_param: TenantIpSubnet

```

- Bond

```

- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
    -
      type: interface
      name: nic2
    -
      type: interface
      name: nic3
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name: bond0
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet

```

- Bond + VLAN

```

- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:

```

```

-
  type: interface
  name: nic2
-
  type: interface
  name: nic3
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: bond0
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
        str_replace:
          template: vlanVLANID
        params:
          VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet

```

- Advanced vRouter DPDK Mode Configurations

In addition to the standard NIC configuration, the vRouter DPDK mode supports the following modes:

- Standard
- VLAN
- Bond
- Bond + VLAN

Network Environment Configuration

Enable the number of hugepages:

```
parameter_defaults:
  ContrailDpdkHugepages1GB: 10
```

NIC Template Configurations

- Standard

```
- type: contrail_vrouter_dpkg
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
```

- VLAN

```
- type: contrail_vrouter_dpkg
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  vlan_id:
    get_param: TenantNetworkVlanID
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
```

- Bond

```
- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  bond_mode: 4
  bond_policy: layer2+3
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
    -
      type: interface
      name: nic3
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
```

- Bond + VLAN

```
- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  vlan_id:
    get_param: TenantNetworkVlanID
  bond_mode: 4
  bond_policy: layer2+3
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
    -
      type: interface
      name: nic3
```

```

        use_dhcp: false
    addresses:
    - ip_netmask:
        get_param: TenantIpSubnet

```

- Advanced vRouter SRIOV Mode Configurations

vRouter SRIOV can be used in the following combinations:

- SRIOV + Kernel mode
 - Standard
 - VLAN
 - Bond
 - Bond + VLAN
- SRIOV + DPDK mode
 - Standard
 - VLAN
 - Bond
 - Bond + VLAN

Network environment configuration

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages

- SRIOV + Kernel mode

```

parameter_defaults:
    ContrailSriovHugepages1GB: 10

```

- SRIOV + DPDK mode

```

parameter_defaults:
    ContrailSriovMode: dpdk

```

```
ContrailDpdkHugepages1GB: 10
ContrailSriovHugepages1GB: 10
```

SRIOV PF/VF settings

```
NovaPCIPassthrough:
- devname: "ens2f1"
  physical_network: "sriov1"
ContrailSriovNumVFs: ["ens2f1:7"]
```

NIC template configurations:

The SRIOV NICs are not configured in the NIC templates. However, vRouter NICs must still be configured.

See following NIC Template Configurations for vRouter kernel mode.

The snippets below only shows the relevant section of the NIC configuration for each mode.

- VLAN

```
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: nic2
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
        str_replace:
          template: vlanVLANID
        params:
          VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
```

- Bond

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
    -
      type: interface
      name: nic2
    -
      type: interface
      name: nic3
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name: bond0
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
```

- Bond + VLAN

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
    -
      type: interface
      name: nic2
    -
      type: interface
      name: nic3
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
```

```

    device: bond0
  - type: contrail_vrouter
    name: vhost0
    use_dhcp: false
    members:
      -
        type: interface
        name:
          str_replace:
            template: vlanVLANID
          params:
            VLANID: {get_param: TenantNetworkVlanID}
        use_dhcp: false
    addresses:
      - ip_netmask:
          get_param: TenantIpSubnet

```

See following NIC Template Configurations for vRouter DPDK mode:

- Standard

```

  - type: contrail_vrouter_dpdk
    name: vhost0
    use_dhcp: false
    driver: uio_pci_generic
    cpu_list: 0x01
    members:
      -
        type: interface
        name: nic2
        use_dhcp: false
    addresses:
      - ip_netmask:
          get_param: TenantIpSubnet

```

- VLAN

```

  - type: contrail_vrouter_dpdk
    name: vhost0
    use_dhcp: false
    driver: uio_pci_generic

```



```

cpu_list: 0x01
vlan_id:
  get_param: TenantNetworkVlanID
members:
  -
    type: interface
    name: nic2
    use_dhcp: false
addresses:
  - ip_netmask:
      get_param: TenantIpSubnet

```

- Bond

```

- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  bond_mode: 4
  bond_policy: layer2+3
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
    -
      type: interface
      name: nic3
      use_dhcp: false
  addresses:
    - ip_netmask:
        get_param: TenantIpSubnet

```

- Bond + VLAN

```

- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01

```

```

vlan_id:
  get_param: TenantNetworkVlanID
bond_mode: 4
bond_policy: layer2+3
members:
  -
    type: interface
    name: nic2
    use_dhcp: false
  -
    type: interface
    name: nic3
    use_dhcp: false
addresses:
  - ip_netmask:
      get_param: TenantIpSubnet

```

- Advanced Scenarios

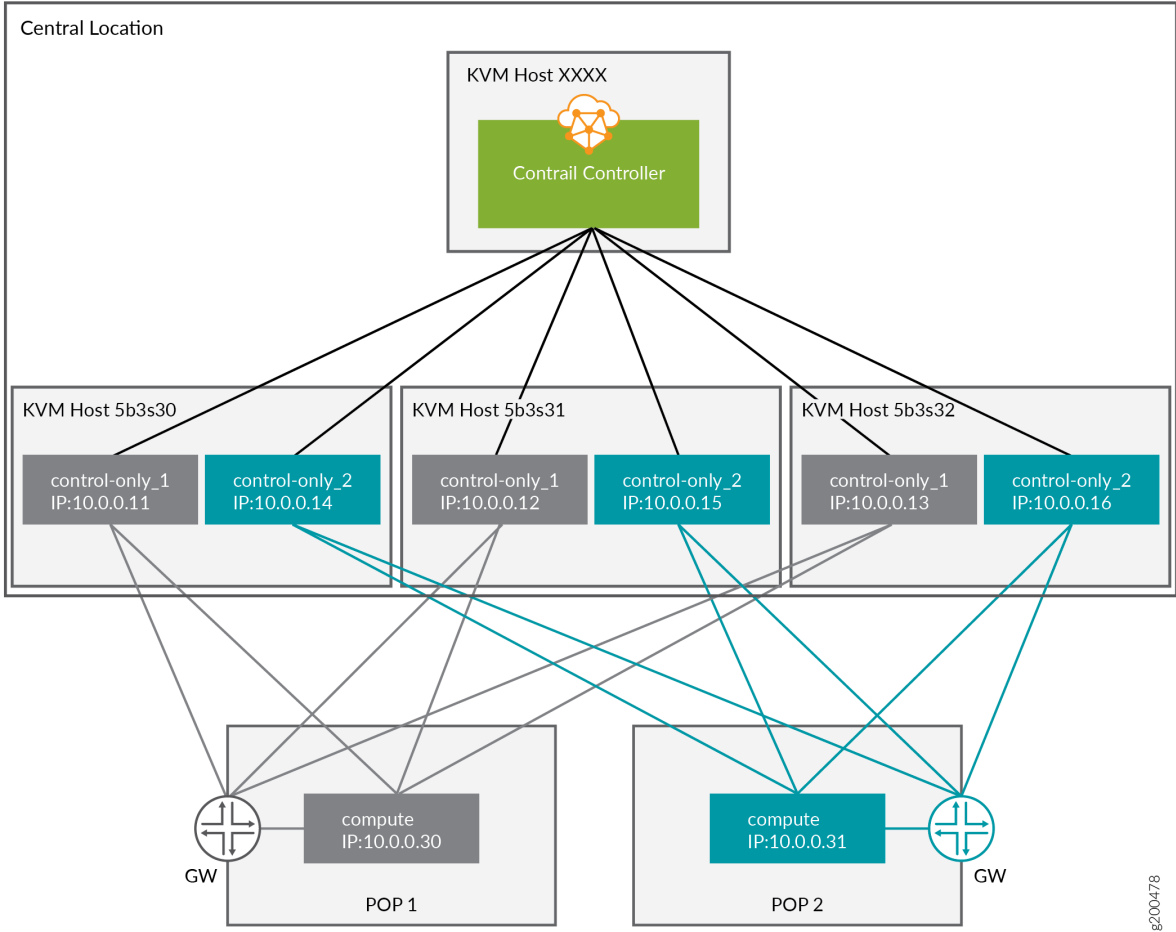
Remote Compute

Remote Compute extends the data plane to remote locations (POP) whilst keeping the control plane central. Each POP will have its own set of Contrail control services, which are running in the central location. The difficulty is to ensure that the compute nodes of a given POP connect to the Control nodes assigned to that POP. The Control nodes must have predictable IP addresses and the compute nodes have to know these IP addresses. In order to achieve that the following methods are used:

- Custom Roles
- Static IP assignment
- Precise Node placement
- Per Node hieradata

Each overcloud node has a unique DMI UUID. This UUID is known on the undercloud node as well as on the overcloud node. Hence, this UUID can be used for mapping node specific information. For each POP, a Control role and a Compute role has to be created.

Overview



g200478

Mapping Table

Table 8: Mapping Table

Nova Name	Ironic Name	UUID	KVM	IP Address	POP
overcloud - contrailcontrolonly-0	control-only-1-5b3s30	<p>Ironic UUID: 7d758dce-2784-45fd-be09-5a41eb53e764</p> <p>DMI UUID: 73F8D030-E896-4A95-A9F5-E1A4FEBE322D</p>	5b3s30	10.0.0.11	POP1

Table 8: Mapping Table (Continued)

Nova Name	Ironic Name	UUID	KVM	IP Address	POP
overcloud - contrailcontrolonly -1	control-only-2- 5b3s30	Ironic UUID: d26abdeb- d514- 4a37-a7fb-2cd2511c351f DMI UUID: 14639A66- D62C- 4408-82EE- FDDC4E509687	5b3s30	10.0.0.14	POP2
overcloud - contrailcontrolonly -2	control-only-1- 5b3s31	Ironic UUID: 91dd9fa9- e8eb- 4b51-8b5e-bbaffb6640e4 DMI UUID: 28AB0B57- D612- 431E- B177-1C578AE0FEA4	5b3s31	10.0.0.12	POP1
overcloud - contrailcontrolonly -3	control-only-2- 5b3s31	Ironic UUID: 09fa57b8-580f- 42ec-bf10-a19573521ed4 DMI UUID: 09BEC8CB-77E9- 42A6- AFF4-6D4880FD87D0	5b3s31	10.0.0.15	POP2
overcloud - contrailcontrolonly -4	control-only-1- 5b3s32	Ironic UUID: 4766799-24c8- 4e3b-af54-353f2b796ca4 DMI UUID: 3993957A- ECBF- 4520-9F49-0AF6EE1667A7	5b3s32	10.0.0.13	POP1

Table 8: Mapping Table *(Continued)*

Nova Name	Ironic Name	UUID	KVM	IP Address	POP
overcloud - contrailcontrolonly -5	control-only-2- 5b3s32	Ironic UUID: 58a803ae- a785- 470e-9789-139abbfa74fb DMI UUID: AF92F485- C30C- 4D0A-BDC4- C6AE97D06A66	5b3s32	10.0.0.16	POP2

ControlOnly preparation

Add ControlOnly overcloud VMs to overcloud KVM host



NOTE: This has to be done on the overcloud KVM hosts

Two ControlOnly overcloud VM definitions will be created on each of the overcloud KVM hosts.

```

ROLES=control-only:2
num=4
ipmi_user=<user>
ipmi_password=<password>
libvirt_path=/var/lib/libvirt/images
port_group=overcloud
prov_switch=br0

/bin/rm ironic_list
IFS=' ' read -ra role_list <<< "${ROLES}"
for role in ${role_list[@]}; do
    role_name=`echo $role|cut -d ":" -f 1`
    role_count=`echo $role|cut -d ":" -f 2`
    for count in `seq 1 ${role_count}`; do
        echo $role_name $count
        qemu-img create -f qcow2 ${libvirt_path}/${role_name}_${count}.qcow2 99G
        virsh define /dev/stdin <<EOF
$(virt-install --name ${role_name}_${count} \
--disk ${libvirt_path}/${role_name}_${count}.qcow2 \
--vcpus=4 \

```

```

--ram=16348 \
--network network=br0,model=virtio,portgroup=${port_group} \
--network network=br1,model=virtio \
--virt-type kvm \
--cpu host \
--import \
--os-variant rhel7 \
--serial pty \
--console pty,target_type=virtio \
--graphics vnc \
--print-xml)
EOF
    vbmc add ${role_name}_${count} --port 1623${num} --username ${ipmi_user} --password $
{ipmi_password}
    vbmc start ${role_name}_${count}
    prov_mac=`virsh domiflist ${role_name}_${count}|grep ${prov_switch}|awk '{print $5}'`
    vm_name=${role_name}-${count}-`hostname -s`
    kvm_ip=`ip route get 1 |grep src |awk '{print $7}'`
    echo ${prov_mac} ${vm_name} ${kvm_ip} ${role_name} 1623${num}>> ironic_list
    num=$((expr $num + 1))
done
done

```



NOTE: The generated *ironic_list* will be needed on the undercloud to import the nodes to Ironic.

Get the *ironic_lists* from the overcloud KVM hosts and combine them.

```

cat ironic_list_control_only
52:54:00:3a:2f:ca control-only-1-5b3s30 10.87.64.31 control-only 16234
52:54:00:31:4f:63 control-only-2-5b3s30 10.87.64.31 control-only 16235
52:54:00:0c:11:74 control-only-1-5b3s31 10.87.64.32 control-only 16234
52:54:00:56:ab:55 control-only-2-5b3s31 10.87.64.32 control-only 16235
52:54:00:c1:f0:9a control-only-1-5b3s32 10.87.64.33 control-only 16234
52:54:00:f3:ce:13 control-only-2-5b3s32 10.87.64.33 control-only 16235

```

Import:

```

ipmi_password=<password>
ipmi_user=<user>

```

```

DEPLOY_KERNEL=$(openstack image show bm-deploy-kernel -f value -c id)
DEPLOY_RAMDISK=$(openstack image show bm-deploy-ramdisk -f value -c id)

num=0
while IFS= read -r line; do
    mac=`echo $line|awk '{print $1}'`
    name=`echo $line|awk '{print $2}'`
    kvm_ip=`echo $line|awk '{print $3}'`
    profile=`echo $line|awk '{print $4}'`
    ipmi_port=`echo $line|awk '{print $5}'`
    uuid=`openstack baremetal node create --driver ipmi \
        --property cpus=4 \
        --property memory_mb=16348 \
        --property local_gb=100 \
        --property cpu_arch=x86_64 \
        --driver-info ipmi_username=${ipmi_user} \
        --driver-info ipmi_address=${kvm_ip} \
        --driver-info ipmi_password=${ipmi_password} \
        --driver-info ipmi_port=${ipmi_port} \
        --name=${name} \
        --property capabilities=boot_option:local \
        -c uuid -f value`

    openstack baremetal node set ${uuid} --driver-info deploy_kernel=$DEPLOY_KERNEL --driver-
info deploy_ramdisk=$DEPLOY_RAMDISK
    openstack baremetal port create --node ${uuid} ${mac}
    openstack baremetal node manage ${uuid}
    num=$((expr $num + 1))
done < <(cat ironic_list_control_only)

```

ControlOnly node introspection

```
openstack overcloud node introspect --all-manageable --provide
```

Get the ironic UUID of the ControlOnly nodes

```

openstack baremetal node list |grep control-only
| 7d758dce-2784-45fd-be09-5a41eb53e764 | control-only-1-5b3s30 | None | power off |
available | False |
| d26abdeb-d514-4a37-a7fb-2cd2511c351f | control-only-2-5b3s30 | None | power off |
available | False |

```

```
| 91dd9fa9-e8eb-4b51-8b5e-bbaffb6640e4 | control-only-1-5b3s31 | None | power off |
available | False |
| 09fa57b8-580f-42ec-bf10-a19573521ed4 | control-only-2-5b3s31 | None | power off |
available | False |
| f4766799-24c8-4e3b-af54-353f2b796ca4 | control-only-1-5b3s32 | None | power off |
available | False |
| 58a803ae-a785-470e-9789-139abbfa74fb | control-only-2-5b3s32 | None | power off |
available | False |
```

The first ControlOnly node on each of the overcloud KVM hosts will be used for POP1, the second for POP2, and so and so forth.

Get the ironic UUID of the POP compute nodes:

```
openstack baremetal node list |grep compute
| 91d6026c-b9db-49cb-a685-99a63da5d81e | compute-3-5b3s30 | None | power off | available |
False |
| 8028eb8c-e1e6-4357-8fcf-0796778bd2f7 | compute-4-5b3s30 | None | power off | available |
False |
| b795b3b9-c4e3-4a76-90af-258d9336d9fb | compute-3-5b3s31 | None | power off | available |
False |
| 2d4be83e-6fcc-4761-86f2-c2615dd15074 | compute-4-5b3s31 | None | power off | available |
False |
```

The first two compute nodes belong to POP1 the second two compute nodes belong to POP2.

Create an input YAML using the ironic UUIDs:

```
~/subcluster_input.yaml
---
- subcluster: subcluster1
  asn: "65413"
  control_nodes:
    - uuid: 7d758dce-2784-45fd-be09-5a41eb53e764
      ipaddress: 10.0.0.11
    - uuid: 91dd9fa9-e8eb-4b51-8b5e-bbaffb6640e4
      ipaddress: 10.0.0.12
    - uuid: f4766799-24c8-4e3b-af54-353f2b796ca4
      ipaddress: 10.0.0.13
  compute_nodes:
    - uuid: 91d6026c-b9db-49cb-a685-99a63da5d81e
      vrouter_gateway: 10.0.0.1
```



```

- uuid: 8028eb8c-e1e6-4357-8fcf-0796778bd2f7
  vrouter_gateway: 10.0.0.1
- subcluster: subcluster2
  asn: "65414"
  control_nodes:
    - uuid: d26abdeb-d514-4a37-a7fb-2cd2511c351f
      ipaddress: 10.0.0.14
    - uuid: 09fa57b8-580f-42ec-bf10-a19573521ed4
      ipaddress: 10.0.0.15
    - uuid: 58a803ae-a785-470e-9789-139abbfa74fb
      ipaddress: 10.0.0.16
  compute_nodes:
    - uuid: b795b3b9-c4e3-4a76-90af-258d9336d9fb
      vrouter_gateway: 10.0.0.1
    - uuid: 2d4be83e-6fcc-4761-86f2-c2615dd15074
      vrouter_gateway: 10.0.0.1

```



NOTE: Only control_nodes, compute_nodes, dpdk_nodes and sriov_nodes are supported.

Generate subcluster environment:

```

~/tripleo-heat-templates/tools/contrail/create_subcluster_environment.py -i ~/
subcluster_input.yaml \
-o ~/tripleo-heat-templates/environments/contrail/contrail-subcluster.yaml

```

Check subcluster environment file:

```

cat ~/tripleo-heat-templates/environments/contrail/contrail-subcluster.yaml
parameter_defaults:
  NodeDataLookup:
    041D7B75-6581-41B3-886E-C06847B9C87E:
      contrail_settings:
        CONTROL_NODES: 10.0.0.14,10.0.0.15,10.0.0.16
        SUBCLUSTER: subcluster2
        VROUTER_GATEWAY: 10.0.0.1
    09BEC8CB-77E9-42A6-AFF4-6D4880FD87D0:
      contrail_settings:
        BGP_ASN: '65414'
        SUBCLUSTER: subcluster2

```

```

14639A66-D62C-4408-82EE-FDDC4E509687:
  contrail_settings:
    BGP_ASN: '65414'
    SUBCLUSTER: subcluster2
28AB0B57-D612-431E-B177-1C578AE0FEA4:
  contrail_settings:
    BGP_ASN: '65413'
    SUBCLUSTER: subcluster1
3993957A-ECBF-4520-9F49-0AF6EE1667A7:
  contrail_settings:
    BGP_ASN: '65413'
    SUBCLUSTER: subcluster1
73F8D030-E896-4A95-A9F5-E1A4FEBE322D:
  contrail_settings:
    BGP_ASN: '65413'
    SUBCLUSTER: subcluster1
7933C2D8-E61E-4752-854E-B7B18A424971:
  contrail_settings:
    CONTROL_NODES: 10.0.0.14,10.0.0.15,10.0.0.16
    SUBCLUSTER: subcluster2
    VROUTER_GATEWAY: 10.0.0.1
AF92F485-C30C-4D0A-BDC4-C6AE97D06A66:
  contrail_settings:
    BGP_ASN: '65414'
    SUBCLUSTER: subcluster2
BB9E9D00-57D1-410B-8B19-17A0DA581044:
  contrail_settings:
    CONTROL_NODES: 10.0.0.11,10.0.0.12,10.0.0.13
    SUBCLUSTER: subcluster1
    VROUTER_GATEWAY: 10.0.0.1
E1A809DE-FDB2-4EB2-A91F-1B3F75B99510:
  contrail_settings:
    CONTROL_NODES: 10.0.0.11,10.0.0.12,10.0.0.13
    SUBCLUSTER: subcluster1
    VROUTER_GATEWAY: 10.0.0.1

```

Deployment

Add contrail-subcluster.yaml, contrail-ips-from-pool-all.yaml and contrail-scheduler-hints.yaml to the OpenStack deploy command:

```
openstack overcloud deploy --templates ~/tripleo-heat-templates \
-e ~/overcloud_images.yaml \
-e ~/tripleo-heat-templates/environments/network-isolation.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-subcluster.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-ips-from-pool-all.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-scheduler-hints.yaml \
--roles-file ~/tripleo-heat-templates/roles_data_contrail_aio.yaml
```

overcloud Installation

Deployment:

```
openstack overcloud deploy --templates ~/tripleo-heat-templates \
-e ~/overcloud_images.yaml \
-e ~/tripleo-heat-templates/environments/network-isolation.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
--roles-file ~/tripleo-heat-templates/roles_data_contrail_aio.yaml
```

Validation Test:

```
source overcloudrc
curl -O http://download.cirros-cloud.net/0.3.5/cirros-0.3.5-x86_64-disk.img
openstack image create --container-format bare --disk-format qcow2 --file cirros-0.3.5-x86_64-disk.img cirros
openstack flavor create --public cirros --id auto --ram 64 --disk 0 --vcpus 1
openstack network create net1
openstack subnet create --subnet-range 1.0.0.0/24 --network net1 sn1
nova boot --image cirros --flavor cirros --nic net-id=openstack network show net1 -c id -f value --availability-zone nova:overcloud-novacompute-0.localdomain c1
nova list
```

RELATED DOCUMENTATION

[Provisioning Red Hat OpenShift Container Platform Clusters Using Ansible Deployer](#) | 193

Provisioning Red Hat OpenShift Container Platform Clusters Using Ansible Deployer

IN THIS SECTION

- [Installing a Standalone OpenShift Cluster Using Ansible Deployer](#) | 193
- [Provisioning of Nested OpenShift Clusters Using Ansible Deployer—Beta](#) | 203

Contrail Release 5.0.2 supports the following ways of installing and provisioning standalone and nested Red Hat OpenShift Container Platform version 3.9 clusters. These instructions are valid for systems with Microsoft Azure, Amazon Web Services (AWS), or bare metal server (BMS).

Installing a Standalone OpenShift Cluster Using Ansible Deployer

Prerequisites

Ensure the following system requirements.

- **Master Node** (x1 or x3 for high availability)
 - Image: RHEL 7.5
 - CPU/RAM: 4 CPU, 32 GB RAM
 - Disk: 250 GB
 - Security Group: Allow all traffic from everywhere
- **Slave Node** (x*n*)
 - Image: RHEL 7.5
 - CPU/RAM: 8 CPU, 64 GB RAM
 - Disk: 250 GB
 - Security Group: Allow all traffic from everywhere

- **Load Balancer Node** (x1, only when using high availability. Not needed for single master node installation.)
 - Image: RHEL 7.5
 - CPU/RAM: 2 CPU, 16 GB RAM
 - Disk: 100 GB
 - Security Group: Allow all traffic from everywhere



NOTE: Ensure that you launch the instances in the same subnet.

Installing a standalone OpenShift cluster using Ansible deployer

Perform the following steps to install a standalone OpenShift cluster with Contrail as networking provider and provision the cluster using contrail-ansible-deployer.

1. Re-image all the servers.

```
/server-manager reimage --server_id server1 redhat-7.5-minimal
```

2. Set up environment nodes:

- a. You must register all nodes in order to subscribe to OpenShift Container Platform. Register all nodes in the cluster using Red Hat Subscription Manager (RHSM).

```
(all-nodes)# subscription-manager register --username username --password password --force
```

- b. List the available subscriptions.

```
(all-nodes)# subscription-manager list --available --matches '*OpenShift*'
```

- c. From the list of available subscriptions, find and attach the pool ID for the OpenShift Container Platform subscription.

```
(all-nodes)# subscription-manager attach --pool=pool-ID
```

- d. Disable all yum repositories.

```
(all-nodes)# subscription-manager repos --disable="*"
```

- e. Enable only the required repositories.

```
(all-nodes)# subscription-manager repos \
  --enable="rhel-7-server-rpms" \
  --enable="rhel-7-server-extras-rpms" \
  --enable="rhel-7-server-ose-3.9-rpms" \
```

```
--enable="rhel-7-fast-datapath-rpms" \
--enable="rhel-7-server-ansible-2.5-rpms"
```

- f. Install Extra Packages for Enterprise Linux (EPEL).

```
(all-nodes)# yum install wget -y && wget -O /tmp/epel-release-latest-7.noarch.rpm https://
dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm && rpm -ivh /tmp/epel-release-
latest-7.noarch.rpm
```

```
(all-nodes)# yum install wget -y && wget -O /tmp/epel-release-latest-7.noarch.rpm
```

- g. Update the system to use the latest packages.

```
(all-nodes)# yum update -y
```

- h. Install the following package which provides OpenShift Container Platform utilities.

```
(all-nodes)# yum install atomic-openshift-excluder atomic-openshift-utils git python-netaddr -y
```

- i. Remove the atomic-openshift packages from the list for the duration of the installation.

```
(all-nodes)# atomic-openshift-excluder unexclude -y
```

- j. Enable SSH access for the root user.

```
(all-nodes)# sudo su
(all-nodes)# passwd
(all-nodes)# sed -i -e 's/#PermitRootLogin yes/PermitRootLogin yes/g' -e 's/
PasswordAuthentication no/PasswordAuthentication yes/g' /etc/ssh/sshd_config
(all-nodes)# service sshd restart
(all-nodes)# logout
```

- k. Log out.

```
(all-nodes)# logout
```

- l. Log in as root user.

```
ssh node-ip -l root
```

- m. Enforce the SELinux security policy.

```
(all-nodes)# vi /etc/selinux/config

SELINUX=enforcing
```

3. Install the supported Ansible version by running the following command:

```
yum install ansible
```

4. Get the files from the latest tar ball. Download the OpenShift Container Platform install package from Juniper software download site and modify the contents of the openshift-ansible inventory file.

- a. Download the Openshift Deployer (contrail-openshift-deployer-5.0.X.tar) installer tar ball from the Juniper software download site: <https://www.juniper.net/support/downloads/?p=contrail#sw>

- b. Copy the install package to the node from where Ansible must be deployed. Ensure that the node has password-free access to the OpenShift master and slave nodes.

```
scp contrail-openshift-deployer-5.0.X.tar openshift-ansible-node:/root/
```

- c. Untar the contrail-openshift-deployer-5.0.X.tar package.

```
tar -xvf contrail-openshift-deployer-5.0.X.tar -C /root/
```

- d. Verify the contents of the **openshift-ansible** directory.

```
cd /root/openshift-ansible/
```

- e. Modify the inventory file to match your OpenShift environment.

Populate the install file with Contrail configuration parameters specific to your system. Refer to the following example.

Add the master nodes in the [nodes] section of the inventory to ensure that the Contrail control pods will come up on the OpenShift master nodes.

```
(ansible-node)# vi /root/openshift-ansible/inventory/ose-install

[OSEv3:vars]
...
contrail_version=5.0
contrail_container_tag=5.0.X-0.X
contrail_registry=hub.juniper.net/contrail
contrail_registry_username=username-for-contrail-container-registry
contrail_registry_password=password-for-contrail-container-registry
...
```

For more information about each of these parameters and for an example for a HA master, see <https://github.com/Juniper/contrail-kubernetes-docs/blob/master/install/openshift/3.9/standalone-openshift.md>.



NOTE: Juniper Networks recommends that you obtain the Ansible source files from the latest release.

This procedure assumes that there is one master node, one infra node, and one compute node.

```
master : server1 (1x.xx.xx.11)

infra : server2 (1x.xx.xx.22)

compute : server3 (1x.xx.xx.33)
```

5. Edit **/etc/hosts** to allow all machines to access all nodes.

```
[root@server1]# cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.84.5.100 puppet
1x.xx.xx.11 server1.contrail.juniper.net server1
1x.xx.xx.22 server2.contrail.juniper.net server2
1x.xx.xx.33 server3.contrail.juniper.net server3
```

6. Set up password free SSH access to the Ansible node and all the nodes.

```
ssh-keygen -t rsa
ssh-copy-id root@1x.xx.xx.11
ssh-copy-id root@1x.xx.xx.22
ssh-copy-id root@1x.xx.xx.33
```

7. Run Ansible playbook to install OpenShift Container Platform with Contrail. Before you run Ansible playbook, ensure that you have edited **inventory/ose-install** file as shown below.

```
(ansible-node)# cd /root/openshift-ansible
(ansible-node)# ansible-playbook -i inventory/ose-install playbooks/prerequisites.yml
(ansible-node)# ansible-playbook -i inventory/ose-install playbooks/deploy_cluster.yml
```


8. Verify that Contrail has been installed and is operational.

```
(master)# oc get ds -n kube-system
(master)# oc get pods -n kube-system
```

9. Install the customized web console that should run on the infra nodes. To do this, disable the OpenShift Web console and enable the Contrail Web console and add the following lines in **ose-install**:

```
openshift_web_console_install=false
openshift_web_console_contrail_install=true
```

10. Create a password for the admin user to log in to the UI from the master node.

```
(master-node)# htpasswd /etc/origin/master/htpasswd admin
```



NOTE: If you are using a load balancer, you must manually copy the htpasswd file into all your master nodes.

11. Assign cluster-admin role to admin user.

```
(master-node)# oc adm policy add-cluster-role-to-user cluster-admin admin
(master-node)# oc login -u admin
```

12. Open a Web browser and type the entire fqdn name of your master node or load balancer node, followed by :8443/console.

```
https://<your host name from your ose-install inventory>:8443/console
```



NOTE: Use the user name and password created above to log into the Web console.



NOTE: Your DNS should resolve the host name for access. If the host name is not resolved, modify the /etc/hosts file to route to the above host.

13. Verify the provisioning process.

```
(master-node)# oc get pods -n kube-system
```

The status of all the pods must be displayed as Running.

```
(master-node)# contrail-status
```

All contrail-services must be displayed as active.

14. Access the Contrail and OpenShift Web user interfaces and attempt to log in to each.

Contrail: <https://master-node-ip:8143> with <admin/cOntrail123> login credentials.

OpenShift: <https://infra-node-ip:8443> with <admin/password created in step 10> login credentials.

You can test the system by launching pods, services, namespaces, network-policies, ingress, and soon. For more information, see the examples listed in <https://github.com/juniper/openshift-contrail/tree/master/openshift/examples>.

Sample ose-install File

Use the following sample **ose-install** file for reference.

```
[OSEv3:children]
masters
nodes
etcd
openshift_ca

[OSEv3:vars]
ansible_ssh_user=root
ansible_become=yes
debug_level=2
deployment_type=origin #openshift-enterprise for Redhat
openshift_release=v3.9
#openshift_repos_enable_testing=true
containerized=false
openshift_install_examples=true
openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge':
'true', 'kind': 'HTPasswdPasswordIdentityProvider', 'filename': '/etc/origin/master/htpasswd'}]
osm_cluster_network_cidr=10.32.0.0/12
openshift_portal_net=10.96.0.0/12
openshift_use_dnsmasq=true
openshift_clock_enabled=true
openshift_hosted_manage_registry=false
openshift_hosted_manage_router=false
openshift_enable_service_catalog=false
```

```

openshift_use_openshift_sdn=false
os_sdn_network_plugin_name='cni'
openshift_disable_check=memory_availability,package_availability,disk_availability,package_version,docker_storage
openshift_docker_insecure_registries=opencontrailnightly
openshift_web_console_install=false
#openshift_web_console_nodeselector={'region':'infra'}

openshift_web_console_contrail_install=true
openshift_use_contrail=true
nested_mode_contrail=false
contrail_version=5.0
contrail_container_tag=queens-5.0-156
contrail_registry=opencontrailnightly
# Username /Password for private Docker registries
#contrail_registry_username=test
#contrail_registry_password=test
# Below option presides over contrail masters if set
#vrouter_physical_interface=ens160
#docker_version=1.13.1
ntpserver=10.1.1.1 # a proper ntpserver is required for contrail.

# Contrail_vars
# below variables are used by contrail kubemanager to configure the cluster,
# you can configure all options below. All values are defaults and can be modified.

#kubernetes_api_server=10.84.13.52      # in our case this is the master, which is default
#kubernetes_api_port=8080
#kubernetes_api_secure_port=8443
#cluster_name=myk8s
#cluster_project={}
#cluster_network={}
#pod_subnets=10.32.0.0/12
#ip_fabric_subnets=10.64.0.0/12
#service_subnets=10.96.0.0/12
#ip_fabric_forwarding=false
#ip_fabric_snat=false
#public_fip_pool={}
#vnc_endpoint_ip=20.1.1.1
#vnc_endpoint_port=8082

[masters]
10.84.13.52 openshift_hostname=openshift-master

```

```
[etcd]
10.84.13.52 openshift_hostname=openshift-master

[nodes]
10.84.13.52 openshift_hostname=openshift-master
10.84.13.53 openshift_hostname=openshift-compute
10.84.13.54 openshift_hostname=openshift-infra openshift_node_labels="{ 'region': 'infra' }"

[openshift_ca]
10.84.13.52 openshift_hostname=openshift-master
```

Sample ose-install File for a HA setup

Use the following sample **ose-install** file for reference.

```
[OSEv3:children]
masters
nodes
etcd
lb
openshift_ca

[OSEv3:vars]
ansible_ssh_user=root
ansible_become=yes
debug_level=2
deployment_type=openshift-enterprise
openshift_release=v3.9
openshift_repos_enable_testing=true
containerized=false
openshift_install_examples=true
openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge':
'true', 'kind': 'HTPasswdPasswordIdentityProvider', 'filename': '/etc/origin/master/htpasswd'}]
osm_cluster_network_cidr=10.32.0.0/12
openshift_portal_net=10.96.0.0/12
openshift_use_dnsmasq=true
openshift_clock_enabled=true
openshift_enable_service_catalog=false
openshift_use_openshift_sdn=false
os_sdn_network_plugin_name='cni'
openshift_disable_check=disk_availability,package_version,docker_storage
```

```

openshift_docker_insecure_registries=ci-repo.englab.juniper.net:5010
openshift_web_console_install=false
openshift_web_console_contrail_install=true
openshift_web_console_nodeselector={'region':'infra'}
openshift_hosted_manage_registry=true
openshift_hosted_registry_selector="region=infra"
openshift_hosted_manage_router=true
openshift_hosted_router_selector="region=infra"
ntpserver=10.84.5.100

# Openshift HA
openshift_master_cluster_method=native
openshift_master_cluster_hostname=lb
openshift_master_cluster_public_hostname=lb

# Below are Contrail variables. Comment them out if you don't want to install Contrail through
# ansible-playbook
contrail_version=5.0
openshift_use_contrail=true
#rhel-queens-5.0-latest
#contrail_container_tag=rhel-queens-5.0-319
#contrail_registry=ci-repo.englab.juniper.net:5010
contrail_registry=hub.juniper.net/contrail
contrail_registry_username=JNPR-Customer200
contrail_registry_password=F*****f
contrail_container_tag=5.0.2-0.309-rhel-queens
contrail_nodes=[10.0.0.7, 10.0.0.8, 10.0.0.13]
vrouters_physical_interface=eth0

[masters]
10.0.0.7 openshift_hostname=master1
10.0.0.8 openshift_hostname=master2
10.0.0.13 openshift_hostname=master3

[lb]
10.0.0.5 openshift_hostname=lb

[etcd]
10.0.0.7 openshift_hostname=master1
10.0.0.8 openshift_hostname=master2
10.0.0.13 openshift_hostname=master3

```

```
[nodes]
10.0.0.7 openshift_hostname=master1
10.0.0.8 openshift_hostname=master2
10.0.0.13 openshift_hostname=master3
10.0.0.10 openshift_hostname=slave1
10.0.0.4 openshift_hostname=slave2
10.0.0.6 openshift_hostname=infra1 openshift_node_labels="{ 'region': 'infra' }"
10.0.0.11 openshift_hostname=infra2 openshift_node_labels="{ 'region': 'infra' }"
10.0.0.12 openshift_hostname=infra3 openshift_node_labels="{ 'region': 'infra' }"

[openshift_ca]
10.0.0.7 openshift_hostname=master1
10.0.0.8 openshift_hostname=master2
10.0.0.13 openshift_hostname=master3
```

Provisioning of Nested OpenShift Clusters Using Ansible Deployer—Beta

IN THIS SECTION

- [Configure network connectivity to Contrail configuration and data plane functions | 203](#)
- [Installing Nested OpenShift Cluster using Ansible Deployer | 207](#)

When Contrail provides networking for an OpenShift cluster that is provisioned on a Contrail-OpenStack cluster, it is called a nested OpenShift cluster. Contrail components are shared between the two clusters.

The following steps describe how to provision a nested OpenShift cluster.



NOTE: Provisioning of nested OpenShift Clusters is supported only as a Beta feature. Ensure that you have an operational Contrail-OpenStack cluster based on Contrail Release 5.0 before provisioning a nested OpenShift cluster.

Configure network connectivity to Contrail configuration and data plane functions

A nested OpenShift cluster is managed by the same Contrail control processes that manage the underlying OpenStack cluster. The nested OpenShift cluster needs IP reachability to the Contrail control processes. Because the OpenShift cluster is actually an overlay on the OpenStack cluster, you can use

the link local service feature or a combination of link local service with fabric Source Network Address Translation (SNAT) feature of Contrail to provide IP reachability to and from the OpenShift cluster on the overlay and the OpenStack cluster.

Use *one* of the following options to create link local services.

- **Fabric SNAT with link local service**

To provide IP reachability to and from the Kubernetes cluster using the fabric SNAT with link local service, perform the following steps.

1. Enable fabric SNAT on the virtual network of the VMs.

The fabric SNAT feature must be enabled on the virtual network of the virtual machines on which the Kubernetes master and minions are running.

2. Create one link local service for the Container Network Interface (CNI) to communicate with its vRouter using the Contrail GUI.

The following link local service is required.

Contrail Process	Service IP	Service Port	Fabric IP	Fabric Port
vRouter	<i>Service_IP for the active node</i>	9091	127.0.0.1	9091



NOTE: Fabric IP address is 127.0.0.1 since you must make the CNI communicate with the vRouter on its underlay node.

For example, the following link local services must be created:

Link Local Service Name	Service IP	Service Port	Fabric IP	Fabric Port
K8s-cni-to-agent	10.10.10.5	9091	127.0.0.1	9091



NOTE: Here 10.10.10.5 is the Service IP address that you chose. This can be any unused IP in the cluster. This IP address is primarily used to identify link local traffic and has no other significance.

- **Link local only**

To configure a Link local service, you need a Service IP address and a Fabric IP address. The fabric IP address is the node IP address on which the Contrail processes are running. Service IP address along with port number is used by the data plane to identify the fabric IP address. Service IP address is required to be a unique and unused IP address in the entire OpenStack cluster. For each node of the OpenStack cluster, one service IP address must be identified.

The following are the link local services are required:

Contrail Process	Service IP	Service Port	Fabric IP	Fabric Port
Contrail Config	<i>Service_IP for the active node</i>	8082	<i>Node_IP for the active node</i>	8082
Contrail Analytics	<i>Service_IP for the active node</i>	8086	<i>Node_IP for the active node</i>	8086
Contrail Msg Queue	<i>Service_IP for the active node</i>	5673	<i>Node_IP for the active node</i>	5673
Contrail VNC DB	<i>Service_IP for the active node</i>	9161	<i>Node_IP for the active node</i>	9161
Keystone	<i>Service_IP for the active node</i>	35357	<i>Node_IP for the active node</i>	35357
vRouter	<i>Service_IP for the active node</i>	9091	127.0.0.1	9091

For example, consider the following hypothetical OpenStack cluster:

```
Contrail Config : 192.168.1.100
Contrail Analytics : 192.168.1.100, 192.168.1.101
Contrail Msg Queue : 192.168.1.100
Contrail VNC DB : 192.168.1.100, 192.168.1.101, 192.168.1.102
Keystone: 192.168.1.200
Vrouter: 192.168.1.300, 192.168.1.400, 192.168.1.500
```


This cluster is made of seven nodes. You must allocate seven unused IP addresses for these nodes:

```
192.168.1.100 --> 10.10.10.1
192.168.1.101 --> 10.10.10.2
192.168.1.102 --> 10.10.10.3
192.168.1.200 --> 10.10.10.4
192.168.1.300 --> 10.10.10.5
192.168.1.400 --> 10.10.10.6
192.168.1.500 --> 10.10.10.7
```

The following link local services must be created:

Link Local Service Name	Service IP	Service Port	Fabric IP	Fabric Port
Contrail Config	10.10.10.1	8082	192.168.1.100	8082
Contrail Analytics	10.10.10.1	8086	192.168.1.100	8086
Contrail Analytics 2	10.10.10.2	8086	192.168.1.101	8086
Contrail Msg Queue	10.10.10.1	5673	192.168.1.100	5673
Contrail VNC DB 1	10.10.10.1	9161	192.168.1.100	9161
Contrail VNC DB 2	10.10.10.2	9161	192.168.1.101	9161
Contrail VNC DB 3	10.10.10.3	9161	192.168.1.102	9161
Keystone	10.10.10.4	35357	192.168.1.200	35357
VRouter-192.168.1.300	10.10.10.5	9091	127.0.0.1	9091
VRouter-192.168.1.400	10.10.10.6	9091	127.0.0.1	9091

VRouter-192.168.1.500	10.10.10.7	9091	127.0.0.1	9091
-----------------------	------------	------	-----------	------

Installing Nested OpenShift Cluster using Ansible Deployer

Perform the steps on [No Link Title](#) to continue installing and provisioning the OpenShift cluster.

Sample ose-install File

Add the following information to the ["No Link Title" on page 199](#).

```
#Nested mode vars
nested_mode_contrail=true
auth_mode=keystone
keystone_auth_host=192.168.24.12
keystone_auth_admin_tenant=admin
keystone_auth_admin_user=admin
keystone_auth_admin_password=MAYffWrX7ZpPrV2AMaA9zAUvG
keystone_auth_admin_port=35357
keystone_auth_url_version=/v3
#k8s_nested_vrouter_vip is a service IP for the running node which we configured above
k8s_nested_vrouter_vip=10.10.10.5
#k8s_vip is kubernetes api server ip
k8s_vip=192.168.1.3
#cluster_network is the one which vm network belongs to
cluster_network="{ 'domain': 'default-domain', 'project': 'admin', 'name': 'net1' }"
```

For more information, see <https://github.com/Juniper/contrail-kubernetes-docs/tree/master/install/openshift/3.9>.

RELATED DOCUMENTATION

| [Deploying Contrail with Red Hat OpenStack Platform Director 13](#) | 138

Using Contrail with Juju Charms

IN THIS CHAPTER

- [Deploying Contrail by Using Juju Charms | 208](#)

Deploying Contrail by Using Juju Charms

IN THIS SECTION

- [Preparing to Deploy Contrail by Using Juju Charms | 209](#)
- [Deploying Contrail Charms | 211](#)
- [Options for Juju Charms | 223](#)

You can deploy Contrail by using Juju Charms. Juju helps you deploy, configure, and efficiently manage applications on private clouds and public clouds. Juju accesses the cloud with the help of a Juju controller. A Charm is a module containing a collection of scripts and metadata and is used with Juju to deploy Contrail.

Contrail supports the following charms:

- `contrail-agent`
- `contrail-analytics`
- `contrail-analyticsdb`
- `contrail-controller`
- `contrail-keystone-auth`
- `contrail-openstack`

These topics describe how to deploy Contrail by using Juju Charms.

Preparing to Deploy Contrail by Using Juju Charms

Follow these steps to prepare for deployment:

1. Install Juju.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install juju
```

2. Configure Juju.

You can add a cloud to Juju, identify clouds supported by Juju, and also manage clouds already added to Juju.

- **Adding a cloud**—Juju recognizes a wide range of cloud types. You can use any one of the following methods to add a cloud to Juju:
 - **Adding a Cloud by Using Interactive Command**

Example: Adding an MAAS cloud to Juju

```
juju add-cloud
Cloud Types
  maas
  manual
  openstack
  oracle
  vsphere

Select cloud type: maas

Enter a name for your maas cloud: maas-cloud

Enter the API endpoint url: http://<ip-address>:<node>/MAAS

Cloud "maas-cloud" successfully added
You may bootstrap with 'juju bootstrap maas-cloud'
```



NOTE: Juju 2.x is compatible with MAAS series 1.x and 2.x.

- **Adding a Cloud Manually**

You use a YAML configuration file to add a cloud manually. Enter the following command:

```
juju add-cloud <cloud-name>
juju add-credential <cloud name>
```

For an example, to add the cloud *junmaas*, assuming that the name of the configuration file in the directory is *maas-clouds.yaml*, you run the following command:

```
juju add-cloud junmaas maas-clouds.yaml
```

The following is the format of the YAML configuration file:

```
clouds:
  <cloud_name>:
    type: <type_of_cloud>
    auth-types: [<authentication_types>]
    regions:
      <region-name>:
        endpoint: <http://<ip-address>:<node>/MAAS>
```



NOTE: The auth-types for a MAAS cloud type is *oauth1*.

- **Identifying a supported cloud**

Juju recognizes the cloud types given below. You use the `juju clouds` command to list cloud types that are supported by Juju.

```
$ juju clouds
```

Cloud	Regions	Default	Type	Description
aws	15	us-east-1	ec2	Amazon Web Services
aws-china	1	cn-north-1	ec2	Amazon China
aws-gov	1	us-gov-west-1	ec2	Amazon (USA Government)
azure	26	centralus	azure	Microsoft Azure
azure-china	2	chinaeast	azure	Microsoft Azure China
cloudsigma	5	hn1	cloudsigma	CloudSigma Cloud
google	13	us-east1	gce	Google Cloud Platform
joyent	6	eu-ams-1	joyent	Joyent Cloud

oracle	5	uscom-central-1	oracle	Oracle Cloud
rackspace	6	dfw	rackspace	Rackspace Cloud
localhost	1	localhost	lxd	LXD Container Hypervisor

3. Create a Juju controller.

```
juju bootstrap --bootstrap-series=xenial <cloud name> <controller name>
```



NOTE: A Juju controller manages and keeps track of applications in the Juju cloud environment.

Deploying Contrail Charms

IN THIS SECTION

- [Deploying Contrail Charms in a Bundle | 211](#)
- [Deploying Juju Charms Manually | 218](#)

You can deploy Contrail Charms in a bundle or manually.

Deploying Contrail Charms in a Bundle

Follow these steps to deploy Contrail Charms in a bundle.

1. Deploy Contrail Charms.

To deploy Contrail Charms in a bundle, use the `juju deploy <bundle_yaml_file>` command.

The following example shows you how to use **bundle_yaml_file** to deploy Contrail on Amazon Web Services (AWS) Cloud.

```
series: xenial
services:
  ubuntu:
    charm: cs:xenial/ubuntu
    num_units: 3
    to: [ "1", "2", "3" ]
```

```

ntp:
  charm: cs:xenial/ntp
  num_units: 0
  options:
    source: ntp.juniper.net
mysql:
  charm: cs:xenial/percona-cluster
  options:
    dataset-size: 15%
    max-connections: 10000
    root-password: password
    sst-password: password
    vip: ip-address
    vip_cidr: 24
  num_units: 3
  to: [ "lxd:1", "lxd:2", "lxd:3" ]
rabbitmq-server:
  charm: cs:xenial/rabbitmq-server
  num_units: 3
  to: [ "lxd:1", "lxd:2", "lxd:3" ]
heat:
  charm: cs:xenial/heat
  num_units: 3
  options:
    vip: ip-address
    vip_cidr: 24
  to: [ "lxd:1", "lxd:2", "lxd:3" ]
keystone:
  charm: cs:xenial/keystone
  options:
    admin-password: password
    admin-role: admin
    openstack-origin: cloud:xenial-newton
    vip: ip-address
    vip_cidr: 24
  num_units: 3
  to: [ "lxd:1", "lxd:2", "lxd:3" ]
nova-cloud-controller:
  charm: cs:xenial/nova-cloud-controller
  options:
    network-manager: Neutron
    openstack-origin: cloud:xenial-newton
    vip: ip-address

```

```

    vip_cidr: 24
    num_units: 3
    to: [ "lxd:1", "lxd:2", "lxd:3" ]
neutron-api:
    charm: cs:xenial/neutron-api
    series: xenial
    options:
        manage-neutron-plugin-legacy-mode: false
        openstack-origin: cloud:xenial-newton
        vip: ip-address
        vip_cidr: 24
    num_units: 3
    to: [ "lxd:1", "lxd:2", "lxd:3" ]
glance:
    charm: cs:xenial/glance
    options:
        openstack-origin: cloud:xenial-newton
        vip: ip-address
        vip_cidr: 24
    num_units: 3
    to: [ "lxd:1", "lxd:2", "lxd:3" ]
openstack-dashboard:
    charm: cs:xenial/openstack-dashboard
    options:
        openstack-origin: cloud:xenial-newton
        vip: ip-address
        vip_cidr: 24
    num_units: 3
    to: [ "lxd:1", "lxd:2", "lxd:3" ]
nova-compute:
    charm: cs:xenial/nova-compute
    options:
        openstack-origin: cloud:xenial-newton
    num_units: 3
    to: [ "4", "5", "6" ]
mysql-hacluster:
    charm: cs:xenial/hacluster
    options:
        cluster_count: 3
    num_units: 0
keystone-hacluster:
    charm: cs:xenial/hacluster
    options:

```



```

        cluster_count: 3
        num_units: 0
ncc-hacluster:
    charm: cs:xenial/hacluster
    options:
        cluster_count: 3
        num_units: 0
neutron-hacluster:
    charm: cs:xenial/hacluster
    options:
        cluster_count: 3
        num_units: 0
glance-hacluster:
    charm: cs:xenial/hacluster
    options:
        cluster_count: 3
        num_units: 0
dashboard-hacluster:
    charm: cs:xenial/hacluster
    options:
        cluster_count: 3
        num_units: 0
heat-hacluster:
    charm: cs:xenial/hacluster
    options:
        cluster_count: 3
        num_units: 0
contrail-openstack:
    charm: cs:~juniper-os-software/contrail-openstack
    series: xenial
    num_units: 0
contrail-agent:
    charm: cs:~juniper-os-software/contrail-agent
    num_units: 0
    series: xenial
    options:
        log-level: "SYS_DEBUG"
contrail-analytics:
    charm: cs:~juniper-os-software/contrail-analytics
    num_units: 3
    series: xenial
    to: [ "1", "2", "3" ]
contrail-analyticsdb:

```

```

charm: cs:~juniper-os-software/contrail-analyticsdb
num_units: 3
series: xenial
options:
  log-level: "SYS_DEBUG"
  cassandra-minimum-diskgb: 4
  cassandra-jvm-extra-opts: "-Xms1g -Xmx2g"
  to: [ "1", "2", "3" ]
contrail-controller:
charm: cs:~juniper-os-software/contrail-controller
series: xenial
options:
  vip: ip-address
  log-level: "SYS_DEBUG"
  cassandra-minimum-diskgb: 4
  cassandra-jvm-extra-opts: "-Xms1g -Xmx2g"
  to: [ "1", "2", "3" ]
contrail-keystone-auth:
charm: cs:~juniper-os-software/contrail-keystone-auth
series: xenial
num_units: 1
to: [ "lxd:1" ]

contrail-keepalived:
charm: cs:~boucherv29/keepalived-19
series: xenial
options:
  virtual_ip: ip-address
contrail-haproxy:
charm: haproxy
series: xenial
expose: true
options:
  peering_mode: "active-active"
  to: [ "1", "2", "3" ]

relations:
# openstack
- [ "ubuntu", "ntp" ]
- [ mysql, mysql-hacluster ]
- [ "keystone", "mysql" ]
- [ keystone, keystone-hacluster ]
- [ "glance", "mysql" ]

```

```

- [ "glance", "keystone" ]
- [ glance, glance-hacluster ]
- [ "nova-cloud-controller", "mysql" ]
- [ "nova-cloud-controller", "rabbitmq-server" ]
- [ "nova-cloud-controller", "keystone" ]
- [ "nova-cloud-controller", "glance" ]
- [ nova-cloud-controller, ncc-hacluster ]
- [ "neutron-api", "mysql" ]
- [ "neutron-api", "rabbitmq-server" ]
- [ "neutron-api", "nova-cloud-controller" ]
- [ "neutron-api", "keystone" ]
- [ neutron-api, neutron-hacluster ]
- [ "nova-compute:amqp", "rabbitmq-server:amqp" ]
- [ "nova-compute", "glance" ]
- [ "nova-compute", "nova-cloud-controller" ]
- [ "nova-compute", "ntp" ]
- [ "openstack-dashboard:identity-service", "keystone" ]
- [ openstack-dashboard, dashboard-hacluster ]
- [ "heat", "mysql" ]
- [ "heat", "rabbitmq-server" ]
- [ "heat", "keystone" ]
- [ "heat", "heat-hacluster" ]

#contrail
- [ "contrail-keystone-auth", "keystone" ]
- [ "contrail-controller", "contrail-keystone-auth" ]
- [ "contrail-analytics", "contrail-analyticsdb" ]
- [ "contrail-controller", "contrail-analytics" ]
- [ "contrail-controller", "contrail-analyticsdb" ]
- [ "contrail-openstack", "nova-compute" ]
- [ "contrail-openstack", "neutron-api" ]
- [ "contrail-openstack", "heat" ]
- [ "contrail-openstack", "contrail-controller" ]
- [ "contrail-agent:juju-info", "nova-compute:juju-info" ]
- [ "contrail-agent", "contrail-controller" ]

#haproxy
- [ "haproxy:juju-info", "keepalived:juju-info" ]
- [ "contrail-analytics", "haproxy" ]
- [ "contrail-controller:http-services", "haproxy" ]
- [ "contrail-controller:https-services", "haproxy" ]

```

machines:

```

"1":
  series: xenial
  #constraints: mem=15G root-disk=40G
  constraints: tags=contrail-controller-vm-1
"2":
  series: xenial
  #constraints: mem=15G root-disk=40G
  constraints: tags=contrail-controller-vm-2
"3":
  series: xenial
  #constraints: mem=15G root-disk=40G
  constraints: tags=contrail-controller-vm-3
"4":
  series: xenial
  #constraints: mem=4G root-disk=20G
  constraints: tags=compute-storage-1
"5":
  series: xenial
  #constraints: mem=4G root-disk=20G
  constraints: tags=compute-storage-2
"6":
  series: xenial
  #constraints: mem=4G root-disk=20G
  constraints: tags=compute-storage-3

```

You can create or modify the Contrail Charm deployment bundle YAML file to:

- Point to machines or instances where the Contrail Charms must be deployed.
- Include the options you need.

Each Contrail Charm has a specific set of options. The options you choose depend on the charms you select. For more information on the options that are available, see ["Options for Juju Charms" on page 223](#).

2. (Optional) Check the status of deployment.

You can check the status of the deployment by using the `juju status` command.

3. Enable configuration statements.

Based on your deployment requirements, you can enable the following configuration statements:

- `contrail-agent`

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-agent/>.

- `contrail-analytics`

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-analytics>.

- contrail-analyticsdb

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-analyticsdb>.

- contrail-controller

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-controller>.

- contrail-keystone-auth

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-keystone-auth>.

- contrail-openstack

For more information see, <https://jaas.ai/u/juniper-os-software/contrail-openstack>.

Deploying Juju Charms Manually

Before you begin deployment, ensure that you have:

- Installed and configured Juju
- Created a Juju controller
- Ubuntu 16.04 or Ubuntu 18.04 installed

Follow these steps to deploy Juju Charms manually:

1. Create machine instances for OpenStack, compute, and Contrail.

```
juju add-machine --constraints mem=8G cores=2 root-disk=40G --series=xenial #for openstack
machine(s) 0
juju add-machine --constraints mem=7G cores=4 root-disk=40G --series=xenial #for compute
machine(s) 1,(3)
juju add-machine --constraints mem=15G cores=2 root-disk=300G --series=xenial #for contrail
machine 2
```

2. Deploy OpenStack services.

You can deploy OpenStack services by using any one of the following methods:

- **By specifying the OpenStack parameters in a YAML file**

The following is an example of a YAML-formatted (**nova-compute-config.yaml**) file.

```
nova-compute:
  openstack-origin: cloud:xenial-ocata
  virt-type: qemu
  enable-resize: True
  enable-live-migration: True
  migration-auth-type: ssh
```

Use this command to deploy OpenStack services by using a YAML-formatted file:

```
juju deploy cs:xenial/nova-compute --config ./nova-compute-config.yaml
```

- **By using CLI**

To deploy OpenStack services through the CLI:

```
juju deploy cs:xenial/nova-cloud-controller --config console-access-protocol=novnc --
config openstack-origin=cloud:xenial-ocata
```

- **By using a combination of YAML-formatted file and CLI**

To deploy OpenStack services by using a combination of YAML-formatted file and CLI:



NOTE: Use the `--to <machine number>` command to point to a machine or container where you want the application to be deployed.

```
juju deploy cs:xenial/ntp
juju deploy cs:xenial/rabbitmq-server --to lxd:0
juju deploy cs:xenial/percona-cluster mysql --config root-password=<root-password> --
config max-connections=1500 --to lxd:0
juju deploy cs:xenial/openstack-dashboard --config openstack-origin=cloud:xenial-ocata --
to lxd:0
juju deploy cs:xenial/nova-cloud-controller --config console-access-protocol=novnc --
config openstack-origin=cloud:xenial-ocata --config network-manager=Neutron --to lxd:0
juju deploy cs:xenial/neutron-api --config manage-neutron-plugin-legacy-mode=false --
config openstack-origin=cloud:xenial-ocata --config neutron-security-groups=true --to lxd:0
juju deploy cs:xenial/glance --config openstack-origin=cloud:xenial-ocata --to lxd:0
```

```
juju deploy cs:xenial/keystone --config admin-password=<admin-password> --config admin-
role=admin --config openstack-origin=cloud:xenial-ocata --to lxd:0
```



NOTE: You set OpenStack services on different machines or on different containers to prevent HAProxy conflicts from applications.

3. Deploy and configure nova-compute.

```
juju deploy cs:xenial/nova-compute --config ./nova-compute-config.yaml --to 1
```



NOTE: You can deploy nova-compute to more than one compute machine.

(Optional) To add additional computes:

```
juju add-unit nova-compute --to 3 # Add one more unit
```

4. Deploy and configure Contrail services.

```
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-keystone-auth --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-controller --config
auth-mode=rbac --config cassandra-minimum-diskgb=4 --config cassandra-jvm-extra-opts="-Xms1g -
Xmx2g" --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-analyticsdb cassandra-
minimum-diskgb=4 --config cassandra-jvm-extra-opts="-Xms1g -Xmx2g" --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-analytics --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-openstack
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-agent
```

5. Enable applications to be available to external traffic:

```
juju expose openstack-dashboard
juju expose nova-cloud-controller
juju expose neutron-api
juju expose glance
juju expose keystone
```

6. Enable contrail-controller and contrail-analytics services to be available to external traffic if you do not use HAProxy.

```
juju expose contrail-controller
juju expose contrail-analytics
```

7. Apply SSL.

You can apply SSL if needed. To use SSL with Contrail services, deploy easy-rsa service and add-relation command to create relations to contrail-controller service and contrail-agent services.

```
juju deploy cs:~containers/xenial/easyrsa --to 0
juju add-relation easyrsa contrail-controller
juju add-relation easyrsa contrail-agent
```

8. (Optional) HA configuration.

If you use more than one controller, follow the HA solution given below:

- a. Deploy HAProxy and Keepalived services.

HAProxy charm is deployed on machines with Contrail controllers. HAProxy charm must have `peering_mode` set to active-active. If `peering_mode` is set to active-passive, HAProxy creates additional listeners on the same ports as other Contrail services. This leads to port conflicts.

Keepalived charm does not require to option.

```
juju deploy cs:xenial/haproxy --to <first contrail-controller machine> --config
peering_mode=active-active
juju add-unit haproxy --to <another contrail-controller machine>
juju deploy cs:~boucherv29/keepalived-19 --config virtual_ip=<vip>
```

- b. Enable HAProxy to be available to external traffic.

```
juju expose haproxy
```



NOTE: If you enable HAProxy to be available to external traffic, do not follow step 6.

c. Add HAProxy and Keepalived relations.

```
juju add-relation haproxy:juju-info keepalived:juju-info
juju add-relation contrail-analytics:http-services haproxy
juju add-relation contrail-controller:http-services haproxy
juju add-relation contrail-controller:https-services haproxy
```

d. Configure contrail-controller service with VIP.

```
juju set contrail-controller vip=<vip>
```

9. Add other necessary relations.

```
juju add-relation keystone:shared-db mysql:shared-db
juju add-relation glance:shared-db mysql:shared-db
juju add-relation keystone:identity-service glance:identity-service
juju add-relation nova-cloud-controller:image-service glance:image-service
juju add-relation nova-cloud-controller:identity-service keystone:identity-service
juju add-relation nova-cloud-controller:cloud-compute nova-compute:cloud-compute
juju add-relation nova-compute:image-service glance:image-service
juju add-relation nova-compute:amqp rabbitmq-server:amqp
juju add-relation nova-cloud-controller:shared-db mysql:shared-db
juju add-relation nova-cloud-controller:amqp rabbitmq-server:amqp
juju add-relation openstack-dashboard:identity-service keystone

juju add-relation neutron-api:shared-db mysql:shared-db
juju add-relation neutron-api:neutron-api nova-cloud-controller:neutron-api
juju add-relation neutron-api:identity-service keystone:identity-service
juju add-relation neutron-api:amqp rabbitmq-server:amqp

juju add-relation contrail-controller ntp
juju add-relation nova-compute:juju info ntp:juju info

juju add-relation contrail-controller contrail-keystone-auth
juju add-relation contrail-keystone-auth keystone
juju add-relation contrail-controller contrail-analytics
juju add-relation contrail-controller contrail-analyticsdb
juju add-relation contrail-analytics contrail-analyticsdb

juju add-relation contrail-openstack neutron-api
juju add-relation contrail-openstack nova-compute
```

```
juju add-relation contrail-openstack contrail-controller

juju add-relation contrail-agent:juju info nova-compute:juju info
juju add-relation contrail-agent contrail-controller
```

Options for Juju Charms

Each Contrail Charm has a specific set of options. The options you choose depend on the charms you select. The following tables list the various options you can choose:

- Options for **contrail-agent** Charms.

Table 9: Options for contrail-agent

Option	Default option	Description
physical-interface		Specify the interface where you want to install vhost0 on. If you do not specify an interface, vhost0 is installed on the default gateway interface.
vhost-gateway	auto	Specify the gateway for vhost0. You can enter either an IP address or the keyword (auto) to automatically set a gateway based on the existing vhost routes.
remove-juju-bridge	true	To install vhost0 directly on the interface, enable this option to remove any bridge created to deploy LXD/LXC and KVM workloads.
dpdk	false	Specify DPDK vRouter.
dpdk-driver	uio_pci_generic	Specify DPDK driver for the physical interface.
dpdk-hugepages	70%	Specify the percentage of huge pages reserved for DPDK vRouter and OpenStack instances.
dpdk-coremask	1	Specify the vRouter CPU affinity mask to determine on which CPU the DPDK vRouter will run.

Table 9: Options for contrail-agent (Continued)

Option	Default option	Description
dpgk-main-mempool-size		Specify the main packet pool size.
dpgk-pmd-td-size		Specify the DPDK PMD Tx Descriptor size.
dpgk-pmd-rxd-size		Specify the DPDK PMD Rx Descriptor size.
docker-registry	opencontrailnightly	Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag	latest	Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services. Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.
no_proxy		Specify the list of destinations that must be directly accessed.

- Options for **contrail-analytics** Charms.

Table 10: Options for contrail-analytics

Option	Default option	Description
control-network		Specify the IP address and network mask of the control network.
docker-registry		Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag		Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services. Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.
no_proxy		Specify the list of destinations that must be directly accessed.

- Options for **contrail-analyticsdb** Charms.

Table 11: Options for contrail-analyticsdb

Option	Default option	Description
control-network		Specify the IP address and network mask of the control network.
cassandra-minimum-diskgb	256	Specify the minimum disk requirement.
cassandra-jvm-extra-opts		Specify the memory limit.
docker-registry		Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag		Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services. Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.
no_proxy		Specify the list of destinations that must be directly accessed.

- Options for **contrail-controller** Charms.

Table 12: Options for contrail-controller

Option	Default option	Description
control-network		Specify the IP address and network mask of the control network.
auth-mode	rbac	Specify the authentication mode. Options: rbac, cloud-admin, no-auth. For more information, see https://github.com/Juniper/contrail-controller/wiki/RBAC .
cassandra-minimum-diskgb	20	Specify the minimum disk requirement.
cassandra-jvm-extra-opts		Specify the memory limit.
cloud-admin-role	admin	Specify the role name in keystone for users who have admin-level access.
global-read-only-role		Specify the role name in keystone for users who have read-only access.
vip		Specify if the Contrail API VIP is used for configuring client-side software. If not specified, private IP of the first Contrail API VIP unit will be used.
use-external-rabbitmq	false	To enable the Charm to use the internal RabbitMQ server, set use-external-rabbitmq to false. To use an external AMQP server, set use-external-rabbitmq to true. NOTE: Do not change the flag after deployment.
flow-export-rate	0	Specify how many flow records are exported by vRouter agent to the Contrail Collector when a flow is created or deleted.

Table 12: Options for contrail-controller (Continued)

Option	Default option	Description
docker-registry		Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag		Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services. Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.
no_proxy		Specify the list of destinations that must be directly accessed.

- Options for **contrail-keystone-auth** Charms.

Table 13: Options for contrail-keystone-auth

Option	Default option	Description
ssl_ca		Specify if the base64-encoded SSL CA certificate is provided to Contrail keystone clients. NOTE: This certificate is required if you use a privately signed ssl_cert and ssl_key.

- Options for **contrail-openstack** Charms.

Table 14: Options for contrail-controller

Option	Default option	Description
enable-metadata-server	true	Set enable-metadata-server to true to configure metadata and enable nova to run a local instance of nova-api-metadata for virtual machines
use-internal-endpoints	false	Set use-internal-endpoints to true for OpenStack to configure services to use internal endpoints.
heat-plugin-dirs	/usr/lib64/heat,/usr /lib/heat/usr/lib/ python2.7/dist-packages/vnc_api/gen/heat/resources	Specify the heat plugin directories.
docker-registry		Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag		Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services. Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.

Table 14: Options for contrail-controller *(Continued)*

Option	Default option	Description
no_proxy		Specify the list of destinations that must be directly accessed.

SEE ALSO

[Understanding Juju Charms](#)

Preparing to Deploy Contrail by Using Juju Charms

Deploying Contrail Charms

3

PART

Configuring Contrail

- [Configuring Virtual Networks | 232](#)
 - [Example of Deploying a Multi-Tier Web Application Using Contrail | 265](#)
 - [Configuring Services | 280](#)
 - [Configuring Service Chaining | 302](#)
 - [Examples: Configuring Service Chaining | 349](#)
-

Configuring Virtual Networks

IN THIS CHAPTER

- Creating Projects in OpenStack for Configuring Tenants in Contrail | 232
- Creating a Virtual Network with Juniper Networks Contrail | 234
- Creating a Virtual Network with OpenStack Contrail | 238
- Creating an Image for a Project in OpenStack Contrail | 240
- Creating a Floating IP Address Pool | 244
- Using Security Groups with Virtual Machines (Instances) | 246
- Support for IPv6 Networks in Contrail | 251
- Configuring EVPN and VXLAN | 255

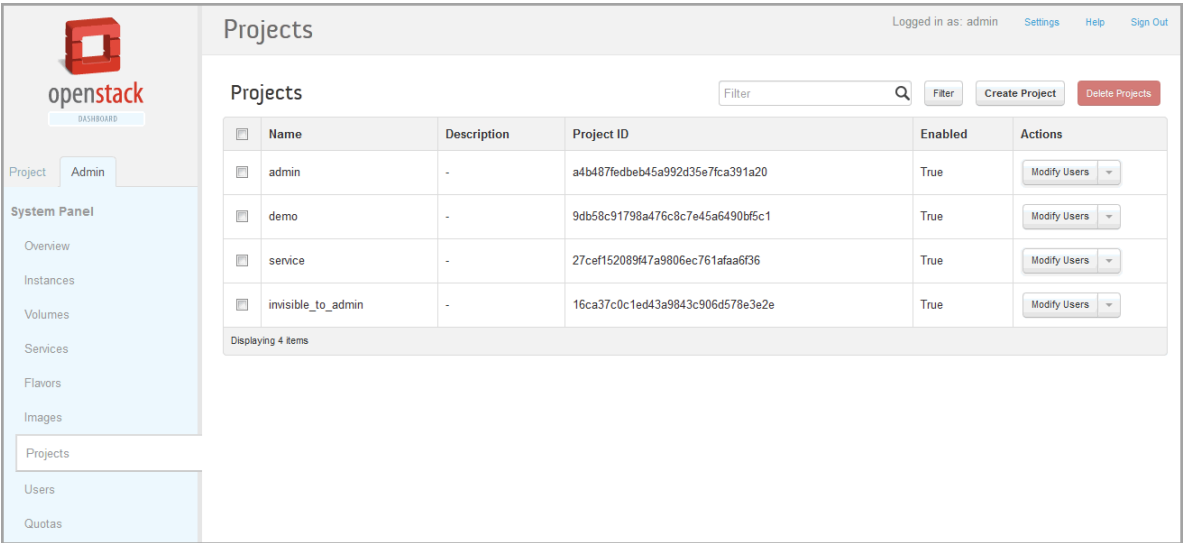
Creating Projects in OpenStack for Configuring Tenants in Contrail

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.

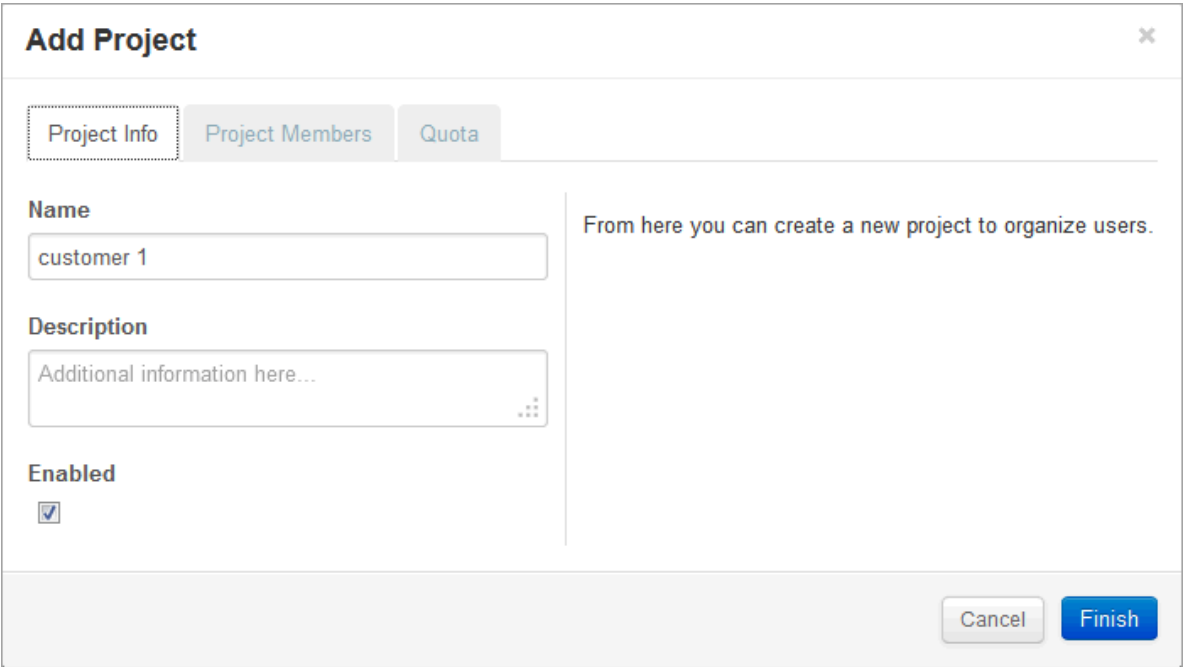
1. Click the **Admin** tab on the OpenStack dashboard, then click the **Projects** link to access the **Projects** page; see [Figure 24 on page 233](#).

Figure 24: OpenStack Projects



2. In the upper right, click the **Create Project** button to access the **Add Project** window; see [Figure 25 on page 233](#).

Figure 25: 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 | 234](#)

[Creating a Virtual Network with OpenStack Contrail | 238](#)

[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 26 on page 235](#).

Figure 26: Add IP Address Management

Add IP Address Management

Name

IPAM Name

DNS Method

Default

NTP Server IP

Domain Name

Cancel

Save

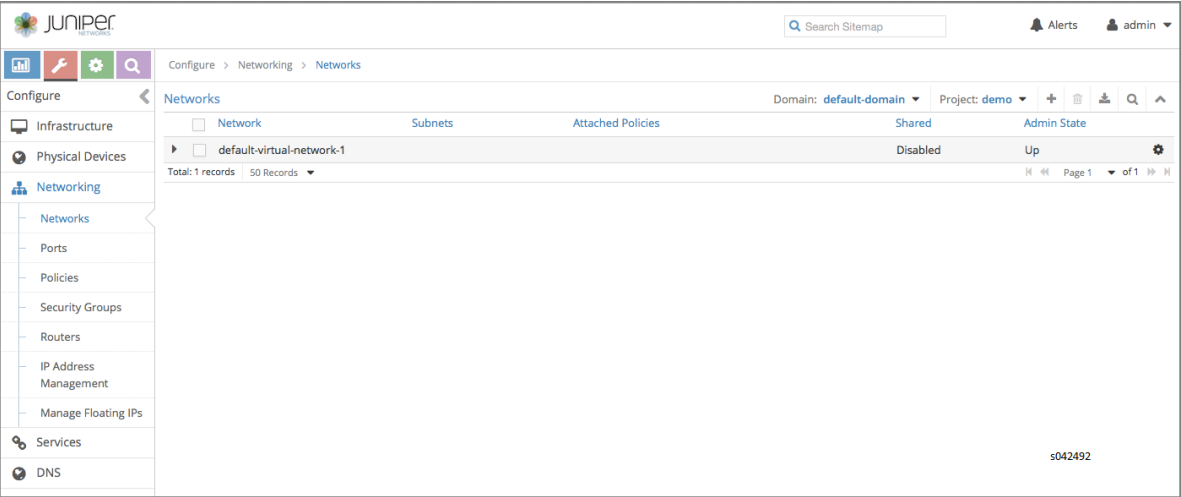
2. Complete the fields in **Add IP Address Management**: The fields are described in [Table 15 on page 235](#).

Table 15: 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. You must create a network policy first, as you need a network policy to create a virtual network. Follow the steps described in [Creating a Network Policy—Juniper Networks Contrail](#) to create a network policy.
4. Select **Configure > Networking > Networks** to access the **Configure Networks** page; see [Figure 27 on page 236](#).

Figure 27: Configure Networks




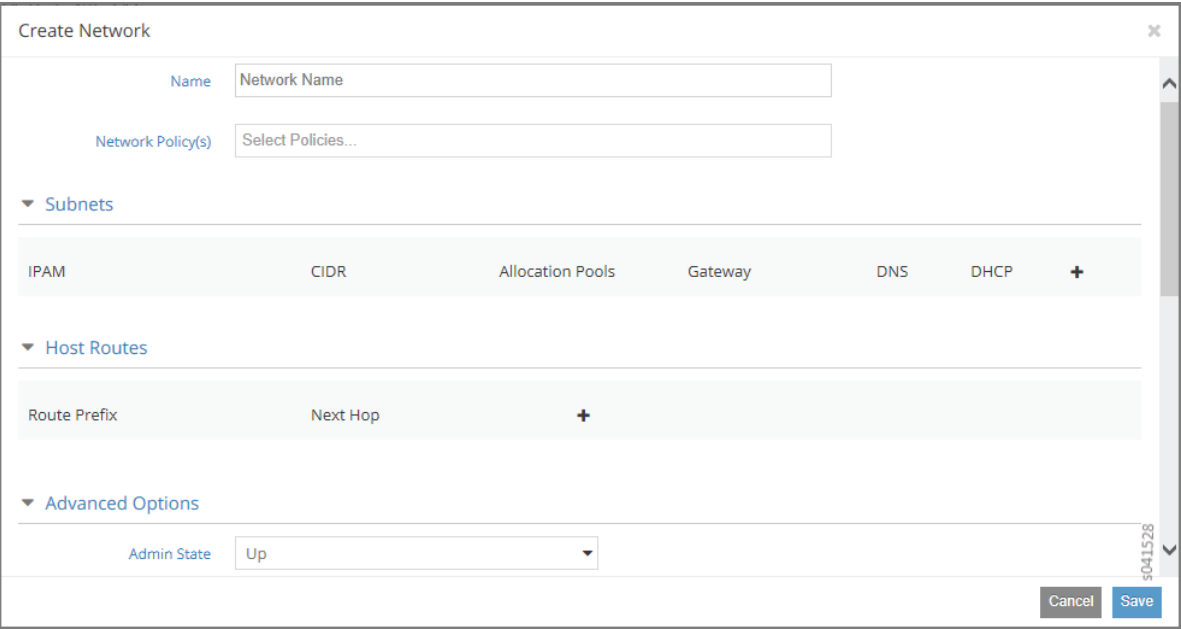
5. Verify that your project is displayed as active in the upper-right field, then click the  icon. The **Create Network** window is displayed. See [Figure 28 on page 236](#). Use the scroll bar to access all sections of this window.

Figure 28: Create Network



6. Complete the fields in the **Create Network** window with values that identify the network name, network policy, and IP options as needed. See field descriptions in [Table 16 on page 237](#).

Table 16: 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.

7. To save your network, click the **Save** button, or click **Cancel** to discard your work and start over.

RELATED DOCUMENTATION

[Creating an Image for a Project in OpenStack Contrail](#) | 240

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 29 on page 238](#).

Figure 29: 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 30 on page 238](#) and [Figure 31 on page 239](#).

Figure 30: Create Network Window

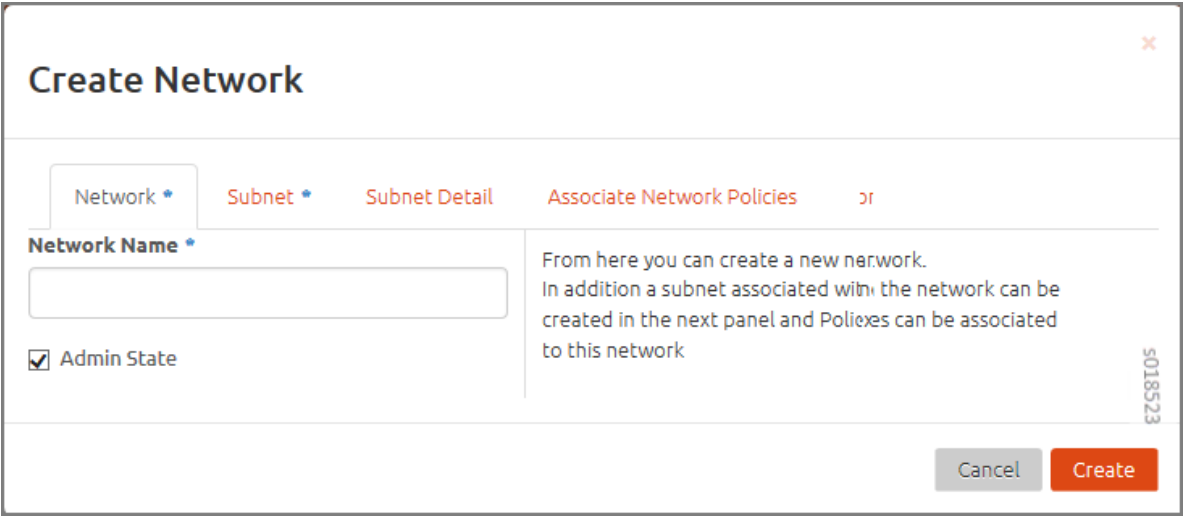


Figure 31: Create Network Window Subnet Tab

Create Network

Network *

Subnet *

Subnet Detail

Associate Network Policies

☒ Create Subnet

Subnet Name

IPAM ?

default-network-ipam (default-projec

+

Network Address ?

IP Version *

IPv4

Gateway IP ?

☐ Disable Gateway

You can create a subnet associated with the new network, in which case "Network Address" must be specified. If you wish to create a network WITHOUT a subnet, uncheck the "Create Subnet" checkbox.

5018524

Cancel

Create

3. Click the **Network**, **Subnet**, **Subnet Detail**, and **Associate Network Policies** tabs to complete the fields in the **Create Network** window. See field descriptions in [Table 17 on page 239](#).

Table 17: Create Network Fields

Field	Description
Network Name	Enter a name for the network.
Subnet Name	Enter a name for the subnetwork.

Table 17: Create Network Fields *(Continued)*

Field	Description
IPAM	<p>Select the IPAM associated with the IP block.</p> <p>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.</p>
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.

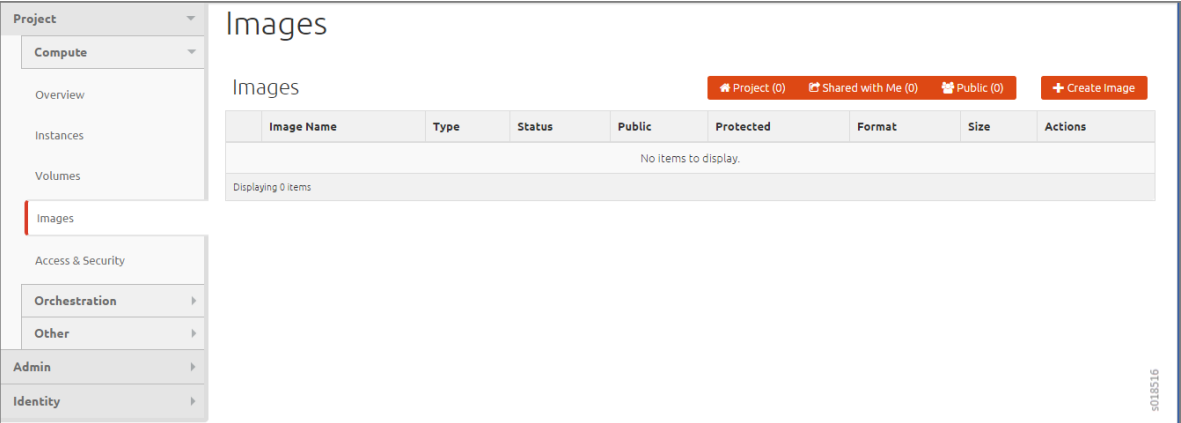
4. Click the **Subnet Details** tab to specify the Allocation Pool, DNS Name Servers, and Host Routes.
5. Click the **Associate Network Policies** tab to associate policies to the network.
6. To save your network, click **Create Network**, or click **Cancel** to discard your work and start over.

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 32 on page 241](#).

Figure 32: 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 33 on page 242](#).

Figure 33: OpenStack Create An Image Window

Create An Image

Name *

Description

Image Source

Image Location ▼

Image Location ?

http://example.com/image.iso

Format *

Select format ▼

Architecture

Minimum Disk (GB) ?

Minimum RAM (MB) ?

☐ Public

☐ Protected

Description:

Currently only images available via an HTTP URL are supported. The image location must be accessible to the Image Service. Compressed image binaries are supported (.zip and .tar.gz.)

Please note: The Image Location field MUST be a valid and direct URL to the image binary. URLs that redirect or serve error pages will result in unusable images.

s018515

Cancel

Create Image

4. Complete the fields to specify your image. [Table 18 on page 243](#) 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 18: 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 Image VHD- Virtual Hard Disk VMDK- Virtual Machine Disk
Architecture	Enter the architecture.
Minimum Disk (GB)	Enter the minimum disk size required to boot the image. If you do not specify a size, the default is 0 (no minimum).

Table 18: Create an Image Fields *(Continued)*

Field	Description
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**.

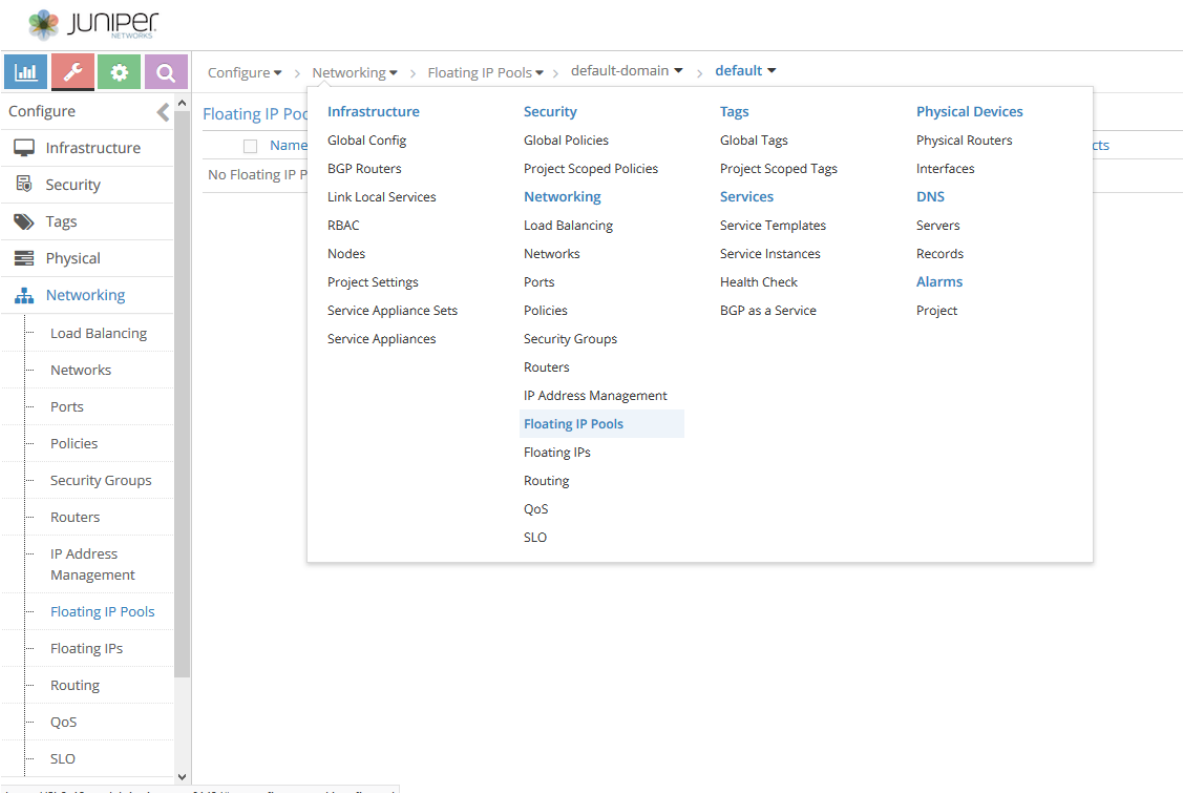
Creating a Floating IP Address Pool

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:

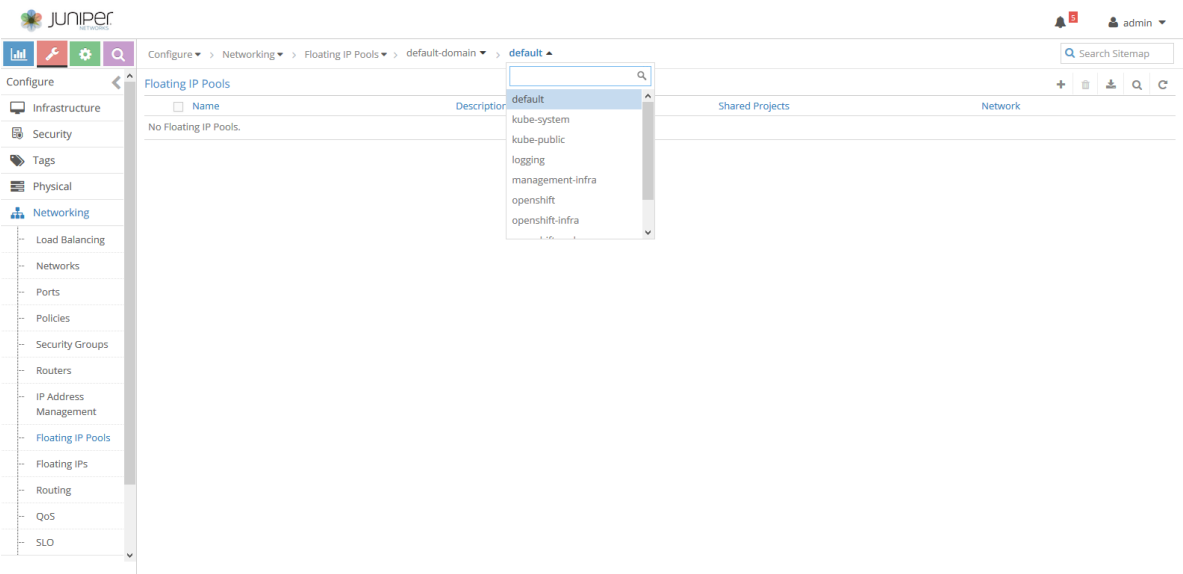
1. Select **Configure > Networking > Floating IP Pools**.

Figure 34: Floating IP Pools Selection



2. Select the network you want to associate with a floating IP pool.

Figure 35: Network Selection

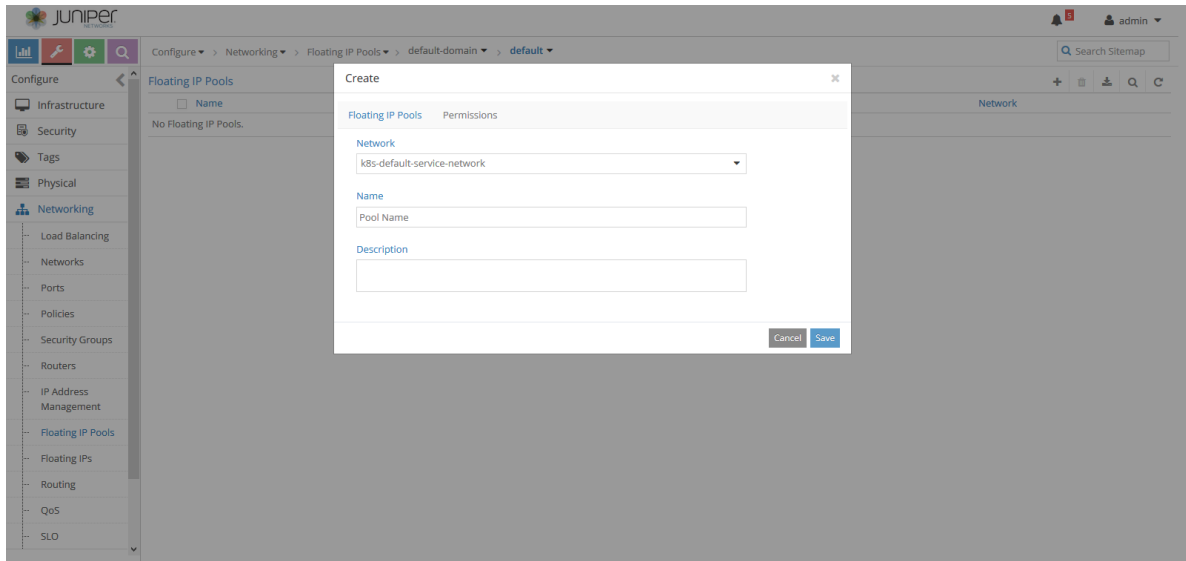


3. Click the add icon (+) to create a floating IP pool.

4. Add a **Name** and **Description** in the **Floating IP Pools** tab.

Click the **Permissions** tab to set **Owner Permissions** and **Global Share Permissions** for the floating IP pool. To associate the floating IP pool with multiple projects, click the add icon (+) in the **Share List**.

Figure 36: Create the Floating IP Pool



5. Click **Save** to create the floating IP address pool, or click **Cancel** to discard your changes and start over.

Using Security Groups with Virtual Machines (Instances)

IN THIS SECTION

- [Security Groups Overview | 246](#)
- [Creating Security Groups and Adding Rules | 247](#)

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

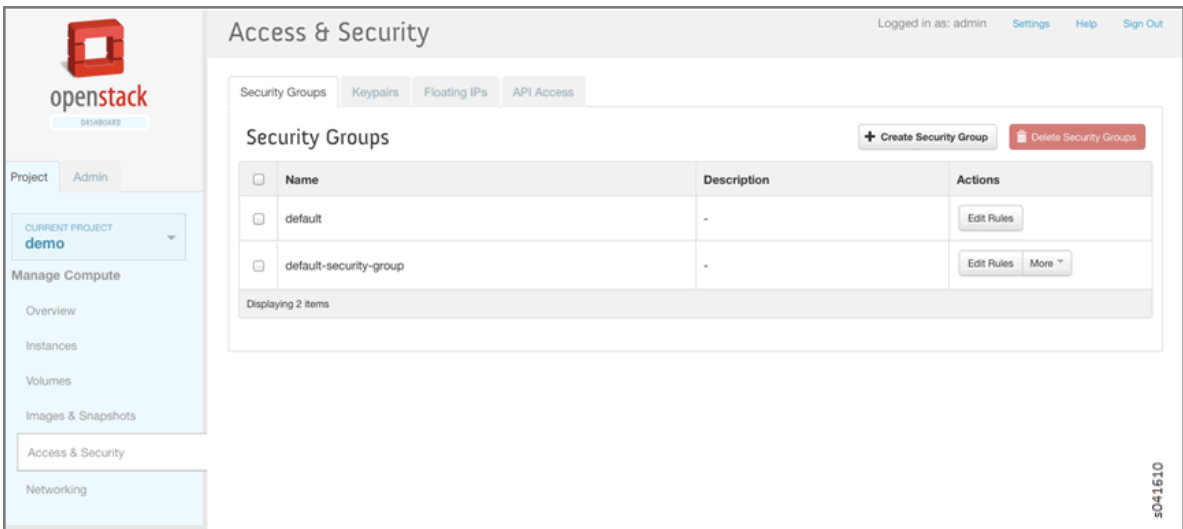
A default security group is created for each project. You can add security rules to the default security group and you can create additional security groups and add rules to them. The security groups are then associated with a VM, when the VM is launched or at a later date.

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 37 on page 247](#).

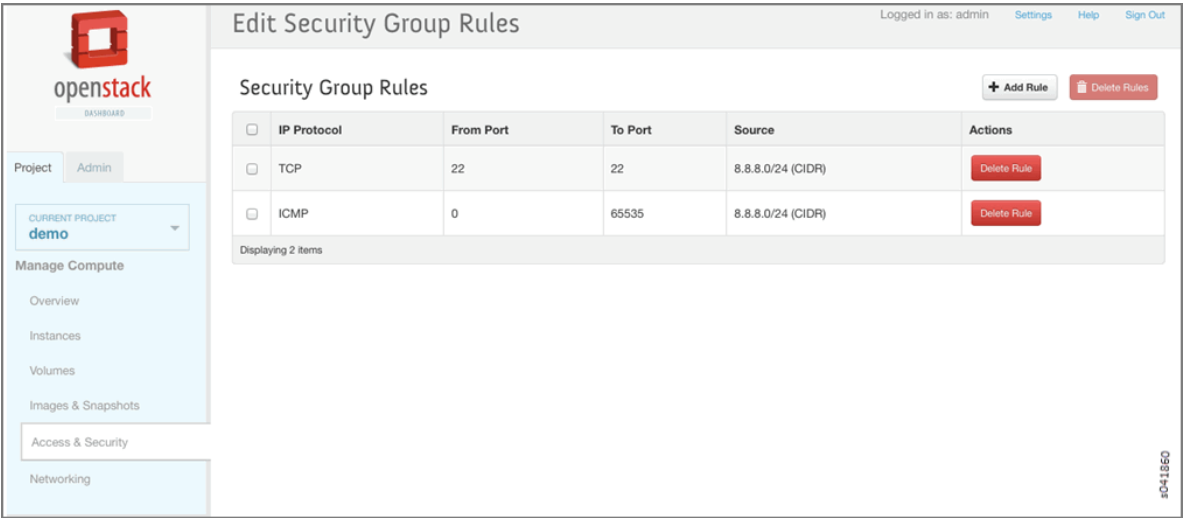
Figure 37: Security Groups



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

The **Edit Security Group Rules** window is displayed; see [Figure 38 on page 248](#). Any rules already associated with the security group are listed.

Figure 38: Edit Security Group Rules



3. Click **Add Rule** to add a new rule; see [Figure 39 on page 249](#).

Figure 39: Add Rule

Add Rule

IP Protocol

ICMP

Type

0

Code

0

Source

CIDR

CIDR

Security Group

0.0.0.0/0

Description:

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

Protocol:

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

Open Port/Port Range:

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

Source:

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

Cancel

Add

Table 19: 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 19: 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 40 on page 250](#).
- Each new security group has a unique 32-bit security group ID and an ACL is associated with the configured rules.

Figure 40: Create Security Group

Create Security Group

Name: SG1

Description: From here you can create a new security group

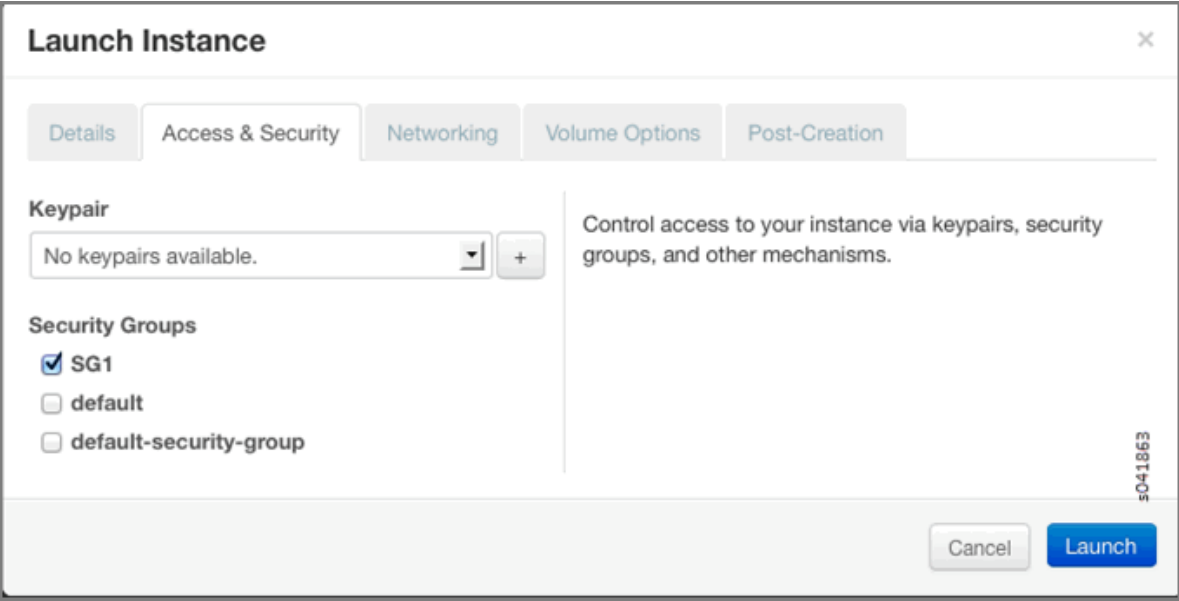
Description: Security Group 1

Cancel Create Security Group

sg-041861

5. When an instance is launched, there is an opportunity to associate a security group; see [Figure 41 on page 251](#).
- In the **Security Groups** list, select the security group name to associate with the instance.

Figure 41: 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 | 251](#)
- [Creating IPv6 Virtual Networks in Contrail | 252](#)
- [Adding IPv6 Peers | 254](#)

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

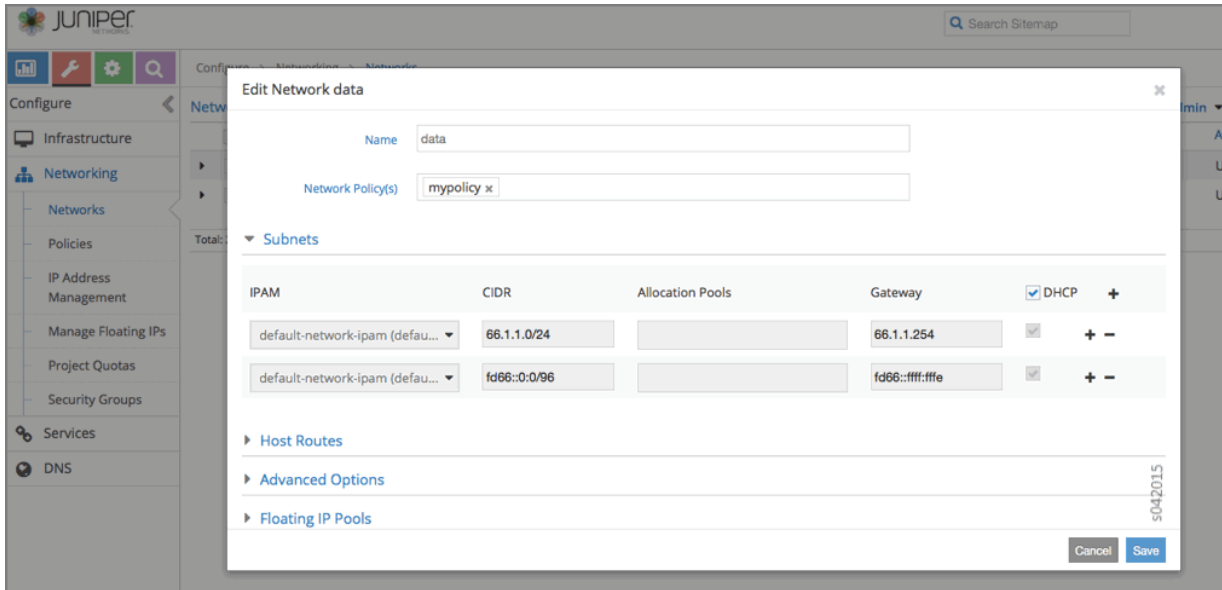
- Virtual machines with IPv6-only interfaces
- DHCPv6 and neighbor discovery
- Policy and Security groups
- IPv6 flow set up, tear down, and aging
- Flow set up and tear down based on TCP state machine
- Protocol-based flow aging
- Fat flow
- Allowed address pair configuration with IPv6 addresses
- IPv6 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.

openstack

SAVED

Project

Admin

CURRENT PROJECT

admin

Manage Compute

Overview

Instances

Volumes

Images & Snapshots

Access & Security

Other

Routers

Network Topology

Load Balancers

Networking

Instances

Logged in as: admin

Settings

Help

Sign Out

Filter

Filter

+ Launch Instance

Soft Reboot Instances

Terminate Instances

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Size	Keypair	Status	Task	Power State	Uptime	Actions
<input type="checkbox"/>	Test-6dba4281-ada9-41fc-8009-bcd89d78ee3	ubuntu-jdof	data 66.1.1.251 fd66::fff:fff vn-jdaf 76.1.1.252	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<div>Create Snapshot</div> <div>More ~</div>
<input type="checkbox"/>	Test-7a3b7c5b-e6a5-46b3-9346-29079a1abdba	ubuntu-jdof	data 66.1.1.250 fd66::fff:fff vn-jdaf 76.1.1.250	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<div>Create Snapshot</div> <div>More ~</div>
<input type="checkbox"/>	Test-663309b7-1765-4cc4-9edc-f9025ecd4ee5	ubuntu-jdof	data 66.1.1.245 fd66::fff:fff vn-jdaf 76.1.1.244	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<div>Create Snapshot</div> <div>More ~</div>
<input type="checkbox"/>	Test-a20de6d7-3d2b-447e-8894-d794eaa620ab	ubuntu-jdof	data 66.1.1.252 fd66::fff:fff vn-jdaf 76.1.1.251	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<div>Create Snapshot</div> <div>More ~</div>
<input type="checkbox"/>	Test-43345608-455f-47a6-9346-5c81f6be2197	ubuntu-jdof	data 66.1.1.247 fd66::fff:fff vn-jdaf 76.1.1.247	m1.medium 4GB RAM 2 VCPU 40.0GB Disk	-	Active	None	Running	4 days, 9 hours	<div>Create Snapshot</div> <div>More ~</div>

s042016

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:

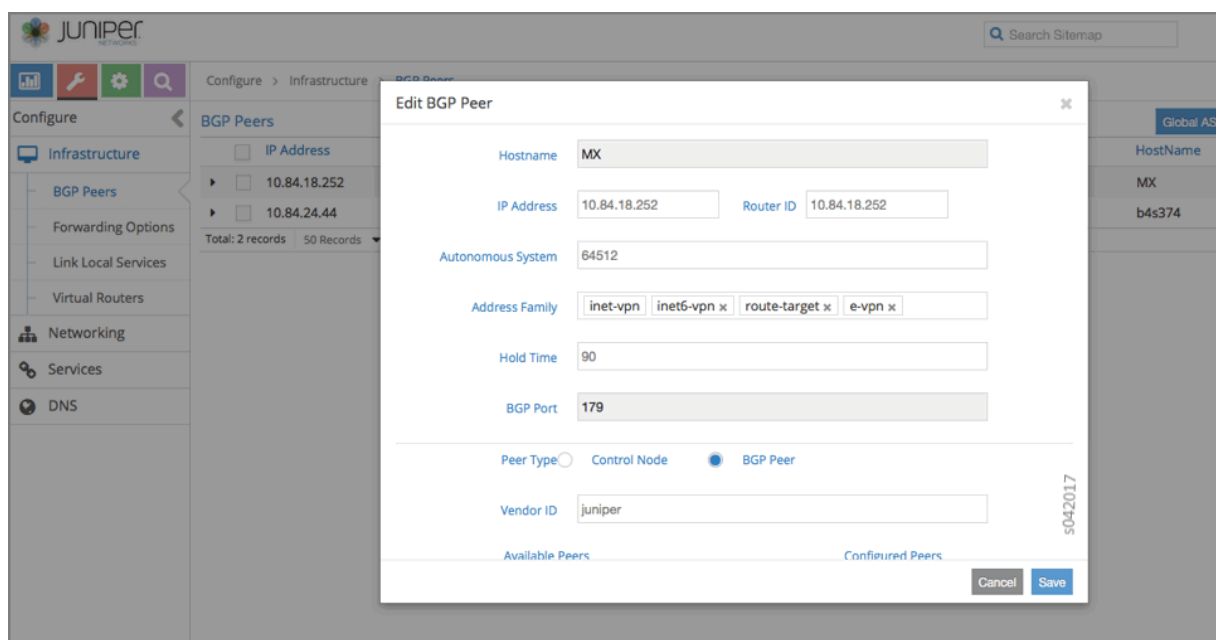
```
iface eht0 inet6 dhcp
```

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 | 257](#)
- [Configuring Forwarding | 259](#)
- [Configuring the VXLAN Identifier | 260](#)
- [Configuring Encapsulation Methods | 261](#)

Contrail supports Ethernet VPNs (EVPN) and Virtual Extensible Local Area Networks (VXLAN).

EVPN is a flexible solution that uses Layer 2 overlays to interconnect multiple edges (virtual machines) within a data center. Traditionally, the data center is built as a flat Layer 2 network with issues such as flooding, limitations in redundancy and provisioning, and high volumes of MAC address learning, which cause churn during node failures. EVPNs are designed to address these issues without disturbing flat MAC connectivity.

In EVPNs, MAC address learning is driven by the control plane, rather than by the data plane, which helps control learned MAC addresses across virtual forwarders, thus avoiding flooding. The forwarders advertise locally learned MAC addresses to the controllers. The controllers use MP-BGP to communicate with peers. The peering of controllers using BGP for EVPN results in better and faster convergence.

With EVPN, MAC learning is confined to the virtual networks to which the virtual machine belongs, thus isolating traffic between multiple virtual networks. In this manner, virtual networks can share the same MAC addresses without any traffic crossover.

Unicast in EVPNs

Unicast forwarding is based on MAC addresses where traffic can terminate on a local endpoint or is encapsulated to reach the remote endpoint. Encapsulation can be MPLS/UDP, MPLS/GRE, or VXLAN.

BUM Traffic in EVPN

Multicast and broadcast traffic is flooded in a virtual network. The replication tree is built by the control plane, based on the advertisements of end nodes (virtual machines) sent by forwarders. Each virtual network has one distribution tree, a method that avoids maintaining multicast states at fabric nodes, so the nodes are unaffected by multicast. The replication happens at the edge forwarders. Per-group subscription is not provided. Broadcast, unknown unicast, and multicast (BUM) traffic is handled the same way, and gets flooded in the virtual network to which the virtual machine belongs.

VXLAN

VXLAN is an overlay technology that encapsulates MAC frames into a UDP header at Layer 2. Communication is established between two virtual tunnel endpoints (VTEPs). VTEPs encapsulate the virtual machine traffic into a VXLAN header, as well as strip off the encapsulation. Virtual machines can only communicate with each other when they belong to the same VXLAN segment. A 24-bit virtual network identifier (VNID) uniquely identifies the VXLAN segment. This enables having the same MAC frames across multiple VXLAN segments without traffic crossover. Multicast in VXLAN is implemented as Layer 3 multicast, in which endpoints subscribe to groups.

Design Details of EVPN and VXLAN

In Contrail Release 1.03 and later, EVPN is enabled by default. The supported forwarding modes include:

- Fallback bridging—IPv4 traffic lookup is performed using the IP FIB. All non-IPv4 traffic is directed to a MAC FIB.
- Layer 2-only— All traffic is forwarded using a MAC FIB lookup.

You can configure the forwarding mode individually on each virtual network.

EVPN is used to share MAC addresses across different control planes in both forwarding models. The result of a MAC address lookup is a next hop, which, similar to IP forwarding, points to a local virtual machine or a tunnel to reach the virtual machine on a remote server. The tunnel encapsulation methods supported for EVPN are MPLSoGRE, MPLSoUDP, and VXLAN. The encapsulation method selected is based on a user-configured priority.

In VXLAN, the VNID is assigned uniquely for every virtual network carried in the VXLAN header. The VNID uniquely identifies a virtual network. When the VXLAN header is received from the fabric at a remote server, the VNID lookup provides the VRF of the virtual machine. This VRF is used for the MAC lookup from the inner header, which then provides the destination virtual machine.

Non-IP multicast traffic uses the same multicast tree as for IP multicast (255.255.255.255). The multicast is matched against the all-broadcast prefix in the bridging table (FF:FF:FF:FF:FF:FF). VXLAN is not supported for IP/non-IP multicast traffic.

The following table summarizes the traffic and encapsulation types supported for EVPN.

		Encapsulation		
		MPLS-GRE	MPLS-UDP	VXLAN
Traffic Type	IP unicast	Yes	Yes	No

	IP-BUM	Yes	Yes	No
	non IP unicast	Yes	Yes	Yes
	non IP-BUM	Yes	Yes	No

Configuring the VXLAN Identifier Mode

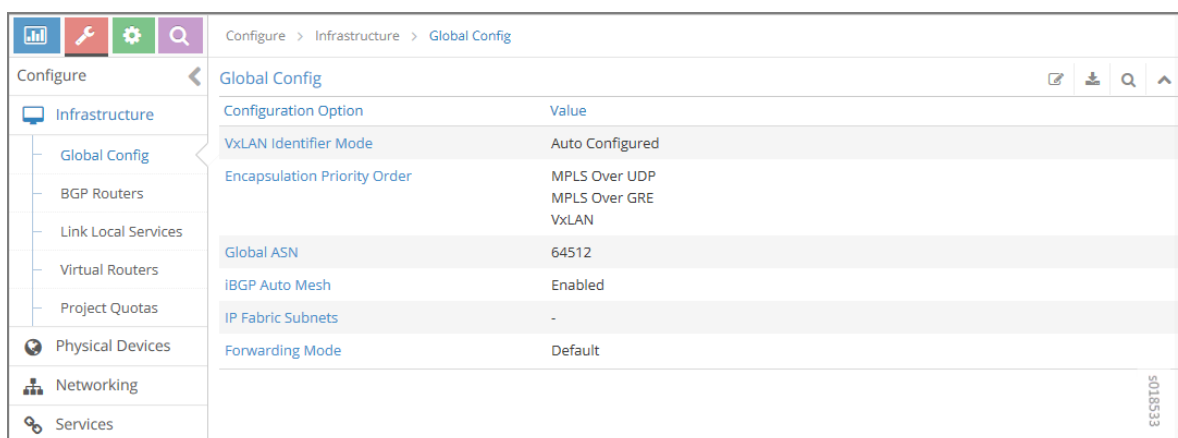
You can configure the global VXLAN identifier mode to select an auto-generated VNID or a user-generated VXLAN ID, either through the Contrail Web UI or by modifying a python file.

To configure the global VXLAN identifier mode:

1. From the Contrail Web UI, select **Configure > Infrastructure > Global Config**.

The Global Config options and values are displayed in the Global Config window.

Figure 42: Global Config Window for VXLAN ID



2. Click the edit icon



.

The Edit Global Config window is displayed as shown in [Figure 43 on page 258](#).

Figure 43: 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 44 on page 259](#).

Figure 44: Edit Network Window

IPAM	CIDR	Allocation Pools	Gateway	DNS	DHCP	+
TestProjectC5Ca5C-ipam655...	31.222.172.0/24		<input checked="" type="checkbox"/> 31.222.172.1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> -

Under the Advanced Options select the forwarding mode from the following choices:

- Select **Default** to enable the default forwarding mode.
- Select **L2 and L3** to enable IP and MAC FIB (fallback bridging).
- Select **L2 Only** to enable only MAC FIB.
- Select **L3 Only** to enable only IP.



NOTE: The full list of forwarding modes are only displayed if you change entries in the `/usr/src/contrail/contrail-web-core/config/config.global.js` file. For example:

1. To make the **L2** selection available locate the following:

```
config.network = {};
config.network.L2_enable = false;
```

2. Change the entry to the following:

```
config.network = {};
config.network.L2_enable = true;
```

3. To make the other selections available, modify the corresponding entries.
4. Save the file and quit the editor.
5. Restart the Contrail Web user interface process (webui).

Alternatively, you can use the following python provisioning command to change the forwarding mode:

```
python provisioning_forwarding_mode --project_fq_name 'defaultdomain: admin' --vn_name vn1 --forwarding_mode <
12_13| 12 >
```

Options:

12_13 = Enable IP FIB and MAC FIB (fallback bridging)

12 = Enable MAC FIB only (Layer 2 only)

Configuring the VXLAN Identifier

The VXLAN identifier can be set only if the VXLAN network identifier mode has been set to User Configured. You can then set the VXLAN ID by either using the Contrail Web UI or by using Python commands.

To configure the global VXLAN identifier:

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

3. Click the gear icon



and select **Edit**.

The Edit Network window is displayed. Select the **Advanced Options** as shown in [Figure 45 on page 261](#).

Figure 45: Edit Network Window for VXLAN Identifier

Edit Network default-virtual-network-1

▼ Advanced Options

Admin State: Up

☐ Shared ☐ External

DNS Servers: DNS Servers +

Forwarding Mode: L2 and L3

VxLAN Identifier: 0-1048575

☐ Allow Transit

☐ Flood unknown unicast

☐ Extend To Physical Router(s)

Cancel Save

4. Type the VXLAN identifier.

5. Click **Save**.

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

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

Configuring Encapsulation Methods

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

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

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

To configure the global VXLAN identifier mode:

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



.

The Edit Global Config window is displayed as shown in [Figure 46 on page 263](#).

Figure 46: Edit Global Config Window for Encapsulation Priority Order

Edit Global Config

▼ **Forwarding Options**

Forwarding Mode: Default

VxLAN Identifier Mode: ☐ Auto Configured ☒ User Configured

Encapsulation Priority Order +

MPLS Over UDP + -

MPLS Over GRE + -

VxLAN + -

▼ **BGP Options**

Global ASN: 64512

Cancel Save

5018508

Under Encapsulation Priority Order select one of the following:

- MPLS over UDP
- MPLS over GRE
- VxLAN

Click the + plus symbol to the right of the first priority to add a second priority or third priority.

Use the following procedure to change the default encapsulation method to VXLAN by editing the `encap.py` file.



NOTE: VXLAN is *only* supported for EVPN unicast. It is not supported for IP traffic or multicast traffic. VXLAN priority and presence in the `encap.py` file or configured in the Web UI is ignored for traffic not supported by VXLAN.

To set the priority of encapsulation methods to VXLAN:

1. Modify the **encap.py** file found in the **/opt/contrail/utils/** directory.

The default encapsulation line is:

```
encap_obj=EncapsulationPrioritiesType(encapsulation=['MPLSoUDP', 'MPLSoGRE'])
```

Modify the line to:

```
encap_obj=EncapsulationPrioritiesType(encapsulation=['VXLAN', 'MPLSoUDP', 'MPLSoGRE'])
```

2. After the status is modified, execute the following script:

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

The configuration is applied globally for all virtual networks.

Example of Deploying a Multi-Tier Web Application Using Contrail

IN THIS CHAPTER

- [Example: Deploying a Multi-Tier Web Application | 265](#)
- [Sample Network Configuration for Devices for Simple Tiered Web Application | 273](#)

Example: Deploying a Multi-Tier Web Application

IN THIS SECTION

- [Multi-Tier Web Application Overview | 265](#)
- [Example: Setting Up Virtual Networks for a Simple Tiered Web Application | 266](#)
- [Verifying the Multi-Tier Web Application | 269](#)
- [Sample Addressing Scheme for Simple Tiered Web Application | 269](#)
- [Sample Physical Topology for Simple Tiered Web Application | 270](#)
- [Sample Physical Topology Addressing | 271](#)

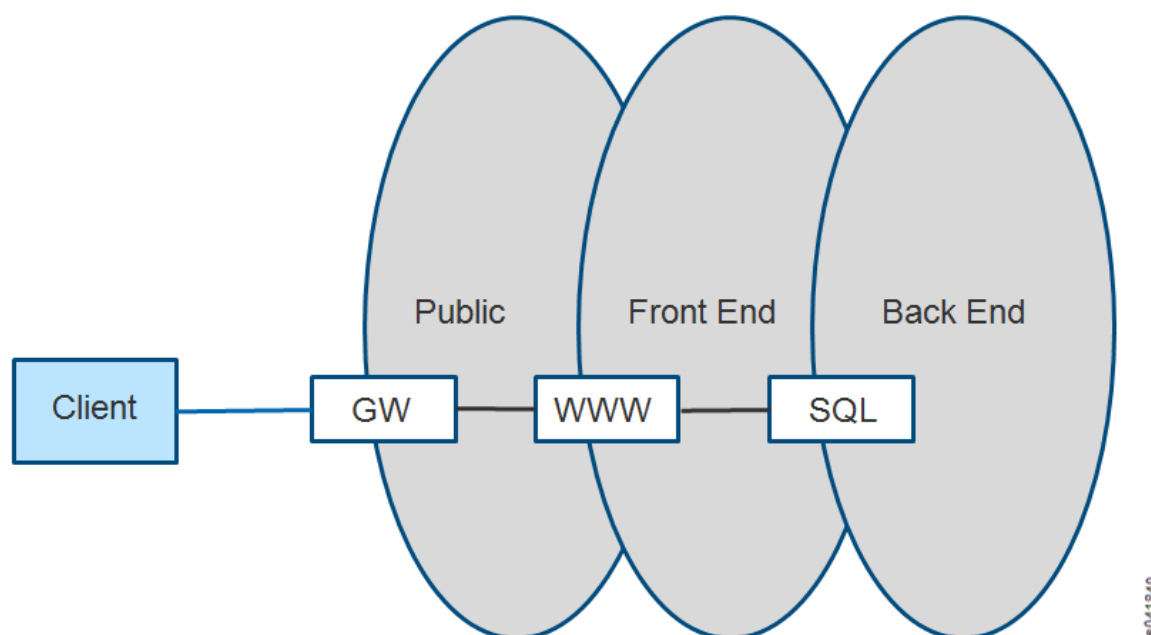
Multi-Tier Web Application Overview

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 47 on page 266](#).

Figure 47: 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 232](#).

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 238](#) or ["Creating a Virtual Network with Juniper Networks Contrail" on page 234](#).

- 3. Create a floating IP pool named **public_pool** for the **public** network within the **demo** project; see [Figure 48 on page 268](#).

Figure 48: Create Floating IP Pool

Edit Network public

Network Name

public

Network Policy(s)

Select Policies...

Address Management

default-network... ▾

xxx.xxx.xxx.xxx/xx

+ -

IPAM	IP Block
default-network-ipam	10.84.41.0/24

Floating IP Pools

public_pool

demo ×

+ -

Pool Name

admin

Cancel

Save

4. Allocate the floating IP pool **public_pool** to the **demo** project; see [Figure 49 on page 268](#).

Figure 49: Allocate Floating IP

Allocate Floating IP

Floating IP Pool

public:public_pool ▾

Cancel

Save

5. Verify that the floating IP pool has been allocated; see **Configure > Networking > Allocate Floating IPs**.
6. Create a policy that allows any host to talk to any host using any IP address, protocol, and port, and apply this policy between the **frontend** network and the **backend** network.
This now allows communication between the web servers in the front-end network and the database servers in the back-end network.
7. Launch the virtual machine instances that represent the web server and the database server.



NOTE: Your installation might not include the virtual machines needed for the web server and the database server. Contact your account team if you need to download the VMs for this setup.

On the **Instances** tab for this project, select **Launch Instance** and for each instance that you launch, complete the fields to make the following associations:

- Web server VM: select **frontend** network and the policy created to allow communication between **frontend** and **backend** networks. Apply the floating IP address pool to the web server.
- Database server VM: select **backend** network and the policy created to allow communication between **frontend** and **backend** networks.

Verifying the Multi-Tier Web Application

Verify your web setup.

- To demonstrate this web application setup, go to the client machine, open a browser, and navigate to the address in the **public** network that is assigned to the web server in the **frontend** network.
The result will display the Contrail interface with various data populated, verifying that the web server is communicating with the database server in the **backend** network and retrieving data.

The client machine only has access to the public IP address. Attempts to browse to any of the addresses assigned to the **frontend** network or to the **backend** network should fail.

Sample Addressing Scheme for Simple Tiered Web Application

Use the information in [Table 20 on page 270](#) as a guide for addressing devices in the simple tiered web example.

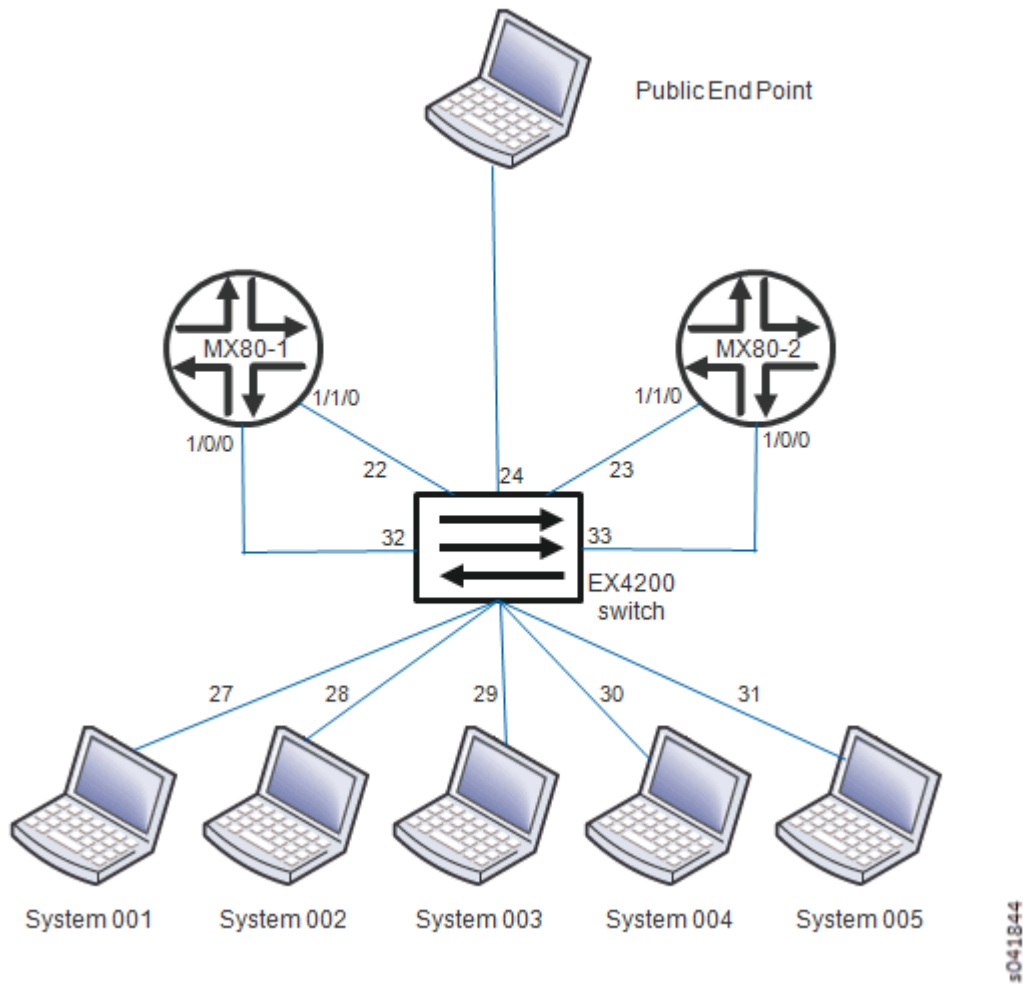
Table 20: Sample Addressing Scheme for Example

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 50 on page 271 provides a guideline diagram for the physical topology for the simple tiered web application example.

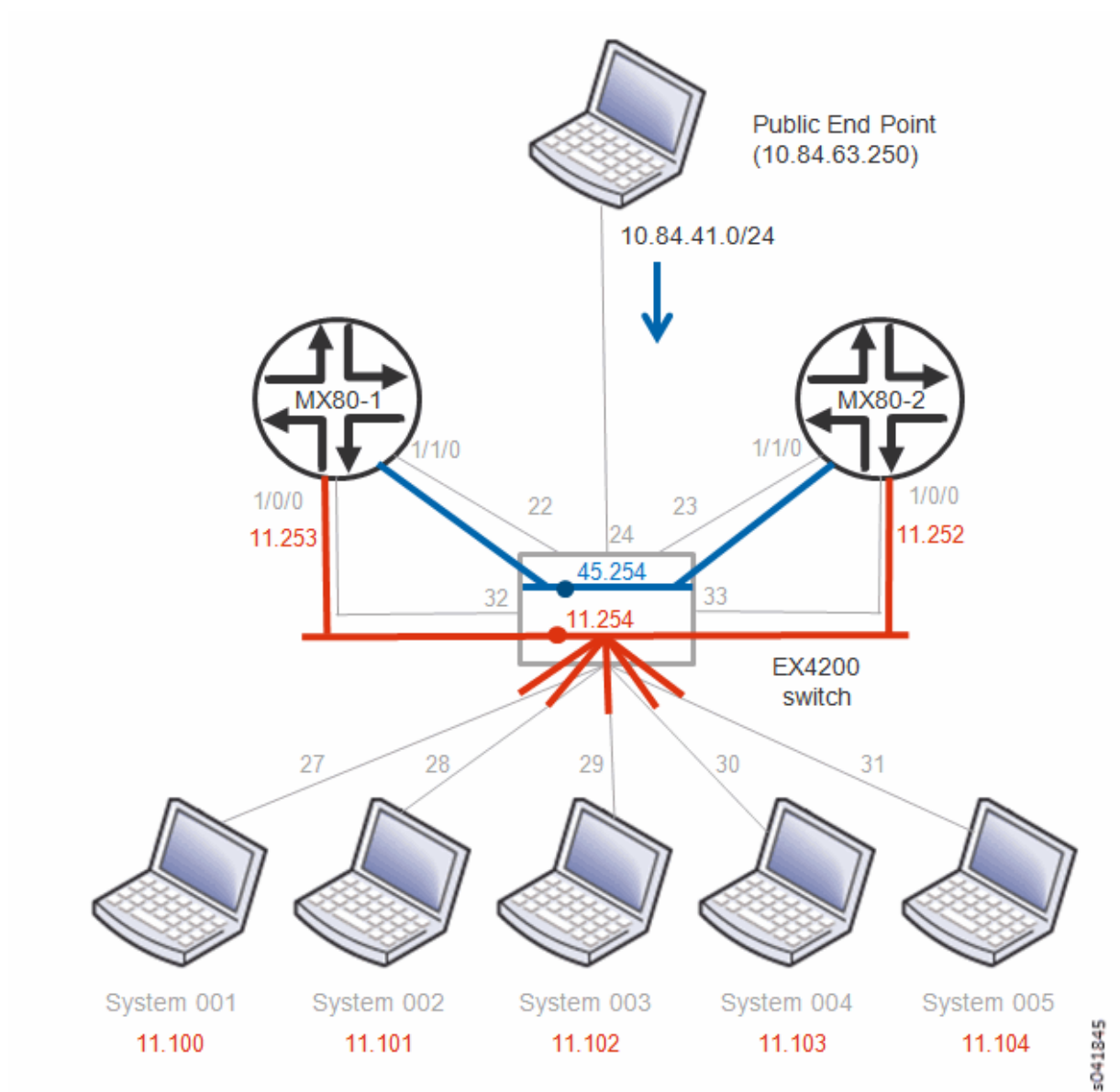
Figure 50: Sample Physical Topology for Simple Tiered Web Application



Sample Physical Topology Addressing

Figure 51 on page 272 provides a guideline diagram for addressing the physical topology for the simple tiered web application example.

Figure 51: Sample Physical Topology Addressing



SEE ALSO

[Sample Network Configuration for Devices for Simple Tiered Web Application](#) | 273

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 265](#). Configurations are shown for Juniper Networks devices: two MX80s and one EX4200.

MX80-1 Configuration

```
version 12.2R1.3;
system {
    root-authentication {
        encrypted-password "xxxxxxxxx"; ## SECRET-DATA
    }
    services {
        ssh {
            root-login allow;
        }
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
    }
}
chassis {
    fpc 1 {
        pic 0 {
            tunnel-services;
        }
    }
}
interfaces {
    ge-1/0/0 {
        unit 0 {
            family inet {
                address 10.84.11.253/24;
            }
        }
    }
}
```

```

    }
  }
}
ge-1/1/0 {
  description "IP Fabric interface";
  unit 0 {
    family inet {
      address 10.84.45.1/24;
    }
  }
}
lo0 {
  unit 0 {
    family inet {
      address 127.0.0.1/32;
    }
  }
}
}
routing-options {
  static {
    route 0.0.0.0/0 next-hop 10.84.45.254;
  }
  route-distinguisher-id 10.84.11.253;
  autonomous-system 64512;
  dynamic-tunnels {
    setup1 {
      source-address 10.84.11.253;
      gre;
      destination-networks {
        10.84.11.0/24;
      }
    }
  }
}
}
protocols {
  bgp {
    group mx {
      type internal;
      local-address 10.84.11.253;
      family inet-vpn {
        unicast;
      }
    }
  }
}

```

```

        neighbor 10.84.11.252;
    }
    group contrail-controller {
        type internal;
        local-address 10.84.11.253;
        family inet-vpn {
            unicast;
        }
        neighbor 10.84.11.101;
        neighbor 10.84.11.102;
    }
}
}
routing-instances {
    customer-public {
        instance-type vrf;
        interface ge-1/1/0.0;
        vrf-target target:64512:10000;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 10.84.45.254;
            }
        }
    }
}
}
}

```

MX80-2 Configuration

```

version 12.2R1.3;
system {
    root-authentication {
        encrypted-password "xxxxxxxx"; ## SECRET-DATA
    }
    services {
        ssh {
            root-login allow;
        }
    }
    syslog {
        user * {
            any emergency;

```

```

    }
    file messages {
        any notice;
        authorization info;
    }
}
chassis {
    fpc 1 {
        pic 0 {
            tunnel-services;
        }
    }
}
interfaces {
    ge-1/0/0 {
        unit 0 {
            family inet {
                address 10.84.11.252/24;
            }
        }
    }
    ge-1/1/0 {
        description "IP Fabric interface";
        unit 0 {
            family inet {
                address 10.84.45.2/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 127.0.0.1/32;
            }
        }
    }
}
routing-options {
    static {
        route 0.0.0.0/0 next-hop 10.84.45.254;
    }
    route-distinguisher-id 10.84.11.252;
}

```

```

autonomous-system 64512;
dynamic-tunnels {
    setup1 {
        source-address 10.84.11.252;
        gre;
        destination-networks {
            10.84.11.0/24;
        }
    }
}
}
protocols {
    bgp {
        group mx {
            type internal;
            local-address 10.84.11.252;
            family inet-vpn {
                unicast;
            }
            neighbor 10.84.11.253;
        }
        group contrail-controller {
            type internal;
            local-address 10.84.11.252;
            family inet-vpn {
                unicast;
            }
            neighbor 10.84.11.101;
            neighbor 10.84.11.102;
        }
    }
}
}
routing-instances {
    customer-public {
        instance-type vrf;
        interface ge-1/1/0.0;
        vrf-target target:64512:10000;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 10.84.45.254;
            }
        }
    }
}

```



```

    }
  }
}

```

EX4200 Configuration

```

system {
  host-name EX4200;
  time-zone America/Los_Angeles;
  root-authentication {
    encrypted-password "xxxxxxxxxxxx"; ## SECRET-DATA
  }
  login {
    class read {
      permissions [ clear interface view view-configuration ];
    }
    user admin {
      uid 2000;
      class super-user;
      authentication {
        encrypted-password "xxxxxxxxxxxx"; ## SECRET-DATA
      }
    }
    user user1 {
      uid 2002;
      class read;
      authentication {
        encrypted-password "xxxxxxxxxxxx"; ## SECRET-DATA
      }
    }
  }
}
services {
  ssh {
    root-login allow;
  }
  telnet;
  netconf {
    ssh;
  }
  web-management {
    http;
  }
}

```

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

Configuring Services

IN THIS CHAPTER

- [Configuring DNS Servers | 280](#)
- [Support for Multicast | 293](#)
- [Using Static Routes with Services | 295](#)
- [Configuring Metadata Service | 300](#)

Configuring DNS Servers

IN THIS SECTION

- [DNS Overview | 280](#)
- [Defining Multiple Virtual Domain Name Servers | 281](#)
- [IPAM and Virtual DNS | 282](#)
- [DNS Record Types | 282](#)
- [Configuring DNS Using the Interface | 283](#)
- [Configuring DNS Using Scripts | 291](#)

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:

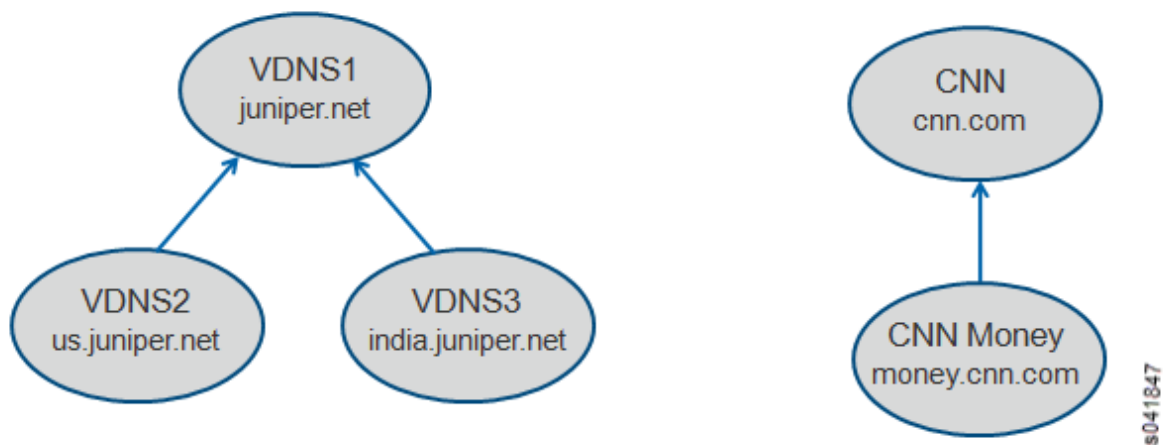
- It should be possible to configure multiple domain name servers to provide name resolution service for the virtual machines spawned in the system.
- It should be possible to configure the domain name servers to form DNS server hierarchies required by each tenant.
 - The hierarchies can be independent 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 52 on page 281](#) shows examples of virtual DNS servers defined in **default-domain**, providing the name service for the DNS domains indicated.

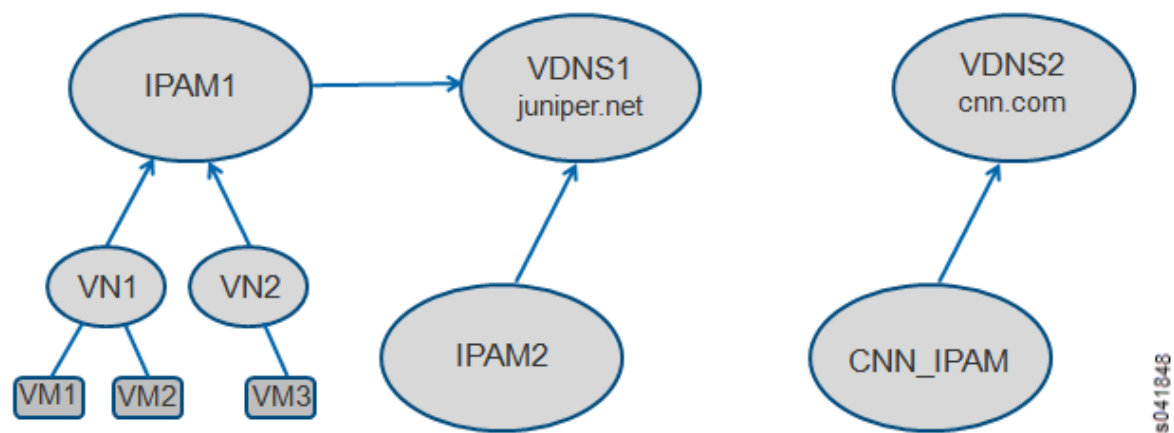
Figure 52: DNS Servers Examples



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 53 on page 282](#)

Figure 53: 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 21 on page 282](#) for descriptions of the record types.

Table 21: DNS Record Types Supported

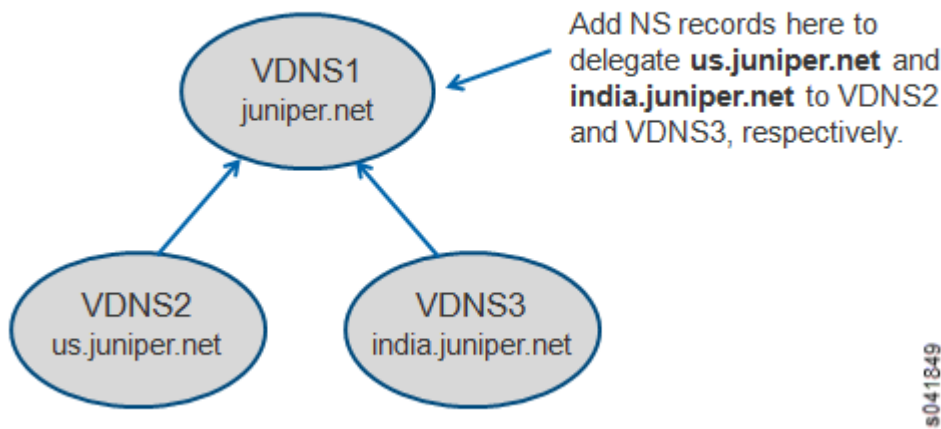
DNS Record Type	Description
A	Used for mapping hostnames to IPv4 addresses. Name refers to the name of the virtual machine, and data is the IPv4 address of the virtual machine.
CNAME	Provides an alias to a name. Name refers to the name of the virtual machine, and data is the new name (alias) for the virtual machine.

Table 21: 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 54 on page 283 shows an example usage for the DNS record type of **NS**.

Figure 54: 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 55 on page 284](#).

Figure 55: Configure DNS Records

Configure > DNS > Servers

Search

Configure DNS Records

default-domainadmin

CreateDelete

Configure Virtual DNS

Virtual DNS Name	DNS Domain Name	Next DNS Server
No Data Found		

DNS RecordsAssociated IPAMs

DNS Records of {{dnsname}}

Add RecordDelete

Name	Type : Data	TTL (secs)	Class
------	-------------	------------	-------

2. To add a new DNS server, click the **Create** button.
- Enter DNS server information in the **Add DNS** window; see [Figure 56 on page 285](#)

Figure 56: Add DNS

Create DNS Server

Server Name

Domain Name

DNS Forwarder

Enter Forwarder IP or Select a DNS Server

Record Resolution Order

Random

Time To Live

TTL (86400 sec)

Associate IPAMs

Cancel

Save

s041864

Complete the fields for the new server; see [Table 22 on page 285](#).

Table 22: 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 22: 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 57 on page 287](#).

Figure 57: Add DNS Record

Add DNS Record

Type

A (IP Address Record)

Host Name

Host Name to be resolved

IP Address

Enter an IP Address

Class

IN (Internet)

Time To Live

TTL(86400 secs)

Cancel

Save

4. Complete the fields for the new record; see [Table 23 on page 287](#).

Table 23: 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 23: 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 58 on page 288](#).

Figure 58: Associate IPAMs to DNS

The screenshot shows a window titled "Edit DNS Server" with a close button (X) in the top right corner. The window contains several input fields and a dropdown menu:

- Server Name:** A text input field containing "vdns1".
- Domain Name:** A text input field containing "juniper.net".
- DNS Forwarder:** A text input field containing "Enter Forwarder IP or Select a DNS Server" and a dropdown arrow.
- Record Resolution Order:** A dropdown menu showing "Random".
- Time To Live:** A text input field containing "86400".
- Associate IPAMs:** A text input field with a dropdown menu open, showing two options: "admin:ipam1" (highlighted) and "default-project:default-network-ipam".

At the bottom right of the window, there are two buttons: "Cancel" and "Save". A vertical text label "5041854" is visible on the right side of the window.

Complete the IPAM associations, using the field descriptions in [Table 24 on page 288](#).

Table 24: Associate IPAMs to DNS Fields

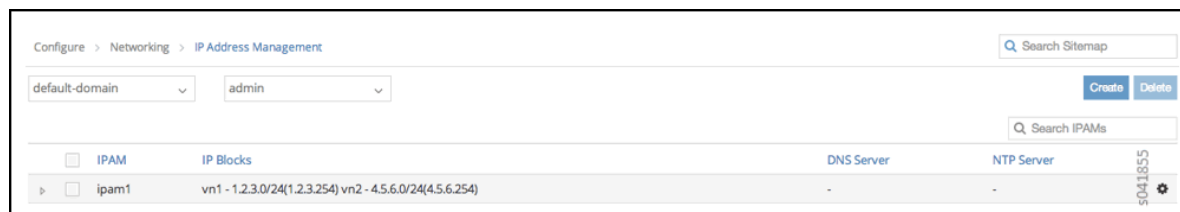
Field	Description
Associate to All IPAMs	Select this box to associate the selected DNS server to all available IPAMs.

Table 24: 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 59 on page 289](#) 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 59: 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 60 on page 290](#).

Figure 60: DNS Server

Add IP Address Management

Name

DNS Method

Virtual DNS

Default

Virtual DNS

Tenant

NTP Server IP

None

Associate IP Blocks to Networks

fip_vn

IP Block

Gateway

+ -

Network	IP Block	Gateway
---------	----------	---------

Cancel

Save

5041857

8. In the first field, select the **DNS Method** from a list (**None**, **Default DNS**, **Tenant DNS**, **Virtual DNS**; see [Table 25 on page 290](#).

Table 25: 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 25: 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

You can configure DNS by using scripts that are available in the `contrail-utils` RPM/DEB package in the `/opt/contrail/utils` directory. The scripts are copied to the `config_api_container` or `config` node when you install the `contrail-utils` RPM/DEB package. You can execute the scripts from either the `config_api` container or the `config` node. The scripts are described in [Table 26 on page 292](#).



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 26: DNS Scripts

Action	Script
Add a virtual DNS server	<p>Script: <code>add_virtual_dns.py</code></p> <p>Sample usage: <code>python add_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --name vdns1 --domain_name default-domain --dns_domain juniper.net --dyn_updates --record_order random --ttl 1200 --next_vdns default-domain:vdns2</code></p>
Delete a virtual DNS server	<p>Script: <code>del_virtual_dns_record.py</code></p> <p>Sample usage: <code>python del_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --fq_name default-domain:vdns1</code></p>
Add a DNS record	<p>Script: <code>add_virtual_dns_record.py</code></p> <p>Sample usage: <code>python add_virtual_dns_record.py --api_server_ip 10.204.216.21 --api_server_port 8082 --name rec1 --vdns_fqname default-domain:vdns1 --rec_name one --rec_type A --rec_class IN --rec_data 1.2.3.4 --rec_ttl 2400</code></p>
Delete a DNS record	<p>Script: <code>del_virtual_dns_record.py</code></p> <p>Sample usage: <code>python del_virtual_dns_record.py --api_server_ip 10.204.216.21 --api_server_port 8082 --fq_name default-domain:vdns1:rec1</code></p>
Associate a virtual DNS server with an IPAM	<p>Script: <code>associate_virtual_dns.py</code></p> <p>Sample usage: <code>python associate_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --ipam_fqname default-domain:demo:ipam1 --vdns_fqname default-domain:vdns1</code></p>
Disassociate a virtual DNS server with an IPAM	<p>Script: <code>disassociate_virtual_dns.py</code></p> <p>Sample usage: <code>python disassociate_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --ipam_fqname default-domain:demo:ipam1 --vdns_fqname default-domain:vdns1</code></p>

Support for Multicast

IN THIS SECTION

- Subnet Broadcast | 293
- All-Broadcast/Limited-Broadcast and Link-Local Multicast | 294
- Host Broadcast | 295

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 | 295](#)
- [Configuring Static Routes on a Service Instance | 296](#)
- [Configuring Static Routes on Service Instance Interfaces | 297](#)
- [Configuring Static Routes as Host Routes | 299](#)

Static Routes for Service Instances

Static routes are manually configured in a network to initiate data transmission between two networks. The traffic generated by a set of devices in a network is directed through a static route, which ensures an efficient flow of traffic towards a specific destination address.

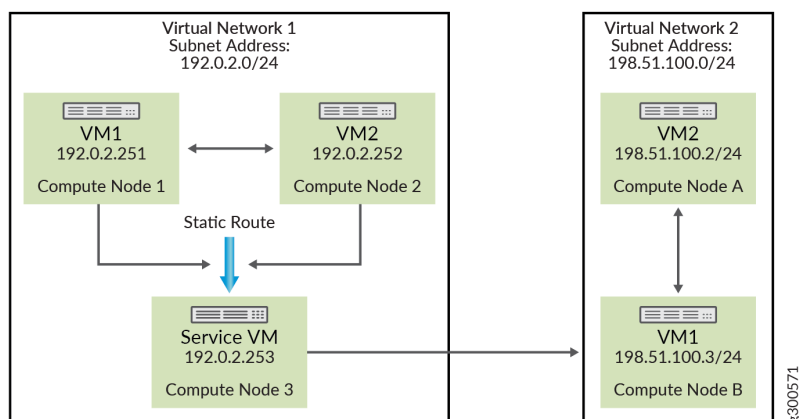
Static routes are used in small networks and networks with simple architecture as a route for direct communication between two networks. Static routes cannot operate in large, dynamic networks due to the frequent change in networks and routes. In such cases, we can use BGPaaS in Contrail for dynamic routing updates. A static route can also be configured as the Default route (a gateway of last resort), which the routers use to send data packets with unknown destination address.

In a virtual network, you can configure static routes towards a service virtual machine (VM) interface to direct all network traffic through the service virtual machine. The configured static routes are advertised to other nodes through BGP, which ensures that traffic is directed through specific virtual machine.

In [Figure 61 on page 296](#), there are three VMs in a virtual network (VN1) with subnet address 192.0.2.0/24. The virtual machines are VM1 (192.0.2.251), VM2 (192.0.2.252), and a Service VM (192.0.2.253). When VM1 or VM2 in VN1 generates traffic targeted towards another virtual network (VN2) with subnet address 198.51.100.0/24, you need to configure a static route. You can configure a static route towards the Service VM interface to direct the traffic generated by VM1 and VM2 destined

towards VN2. Once configured, all traffic targeted towards VN2 from VN1 is directed through the static route (192.0.2.253).

Figure 61: Static Route in a Virtual Network



Configuring Static Routes on a Service Instance

To configure static routes on a service instance, first enable the static route option in the service template to be used for the service instance.

To enable the static route option in a service template:

1. Go to **Configure > Services > Service Templates** and click **Create**.
2. At **Add Service Template**, complete the fields for **Name**, **Service Mode**, and **Image Name**.
3. Select the **Interface Types** to use for the template, then for each interface type that might have a static route configured, click the check box under the **Static Routes** column to enable the static route option for that interface.

The following figure shows a service template in which the left and right interfaces of service instances have the static routes option enabled. Now a user can configure a static route on a corresponding interface on a service instance that is based on the service template shown.

Add Service Template
✕

Name

Service Mode

Image Name

Interface Types	Shared IP	Static Routes	+
Management <input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>	+ -
Left <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	+ -
Right <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	+ -

▶ Advanced options

Cancel Save

s041915

Configuring Static Routes on Service Instance Interfaces

To configure static routes on a service instance interface:

1. Go to **Configure > Services > Service Instances** and click **Create**.
2. At **Create Service Instances**, complete the fields for **Instance Name** and **Services Template**.
3. Select the virtual network for each of the interfaces
4. Click the **Static Routes** dropdown menu under each interface field for which the static routes option is enabled to open the **Static Routes** menu and configure the static routes in the fields provided.



NOTE: If the **Auto Configured** option is selected, traffic destined to the static route subnet is load balanced across service instances.

The following figure shows a configuration to apply a service instance between VN1 (192.0.2.0/24) and VN2 (198.51.100.0/24). The left interface of the service instance is configured with VN1 and the right interface is configured to be VN2 (198.51.100.0/24). The static route 192.0.2.253 is configured on the left interface, so that all traffic from VN1 that is destined to VN2 reaches the left interface of the service instance.

Create Service Instances

Instance Name:

Services Template:

Interface 1:

Interface 2:

▼ Static Routes

Prefix	Next hop	
<input type="text" value="11.1.1.0/24"/>	Interface 2	+ -

Interface 3:

▼ Static Routes

Cancel Save

5041916

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

Create Service Instances

Interface 2: Left, vn1

▼ Static Routes

Prefix	Next hop	+	-
11.1.1.0/24	Interface 2	+	-

Interface 3: Right, vn2

▼ Static Routes

Prefix	Next hop	+	-
10.1.1.0/24	Interface 3	+	-

Cancel Save

s041917

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

Configuring Static Routes as Host Routes

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

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

To configure static routes as host routes:

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

An example is shown in the following figure.

Create Network

Address Management

ipam1

IP Block

Gateway

+

-

IPAM	IP Block	Gateway
ipam1	1.2.3.0/24	1.2.3.254

Route Targets

Floating IP Pools

Host Routes

IPAM	Route Prefix	+	-
ipam1	1.1.1.0/24	+	-
ipam1	2.2.2.0/24	+	-

Cancel

Save

Configuring Metadata Service

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

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

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

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

- `linklocal-service-name` = `metadata`
- `linklocal-service-ip` = 169.254.169.254

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

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

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


Configuring Service Chaining

IN THIS CHAPTER

- [Service Chaining | 302](#)
- [Service Chaining MX Series Configuration | 306](#)
- [ECMP Load Balancing in the Service Chain | 308](#)
- [Customized Hash Field Selection for ECMP Load Balancing | 309](#)
- [Using the Contrail Heat Template | 314](#)
- [Service Chain Route Reorigination | 319](#)
- [Service Instance Health Checks | 341](#)

Service Chaining

IN THIS SECTION

- [Service Chaining Basics | 302](#)
- [Service Chaining Configuration Elements | 304](#)

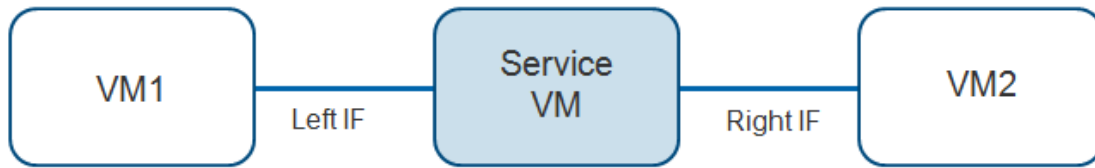
Contrail Controller supports chaining of various Layer 2 through Layer 7 services such as firewall, NAT, IDP, and so on.

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.

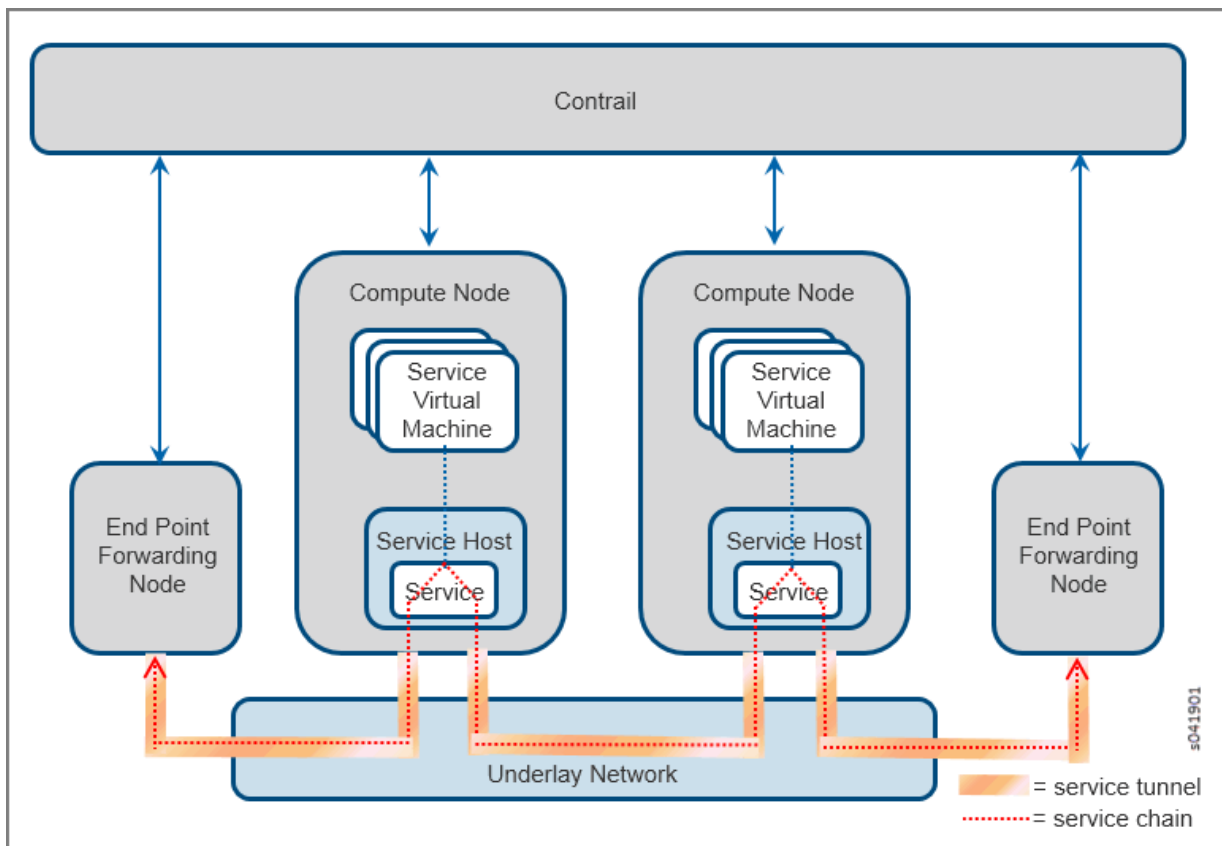
[Figure 62 on page 303](#) shows the basic service chain schema, with a single service. The service VM spawns the service, using the convention of left interface (left IF) and right interface (right IF). Multiple services can also be chained together.

Figure 62: 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 63 on page 303](#) 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 63: 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 | 349](#)

[Example: Creating an In-Network-NAT Service Chain by Using Contrail Command | 357](#)

[Example: Creating a Transparent Service Chain by Using Contrail Command | 366](#)

[ECMP Load Balancing in the Service Chain | 308](#)

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;
}
}

```

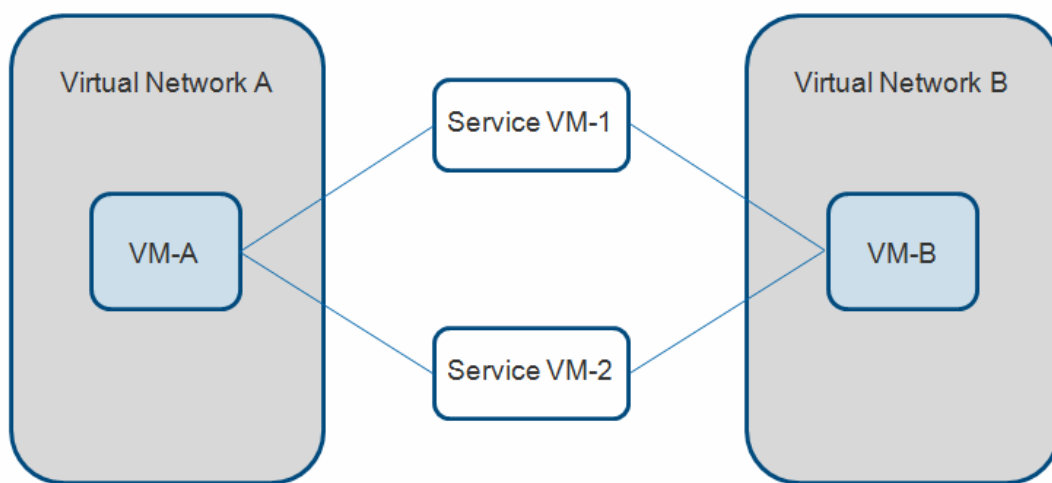
4. The final step is to add target:100:10000 to the public virtual network and target:100:20000 to the enterprise virtual network, using the Contrail Juniper Networks interface.

A full MX Series router configuration for Contrail can be seen in ["Sample Network Configuration for Devices for Simple Tiered Web Application" on page 273.](#)

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 64 on page 308](#), 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 64: Load Balancing a Service Chain



s041830

The following are the major features of load balancing in the service chain:

- Load balancing can be configured at every level of the service chain.
- Load balancing is supported in routed and bridged service chain modes.
- Load balancing can be used to achieve high availability—if a service VM goes down, the traffic passing through that service VM can be distributed through another service VM.
- A load balanced traffic stream always follows the same path through the chain of service VM.

RELATED DOCUMENTATION

[Service Chaining | 302](#)

[Customized Hash Field Selection for ECMP Load Balancing | 309](#)

Customized Hash Field Selection for ECMP Load Balancing

IN THIS SECTION

- [Overview: Custom Hash Feature | 309](#)
- [Using ECMP Hash Fields Selection | 311](#)
- [Sample Flows | 312](#)

Overview: Custom Hash Feature

Starting with Contrail Release 3.0, it is possible to configure the set of fields used to hash upon during equal-cost multipath (ECMP) load balancing.

Earlier versions of Contrail had this set of fields fixed to the standard 5-tuple set of: source L3 address, destination L3 address, L4 protocol, L4 SourcePort, and L4 DestinationPort.

With the custom hash feature, users can configure an exact subset of fields to hash upon when choosing the forwarding path among a set of eligible ECMP candidates.

The custom hash configuration can be applied in the following ways:

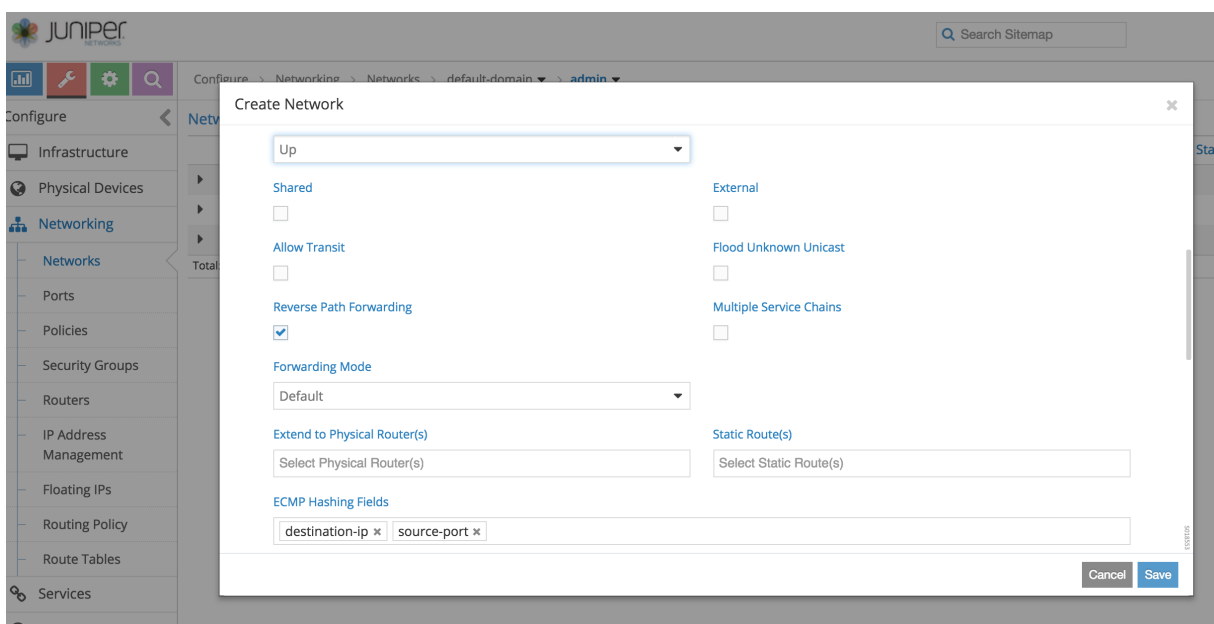
- globally

- per virtual network (VN)
- per virtual network interface (VMI)

VMI configurations take precedence over VN configurations, and VN configurations take precedence over global level configuration (if present).

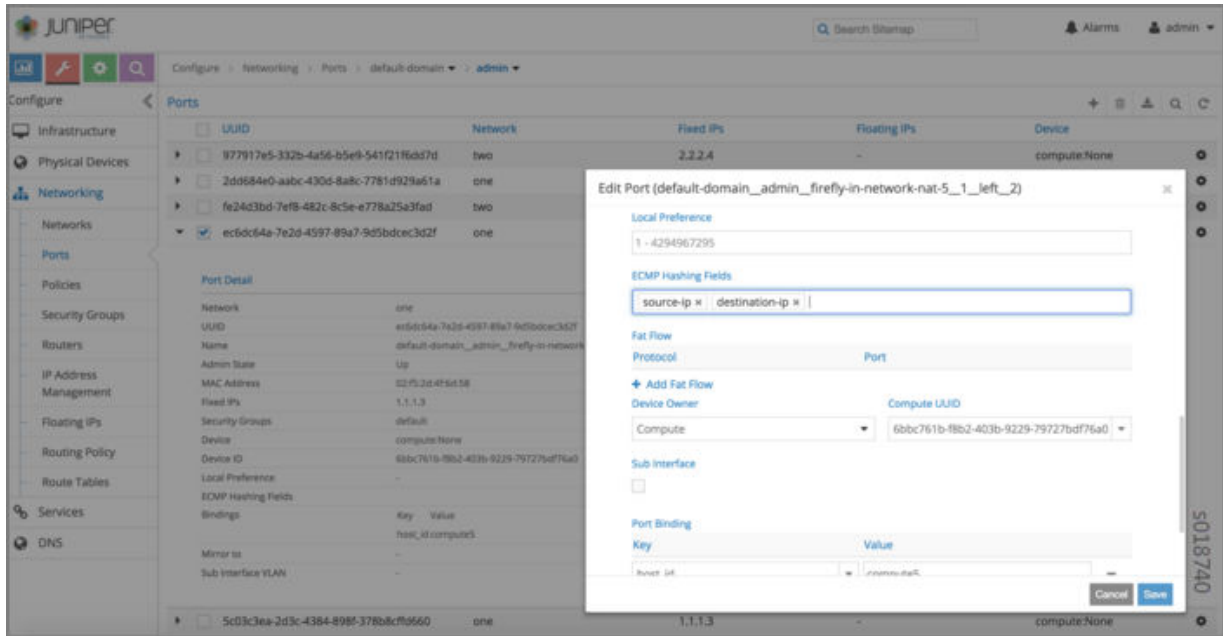
Custom hash is useful whenever packets originating from a particular source and addressed to a particular destination must go through the same set of service instances during transit. This might be required if source, destination, or transit nodes maintain a certain state based on the flow, and the state behavior could also be used for subsequent new flows, between the same pair of source and destination addresses. In such cases, subsequent flows must follow the same set of service nodes followed by the initial flow.

You can use the Contrail UI to identify specific fields in the network upon which to hash at the **Configure > Networking > Network, Create Network** window, in the **ECMP Hashing Fields** section as shown in the following figure.



If the hashing fields are configured for a virtual network, all traffic destined to that VN will be subject to the customized hash field selection during forwarding over ECMP paths by vRouters. This may not be desirable in all cases, as it could potentially skew all traffic to the destination network over a smaller set of paths across the IP fabric.

A more practical scenario is one in which flows between a source and destination must go through the same service instance in between, where one could configure customized ECMP fields for the virtual machine interface (VMI) of the service instance. Then, each service chain route originating from that VMI would get the desired ECMP field selection applied as its path attribute, and eventually get propagated to the ingress vRouter node. See the following example.



Using ECMP Hash Fields Selection

Custom hash fields selection is most useful in scenarios where multiple ECMP paths exist for a destination. Typically, the multiple ECMP paths point to ingress service instance nodes, which could be running anywhere in the Contrail cloud.

Configuring ECMP Hash Fields Over Service Chains

Use the following steps to create customized hash fields with ECMP over service chains.

1. Create the virtual networks needed to interconnect using service chaining, with ECMP load-balancing.
2. Create a service template and enable scaling.
3. Create a service instance, and using the service template, configure by selecting:
 - the desired number of instances for scale-out
 - the left and right virtual network to connect
 - the shared address space, to make sure that instantiated services come up with the same IP address for left and right, respectively

This configuration enables ECMP among all those service instances during forwarding.

4. Create a policy, then select the service instance previously created and apply the policy to the desired VMIs or VNs.

5. After the service VMs are instantiated, the ports of the left and right interfaces are available for further configuration. At the Contrail UI Ports section under Networking, select the left port (VMI) of the service instance and apply the desired ECMP hash field configuration.



NOTE: Currently the ECMP field selection configuration for the service instance left or right interface must be applied by using the Ports (VMIs) section under Networking and explicitly configuring the ECMP fields selection for each of the instantiated service instances' VMIs. This must be done for all service interfaces of the group, to ensure the end result is as expected, because the load balance attribute of only the best path is carried over to the ingress vRouter. If the load balance attribute is not configured, it is not propagated to the ingress vRouter, even if other paths have that configuration.

When the configuration is finished, the vRouters get programmed with routing tables with the ECMP paths to the various service instances. The vRouters are also programmed with the desired ECMP hash fields to be used during load balancing of the traffic.

Sample Flows

This section provides sample flows with and without ECMP custom hash field selection.

Sample Traffic Flow Path Without Custom ECMP Hash Fields

The following is an example of a traffic flow path without using a customized ECMP hash fields selection configuration. The flow is configured with standard 5-tuple flow fields.

```
tcpdump -i eth0 'port 1023 and tcp[tcpflags] & (tcp-syn) != 0 and tcp[tcpflags] & (tcp-ack) == 0'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
14:55:10.115122 IP 2.2.2.5.18337 > 2.2.2.100.1023: Flags [S], seq 2276852196, win 29200, options
[mss 1398,sackOK,TS val 25208882 ecr 0,nop,wscale 7], length 0
14:55:10.132753 IP 2.2.2.4.21193 > 2.2.2.100.1023: Flags [S], seq 4161487314, win 29200, options
[mss 1398,sackOK,TS val 25208886 ecr 0,nop,wscale 7], length 0
14:55:10.152053 IP 2.2.2.5.24230 > 2.2.2.100.1023: Flags [S], seq 2466454857, win 29200, options
[mss 1398,sackOK,TS val 25208892 ecr 0,nop,wscale 7], length 0
14:55:11.146029 IP 2.2.2.5.24230 > 2.2.2.100.1023: Flags [S], seq 2466454857, win 29200, options
[mss 1398,sackOK,TS val 25209142 ecr 0,nop,wscale 7], length 0
14:55:13.147616 IP 2.2.2.5.24230 > 2.2.2.100.1023: Flags [S], seq 2466454857, win 29200, options
[mss 1398,sackOK,TS val 25209643 ecr 0,nop,wscale 7], length 0
14:55:13.164367 IP 2.2.2.3.25582 > 2.2.2.100.1023: Flags [S], seq 2259034580, win 29200, options
[mss 1398,sackOK,TS val 25209644 ecr 0,nop,wscale 7], length 0
14:55:13.179939 IP 2.2.2.5.24895 > 2.2.2.100.1023: Flags [S], seq 2174031724, win 29200, options
```

```
[mss 1398,sackOK,TS val 25209648 ecr 0,nop,wscale 7], length 0
14:55:14.168282 IP 2.2.2.5.24895 > 2.2.2.100.1023: Flags [S], seq 2174031724, win 29200, options
[mss 1398,sackOK,TS val 25209898 ecr 0,nop,wscale 7], length 0
14:55:16.172384 IP 2.2.2.5.24895 > 2.2.2.100.1023: Flags [S], seq 2174031724, win 29200, options
[mss 1398,sackOK,TS val 25210399 ecr 0,nop,wscale 7], length 0
14:55:16.189864 IP 2.2.2.5.22952 > 2.2.2.100.1023: Flags [S], seq 3099816842, win 29200, options
[mss 1398,sackOK,TS val 25210401 ecr 0,nop,wscale 7], length 0
14:55:16.205142 IP 2.2.2.4.16487 > 2.2.2.100.1023: Flags [S], seq 3961114202, win 29200, options
[mss 1398,sackOK,TS val 25210405 ecr 0,nop,wscale 7], length 0
14:55:17.196763 IP 2.2.2.4.16487 > 2.2.2.100.1023: Flags [S], seq 3961114202, win 29200, options
[mss 1398,sackOK,TS val 25210655 ecr 0,nop,wscale 7], length 0
14:55:19.200623 IP 2.2.2.4.16487 > 2.2.2.100.1023: Flags [S], seq 3961114202, win 29200, options
[mss 1398,sackOK,TS val 25211156 ecr 0,nop,wscale 7], length 0
14:55:19.215809 IP 2.2.2.3.18914 > 2.2.2.100.1023: Flags [S], seq 3157557440, win 29200, options
[mss 1398,sackOK,TS val 25211158 ecr 0,nop,wscale 7], length 0
14:55:19.228405 IP 2.2.2.7.15569 > 2.2.2.100.1023: Flags [S], seq 3850648420, win 29200, options
[mss 1398,sackOK,TS val 25211161 ecr 0,nop,wscale 7], length 0
14:55:20.223482 IP 2.2.2.7.15569 > 2.2.2.100.1023: Flags [S], seq 3850648420, win 29200, options
[mss 1398,sackOK,TS val 25211412 ecr 0,nop,wscale 7], length 0
14:55:22.232068 IP 2.2.2.7.15569 > 2.2.2.100.1023: Flags [S], seq 3850648420, win 29200, options
[mss 1398,sackOK,TS val 25211913 ecr 0,nop,wscale 7], length 0
14:55:22.247325 IP 2.2.2.4.28388 > 2.2.2.100.1023: Flags [S], seq 3609240658, win 29200, options
[mss 1398,sackOK,TS val 25211915 ecr 0,nop,wscale 7], length 0
```

Sample Traffic Flow Path With Custom ECMP Hash Fields

The following is an example of a traffic flow path using a customized ECMP hash fields selection configuration, for source-ip and destination-ip only.

```
tcpdump -i eth0 'port 1023 and tcp[tcpflags] & (tcp-syn) != 0 and tcp[tcpflags] & (tcp-ack) == 0'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
15:57:18.680853 IP 2.2.2.4.21718 > 2.2.2.100.1023: Flags [S], seq 2052086108, win 29200, options
[mss 1398,sackOK,TS val 26141024 ecr 0,nop,wscale 7], length 0
15:57:18.696114 IP 2.2.2.4.13585 > 2.2.2.100.1023: Flags [S], seq 2039627277, win 29200, options
[mss 1398,sackOK,TS val 26141028 ecr 0,nop,wscale 7], length 0
15:57:18.714846 IP 2.2.2.4.16414 > 2.2.2.100.1023: Flags [S], seq 3252526560, win 29200, options
[mss 1398,sackOK,TS val 26141033 ecr 0,nop,wscale 7], length 0
15:57:18.731281 IP 2.2.2.4.32499 > 2.2.2.100.1023: Flags [S], seq 1389133175, win 29200, options
[mss 1398,sackOK,TS val 26141037 ecr 0,nop,wscale 7], length 0
15:57:18.747051 IP 2.2.2.4.6081 > 2.2.2.100.1023: Flags [S], seq 427936299, win 29200, options
```

```
[mss 1398,sackOK,TS val 26141041 ecr 0,nop,wscale 7], length 0
15:57:19.740204 IP 2.2.2.4.6081 > 2.2.2.100.1023: Flags [S], seq 427936299, win 29200, options
[mss 1398,sackOK,TS val 26141291 ecr 0,nop,wscale 7], length 0
15:57:21.743951 IP 2.2.2.4.6081 > 2.2.2.100.1023: Flags [S], seq 427936299, win 29200, options
[mss 1398,sackOK,TS val 26141792 ecr 0,nop,wscale 7], length 0
15:57:21.758532 IP 2.2.2.4.13800 > 2.2.2.100.1023: Flags [S], seq 3020971712, win 29200, options
[mss 1398,sackOK,TS val 26141794 ecr 0,nop,wscale 7], length 0
15:57:21.772646 IP 2.2.2.4.23894 > 2.2.2.100.1023: Flags [S], seq 3373734307, win 29200, options
[mss 1398,sackOK,TS val 26141797 ecr 0,nop,wscale 7], length 0
15:57:22.764469 IP 2.2.2.4.23894 > 2.2.2.100.1023: Flags [S], seq 3373734307, win 29200, options
[mss 1398,sackOK,TS val 26142047 ecr 0,nop,wscale 7], length 0
15:57:24.768511 IP 2.2.2.4.23894 > 2.2.2.100.1023: Flags [S], seq 3373734307, win 29200, options
[mss 1398,sackOK,TS val 26142548 ecr 0,nop,wscale 7], length 0
15:57:24.784119 IP 2.2.2.4.21858 > 2.2.2.100.1023: Flags [S], seq 2212369297, win 29200, options
[mss 1398,sackOK,TS val 26142550 ecr 0,nop,wscale 7], length 0
15:57:24.797149 IP 2.2.2.4.29440 > 2.2.2.100.1023: Flags [S], seq 2007897735, win 29200, options
[mss 1398,sackOK,TS val 26142554 ecr 0,nop,wscale 7], length 0
15:57:25.792816 IP 2.2.2.4.29440 > 2.2.2.100.1023: Flags [S], seq 2007897735, win 29200, options
[mss 1398,sackOK,TS val 26142804 ecr 0,nop,wscale 7], length 0
15:57:27.797538 IP 2.2.2.4.29440 > 2.2.2.100.1023: Flags [S], seq 2007897735, win 29200, options
[mss 1398,sackOK,TS val 26143305 ecr 0,nop,wscale 7], length 0
15:57:27.814002 IP 2.2.2.4.23452 > 2.2.2.100.1023: Flags [S], seq 1659332655, win 29200, options
[mss 1398,sackOK,TS val 26143307 ecr 0,nop,wscale 7], length 0
```

Using the Contrail Heat Template

IN THIS SECTION

- [Introduction to Heat | 315](#)
- [Heat Architecture | 315](#)
- [Support for Heat Version 2 Resources | 315](#)
- [Heat Version 2 with Service Chaining and Port Tuple Sample Workflow | 316](#)
- [Example: Creating a Service Template Using Heat | 317](#)

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 | 319](#)
- [Route Aggregation | 321](#)
- [Routing Policy | 328](#)
- [Control for Route Reorigination | 339](#)

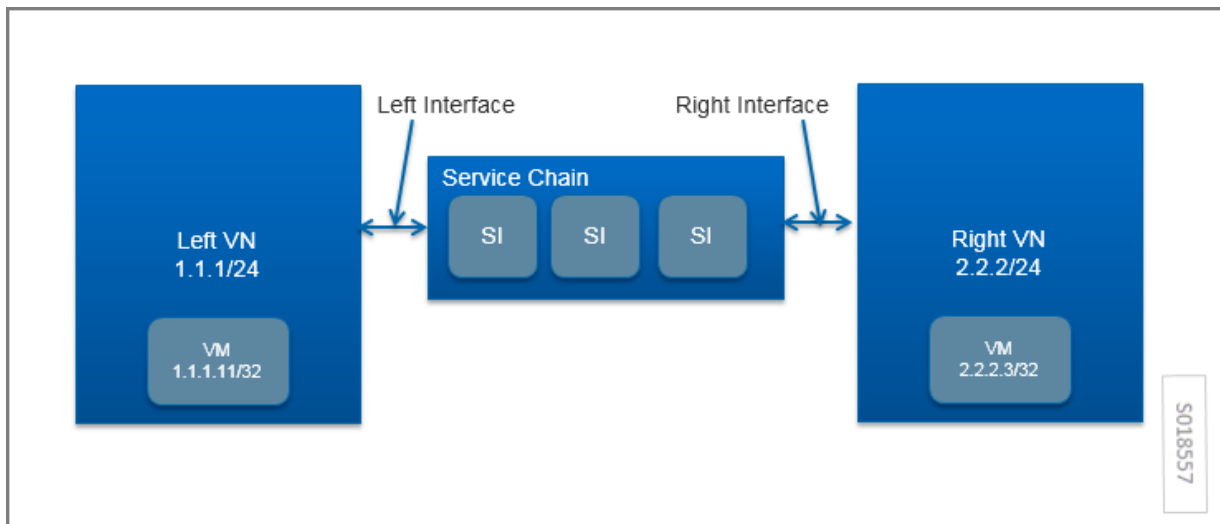
Overview: Service Chaining in Contrail

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 65 on page 320](#), 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 65: Route Reorigination



Using reorigination of routes for service chaining (for example, putting the route for the right network in the left routing table) requires the following features:

- **Route aggregation**

For scaling purposes, it is useful to publish an aggregated route as the service chain route, rather than publishing every route of each VM (/32). This reduces the memory footprint for the route table in the gateway router and also reduces route exchanges between control nodes and the gateway router. The route can be aggregated to the default route (0/0), to the VN subnet prefix, or to any arbitrary route prefix.

- **Path attribute modification for reoriginated routes**

There are cases where the BgpPath attribute for the service chain route needs to be modified. An example is the case of service chain failover, in which there are two service chains with identical services that are connected between the same two VNs. The operator needs to control which service chain is used for traffic between two networks, in addition to ensuring redundancy and high availability by providing failover support. Path attribute modification for reoriginated routes is implemented by means of routing policy, by providing an option to alter the MED (multi-exit discriminator) or local-pref of the reoriginated service chain route.

- **Control to enable and disable reorigination of the route**

In some scenarios, the operator needs a control to stop reorigination of the route as the service chain route, for example, when static routes are configured on service VM interfaces. Control to enable or disable reorigination of the route is implemented by tagging the routes with the no-reoriginate community. Routes with the no-reoriginate community tag are skipped for route reorigination.

Starting in Contrail Release 5.0, when one or more than one service instance in a service chain fails, reorigination of routes on both sides of the service chain is stopped and routes automatically converge to a backup service chain that is part of another Contrail cluster. For more information, see "[Service Instance Health Checks](#)" on page 341.

Route Aggregation

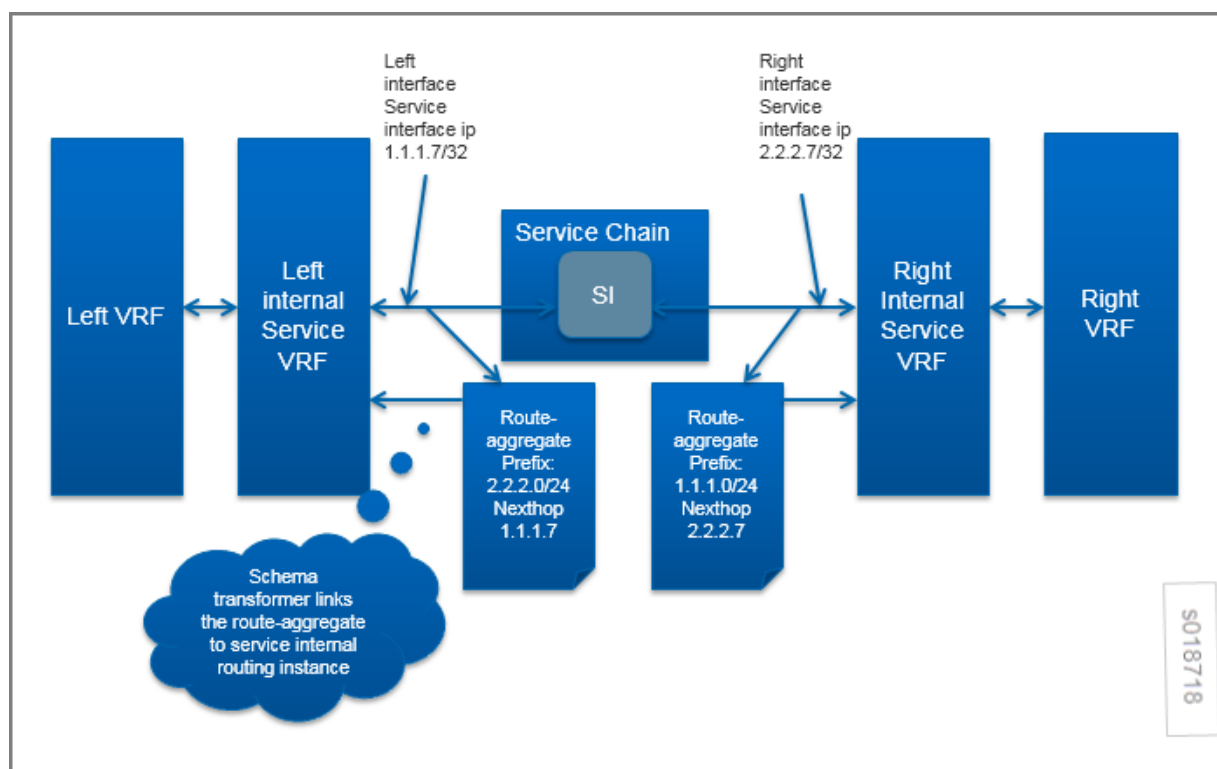
The route aggregation configuration object contains a list of prefixes to aggregate. The next-hop field in the route aggregate object contains the address of the route whose next hop is stitched as a next hop of the aggregate route.

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 66 on page 321](#).

Figure 66: 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 67 on page 324](#).

Figure 67: 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 68 on page 325](#).

Figure 68: Create Service Instance

Create Service Instance

si-aggregate st-with-aggregate - (transparent (left, right)...

▼ **Interface Details**

Interface Type	Virtual Network
left	Auto Configured
right	Auto Configured

▼ **Advanced Options**

► Routing Policy

▼ **Route Aggregate**

Interface Type	Route Aggregate
right	left-to-right

5018720

Cancel Save

Create a Virtual Network and Network Policy

Create a left and right virtual network with the subnets 1.1.1/24 and 2.2.2/24, respectively. Create a network policy to apply a service chain between the left VN and the right VN. See the following example.

Create Policy

Policy Name
service-chain-policy

Policy Rules

Action	Protocol	Source	Ports	Direction	Destination	Ports	Log	Services	Mirror
PASS	ANY	left	ANY	right	ANY		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Service Instance
si-aggregate

+ Add Rule

Cancel Save

Attach the network policy to create the service chain between the left and right VNs. See the following example.

Edit Network

Name
left

Network Policy(s)
default-domain:admin:service-chain-policy

Subnets

IPAM	CIDR	Allocation Pools	Gateway	DNS	DHCP
default-network-ip...	1.1.1.0/24	start-end	1.1.1.1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Host Route(s)

Route Prefix	Next Hop
--------------	----------

Advanced Options

Cancel Save

Validate the Route Aggregate Object in the API Server

Validate the route aggregate object in the API server configuration database. Verify the routing instance reference and the service instance reference for the aggregate object. The `aggregate_route_nexthop` field in the route aggregate object is initialized by the schema transformer to the service chain address. See the following example.

```

{
  - route-aggregate: {
    - fq_name: {
      "default-domain",
      "admin",
      "left-to-right"
    },
    uuid: "872b1fbd-b36c-4165-8723-7e10806d7716",
    parent_uuid: "6861d89d-a02f-4215-b329-1864084c8a75",
    aggregate_route_nexthop: "1.1.1.3",
    - routing_instance_refs: [
      - {
        - to: {
          "default-domain",
          "admin",
          "right",
          "service-ace7ae00-56e3-42d1-96ec-7fe77088d97f-default-domain_admin_si-aggregate"
        },
        href: "http://nodes27.englab.juniper.net:8082/routing-instance/d291a95a-1a5a-4fce-94c8-4abd0968d992",
        attr: null,
        uuid: "d291a95a-1a5a-4fce-94c8-4abd0968d992"
      }
    ],
    parent_href: "http://nodes27.englab.juniper.net:8082/project/6861d89d-a02f-4215-b329-1864084c8a75",
    parent_type: "project",
    + perms2: {-.},
    href: "http://nodes27.englab.juniper.net:8082/route-aggregate/872b1fbd-b36c-4165-8723-7e10806d7716",
    - id_perms: {-.},
    - aggregate_route_entries: {
      - route: [
        "1.1.1.0/24"
      ]
    },
    display_name: "left-to-right",
    - service_instance_refs: [
      - {
        - to: {
          "default-domain",
          "admin",
          "si-aggregate"
        },
        href: "http://nodes27.englab.juniper.net:8082/service-instance/62accf30-8cc8-4148-b7b8-975573b0d950",
        - attr: {
          interface_type: "right"
        },
        uuid: "62accf30-8cc8-4148-b7b8-975573b0d950"
      }
    ],
    name: "left-to-right"
  }
}

```

s018723

Validate the Route Aggregate Object in the Control Node

Validate the instance configurations of the route aggregate by checking the control node introspect for the service instance internal routing instance. For example:

`http://<control-node>:8083/Snh_ShowBgpInstanceConfigReq?search_string=default-domain:admin:right:service-ace7ae00-56e3-42d1-96ec-7fe77088d97f-default-domain_admin_si-aggregate`

See the following example.

service_chain_infos					static_routes aggregate_routes	
service_chain_infos					static_routes	aggregate_routes
family	routing_instance	chain_address	prefixes	service_instance	prefix	nexthop
inet	default-domain:admin:left:left	1.1.1.3	prefixes 1.1.1.0/24	default-domain:admin:si-aggregate	1.1.1.0/24	1.1.1.3

s018724

To check the state of the route aggregate object on the control node, point your browser to:

http://<control-node>:8083/Snh_ShowRouteAggregateReq

See the following example.

service_chain_infos					static_routes aggregate_routes	
service_chain_infos					static_routes	aggregate_routes
family	routing_instance	chain_address	prefixes	service_instance	prefix	nexthop
inet	default-domain:admin:left:left	1.1.1.3	prefixes 1.1.1.0/24	default-domain:admin:si-aggregate	1.1.1.0/24	1.1.1.3

You can also check the route table for the aggregate route in the right VN BGP table. For example:

http://<control-node>:8083/Snh_ShowRouteReq?x=default-domain:admin:right:right.inet.0

See the following example.

routes									
routes									
prefix	last_modified	paths							
1.1.1.0/24	2016-Feb-18 05:00:29.211876	protocol	last_modified	local_preference	local_as	peer_as	peer_router_id	source_as_path	next_hop
		Aggregate	2016-Feb-18 05:00:29.211876	100	0	0	--	--	10.204.216.23
									22

Routing Policy

Contrail uses routing policy infrastructure to manipulate the route and path attribute dynamically. Contrail also supports attaching the import routing policy on the service instances.

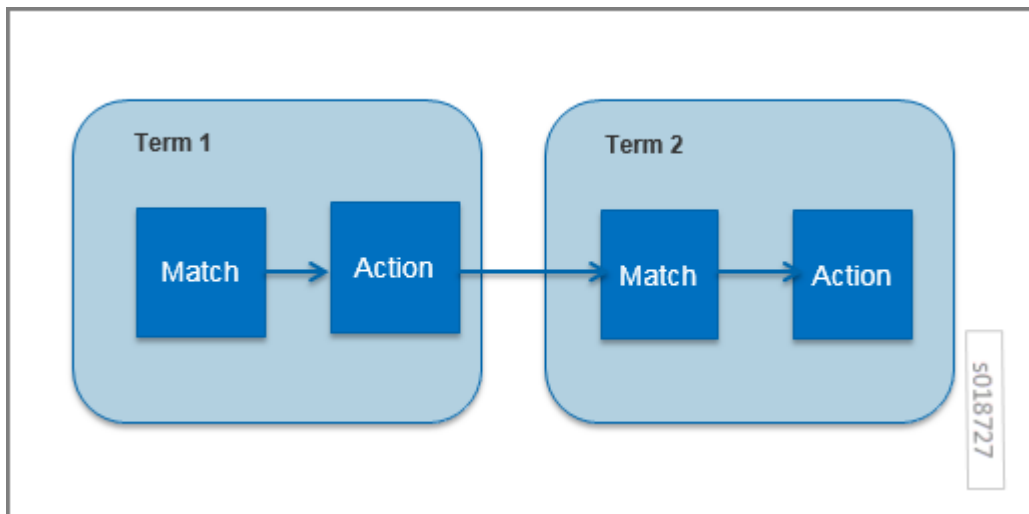
The routing policy contains list terms. A term can be a terminal rule, meaning that upon a match on the specified term, no further terms are evaluated and the route is dropped or accepted, based on the action in that term.

If the term is not a terminal rule, subsequent terms are evaluated for the given route.

The list terms are structured as in the following example.

```
Policy {
  Term-1
  Term-2
}
```

The matches and actions of the policy term lists operate similarly to the Junos language match and actions operations. A visual representation is the following.



Each term is represented as in the following:

```

from {
    match-condition-1
    match-condition-2
    ..
    ..
}
then {
    action
    update-action-1
    update-action-2
    ..
    ..
}

```

The term should not contain an any match condition, for example, an empty `from` should not be present.

If an any match condition is present, all routes are considered as matching the term.

However, the `then` condition can be empty or the action can be unspecified.

Applying Routing Policy

The routing policy evaluation has the following key points:

- If the term of a routing policy consists of multiple match conditions, a route must satisfy all match conditions to apply the action specified in the term.
- If a term in the policy does not specify a match condition, all routes are evaluated against the match.
- If a match occurs but the policy does not specify an accept, reject, or next term action, one of the following occurs:
 - The next term, if present, is evaluated.
 - If no other terms are present, the next policy is evaluated.
 - If no other policies are present, the route is accepted. The default routing policy action is “accept”.
- If a match does not occur with a term in a policy, and subsequent terms in the same policy exist, the next term is evaluated.
- If a match does not occur with any terms in a policy, and subsequent policies exist, the next policy is evaluated.
- If a match does not occur by the end of a policy or all policies, the route is accepted.

A routing policy can consist of multiple terms. Each term consists of match conditions and actions to apply to matching routes.

Each route is evaluated against the policy as follows:

1. The route is evaluated against the first term. If it matches, the specified action is taken. If the action is to accept or reject the route, that action is taken and the evaluation of the route ends. If the next term action is specified or if no action is specified, or if the route does not match, the evaluation continues as described above to subsequent terms.
2. Upon hitting the last non-terminal term of the given routing policy, the route is evaluated against the next policy, if present, in the same manner as described in step 1.

Match Condition: From

The match condition `from` contains a list of match conditions to be satisfied for applying the action specified in the term. It is possible that the term doesn't have any match condition. This indicates that all routes match this term and action is applied according to the action specified in the term.

The following table describes the match conditions supported by Contrail.

Match Condition	User Input	Description
Prefix	List of prefixes to match	<p>Each prefix in the list is represented as prefix and match type, where the prefix match type can be:</p> <ul style="list-style-type: none"> • exact • orlonger • longer <p>Example: 1.1.0.0/16 orlonger</p> <p>A route matches this condition if its prefix matches any of the prefixes in the list.</p>
Community	Community string to match	<p>Represented as either a well-known community string with no export or no reoriginate, or a string representation of a community (64512:11).</p>
Protocol	Array of path source or path protocol to match	<p>BGP XMPP StaticRoute ServiceChain Aggregate. A path is considered as matching this condition if the path protocol is one of protocols in the list.</p>

Routing Policy Action and Update Action

The policy action contains two parts, action and update action.

The following table describes action as supported by Contrail.

Action	Terminal?	Description
Reject	Yes	Reject the route that matches this term. No more terms are evaluated after hitting this term.
Accept	Yes	Accept the route that matches this term. No more terms are evaluated after hitting this term. The route is updated using the update specified in the policy action.

(Continued)

Action	Terminal?	Description
Next Term	No	This is the default action taken upon matching the policy term. The route is updated according to the update specified in the policy action. Next terms present in the routing policy are processed on the route. If there are no more terms in the policy, the next routing policy is processed, if present.

The update action section specifies the route modification to be performed on the matching route.

The following table describes update action as supported by Contrail.

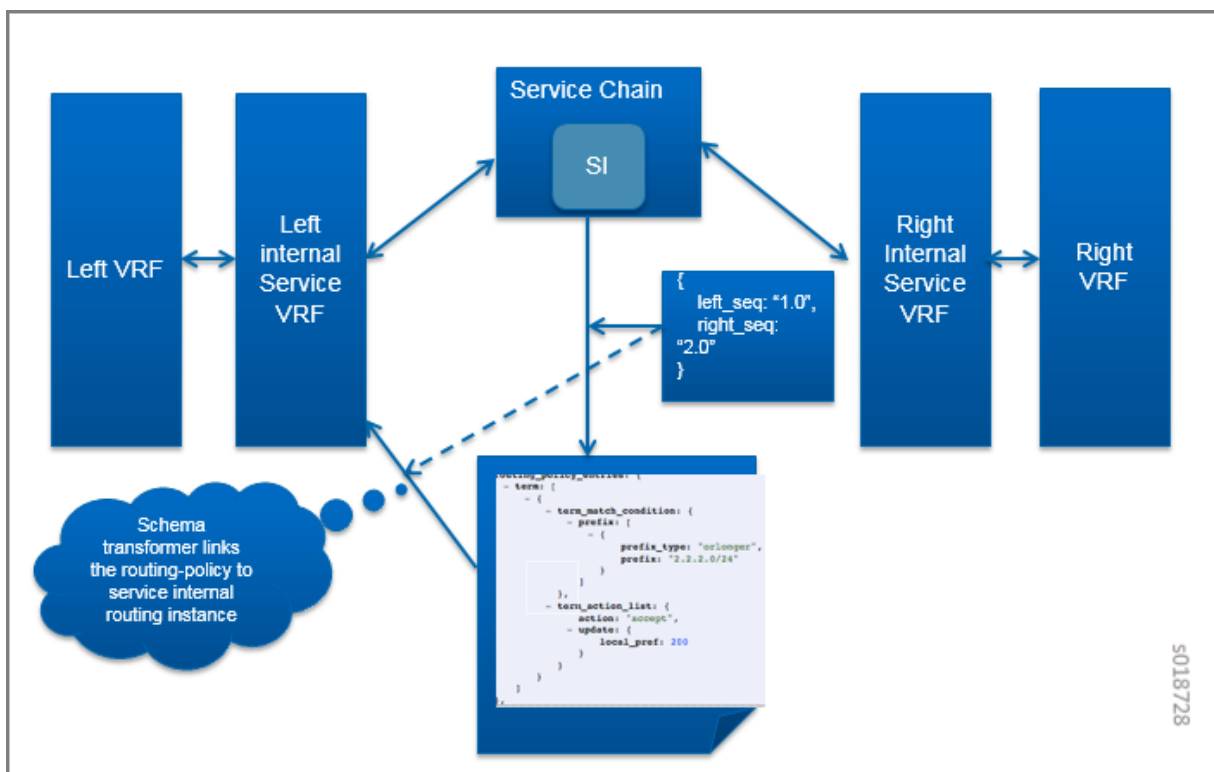
Update Action	User Input	Description
Community	List of community	As part of the policy update, the following actions can be taken for community: <ul style="list-style-type: none"> • Add a list of community to the existing community. • Set a list of community. • Remove a list of community (if present) 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.

Create Routing Policy

Name
failover

Term(s)
from: { prefix 2.2.2.0/24 orlonger } then: { local-preference 200 }

From
prefix 2.2.2.0/24 orlonger

Then
local-preference 200

Cancel Save

s018729



NOTE: The Contrail UI and REST APIs enable you to configure a BGP routing policy and then assign it to a virtual network, but the routing policy will not be applied if the virtual network is attached to an L3VPN.

Configure Service Instance

Create a service instance and attach the routing policy to both the left and right interfaces. The order of the policy is calculated by the UI, based on the order of the policy specified in the list.

Create Service Instance

ha-chain st-with-policy - [transparent (left, right)] - v1

Interface Details

Interface Type	Virtual Network
left	Auto Configured
right	Auto Configured

Advanced Options

Routing Policy

Interface Type	Routing Policy
left	failover
right	failover

Cancel Save

Configure the Network Policy for the Service Chain

At **Edit Policy**, create a policy for the service chain, see the following example.

Edit Policy (service-chain-policy)

Policy Name: service-chain-policy

Policy Rules

Action	Protocol	Source	Ports	Direction	Destination	Ports	Log	Services	Mirror
PASS	ANY	left	ANY	left to right	right	ANY	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Service Instance: si-aggregate ha-chain

+ Add Rule

Cancel Save

Using a VNC Script to Create Routing Policy

The following example shows use of a VNC API script to create a routing policy.

```
from vnc_api.vnc_api import *
vnc_lib = VncApi("admin", "<password>", "admin")
project=vnc_lib.project_read(fq_name=["default-domain", "admin"])
routing_policy=RoutingPolicy(name="vnc_3", parent_obj=project)
policy_term=PolicyTermType()
policy_statement=PolicyStatementType()
```

```

match_condition=TermMatchConditionType(protocol=["bgp"], community="22:33")
prefix_match=PrefixMatchType(prefix="1.1.1.0/24", prefix_type="orlonger")
match_condition.set_prefix([prefix_match])

term_action=TermActionListType(action="accept")
action_update=ActionUpdateType(local_pref=101, med=10)
add_community=ActionCommunityType()
comm_list=CommunityListType(["11:22"])
add_community.set_add(comm_list)
action_update.set_community(add_community)
term_action.set_update(action_update)

policy_term.set_term_action_list(term_action)
policy_term.set_term_match_condition(match_condition)

policy_statement.add_term(policy_term)
routing_policy.set_routing_policy_entries(policy_statement)
vnc_lib.routing_policy_create(routing_policy)

```

Verify Routing Policy in API Server

You can verify the service instance references and the routing instance references for the routing policy by looking in the API server configuration database. See the following example.

```

- routing_policy_entries: {
  - term: {
    - {
      - term_match_condition: {
        - prefix: {
          - {
            prefix_type: "orlonger",
            prefix: "2.2.2.0/24"
          }
        }
      },
      - term_action_list: {
        action: "accept",
        - update: {
          local_pref: 200
        }
      }
    }
  }
},
+ id_perms: {...},
- routing_instance_refs: [
  - {
    - to: [
      "default-domain",
      "admin",
      "right",
      "service-ace7ae00-56e3-42d1-96ec-7fe7708d97f-default-domain_admin_ha-chain"
    ],
    href: "http://nodes27.englab.juniper.net:8082/routing-instance/32b7eed4-57ce-4c44-bbb0-513f78db6068",
    - attr: {
      sequence: "1"
    },
    uuid: "32b7eed4-57ce-4c44-bbb0-513f78db6068"
  },
  - {
    - to: [
      "default-domain",
      "admin",
      "left",
      "service-ace7ae00-56e3-42d1-96ec-7fe7708d97f-default-domain_admin_ha-chain"
    ],
    href: "http://nodes27.englab.juniper.net:8082/routing-instance/6ad868d1-a412-4765-b8c4-f93ec5d9f4b2",
    - attr: {
      sequence: "1"
    },
    uuid: "6ad868d1-a412-4765-b8c4-f93ec5d9f4b2"
  }
],
- service_instance_refs: [
  - {
    - to: [
      "default-domain",
      "admin",
      "ha-chain"
    ],
    href: "http://nodes27.englab.juniper.net:8082/service-instance/983bb90b-b3f4-4d6c-be54-33a474eee7de",
    - attr: {
      left_sequence: "1",
      right_sequence: "1"
    },
    uuid: "983bb90b-b3f4-4d6c-be54-33a474eee7de"
  }
],
name: "failover"

```

s018732

Verify Routing Policy in the Control Node

You can verify the routing policy in the control node.

Point your browser to:

http://<control-node>:8083/Snh_ShowRoutingPolicyReq?search_string=failover

See the following example.

routing_policies

name	generation	ref_count	terms	deleted
default-domain:admin:failover	0	2	<div><div>terms</div><div><div><div>terminal</div><div>matches</div><div>actions</div></div><div><div>true</div><div><div>prefix [2.2.2.0/24 orlonger]</div></div><div><div>accept</div><div>local-pref 200</div></div></div></div></div>	false
default-domain:default-project:default-routing-policy	0	0	<div><div>terms</div></div>	false

5018745

Verify Routing Policy Configuration in the Control Node

You can verify the routing policy configuration in the control node.

Point your browser to:

http://<control-node>:8083/Snh_ShowBgpRoutingPolicyConfigReq?search_string=failover

See the following example.

ShowBgpRoutingPolicyConfigResp

routing_policies

name	terms				
default-domain:admin:failover	<table><thead><tr><th>match</th><th>action</th></tr></thead><tbody><tr><td>from { prefix 2.2.2.0/24 orlonger }</td><td>then { local-preference 200 accept }</td></tr></tbody></table>	match	action	from { prefix 2.2.2.0/24 orlonger }	then { local-preference 200 accept }
match	action				
from { prefix 2.2.2.0/24 orlonger }	then { local-preference 200 accept }				

5018733

Verify Routing Policy Configuration on the Routing Instance

You can verify the routing policy configuration on the internal routing instance.

Point your browser to:

http://<control-node>:8083/Snh_ShowBgpInstanceConfigReq?search_string=<name-of-internal-vrf>

See the following example.

service_chain_info					static_routes aggregate_routes routing_policies		
2 service_chain_info					static_routes	aggregate_routes	routing_policies
family	routing_instance	chain_address	prefixes	service_instance			policy_name sequence
inet	default-domain:admin:right:right	1.1.1.6	prefixes 2.2.2.0/24	default-domain:admin:ha-chain			default-domain:admin:failover 1

You can also verify the routing policy on the routing instance operational object.

Point your browser to:

http://<control-node>:8083/Snh_ShowRoutingInstanceReq?x=<name-of-internal-vrf>

See the following example.

routing_policies	
routing_policies	
policy_name	generation
default-domain:admin:failover	0

Control for Route Reorigination

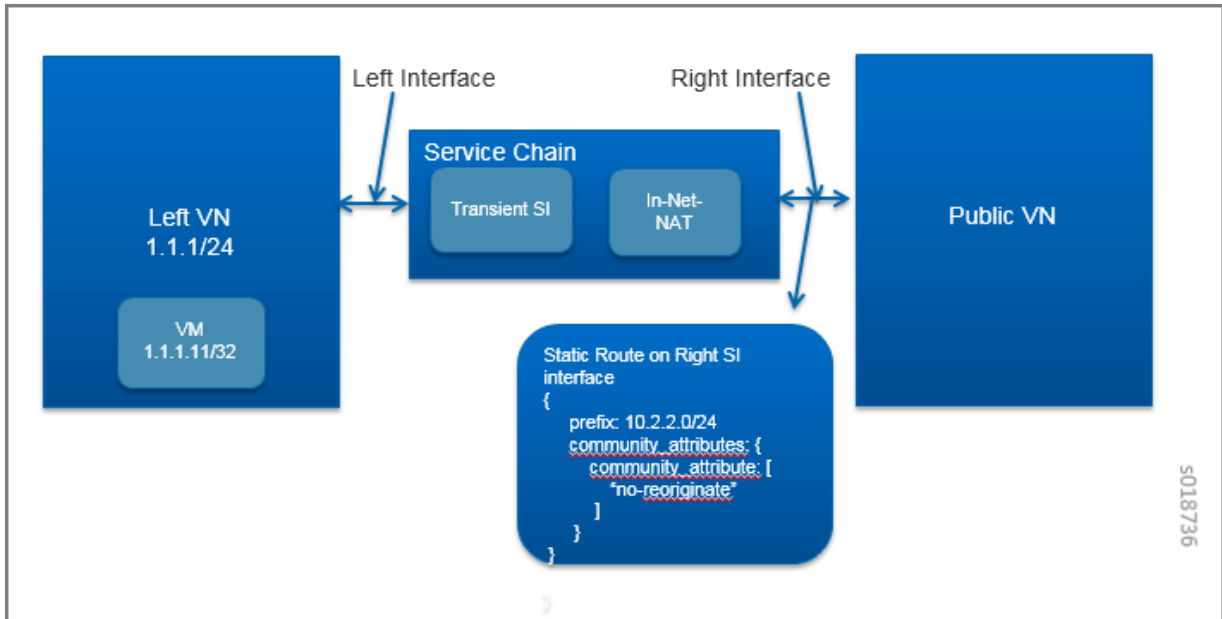
The ability to prevent reorigination of interface static routes is typically required when routes are configured on an interface that belongs to a service VM.

As an example, the following image shows a service chain that has multiple service instances, with an in-net-nat service instance as the last service VM, also with the right VN as the public VN.

The last service instance performs NAT by using a NAT pool. The right interface of the service VM must be configured with an interface static route for the NAT pool so that the destination in the right VN knows how to reach addresses in the NAT pool. However, the NAT pool prefix should not be reoriginated into the left VN.

To prevent route reorigination, the interface static route is tagged with a well-known BGP community called no-reoriginate.

When the control node is reoriginating the route, it skips the routes that are tagged with the BGP community.



Configuring and Troubleshooting Reorigination Control

The community attribute on the static routes for the interface static route of the service instance is specified during creation of the service instance. See the following example.

Create Service Instance

Name:

Service Template:

Interface Type:

Virtual Network:

Interface Type:

Virtual Network:

+ Add Static Routes

Prefix:

Next Hop:

Community:

Routing Policy

Interface Type:

Routing Policy:

Cancel Save

5018737

Use the following example to verify that the service instance configuration object in the API server has the correct community set for the static route. See the following example.

```

{
  - service-instance: {
    + virtual_machine_back_refs: [...],
    + fq_name: [...],
    uuid: "a6e1e71f-f828-43de-a493-b193bdb73ded",
    parent_type: "project",
    parent_uuid: "634f90d9-da62-4c2f-a238-7cc1c1a055a5",
    parent_href: "http://nodeq2:8082/project/634f90d9-da62-4c2f-a2
  - service_instance_properties: {
    right_virtual_network: "default-domain:admin:twig",
    - interface_list: [
      - {
        virtual_network: "default-domain:admin:fifo"
      },
      - {
        virtual_network: "default-domain:admin:twig",
        - static_routes: {
          - route: {
            - {
              prefix: "10.2.2.0/24",
              next_hop: null,
              - community_attributes: {
                - community_attribute: [
                  "no-reoriginate"
                ],
              },
              next_hop_type: null
            },
          },
        },
      },
    ],
    left_virtual_network: "default-domain:admin:fifo",
    - scale_out: {
      max_instances: 1
    },
  },
}

```

s018738

Service Instance Health Checks

IN THIS SECTION

- [Health Check Object | 342](#)
- [Bidirectional Forwarding and Detection Health Check over Virtual Machine Interfaces | 346](#)
- [Bidirectional Forwarding and Detection Health Check for BGPaaS | 347](#)
- [Health Check of Transparent Service Chain | 347](#)
- [Service Instance Fate Sharing | 347](#)

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 | 342](#)
- [Health Check Object Configuration | 342](#)
- [Creating a Health Check with the Contrail User Interface | 344](#)
- [Using the Health Check | 345](#)
- [Health Check Process | 346](#)

Health Check Overview

The service instance health check is used to determine the liveness of a service provided by a VM, checking whether the service is operationally up or down. The vRouter agent uses ping and an HTTP URL to the link-local address to check the liveness of the interface.

If the health check determines that a service is no longer operational, it removes the routes for the VM, thereby disabling packet forwarding to the VM.

The service instance health check is used with service template version 2.

Health Check Object Configuration

[Table 27 on page 342](#) shows the configurable properties of the health check object.

Table 27: Health Check Configurable Parameters

Field	Description
- enabled	Indicates that health check is enabled. The default is False.
- health-check-type	Indicates the health check type: link-local, end-to-end, bgp-as-a-service, and so on.. The default is link-local.
- monitor-type	The protocol type to be used: PING or HTTP.

Table 27: Health Check Configurable Parameters (Continued)

Field	Description
- delay	The delay, in seconds, to repeat the health check.
- timeout	The number of seconds to wait for a response.
- max-retries	The number of retries to attempt before declaring an instance health down.
- http-method	When the monitor protocol is HTTP, the type of HTTP method used, such as GET, PUT, POST, and so on.
- url-path	When the monitor protocol is HTTP, the URL to be used. For all other cases, such as ICMP, the destination IP address.
- expected-codes	When the monitor protocol is HTTP, the expected return code for HTTP operations.

Health Check Modes

The following modes are supported for the service instance health check:

- **link-local**—A local check for the service VM on the vRouter where the VM is running. In this case, the source IP of the packet is the service chain IP.
- **end-to-end**—A remote address or URL is provided for a service health check through a chain of services. The destination of the health check probe is allowed to be outside the service instance. However, the health check probe must be reachable through the interface of the service instance where the health check is attached. The end-to-end health check probe is transmitted all the way to the actual destination outside the service instance. The response to the health check probe is received and processed by the service health check to evaluate the status.

Restrictions include:

- This check is applicable for a chain where the services are not scaled out.
- When this mode is configured, a new health check IP is allocated and used as the source IP of the packet.

- The health check IP is allocated per virtual-machine-interface of the service VM where the health check is attached.
- The agent relies on the service-health-check-ip flag to use as the source IP.



NOTE: In versions prior to Contrail 4.1, end-to-end health check is not supported on a transparent service chain. However, a link-local health check is possible on a transparent service instance if the corresponding service instance interface is configured with its IP address. Contrail 4.1 supports a segment-based health check for transparent service chain.

Creating a Health Check with the Contrail User Interface

To create a health check with the Contrail Web UI:

1. Navigate to **Configure > Services > Health Check Service**, and click to open the **Create** screen. See [Figure 69 on page 344](#).

Figure 69: Create Health Check Screen

The screenshot shows a 'Create' dialog box for a 'Health Check Service'. The dialog has a title bar with a close button. Below the title bar are two tabs: 'Health Check Service' and 'Permissions'. The 'Health Check Service' tab is active and contains the following fields:

- Name:** A text input field containing 'ext_hc_service'.
- Protocol:** A dropdown menu with 'PING' selected.
- Monitor Target:** A text input field containing '8.8.8.8'.
- Delay (secs):** A text input field containing '3'.
- Timeout (secs):** A text input field containing '5'.
- Retries:** A text input field containing '2'.
- Health Check Type:** A dropdown menu with 'End-To-End' selected.

At the bottom right of the dialog are two buttons: 'Cancel' and 'Save'. A vertical text '5018766' is visible on the right side of the dialog.

2. Complete the fields to define the permissions for the health check, see [Table 28 on page 345](#).

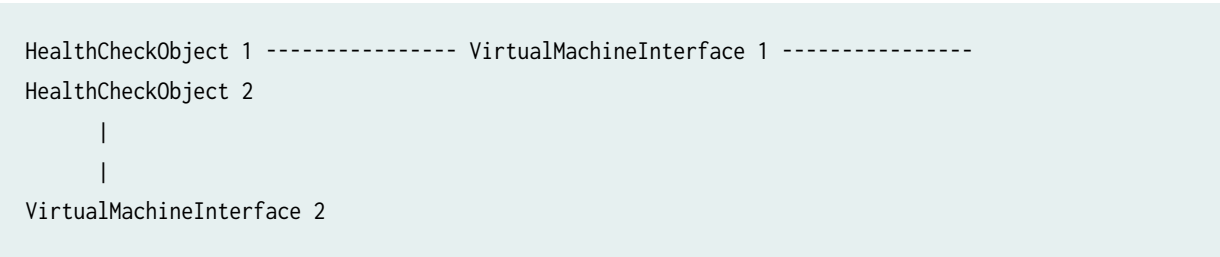
Table 28: Create Health Check Fields

Field	Description
Name	Enter a name for the health check service you are creating.
Protocol	Select from the list the protocol to use for the health check, PING, HTTP, BFD, and so on.
Monitor Target	Select from the list the address of the target to be monitored by the health check.
Delay (secs)	The delay, in seconds, to repeat the health check.
Timeout (secs)	The number of seconds to wait for a response.
Retries	The number of retries to attempt before declaring an instance health down.
Health Check Type	Select from the list the type of health check—link-local, end-to-end, segment-based, bgp-as-a-service, and so on.

Using the Health Check

A REST API can be used to create a health check object and define its associated properties, then a link is added to the VM interface.

The health check object can be linked to multiple VM interfaces. Additionally, a VM interface can be associated with multiple health check objects. The following is an example:



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 releases prior to Contrail Release 5.0, when an SI fails, the service chain continues to forward packets and routes reoriginate on both sides of the service chain. The packets are dropped in the SI or by the vRouter causing a black hole.

Starting in Contrail Release 5.0, when one or more than one SI in a service chain fails, reorigination of routes on both sides of the service chain is stopped and routes automatically converge to a backup

service chain that is part of another Contrail cluster. SI fate sharing brings down the service chain and the gateway nodes automatically reroutes traffic to an alternate cluster.

Starting in Contrail Release 4.1, **segment-based** health check type is used to verify the health of a SI in a service chain. To identify a failure of an SI, segment-based health check is configured either on the egress or ingress interface of the SI. When SI health check fails, the vRouter agent drops an SI route or a connected route. A connected route is also dropped if the vRouter agent restarts due to a software failure, when a compute node reboots, or when long-lived graceful restart (LLGR) is not enabled. You can detect an SI failure by keeping track of corresponding connected routes of the service chain address.



NOTE: When an SI is scaled out, the connected route for an SI interface goes down only when all associated VMs have failed.

The control node uses the `service-chain-id` in `ServiceChainInfo` to link all SIs in a service chain. When the control node detects that any SI of the same `service-chain-id` is down, it stops reoriginating routes in egress and ingress directions for all SIs. The control node reoriginates routes only when the connected routes of all the SIs are up.

Examples: Configuring Service Chaining

IN THIS CHAPTER

- [Example: Creating an In-Network Service Chain by Using Contrail Command | 349](#)
- [Example: Creating an In-Network-NAT Service Chain by Using Contrail Command | 357](#)
- [Example: Creating a Transparent Service Chain by Using Contrail Command | 366](#)

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

IN THIS SECTION

- [Hardware and Software Requirements | 349](#)
- [Overview | 350](#)
- [Configuration | 350](#)

This example provides instructions to create an in-network service chain by using the Contrail Command user interface (UI).

Hardware and Software Requirements

The following are the minimum requirements needed:

Hardware

- Processor: 4 core x86
- Memory: 32GB RAM
- Storage: at least 128GB hard disk

Software

- Contrail Release 3.2 or later



NOTE: For Contrail Networking Release 3.2 through Release 4.1, you use the Contrail Web UI. For more information, see [Example: Creating an In-Network Service Chain by Using Contrail Web UI](#).

Overview

A service chain is a set of services that are connected across networks. A service chain consists of service instances, left and right virtual networks, and a service policy attached to the networks. A service chain can have in-network services, in-network-nat services, and transparent services.

In an in-network service chain, packets are routed between service instance interfaces. When a packet is routed through the service chain, the source address of the packet entering the left interface of the service chain and source address of the packet exiting the right interface is the same. For more information, see ["Service Chaining" on page 302](#).

Configuration

IN THIS SECTION

- [Create Virtual Network | 351](#)
- [Create Virtual Machine | 352](#)
- [Configure Service Template | 353](#)
- [Add Service Instance | 354](#)
- [Create Service Policy | 355](#)
- [Attach Service Policy | 356](#)
- [Launch Virtual Machine | 356](#)

These topics provide instructions to create an in-network service chain.

Create Virtual Network

Step-by-Step Procedure

Use the Contrail Command UI to create a left virtual network, right virtual network, and management virtual network.

To create a left virtual network:

1. Click **Overlay>Virtual Networks**.

The All Networks page is displayed.

2. Click **Create** to create a network.

The Create Virtual Network page is displayed.

3. In the **Name** field enter **test-left-VN** for the left virtual network.

4. Select **(Default) User defined subnet only** from the **Allocation Mode** list.

5. Click **+Add** in the Subnets section to add subnets.

Step-by-Step Procedure

In the row that is displayed,

- a. Click the arrow in the Network IPAM field and select **left-ipam** for the left virtual network.

For the right virtual network, select **right-ipam** and for the management network, select **mgmt-ipam**.



NOTE: Management network is not used to route packets. This network is used to help debug issues with the virtual machine.

6. Enter **192.0.2.0/24** in the **CIDR** field.

7. Click **Create**.

The All Networks page is displayed. All virtual networks that you created are displayed in this page.

Repeat steps "2" on page 351 through "7" on page 351 to create the right virtual network (**test-right-VN**) and management virtual network (**test-mgmt-VN**).

Create Virtual Machine

Step-by-Step Procedure

Follow these steps to create a left virtual machine by using the Contrail Command UI.

1. Click **Workloads > Instances**.

The Instances page is displayed.

2. Click **Create**.

The Create Instance page is displayed.

3. Select **Virtual Machine** option button as the serve type.

4. Enter **test-left-VM** for the left virtual machine in the **Instance Name** field.

5. Select **Image** as the boot source from the **Select Boot Source** list.



NOTE: vSRX image with M1.large flavor is recommended for in-network virtual machine.

6. Select **vSRX image** file from the **Select Image** list.

7. Select **M1.large** flavor from the **Select Flavor** list.

8. Select the network you want to associate with the left virtual machine by clicking > next to the name of the virtual machine listed in the Available Networks table.

For the left virtual machine, select **test-left-VN**. For the right virtual machine, select **test-right-VN**. For the management virtual machine, select **test-mgmt-VN**.

The network is added to the Allocated Networks table.

9. Select **nova** from the **Availability Zone** list.



NOTE: You can choose any other availability zone.

10. Select **5** from the **Count (1-10)** list.



NOTE: You can choose any value from 1 through 10.

11. Click **Create** to launch the left virtual machine instance.

The Instances page is displayed. The virtual machine instances that you created are listed on the Instances page.

Repeat steps "2" on page 352 through "11" on page 352 to create right virtual machine instance (**test-right-VM**) and management virtual machine instance (**test-mgmt-VM**).

Configure Service Template

Step-by-Step Procedure

Follow these steps to create a service template by using the Contrail Command UI:

1. Click **Services>Catalog**.

The VNF Service Templates page is displayed.

2. Click **Create**.

The Create VNF Service Template page is displayed.

3. Enter **test-service-template** in the **Name** field.

4. Select **v2** as the version type.



NOTE: Starting with Release 3.2, Contrail supports only *Service Chain Version 2 (v2)*.

5. Select **Virtual Machine** as the virtualization type.

6. Select **In-Network** as the service mode.

7. Select **Firewall** as the service type.

8. From the Interface section,

- Select **left** as the interface type from the **Interface Type** list.
- Click **+ Add**.

The Interface Type list is added to the table.

Select **right** as the interface type.

- Click **+ Add** again.

Another Interface Type list is added to the table.

Select **management** as the interface type.



NOTE: The interfaces created on the virtual machine must follow the same sequence as that of the interfaces in the service template.

9. Click **Create** to create the service template.

The VNF Service Templates page is displayed. The service template that you created is displayed in the VNF Service Templates page.

Add Service Instance

Step-by-Step Procedure

Follow these steps to add a service instance by using the Contrail Command UI:

1. Click **Services>Deployments**.

The VNF Service Instances page is displayed.

2. Click **Create**.

The Create VNF Service Instance page is displayed.

3. Enter **test-service-instance** in the **Name** field.

4. Select **test-service-template - [in-network, (left, right, management)] - v2** from the **Service Template** list.

The **Interface Type** and **Virtual Network** fields are displayed.

5. Select the virtual network for each interface type as given below.

- **left**—Select the left virtual network (**test-left-VN**) that you created.
- **right**—Select the right virtual network (**test-right-VN**) that you created.
- **management**—Select the management virtual network (**test-management-VN**) that you created.

6. Click the **Port Tuples** section and click **+Add**.

Select the virtual machine instance for each interface type as given below.

- **left**—Select the left virtual machine instance that you created.
- **right**—Select the right virtual machine instance that you created.
- **management**—Select the management virtual machine instance that you created.

7. Click **Create** to create the service instance.

The VNF Service Instances page is displayed. The service instance that you created is displayed in the VNF Service Instances page.

Create Service Policy

Step-by-Step Procedure

Follow these steps to create a service policy by using the Contrail Command UI.

1. Click **Overlay > **Network Policies**.**

The Network Policies page is displayed.

2. Click **Create.**

The Network Policy tab of the Create Network Policy page is displayed.

3. Enter **test-network-policy in the **Policy Name** field.**

4. In the **Policy Rule(s) section,**

- Select **pass** from the **Action** list.
- Select **ANY** from the **Protocol** list.
- Select **Network** from the **Source Type** list.
- Select the **test-left-VN** from the **Source** list.
- In the **Source Port** field, leave the default option, **Any**, as is.
- Select **< >** from the **Direction** list.
- Select **Network** from the **Destination Type** list.
- Select the **test-right-VN** from the **Destination** list.
- In the **Destination Ports** field, leave the default option, **Any**, as is.

5. Click **Create to create the service policy.**

The Network Policies page is displayed. All policies that you created are displayed in the Network Policies page.

Attach Service Policy

Step-by-Step Procedure

Follow these steps to attach a service policy:

1. Click **Overlay>Virtual Networks**.

The All networks page is displayed.

2. Attach service policy to the left virtual network (**test-left-VN**) and right virtual network (**test-right-VN**) that you created.

Step-by-Step Procedure

To attach service policy,

- a. Select the check box next to the name of the virtual network.
- b. Hover over to the end of the selected row and click the **Edit** icon.

The Edit Virtual Network page is displayed.

- c. Select the network policy from the Network Policies list.

3. Click **Save** to save the changes.

The Virtual Networks page is displayed.

Launch Virtual Machine

Step-by-Step Procedure

You can launch virtual machines from Contrail Command and test the traffic through the service chain by doing the following:

1. Launch the left virtual machine in left virtual network. See ["Create Virtual Machine" on page 352](#).
2. Launch the right virtual machine in right virtual network. See ["Create Virtual Machine" on page 352](#).
3. Ping the left virtual machine IP address from the right virtual machine.

Follow these steps to ping a virtual machine:

Step-by-Step Procedure

- a. Click **Workloads>Instances**.

The Instances page is displayed.

- b. Click the open console icon next to **test-right-VM**.

The Console page is displayed.

- c. Log in using root user credentials.
- d. Ping the left virtual machine IP address (**190.0.2.3**) from the Console.

See [Figure 70 on page 357](#) for a sample output.

Figure 70: Ping test-left-VM

```
root@test-right-vm:~# ping -c 5 192.0.2.3
PING 192.0.2.3 (192.0.2.3) 56(84) bytes of data:
64 bytes from 192.0.2.3: icmp_seq=1 ttl=63 time=0.238 ms
64 bytes from 192.0.2.3: icmp_seq=2 ttl=63 time=0.208 ms
64 bytes from 192.0.2.3: icmp_seq=3 ttl=63 time=0.231 ms
64 bytes from 192.0.2.3: icmp_seq=4 ttl=63 time=0.210 ms
64 bytes from 192.0.2.3: icmp_seq=5 ttl=63 time=0.210 ms

--- 192.0.2.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 0.208/0.219/0.238/0.018 ms
root@test-right-vm:~#
```

RELATED DOCUMENTATION

[Service Chaining | 302](#)

[Example: Creating an In-Network-NAT Service Chain by Using Contrail Command | 357](#)

[Example: Creating a Transparent Service Chain by Using Contrail Command | 366](#)

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

IN THIS SECTION

- [Prerequisites | 358](#)
- [Overview | 359](#)

This example provides instructions to create an in-network-nat service chain by using the Contrail Command user interface (UI).

Prerequisites

- **Hardware and Software Requirements**

Hardware

- Processor: 4 core x86
- Memory: 32GB RAM
- Storage: at least 128GB hard disk

Software

- Contrail Release 3.2 or later



NOTE: For Contrail Networking Release 3.2 through Release 4.1, you use the Contrail Web UI. For more information, see [Example: Creating a In-Network-NAT Service Chain by Using Contrail Web UI](#).

- **Create Network IPAM (IP Address Management)**

1. Click **Overlay>IPAM**.

The IP Address Management page is displayed.

2. Click **Create** to create a new network IPAM.
3. Enter a name for the IPAM in the name field.
4. Select **Default** from the DNS list.
5. Enter valid IP address in the NTP Server IP field.
6. Enter domain name in the Domain Name field.
7. Click **Create**.

The IP Address Management page is displayed.

Overview

A service chain is a set of services that are connected across networks. A service chain consists of service instances, left and right virtual networks, and a service policy attached to the networks. A service chain can have in-network services, in-network-nat services, and transparent services.

In an in-network-nat service chain, packets are routed between service instance interfaces. In-network-nat service chain does not require return traffic to be routed to the source network. When a packet is routed through the service chain, the source address of the packet entering the left interface of the service chain is updated and is not the same as the source address of the packet exiting the right interface. For more information, see ["Service Chaining" on page 302](#).

Configuration

IN THIS SECTION

- [Create Virtual Network | 359](#)
- [Create Virtual Machine | 360](#)
- [Configure Service Template | 361](#)
- [Add Service Instance | 362](#)
- [Create Service Policy | 363](#)
- [Attach Service Policy | 364](#)
- [Launch Virtual Machine | 365](#)

These topics provide instructions to create an in-network-nat service chain.

Create Virtual Network

Step-by-Step Procedure

Use the Contrail Command UI to create a left virtual network, right virtual network, and management virtual network.

To create a left virtual network:

1. Click **Overlay>Virtual Networks**.

The All Networks page is displayed.

2. Click **Create** to create a network.

The Create Virtual Network page is displayed.

3. In the **Name** field enter **test-left-VN** for the left virtual network.
4. Select **(Default) User defined subnet only** from the **Allocation Mode** list.
5. Click **+Add** in the Subnets section to add subnets.

Step-by-Step Procedure

In the row that is displayed,

- a. Select an IPAM for the virtual network from the Network IPAM list.
 - b. Enter **192.0.2.0/24** in the **CIDR** field.
6. Click **Create**.

The All Networks page is displayed. All virtual networks that you created are displayed in this page.



NOTE: Management network is not used to route packets. This network is used to help debug issues with the virtual machine.

Repeat steps "2" on page 359 through "6" on page 360 to create the right virtual network (**test-right-VN**) and management virtual network (**test-mgmt-VN**).

Create Virtual Machine

Step-by-Step Procedure

Follow these steps to create a left virtual machine by using the Contrail Command UI.

1. Click **Workloads > Instances**.

The Instances page is displayed.

2. Click **Create**.

The Create Instance page is displayed.

3. Select **Virtual Machine** option button as the serve type.
4. Enter **test-left-VM** for the left virtual machine in the **Instance Name** field.
5. Select **Image** as the boot source from the **Select Boot Source** list.



NOTE: vSRX image with M1.large flavor is recommended for in-network-nat virtual machine.

6. Select **vSRX image** file from the **Select Image** list.
7. Select **M1.large** flavor from the **Select Flavor** list.
8. Select the network you want to associate with the left virtual machine by clicking > next to the name of the virtual machine listed in the Available Networks table.

For the left virtual machine, select **test-left-VN**. For the right virtual machine, select **test-right-VN**. For the management virtual machine, select **test-mgmt-VN**.

The network is added to the Allocated Networks table.

9. Select **nova** from the **Availability Zone** list.



NOTE: You can choose any other availability zone.

10. Select **5** from the **Count (1-10)** list.



NOTE: You can choose any value from 1 through 10.

11. Click **Create** to launch the left virtual machine instance.

The Instances page is displayed. The virtual machine instances that you created are listed on the Instances page.

Repeat steps "2" on page 360 through "11" on page 361 to create right virtual machine instance (**test-right-VM**) and management virtual machine instance (**test-mgmt-VM**).

Configure Service Template

Step-by-Step Procedure

Follow these steps to create a service template by using the Contrail Command UI:

1. Click **Services>Catalog**.

The VNF Service Templates page is displayed.

2. Click **Create**.

The Create VNF Service Template page is displayed.

3. Enter **test-service-template** in the **Name** field.
4. Select **v2** as the version type.



NOTE: Starting with Release 3.2, Contrail supports only *Service Chain Version 2 (v2)*.

5. Select **Virtual Machine** as the virtualization type.
6. Select **In-Network Nat** as the service mode.
7. Select **Firewall** as the service type.
8. From the Interface section,
 - Select **left** as the interface type from the **Interface Type** list.
 - Click **+ Add**.

The Interface Type list is added to the table.

Select **right** as the interface type.

- Click **+ Add** again.

Another Interface Type list is added to the table.

Select **management** as the interface type.



NOTE: The interfaces created on the virtual machine must follow the same sequence as that of the interfaces in the service template.

9. Click **Create** to create the service template.

The VNF Service Templates page is displayed. The service template that you created is displayed in the VNF Service Templates page.

Add Service Instance

Step-by-Step Procedure

Follow these steps to add a service instance by using the Contrail Command UI:

1. Click **Services>Deployments**.

The VNF Service Instances page is displayed.

2. Click **Create**.

The Create VNF Service Instance page is displayed.

3. Enter **test-service-instance** in the **Name** field.

4. Select **test-service-template - [in-network-nat, (left, right, management)] - v2** from the **Service Template** list.

The **Interface Type** and **Virtual Network** fields are displayed.

5. Select the virtual network for each interface type as given below.

- **left**—Select the left virtual network (**test-left-VN**) that you created.
- **right**—Select the right virtual network (**test-right-VN**) that you created.
- **management**—Select the management virtual network (**test-management-VN**) that you created.

6. Click the **Port Tuples** section and click **+Add**.

Select the virtual machine instance for each interface type as given below.

- **left**—Select the left virtual machine instance that you created.
- **right**—Select the right virtual machine instance that you created.
- **management**—Select the management virtual machine instance that you created.

7. Click **Create** to create the service instance.

The VNF Service Instances page is displayed. The service instance that you created is displayed in the VNF Service Instances page.

Create Service Policy

Step-by-Step Procedure

Follow these steps to create a service policy by using the Contrail Command UI.

1. Click **Overlay > Network Policies**.

The Network Policies page is displayed.

2. Click **Create**.

The Network Policy tab of the Create Network Policy page is displayed.

3. Enter **test-network-policy** in the **Policy Name** field.

4. In the **Policy Rule(s)** section,

- Select **pass** from the **Action** list.
- Select **ANY** from the **Protocol** list.
- Select **Network** from the **Source Type** list.
- Select the **test-left-VN** from the **Source** list.
- In the **Source Port** field, leave the default option, **Any**, as is.
- Select **< >** from the **Direction** list.
- Select **Network** from the **Destination Type** list.
- Select the **test-right-VN** from the **Destination** list.
- In the **Destination Ports** field, leave the default option, **Any**, as is.

5. Click **Create** to create the service policy.

The Network Policies page is displayed. All policies that you created are displayed in the Network Policies page.

Attach Service Policy

Step-by-Step Procedure

Follow these steps to attach a service policy:

1. Click **Overlay>Virtual Networks**.

The All networks page is displayed.

2. Attach service policy to the left virtual network (**test-left-VN**) and right virtual network (**test-right-VN**) that you created.

Step-by-Step Procedure

To attach service policy,

- a. Select the check box next to the name of the virtual network.
- b. Hover over to the end of the selected row and click the **Edit** icon.

The Edit Virtual Network page is displayed.

- c. Select the network policy from the Network Policies list.

3. Click **Save** to save the changes.

The Virtual Networks page is displayed.

Launch Virtual Machine

Step-by-Step Procedure

You can launch virtual machines from Contrail Command and test the traffic through the service chain by doing the following:

1. Launch the left virtual machine in left virtual network. See ["Create Virtual Machine" on page 360](#).
2. Launch the right virtual machine in right virtual network. See ["Create Virtual Machine" on page 360](#).
3. Ping the left virtual machine IP address from the right virtual machine.

Follow these steps to ping a virtual machine:

Step-by-Step Procedure

- a. Click **Workloads>Instances**.

The Instances page is displayed.

- b. Click the open console icon next to **test-right-VM**.

The Console page is displayed.

- c. Log in using root user credentials.

- d. Ping the left virtual machine IP address (**190.0.2.3**) from the Console.

RELATED DOCUMENTATION

[Service Chaining | 302](#)

[Example: Creating an In-Network Service Chain by Using Contrail Command | 349](#)

[Example: Creating a Transparent Service Chain by Using Contrail Command | 366](#)

Example: Creating a Transparent Service Chain by Using Contrail Command

IN THIS SECTION

- [Prerequisites | 366](#)
- [Overview | 367](#)
- [Configuration | 367](#)

This example provides step-by-step instructions to create a transparent service chain by using the Contrail Command user interface (UI).

Prerequisites

- **Hardware and Software Requirements**

Hardware

- Processor: 4 core x86
- Memory: 32GB RAM
- Storage: at least 128GB hard disk

Software

- Contrail Release 3.2 or later



NOTE: For Contrail Networking Release 3.2 through Release 4.1, you use the Contrail Web UI. For more information, see [Example: Creating a Transparent Service Chain by Using Contrail Web UI](#).

- **Create Network IPAM (IP Address Management)**

1. Click **Overlay>IPAM**.

The IP Address Management page is displayed.

2. Click **Create** to create a new network IPAM.
3. Enter a name for the IPAM in the name field.

4. Select **Default** from the DNS list.
5. Enter valid IP address in the NTP Server IP field.
6. Enter domain name in the Domain Name field.
7. Click **Create**.

The IP Address Management page is displayed.

Overview

A service chain is a set of services that are connected across networks. A service chain consists of service instances, left and right virtual networks, and a service policy attached to the networks. A service chain can have in-network services, in-network-nat services, and transparent services. A transparent service chain is used for services that do not modify packets that are bridged between service instance interfaces. For more information, see ["Service Chaining" on page 302](#).

Configuration

IN THIS SECTION

- [Create Primary Virtual Networks | 368](#)
- [Create Secondary Virtual Network | 369](#)
- [Create Service Virtual Machine | 369](#)
- [Create Virtual Machine | 370](#)
- [Configure Service Template | 371](#)
- [Add Service Instance | 372](#)
- [Create Service Policy | 373](#)
- [Attach Service Policy | 374](#)
- [Launch Virtual Machine | 374](#)

These topics provide instructions to create a transparent service chain.

Create Primary Virtual Networks

Step-by-Step Procedure

Use the Contrail Command UI to create three primary virtual networks: left virtual network, right virtual network, and management virtual network. You attach service policies to the primary virtual networks that you create.

Follow these steps To create a left virtual network:

1. Click **Overlay>Virtual Networks**.

The All Networks page is displayed.

2. Click **Create** to create a network.

The Create Virtual Network page is displayed.

3. In the **Name** field enter **test-left-VN** for the left virtual network.

4. Select **(Default) User defined subnet only** from the **Allocation Mode** list.

5. Click **+Add** in the Subnets section to add subnets.

Step-by-Step Procedure

In the row that is displayed,

- a. Select an IPAM for the virtual network from the Network IPAM list.

- b. Enter **192.0.2.0/24** in the **CIDR** field.

6. Click **Create**.

The All Networks page is displayed. All virtual networks that you created are displayed in this page.



NOTE: Management network is not used to route packets. This network is used to help debug issues with the virtual machine.

Repeat steps "2" on page 368 through "6" on page 368 to create the right virtual network (**test-right-VN**) and management virtual network (**test-mgmt-VN**).

Create Secondary Virtual Network

Step-by-Step Procedure

Use the Contrail Command UI to create three secondary virtual networks: left virtual network (**trans-left-VN**), right virtual network (**trans-right-VN**), and management virtual network (**trans-mgmt-VN**). You associate the secondary virtual network to the transparent service instance that you create. For more information on creating virtual networks, see ["Create Primary Virtual Networks" on page 368](#).

Create Service Virtual Machine

Step-by-Step Procedure

Follow these steps to create a service virtual machine (SVM) by using the Contrail Command UI.

1. Click **Workloads > Instances**.
The Instances page is displayed.
2. Click **Create**.
The Create Instance page is displayed.
3. Select **Virtual Machine** option button as the serve type.
4. Enter **test-SVM** in the **Instance Name** field.
5. Select **Image** as the boot source from the **Select Boot Source** list.



NOTE: vSRX image with M1.large flavor is recommended for in-network virtual machine.

6. Select **vSRX image** file from the **Select Image** list.
7. Select **M1.large** flavor from the **Select Flavor** list.
8. From the Available Networks table, select **trans-left-VN**, **trans-right-VN**, and **trans-mgmt-VN** networks that you want to associate with the SVM by clicking **>** next to the name of the virtual machine.

The network is added to the Allocated Networks table.
9. Select **nova** from the **Availability Zone** list.



NOTE: You can choose any other availability zone.

10. Select **5** from the **Count (1-10)** list.



NOTE: You can choose any value from 1 through 10.

11. Click **Create** to launch the left virtual machine instance.

The Instances page is displayed. The virtual machine instances that you created are listed on the Instances page.

Create Virtual Machine

Step-by-Step Procedure

Follow these steps to create a left virtual machine by using the Contrail Command UI.

1. Click **Workloads > Instances**.

The Instances page is displayed.

2. Click **Create**.

The Create Instance page is displayed.

3. Select **Virtual Machine** option button as the serve type.

4. Enter **test-left-VM** for the left virtual machine in the **Instance Name** field.

5. Select **Image** as the boot source from the **Select Boot Source** list.



NOTE: vSRX image with M1.large flavor is recommended for in-network virtual machine.

6. Select **vSRX image** file from the **Select Image** list.

7. Select **M1.large** flavor from the **Select Flavor** list.

8. From the Available Networks table, select **test-left-VN** network that you want to associate with the left virtual machine by clicking **>** next to the name of the virtual machine.

For the right virtual machine, select **test-right-VN**.

The network is added to the Allocated Networks table.

9. Select **nova** from the **Availability Zone** list.



NOTE: You can choose any other availability zone.

10. Select **5** from the **Count (1-10)** list.



NOTE: You can choose any value from 1 through 10.

11. Click **Create** to launch the left virtual machine instance.

The Instances page is displayed. The virtual machine instances that you created are listed on the Instances page.

Repeat steps "2" on page 370 through "11" on page 371 to create right virtual machine instance (**test-right-VM**).

Configure Service Template

Step-by-Step Procedure

Follow these steps to create a service template by using the Contrail Command UI:

1. Click **Services>Catalog**.

The VNF Service Templates page is displayed.

2. Click **Create**.

The Create VNF Service Template page is displayed.

3. Enter **test-service-template** in the **Name** field.

4. Select **v2** as the version type.



NOTE: Starting with Release 3.2, Contrail supports only *Service Chain Version 2 (v2)*.

5. Select **Virtual Machine** as the virtualization type.

6. Select **Transparent** as the service mode.

7. Select **Firewall** as the service type.

8. From the Interface section,

- Select **left** as the interface type from the **Interface Type** list.
- Click **+ Add**.

The Interface Type list is added to the table.

Select **right** as the interface type.

- Click **+ Add** again.

Another Interface Type list is added to the table.

Select **management** as the interface type.



NOTE: The interfaces created on the virtual machine must follow the same sequence as that of the interfaces in the service template.

9. Click **Create** to create the service template.

The VNF Service Templates page is displayed. The service template that you created is displayed in the VNF Service Templates page.

Add Service Instance

Step-by-Step Procedure

Follow these steps to add a service instance by using the Contrail Command UI:

1. Click **Services>Deployments**.

The VNF Service Instances page is displayed.

2. Click **Create**.

The Create VNF Service Instance page is displayed.

3. Enter **test-service-instance** in the **Name** field.

4. Select **test-service-template - [transparent, (left, right, management)] - v2** from the **Service Template** list.

The **Interface Type** and **Virtual Network** fields are displayed.

5. Select the virtual network for each interface type as given below.

- **left**—Select **trans-left-VN** virtual network that you created.

- **right**—Select the **trans-right-VN** virtual network that you created.
- **management**—Select the **trans-mgmt-VN** virtual network that you created.

6. Click the **Port Tuples** section and click **+Add**.

Select the virtual machine instance for each interface type as given below. The port tuples should match the interfaces of the SVM. See ["Create Service Virtual Machine" on page 369](#).

- **left**—Select the left virtual machine instance that you created.
- **right**—Select the right virtual machine instance that you created.
- **management**—Select the management virtual machine instance that you created.

7. Click **Create** to create the service instance.

The VNF Service Instances page is displayed. The service instance that you created is displayed in the VNF Service Instances page.

Create Service Policy

Step-by-Step Procedure

Follow these steps to create a service policy by using the Contrail Command UI.

1. Click **Overlay > Network Policies**.

The Network Policies page is displayed.

2. Click **Create**.

The Network Policy tab of the Create Network Policy page is displayed.

3. Enter **test-network-policy** in the **Policy Name** field.

4. In the **Policy Rule(s)** section,

- Select **pass** from the **Action** list.
- Select **ANY** from the **Protocol** list.
- Select **Network** from the **Source Type** list.
- Select the **test-left-VN** from the **Source** list.
- In the **Source Port** field, leave the default option, **Any**, as is.
- Select **< >** from the **Direction** list.

- Select **Network** from the **Destination Type** list.
- Select the **test-right-VN** from the **Destination** list.
- In the **Destination Ports** field, leave the default option, **Any**, as is.

5. Click **Create** to create the service policy.

The Network Policies page is displayed. All policies that you created are displayed in the Network Policies page.

Attach Service Policy

Step-by-Step Procedure

Follow these steps to attach a service policy:

1. Click **Overlay>Virtual Networks**.

The All networks page is displayed.

2. Select the **test-left-VN** network that you want to edit, and click the **Edit** icon.

The Edit Virtual Network page is displayed.



NOTE: For the right virtual network, edit **test-right-VN**.

3. Select **test-network-policy** from the Network Policies list.

4. Click **Save** to save the changes.

The Virtual Networks page is displayed.

Repeat steps "2" on page 374 through "4" on page 374 to attach the service policy to **test-right-VN**.

Launch Virtual Machine

RELATED DOCUMENTATION

[Service Chaining | 302](#)

[Example: Creating an In-Network-NAT Service Chain by Using Contrail Command | 357](#)

[Example: Creating an In-Network Service Chain by Using Contrail Command | 349](#)

4

PART

Monitoring and Troubleshooting the Network Using Contrail Analytics

- Understanding Contrail Analytics | **376**
 - Configuring Contrail Analytics | **404**
 - Using Contrail Analytics to Monitor and Troubleshoot the Network | **418**
-

Understanding Contrail Analytics

IN THIS CHAPTER

- [Understanding Contrail Analytics | 376](#)
- [Contrail Alerts | 377](#)
- [Underlay Overlay Mapping in Contrail | 381](#)

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 377
Analytics Scalability 404
High Availability for Analytics 405
Ceilometer Support in a Contrail Cloud 410
Underlay Overlay Mapping in Contrail 381
Monitoring the System 418
Debugging Processes Using the Contrail Introspect Feature 422
Monitor > Infrastructure > Dashboard 427
Monitor > Infrastructure > Control Nodes 431
Monitor > Infrastructure > Virtual Routers 442
Monitor > Infrastructure > Analytics Nodes 456
Monitor > Infrastructure > Config Nodes 464
Monitor > Networking 468
Understanding Flow Sampling
Query > Flows 480
Query > Logs 490
System Log Receiver in Contrail Analytics 407
Example: Debugging Connectivity Using Monitoring for Troubleshooting 497

Contrail Alerts

IN THIS SECTION

- [Alert API Format | 378](#)
- [Analytics APIs for Alerts | 379](#)
- [Analytics APIs for SSE Streaming | 380](#)

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: "eyJ0aW1lc3RhbnRhaXNiMTAuODQuMTMuNDAlfQ==",
        NTk5NSwgImhvc3RfaXNiMTAuODQuMTMuNDAlfQ==",

```

```

        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 | 382](#)
- [Underlay Overlay Analytics Available in Contrail | 382](#)
- [Architecture and Data Collection | 383](#)
- [New Processes/Services for Underlay Overlay Mapping | 383](#)
- [External Interfaces Configuration for Underlay Overlay Mapping | 384](#)
- [Physical Topology | 384](#)
- [SNMP Configuration | 385](#)

- [Link Layer Discovery Protocol \(LLDP\) Configuration | 385](#)
- [IPFIX and sFlow Configuration | 385](#)
- [Sending pRouter Information to the SNMP Collector in Contrail | 388](#)
- [pRouter UVEs | 388](#)
- [Contrail User Interface for Underlay Overlay Analytics | 390](#)
- [Enabling Physical Topology on the Web UI | 391](#)
- [Viewing Topology to the Virtual Machine Level | 391](#)
- [Viewing the Traffic of any Link | 391](#)
- [Trace Flows | 392](#)
- [Search Flows and Map Flows | 393](#)
- [Overlay to Underlay Flow Map Schemas | 394](#)
- [Module Operations for Overlay Underlay Mapping | 397](#)
- [SNMP Collector Operation | 397](#)
- [Topology Module Operation | 399](#)
- [IPFIX and sFlow Collector Operation | 400](#)
- [Troubleshooting Underlay Overlay Mapping | 401](#)
- [Script to add pRouter Objects | 401](#)

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.

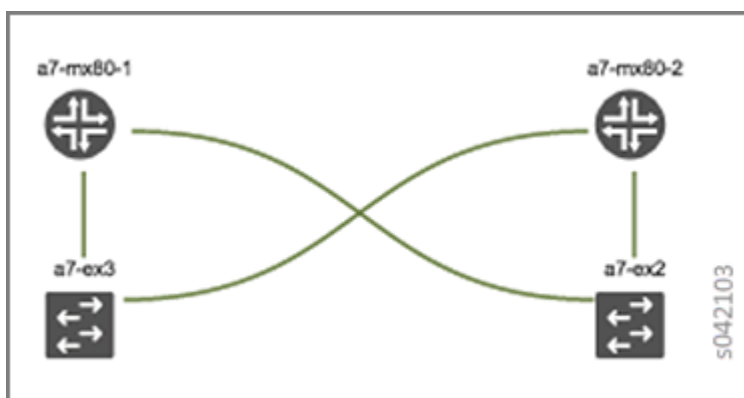
Physical Topology

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 71: Analytics Topology



SNMP Configuration

Configure SNMP on the physical devices so that the `contrail-snmp-collector` can read SNMP data.

The following shows an example SNMP configuration from a Juniper Networks device.

```
set snmp community public authorization read-only
```

Link Layer Discovery Protocol (LLDP) Configuration

Configure LLDP on the physical device so that the `contrail-snmp-collector` can read the neighbor information of the routers.

The following is an example of LLDP configuration on a Juniper Networks device.

```
set protocols lldp interface all
```

```
set protocols lldp-med interface all
```

IPFIX and sFlow Configuration

Flow samples are sent to the `contrail-collector` by the physical devices. Because the `contrail-collector` supports the sFlow and IPFIX protocols for receiving flow samples, the physical devices, such as MX Series devices or ToR switches, must be configured to send samples using one of those protocols.

Example: sFlow Configuration

The following shows a sample sFlow configuration. In the sample, the IP variable *<source ip>* refers to the loopback or IP that can be reachable of the device that acts as an sflow source, and the other IP variable *<collector_IP_data>* is the address of the collector device.

```
root@host> show configuration protocols sflow | display set

set protocols sflow polling-interval 0

set protocols sflow sample-rate ingress 10

set protocols sflow source-ip <source ip>4

set protocols sflow collector <collector_IP_data> udp-port 6343

set protocols sflow interfaces ge-0/0/0.0

set protocols sflow interfaces ge-0/0/1.0

set protocols sflow interfaces ge-0/0/2.0

set protocols sflow interfaces ge-0/0/3.0

set protocols sflow interfaces ge-0/0/4.0
```

Example: IPFIX Configuration

The following is a sample IPFIX configuration from a Juniper Networks device. The IP address variable *<ip_sflow collector>* represents the sflow collector (control-collector analytics node) and *<source ip>* represents the source (outgoing) interface on the router/switch device used for sending flow data to the collector. This could also be the lo0 address, if it is reachable from the Contrail cluster.

```
root@host> show configuration chassis | display set

set chassis tfeb slot 0 sampling-instance sample-ins1

set chassis network-services

root@host> show configuration chassis tfeb | display set
```

```
set chassis tfeb slot 0 sampling-instance sample-ins1
```

```
root@host > show configuration services flow-monitoring | display set
```

```
set services flow-monitoring version-ipfix template t1 flow-active-timeout 30
```

```
set services flow-monitoring version-ipfix template t1 flow-inactive-timeout 30
```

```
set services flow-monitoring version-ipfix template t1 template-refresh-rate packets 10
```

```
set services flow-monitoring version-ipfix template t1 ipv4-template
```

```
root@host > show configuration interfaces | display set | match sampling
```

```
set interfaces ge-1/0/0 unit 0 family inet sampling input
```

```
set interfaces ge-1/0/1 unit 0 family inet sampling input
```

```
root@host> show configuration forwarding-options sampling | display set
```

```
set forwarding-options sampling instance sample-ins1 input rate 1
```

```
set forwarding-options sampling instance sample-ins1 family inet output flow-server <ip_sflow  
collector> port 4739
```

```
set forwarding-options sampling instance sample-ins1 family inet output flow-server <ip_sflow  
collector> version-ipfix template t1
```

```
set forwarding-options sampling instance sample-ins1 family inet output inline-jflow source-  
address <source ip>
```

Sending pRouter Information to the SNMP Collector in Contrail

Information about the physical devices must be sent to the SNMP collector before the full analytics information can be read and displayed. Typically, the pRouter information is taken from the `contrail-config` file.

SNMP collector getting pRouter information from contrail-config file

The physical routers are added to the `contrail-config` by using the Contrail user interface or by using direct API, by means of provisioning or other scripts. Once the configuration is in the `contrail-config`, the `contrail-snmp-collector` gets the physical router information from `contrail-config`. The SNMP collector uses this list and the other configuration parameters to perform SNMP queries and to populate pRouter UVEs.

Figure 72: Add Physical Router Window

The screenshot displays the Juniper Contrail configuration interface. On the left, a sidebar shows the navigation menu with categories like Infrastructure, Physical Devices, and Networking. The main area shows the 'Physical Routers' configuration page. A modal window titled 'Add Physical Router' is open, allowing the user to add a new router. The form includes the following fields and sections:

- Name:** A text input field containing 'new-prouter'.
- Vendor:** A text input field.
- Model:** A text input field.
- Management IP:** A text input field containing '1.1.1.1'.
- Tunnel Source IP:** A text input field.
- User Credentials:** An expandable section.
- Virtual Router:** An expandable section.
- BGP Router:** An expandable section.
- SNMP Credentials:** An expanded section showing:
 - Version:** Radio buttons for 2 (selected) and 3.
 - Community:** A text input field containing 'public'.

At the bottom right of the modal, there are 'Cancel' and 'Save' buttons. The background interface shows a table of existing physical routers with columns for Name, a7-ex2, a7-ex3, a7-mx80-1, and a7-mx80-2.

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 73: 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 74: 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.

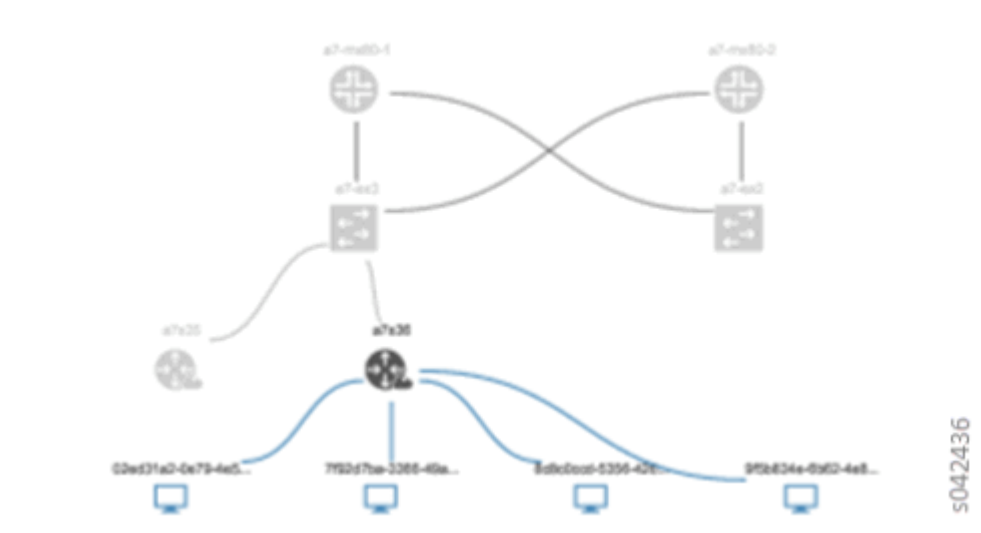
```
service supervisor-webui restart
```

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.

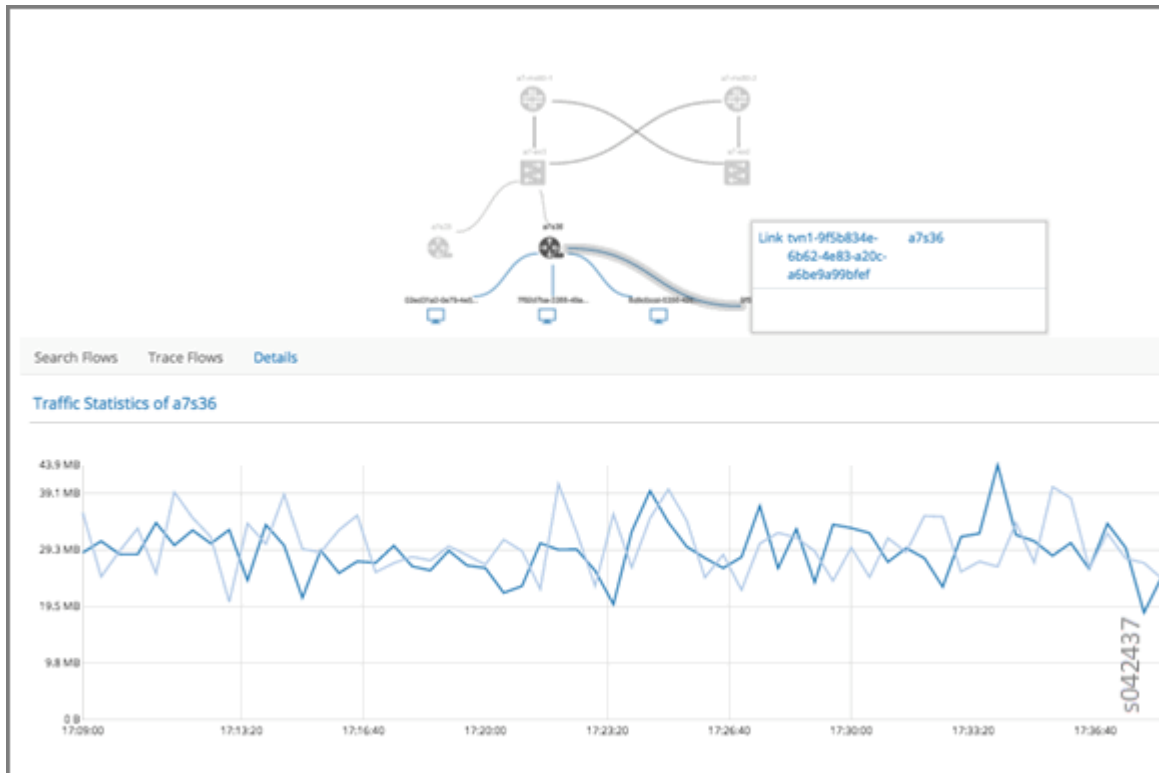
Figure 75: Physical Topology Related to a vRouter



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.

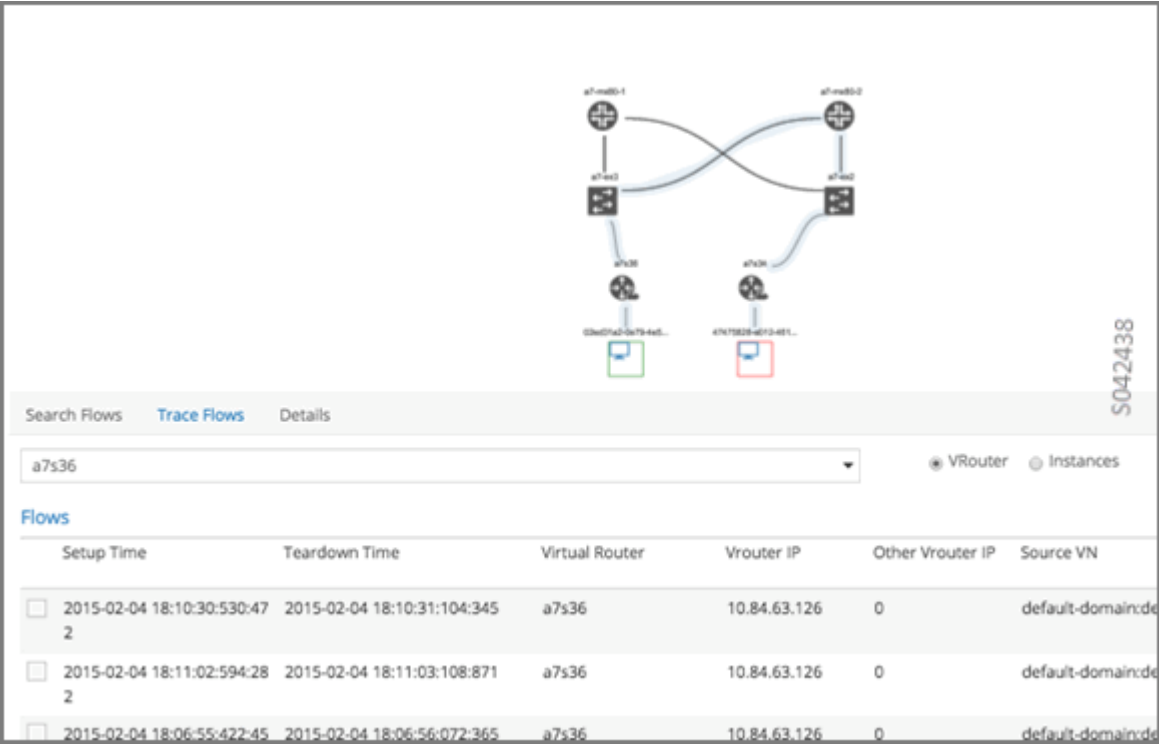
Figure 76: Traffic Statistics Graph



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 77: List of Active Flows



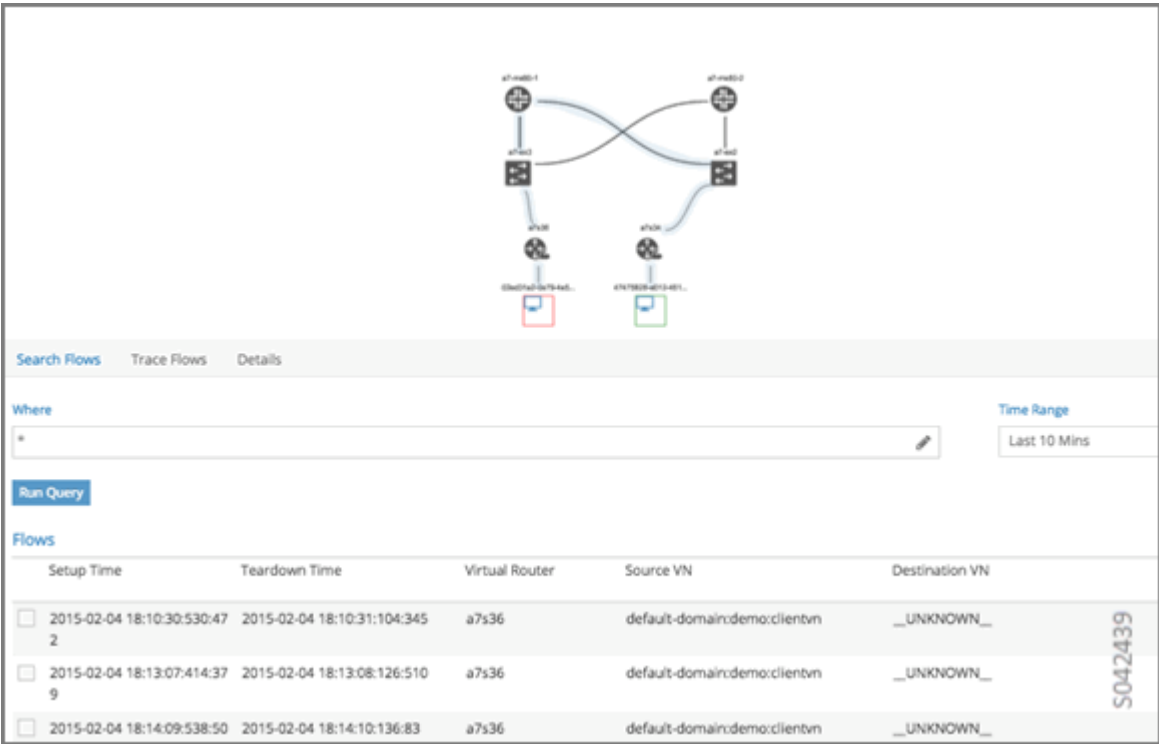
Limitations of Trace Flow Feature

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 78: 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-snmpp.py`, and then execute the command as shown:

```
python cfg-snmpp.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 | 376](#)

[Contrail Alerts | 377](#)

Configuring Contrail Analytics

IN THIS CHAPTER

- [Analytics Scalability | 404](#)
- [High Availability for Analytics | 405](#)
- [Role-Based Access Control for Analytics | 406](#)
- [System Log Receiver in Contrail Analytics | 407](#)
- [Sending Flow Messages to the Contrail System Log | 408](#)
- [More Efficient Flow Queries | 409](#)
- [Ceilometer Support in a Contrail Cloud | 410](#)

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.

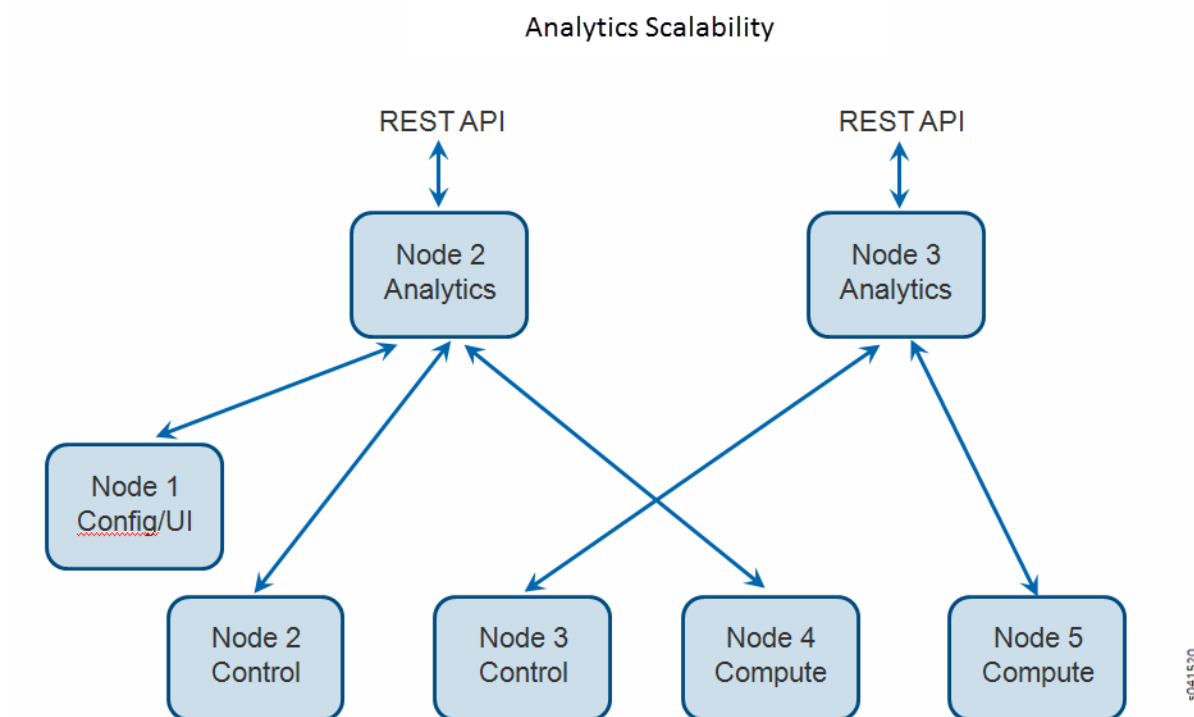
The following is the recommended locations for the various component roles of the Contrail system for a 5-node configuration.

- 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 79 on page 405 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 79: 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 analytics 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
- application_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 | 408](#)
- [Redirecting System Logs to Contrail Collector | 408](#)
- [Exporting Logs from Contrail Analytics | 408](#)

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 | 410](#)
- [Ceilometer Details | 411](#)
- [Verification of Ceilometer Operation | 411](#)
- [Contrail Ceilometer Plugin | 414](#)
- [Ceilometer Installation and Provisioning | 417](#)

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	<code>ceilometer-agent-central</code> and <code>ceilometer-agent-compute</code>
Notification agent	<code>ceilometer-agent-notification</code>
Collector	<code>ceilometer-collector</code>
API Server	<code>ceilometer-api</code>

Alarms

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

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-abf2-880eefc25ceb
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:


```

+-----+-----+
| 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.

Using Contrail Analytics to Monitor and Troubleshoot the Network

IN THIS CHAPTER

- [Monitoring the System | 418](#)
- [Debugging Processes Using the Contrail Introspect Feature | 422](#)
- [Monitor > Infrastructure > Dashboard | 427](#)
- [Monitor > Infrastructure > Control Nodes | 431](#)
- [Monitor > Infrastructure > Virtual Routers | 442](#)
- [Monitor > Infrastructure > Analytics Nodes | 456](#)
- [Monitor > Infrastructure > Config Nodes | 464](#)
- [Monitor > Networking | 468](#)
- [Query > Flows | 480](#)
- [Query > Logs | 490](#)
- [Example: Debugging Connectivity Using Monitoring for Troubleshooting | 497](#)

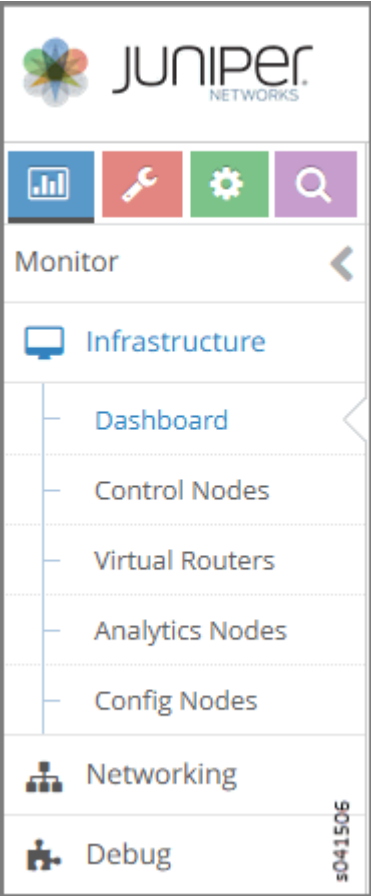
Monitoring the System

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 80 on page 419](#)

Figure 80: Monitor Menu



See [Table 29 on page 419](#) for descriptions of the items available under each of the menu options from the **Monitor** icon.

Table 29: 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 427 .

Table 29: Monitor Menu Options *(Continued)*

Option	Description
Infrastructure > Control Nodes	<p>View a summary for all control nodes in the system, and for each control node, view:</p> <ul style="list-style-type: none"> • Graphical reports of memory usage and average CPU load. • Console information for a specified time period. • A list of all peers with details about type, ASN, and the like. • A list of all routes, including next hop, source, local preference, and the like. <p>See "Monitor > Infrastructure > Control Nodes" on page 431.</p>
Infrastructure > Virtual Routers	<p>View a summary of all vRouters in the system, and for each vRouter, view:</p> <ul style="list-style-type: none"> • Graphical reports of memory usage and average CPU load. • Console information for a specified time period. • A list of all interfaces with details such as label, status, associated network, IP address, and the like. • A list of all associated networks with their ACLs and VRFs. • A list of all active flows with source and destination details, size, and time. <p>See "Monitor > Infrastructure > Virtual Routers" on page 442.</p>
Infrastructure > Analytics Nodes	<p>View activity for the analytics nodes, including memory and CPU usage, analytics host names, IP address, status, and more. See "Monitor > Infrastructure > Analytics Nodes" on page 456.</p>
Infrastructure > Config Nodes	<p>View activity for the config nodes, including memory and CPU usage, config host names, IP address, status, and more. See "Monitor > Infrastructure > Config Nodes" on page 464.</p>

Table 29: Monitor Menu Options *(Continued)*

Option	Description
Networking > Networks	<p>For all virtual networks for all projects in the system, view graphical traffic statistics, including:</p> <ul style="list-style-type: none"> • Total traffic in and out. • Inter VN traffic in and out. • The most active ports, peers, and flows for a specified duration. • All traffic ingress and egress from connected networks, including their attached policies. <p>See "Monitor > Networking" on page 468.</p>
Networking > Dashboard	<p>For all virtual networks for all projects in the system, view graphical traffic statistics, including:</p> <ul style="list-style-type: none"> • Total traffic in and out. • Inter VN traffic in and out. <p>You can view the statistics in varying levels of granularity, for example, for a whole project, or for a single network. See "Monitor > Networking" on page 468.</p>
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 29: Monitor Menu Options *(Continued)*

Option	Description
Debug > Packet Capture	<ul style="list-style-type: none"> • Add and manage packet analyzers. • Attach packet captures and configure their details. • View a list of all packet analyzers in the system and the details of their configurations, including source and destination networks, ports, and IP addresses.

RELATED DOCUMENTATION

[Monitor > Infrastructure > Dashboard | 427](#)

[Monitor > Infrastructure > Control Nodes | 431](#)

[Monitor > Infrastructure > Virtual Routers | 442](#)

[Monitor > Networking | 468](#)

[Query > Logs | 490](#)

[Query > Flows | 480](#)

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.

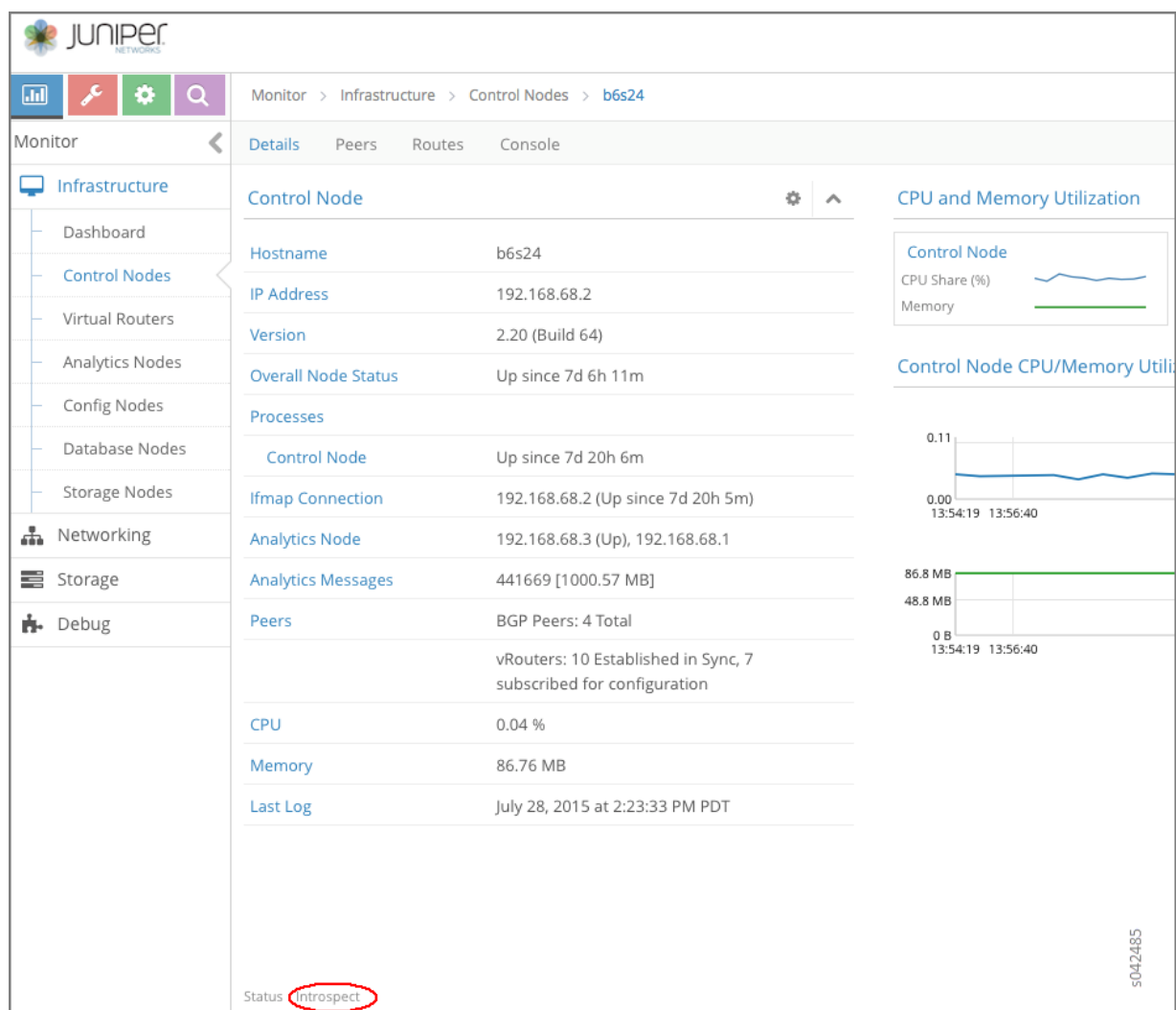
You can display the HTTP introspect of a Contrail daemon directly by accessing the following Introspect ports:

- *<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 81 on page 424](#) shows the contrail-control infrastructure page. Notice the Introspect link at the bottom of the Control Nodes Details tab window.

Figure 81: 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 82 on page 425 shows the Controller Introspect window.

Figure 82: Controller Introspect Window

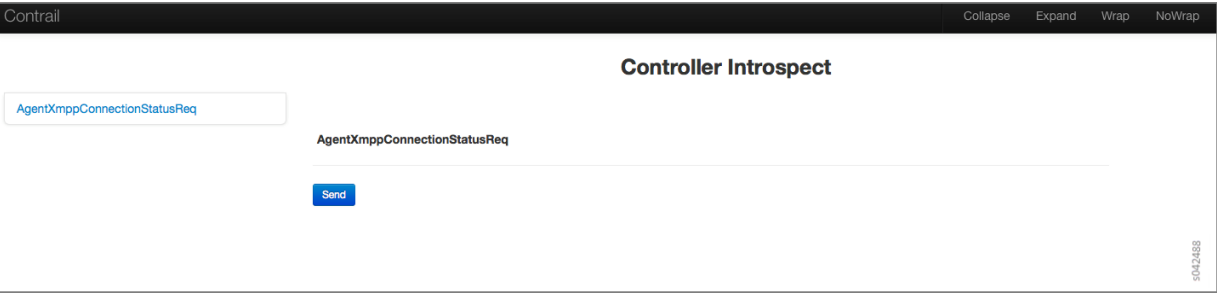


Figure 83 on page 425 shows an example of the BGP Peer (bgp_peer.xml) Introspect page.

Figure 83: BGP Peer Introspect Page

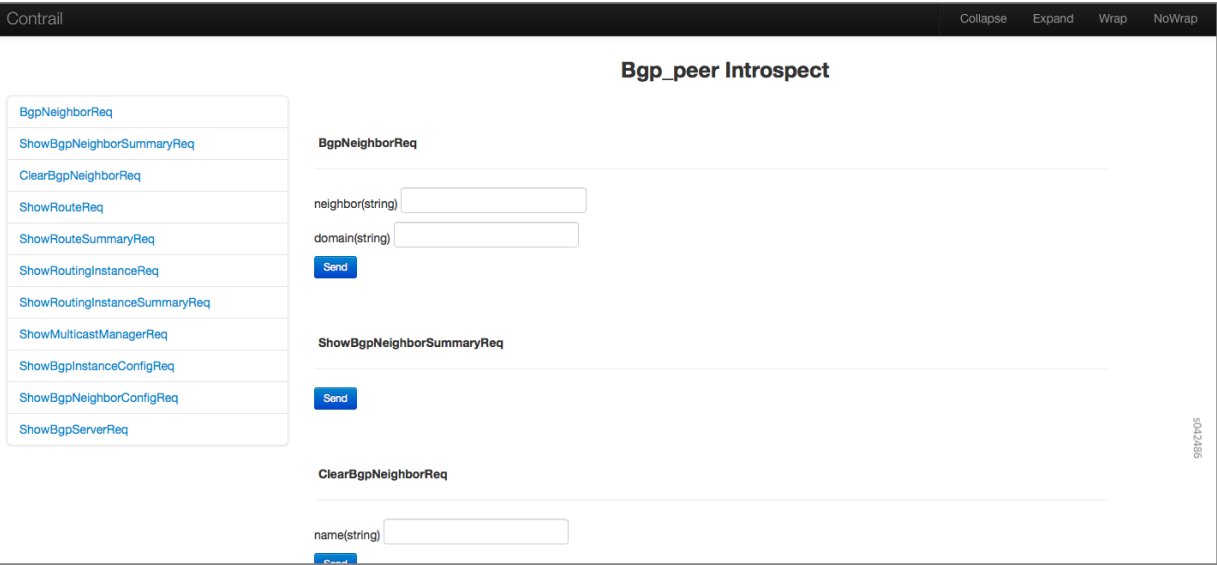


Figure 84 on page 426 shows an example of the BGP Neighbor Summary Introspect page.

Figure 84: BGP Neighbor Summary Introspect Page

Contrail										
ShowBgpNeighborSummaryResp										
neighbors										
peer	deleted	deleted_at	peer_address	peer_id	peer_asn	encoding	peer_type	state	local_address	local_id
b6s23	false	-	192.168.68.1	192.168.68.1	64512	BGP	internal	Established	192.168.68.2	192.168.68.2
b6s25	false	-	192.168.68.3	192.168.68.3	64512	BGP	internal	Established	192.168.68.2	192.168.68.2
mx1	false	-	192.168.100.1	192.168.100.1	64512	BGP	internal	Established	192.168.68.2	192.168.68.2
mx2	false	-	192.168.100.2	192.168.100.2	64512	BGP	internal	Established	192.168.68.2	192.168.68.2
b6s28	false	-	192.168.68.6	-	0	XMPP	internal	Established	192.168.68.2	-
b6s18	false	-	192.168.69.5	-	0	XMPP	internal	Established	192.168.68.2	-
b6s13	false	-	192.168.69.8	-	0	XMPP	internal	Established	192.168.68.2	-
b6s7	false	-	192.168.69.11	-	0	XMPP	internal	Established	192.168.68.2	-
b6s33	false	-	192.168.68.11	-	0	XMPP	internal	Established	192.168.68.2	-
b6s9	false	-	192.168.69.10	-	0	XMPP	internal	Established	192.168.68.2	-
b6s26	false	-	192.168.68.4	-	0	XMPP	internal	Established	192.168.68.2	-

The following are the Sandesh modules for the Contrail vRouter agent (**contrail-vrouter-agent**) Introspect port.

- agent.xml
- agent_stats_interval.xml
- cfg.xml
- controller.xml
- cpuinfo.xml
- diag.xml
- discovery_client_stats.xml
- flow_stats_interval.xml
- ifmap_agent.xml
- kstate.xml
- multicast.xml
- pkt.xml
- port_ipc.xml
- sandesh_trace.xml

- sandesh_uve.xml
- services.xml
- stats_interval.xml
- task.xml
- xmpp_server.xml

Figure 85 on page 427 shows an example of the Agent (agent.xml) Introspect page.

Figure 85: Agent Introspect Page

Contrail

CollapseExpandWrapNoWrap

AgentXmppConnectionStatus

peer									
controller_ip	state	cfg_controller	mcast_controller	last_state	last_event	last_state_at	flap_count	flap_time	rx
192.168.68.3	Established	Yes	No	OpenSent	xmsm::EvXmppKeepalive	2015-Jul-21 01:20:57.616019	2	2015-Jul-21 01:20:57.555077	rx
192.168.68.2	Established	No	Yes	OpenSent	xmsm::EvXmppKeepalive	2015-Jul-21 01:20:59.599875	2	2015-Jul-21 01:20:59.548692	rx

Monitor > Infrastructure > Dashboard

IN THIS SECTION

- [Monitor Dashboard | 428](#)
- [Monitor Individual Details from the Dashboard | 428](#)
- [Using Bubble Charts | 429](#)
- [Color-Coding of Bubble Charts | 430](#)

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 86 on page 428](#).

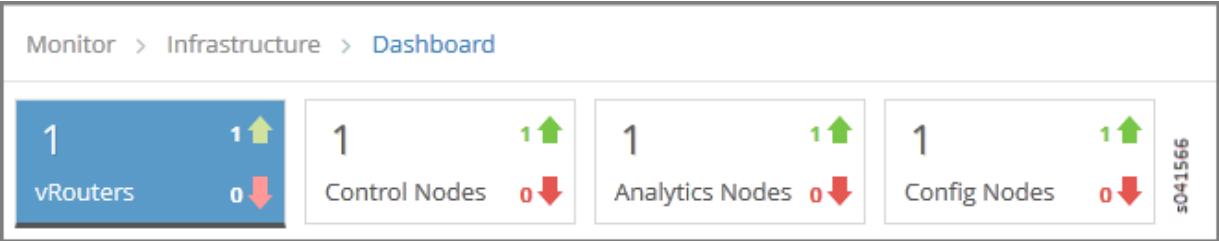
Figure 86: Monitor > Infrastructure > Dashboard



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 87 on page 429](#). 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 87: Dashboard Summary Boxes



Detailed information about monitoring each of the areas represented by the boxes is provided in the links in [Table 30 on page 429](#).

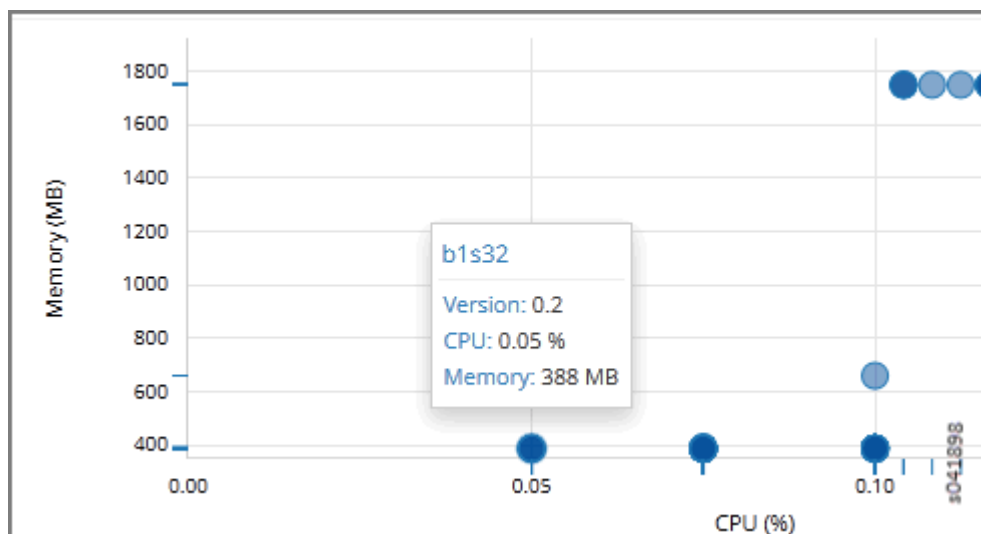
Table 30: Dashboard Summary Boxes

Box	For More Information
vRouters	"Monitor > Infrastructure > Virtual Routers" on page 442
Control Nodes	"Monitor > Infrastructure > Control Nodes" on page 431
Analytics Nodes	"Monitor > Infrastructure > Analytics Nodes" on page 456
Config Nodes	"Monitor > Infrastructure > Config Nodes" on page 464

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 88 on page 430](#). You can click through the summary information to get more details about the component.

Figure 88: 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](#) | 442

[Monitor > Infrastructure > Control Nodes](#) | 431

[Monitor > Infrastructure > Analytics Nodes](#) | 456

[Monitor > Infrastructure > Config Nodes](#) | 464

Monitor > Infrastructure > Control Nodes

IN THIS SECTION

- [Monitor Control Nodes Summary | 431](#)
- [Monitor Individual Control Node Details | 432](#)
- [Monitor Individual Control Node Console | 434](#)
- [Monitor Individual Control Node Peers | 437](#)
- [Monitor Individual Control Node Routes | 439](#)

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 89 on page 431](#). See [Table 31 on page 432](#) for descriptions of the fields on this screen.

Figure 89: Control Nodes Summary



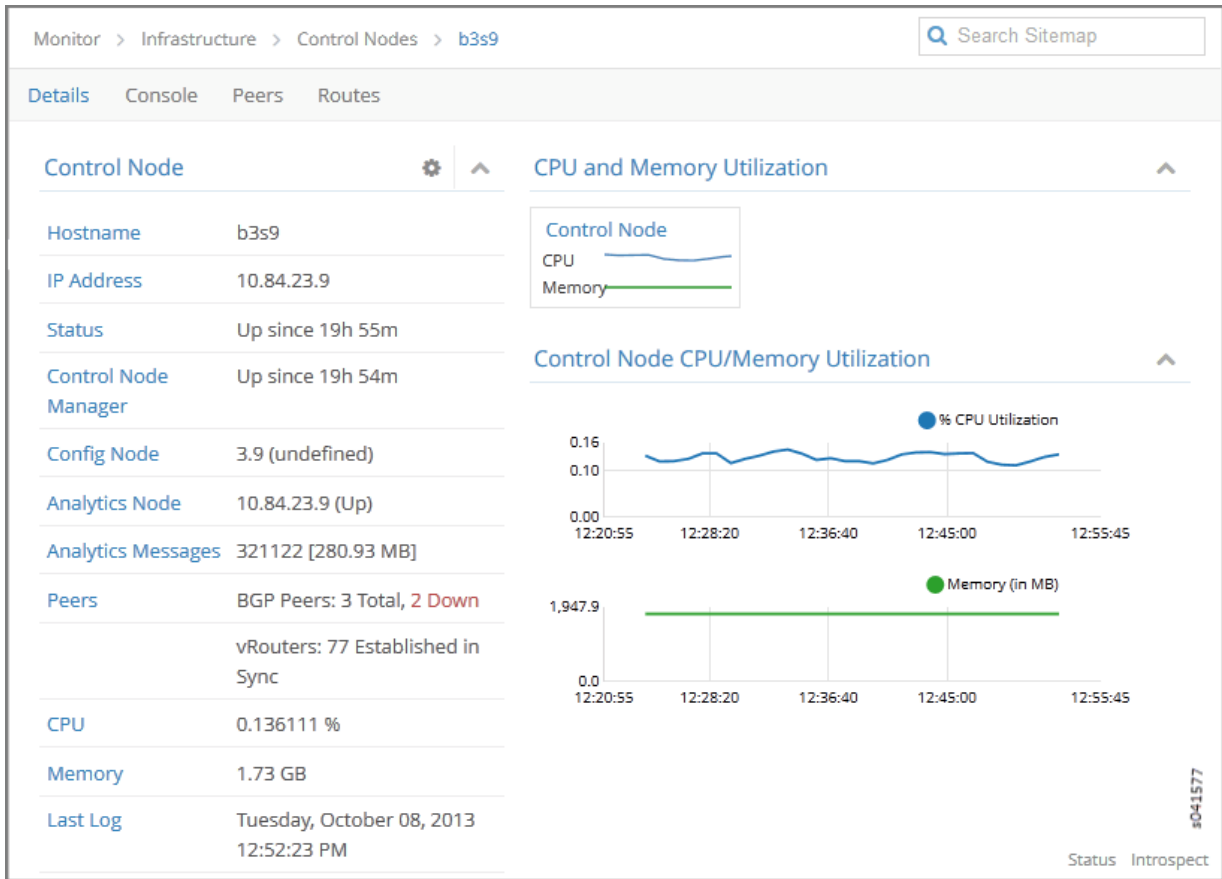
Table 31: Control Nodes Summary Fields

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 90 on page 433](#).

Figure 90: 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 32 on page 433](#) for descriptions of the fields on this tab.

Table 32: 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 32: 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 91 on page 435](#).

Figure 91: Individual Control Node—Console Tab

Monitor > Infrastructure > Control Nodes > b3s9

Search Sitemap

Details
Console
Peers
Routes

Console Logs

Time Range

Custom

From Time

Oct 08, 2013 02:26:33 PM

To Time

Oct 08, 2013 02:31:33 PM

Log Category

All

Log Type

any

Log Level

SYS_DEBUG

Limit

Limit 10 mess

Auto Refresh

☒

Display Logs
Reset

Time	Category	Log Type	Log
2013-10-08 14:31:30:351:353	BGP	BgpStateMachineSessionMessageLog	Bgp Peer 10.84.23.252 : P fsm::EvConnectTimerExp
2013-10-08 14:31:27:971:482	BGP	BgpStateMachineSessionMessageLog	Bgp Peer 10.84.23.253 : P state Connect
2013-10-08 14:31:24:970:157	BGP	BgpStateMachineSessionMessageLog	Bgp Peer 10.84.23.253 : P fsm::EvConnectTimerExp
2013-10-08 14:30:58:220:866	BGP	BgpStateMachineSessionMessageLog	Bgp Peer 10.84.23.252 : P state Connect

See [Table 33 on page 435](#) for descriptions of the fields on the **Console** tab screen.

Table 33: 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: <ol style="list-style-type: none"> All _default_ XMPP TCP

Table 33: Control Node: Console Tab Fields *(Continued)*

Field	Description
Log Type	Select a log type to display.
Log Level	<p>Select a log severity level to display:</p> <ol style="list-style-type: none"> 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	<p>Select from a list an amount to limit the number of messages displayed:</p> <ol style="list-style-type: none"> 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 33: 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 92 on page 438](#).

Figure 92: Individual Control Node—Peers Tab

Monitor > Infrastructure > Control Nodes > b3s9

Search Sitemap

Details
Console
Peers
Routes

Peers

Peer	Peer Type	Peer ASN	Status	Last flap	Messages (Recv/Sent)
10.84.23.252	BGP	64512	Active, -	-	0/ 0
10.84.23.8	BGP	64512	Established, in sync	-	3754/ 3758
10.84.23.253	BGP	64512	Connect, -	-	0/ 0
10.84.21.4	XMPP	-	Established, in sync	-	2751/ 5189
10.84.21.5	XMPP	-	Established, in sync	-	2753/ 5802
10.84.21.6	XMPP	-	Established, in sync	-	2752/ 4264
10.84.21.34	XMPP	-	Established, in sync	-	2753/ 5659

Details :

```

- {
  name: "b3s9:10.84.21.34",
  value: - {
    xmppPeerInfoData: - {
      state_info: - {
        last_state: "Active",
        state: "Established",
        last_state_at: 1381190447915913
      },
      peer_stats_info: - {

```

See Table 34 on page 438 for descriptions of the fields on the **Peers** tab screen.

Table 34: 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 34: 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 93 on page 439](#).

Figure 93: Individual Control Node—Routes Tab

Routing Table	Prefix	Protocol	Source	Next hop	Label	Secur...	Origin VN
bgp.l3vpn.0	10.84.21.1:13:192.168.30.240/32	XMPP	b1s1	10.84.21.1	28	3	default-domain:demo.v n30
		BGP	10.84.23.9	10.84.21.1	28	3	default-domain:demo.v n30
	10.84.21.1:14:192.168.31.242/32	XMPP	b1s1	10.84.21.1	29	3	default-domain:demo.v n31
		BGP	10.84.23.9	10.84.21.1	29	3	default-domain:demo.v n31
	10.84.21.1:1:192.168.2.231/32	XMPP	b1s1	10.84.21.1	16	3	default-domain:demo.v n2

See [Table 35 on page 440](#) for descriptions of the fields on the **Routes** tab screen.

Table 35: 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	<p>Select an address family for which to display the active routes:</p> <ol style="list-style-type: none"> 1. All (default) 2. l3vpn 3. inet 4. inetmcast
(Limit Field)	<p>Select to limit the display of active routes:</p> <ol style="list-style-type: none"> 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	<p>Select a protocol for which to display the active routes:</p> <ol style="list-style-type: none"> 1. All (default) 2. XMPP 3. BGP 4. ServiceChain 5. Static

Table 35: 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.
<i>Column</i>	<i>Description</i>
Routing Table	The name of the routing table that stores this route.
Prefix	The route prefix for each active route displayed.
Protocol	The protocol used by the route.
Source	The host source for each active route displayed.
Next hop	The IP address of the next hop for each active route displayed.
Label	The label for each active route displayed.
Security	The security value for each active route displayed.
Origin VN	The virtual network from which the route originates.
AS Path	The AS path for each active route displayed.

Monitor > Infrastructure > Virtual Routers

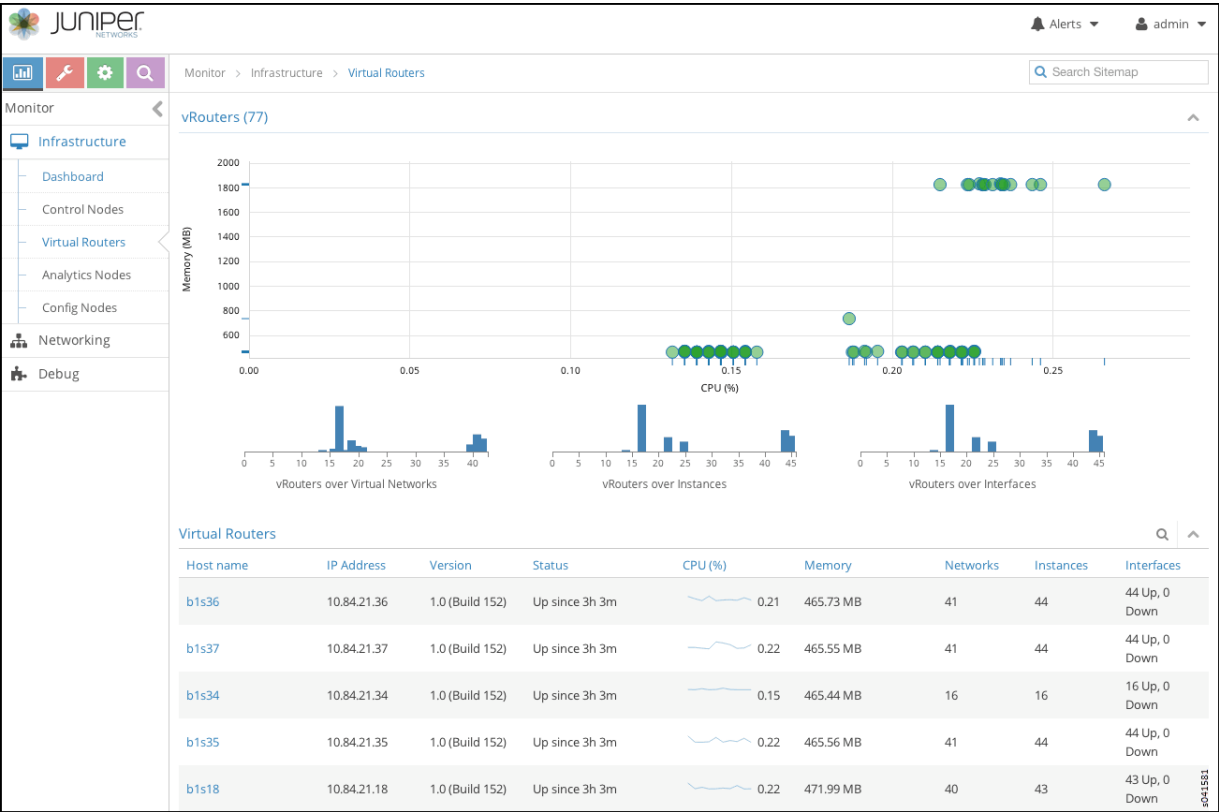
IN THIS SECTION

- [Monitor vRouters Summary | 442](#)
- [Monitor Individual vRouters Tabs | 444](#)
- [Monitor Individual vRouter Details Tab | 444](#)
- [Monitor Individual vRouters Interfaces Tab | 446](#)
- [Monitor Individual vRouters Networks Tab | 448](#)
- [Monitor Individual vRouters ACL Tab | 449](#)
- [Monitor Individual vRouters Flows Tab | 451](#)
- [Monitor Individual vRouters Routes Tab | 452](#)
- [Monitor Individual vRouter Console Tab | 453](#)

Monitor vRouters Summary

Click **Monitor > Infrastructure > Virtual Routers** to view the **vRouters** summary screen. See [Figure 94 on page 443](#).

Figure 94: vRouters Summary



See [Table 36 on page 443](#) for descriptions of the fields on the **vRouters Summary** screen.

Table 36: 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 36: 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 95 on page 445](#).
- **Console**—similar display of information as on individual control nodes **Console** tab. See [Figure 101 on page 454](#).
- **Interfaces**—details about associated interfaces. See [Figure 96 on page 447](#).
- **Networks**—details about associated networks. See [Figure 97 on page 448](#).
- **ACL**—details about access control lists. See [Figure 98 on page 450](#).
- **Flows**—details about associated traffic flows. See [Figure 99 on page 451](#).
- **Routes**—details about associated routes. See [Figure 100 on page 453](#).

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 95 on page 445](#). See [Table 37 on page 445](#) for descriptions of the fields on this tab.

Figure 95: Individual vRouters—Details Tab

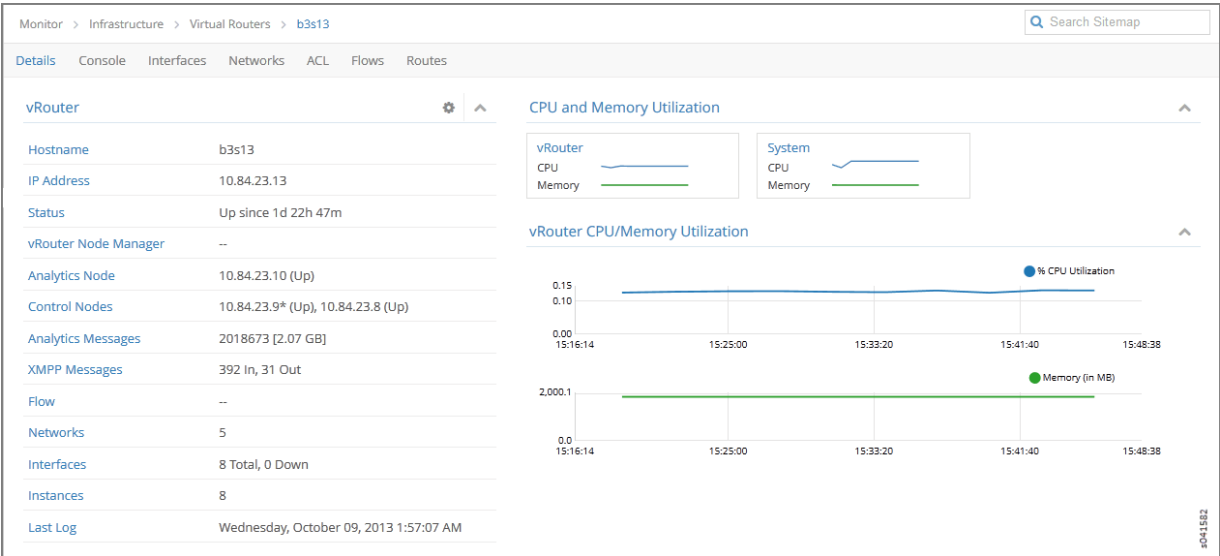


Table 37: vRouters Details Tab Fields

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 37: 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 96 on page 447](#). See [Table 38 on page 447](#) for descriptions of the fields on the **Interfaces** tab screen.

Figure 96: Individual vRouters—Interfaces Tab

Monitor > Infrastructure > Virtual Routers > b1s36

Q

Search Sitemap

Details
Console
Interfaces
Networks
ACL
Flows
Routes

Interfaces

Q ^

Name	Label	Status	Network	IP Address	Floating IP	Instance	
▶ tap25e5cee3-07	18	Up	default-domain:demo:vn30	192.168.30.247	None	005132fd-0d83-4db7-88c8-bd49d68e9480	⚙
▶ tap4d91aab1-f1	25	Up	default-domain:demo:vn26	192.168.26.247	None	65d6c6e9-7a82-43d8-a706-f74d81715920	⚙
▶ tap5a8cd9dd-5b	27	Up	default-domain:demo:vn23	192.168.23.249	None	a159c518-4fb6-402a-ae0d-eb5b4457b551	⚙
▶ tap603a5e0b-8b	16	Up	default-domain:demo:vn19	192.168.19.247	None	fe622580-b0cf-4c6d-89e5-d2065e7e87e4	⚙
▾ tap68ad232c-76	19	Up	default-domain:demo:vn28	192.168.28.247	None	91089d89-76b5-46c2-abc9-b9693bcb37ac	⚙

Details :

```

- {
  index: "6",
  name: "tap68ad232c-76",
  uuid: "68ad232c-76d1-4fe2-a200-42182497545e",
  vrf_name: "default-domain:demo:vn28:vn28",
  active: "Active",
  dhcp_service: "Enable",

```

#041583

Table 38: 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 38: 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 97 on page 448](#). See [Table 39 on page 449](#) for descriptions of the fields on the **Networks** tab screen.

Figure 97: Individual vRouters—Networks Tab

Monitor > Infrastructure > Virtual Routers > b1s36

Search Sitemap

Details Console Interfaces **Networks** ACL Flows Routes

Networks

Name	ACLs	VRF
▶ default-domain:demo:vn24	a372751f-6497-41e9-b409-fa4ab5ce6b7f	default-domain:demo:vn24:vn24
▶ default-domain:demo:vn22	195af177-0a28-49a1-9cf0-2ceac22af5a1	default-domain:demo:vn22:vn22
▶ default-domain:demo:vn30	362cce6e-2894-42d6-ba03-3ee98cac8809	default-domain:demo:vn30:vn30
▶ default-domain:demo:vn21	5918a068-1cd5-4993-9cff-386a807940ca	default-domain:demo:vn21:vn21
▶ default-domain:demo:vn28	dd87c461-97c0-4d47-bff0-89040e7d6ab0	default-domain:demo:vn28:vn28
▶ default-domain:demo:vn19	f0465432-6fc0-4fb3-967c-392100617408	default-domain:demo:vn19:vn19
▶ default-domain:demo:vn2	1c46e7e0-f799-4bc6-ae09-e4654c263aa6	default-domain:demo:vn2:vn2

Details :

```

- {
  name: "default-domain:demo:vn2",
  uuid: "63d08f7a-b342-4892-9171-edab9f4c397f",
  acl_uuid: "1c46e7e0-f799-4bc6-ae09-e4654c263aa6",
  mirror_acl_uuid: - {},
  mirror_cfg_acl_uuid: - {},
  vrf_name: "default-domain:demo:vn2:vn2",
  ipam_data: - {
    list: - {

```

s041584

Table 39: 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 98 on page 450](#). See [Table 40 on page 450](#) for descriptions of the fields on the **ACL** tab screen.

Figure 98: Individual vRouters—ACL Tab

Monitor > Infrastructure > Virtual Routers > b1s36

Search Sitemap

Details Console Interfaces Networks **ACL** Flows Routes

ACL

UUID	Flows	Action	Protocol	Source Network or Prefix	Source Port	Destination Network or Prefix	D
195af177-0a28-49a1-9cf0-2ceac22af5a1	8	pass	any	-	any	-	a
		pass	any	-	any	-	a
		pass	any	-	any	-	a
1c46e7e0-f799-4bc6-ae09-e4654c263aa6	8	pass	any	-	any	-	a

Details:

```
- {
  uuid: "1c46e7e0-f799-4bc6-ae09-e4654c263aa6",
  dynamic_acl: "false",
  entries: - {
    list: - {
      AclEntrySandeshData: - [
        - {
          ace_id: "1",
```

Table 40: 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 40: 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 arrow next to any ACL/SG UUID to reveal more details. Use the horizontal and vertical scroll bars to access all portions of the screen. See [Figure 99 on page 451](#). See [Table 41 on page 452](#) for descriptions of the fields on the **Flows** tab screen.

Figure 99: Individual vRouters—Flows Tab

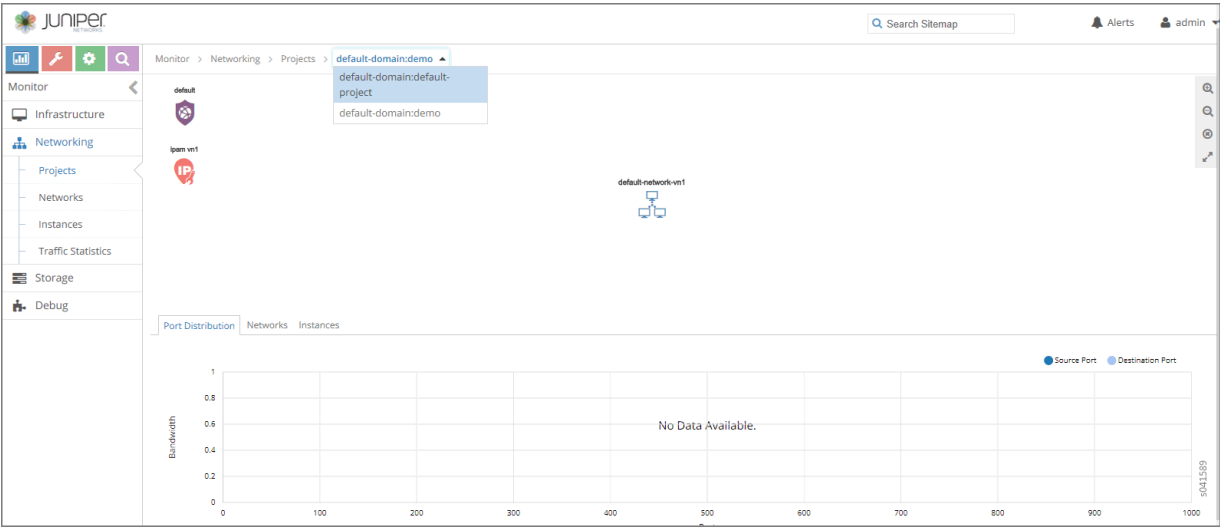


Table 41: vRouters: Flows Tab Fields

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 100 on page 453](#). See [Table 42 on page 453](#) for descriptions of the fields on the **Routes** tab screen.

Figure 100: Individual vRouters—Routes Tab

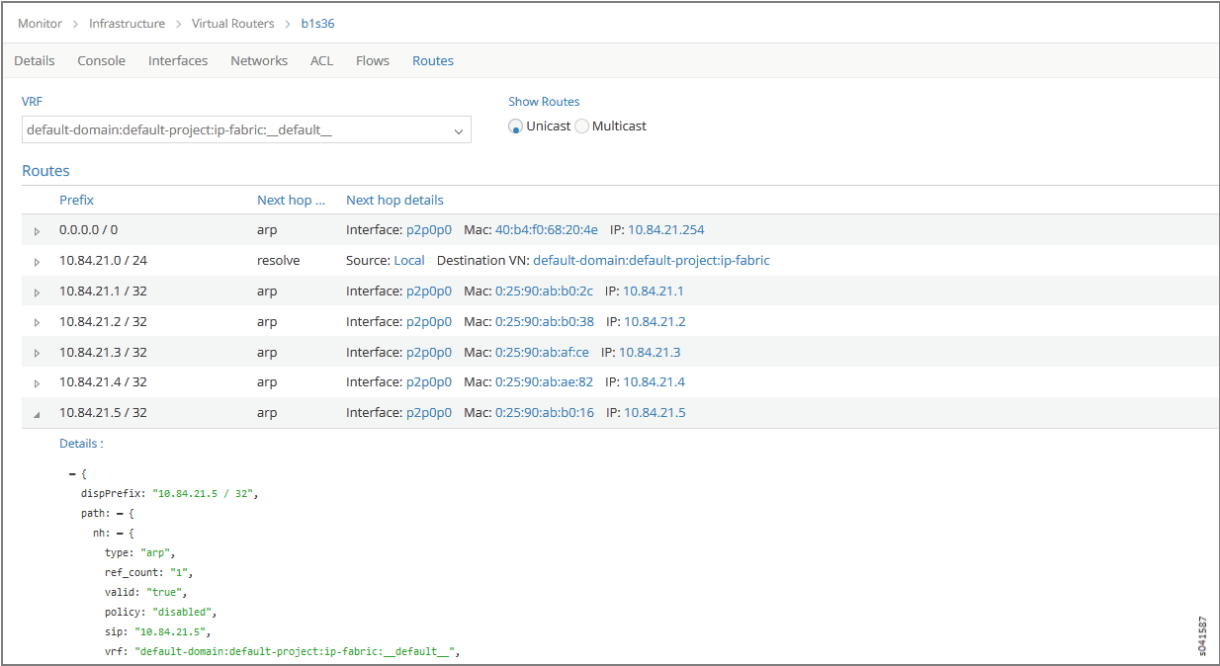


Table 42: vRouters: Routes Tab Fields

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 101 on page 454](#). See [Table 43 on page 454](#) for descriptions of the fields on the **Console** tab screen.

Figure 101: Individual vRouter—Console Tab

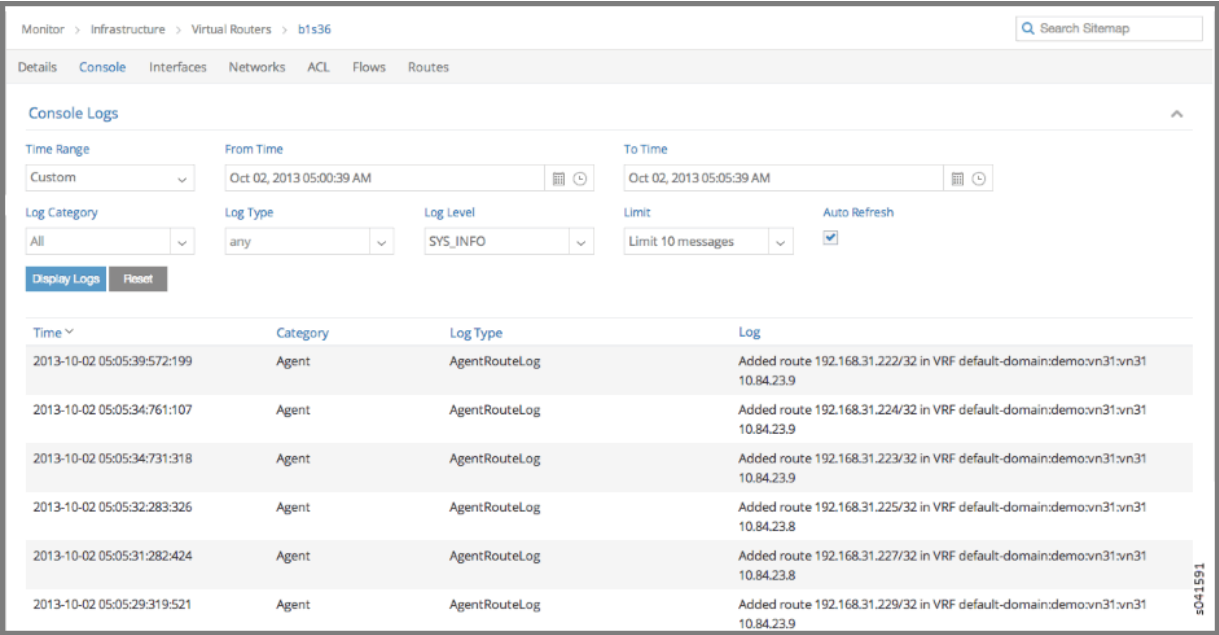


Table 43: Control Node: Console Tab Fields

Field	Description
Time Range	Select a timeframe for which to review logging information as sent to the console. There are several options, ranging from Last 5 mins through to the Last 24 hrs , plus a Custom time range.
From Time	If you select Custom in Time Range , enter the start time.
To Time	If you select Custom in Time Range , enter the end time.
Log Category	Select a log category to display: <ul style="list-style-type: none">All_default_XMPPTCP
Log Type	Select a log type to display.

Table 43: Control Node: Console Tab Fields *(Continued)*

Field	Description
Log Level	Select a log severity level to display: <ul style="list-style-type: none">• SYS_EMERG• SYS_ALERT• SYS_CRIT• SYS_ERR• SYS_WARN• SYS_NOTICE• SYS_INFO• SYS_DEBUG
Limit	Select from a list an amount to limit the number of messages displayed: <ul style="list-style-type: none">• No Limit• Limit 10 messages• Limit 50 messages• Limit 100 messages• Limit 200 messages• Limit 500 messages
Auto Refresh	Click the check box to automatically refresh the display if more messages occur.
Display Logs	Click this button to refresh the display if you change the display criteria.
Reset	Click this button to clear any selected display criteria and reset all criteria to their default settings.

Columns

Table 43: 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 | 456](#)
- [Monitor Analytics Individual Node Details Tab | 458](#)
- [Monitor Analytics Individual Node Generators Tab | 459](#)
- [Monitor Analytics Individual Node QE Queries Tab | 460](#)
- [Monitor Analytics Individual Node Console Tab | 461](#)

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 102 on page 457](#). See [Table 44 on page 457](#) for descriptions of the fields on the analytics summary.

Figure 102: Analytics Nodes Summary



Table 44: 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 103 on page 458](#).

See [Table 45 on page 458](#) for descriptions of the fields on this screen.

Figure 103: Monitor Analytics Individual Node Details Tab

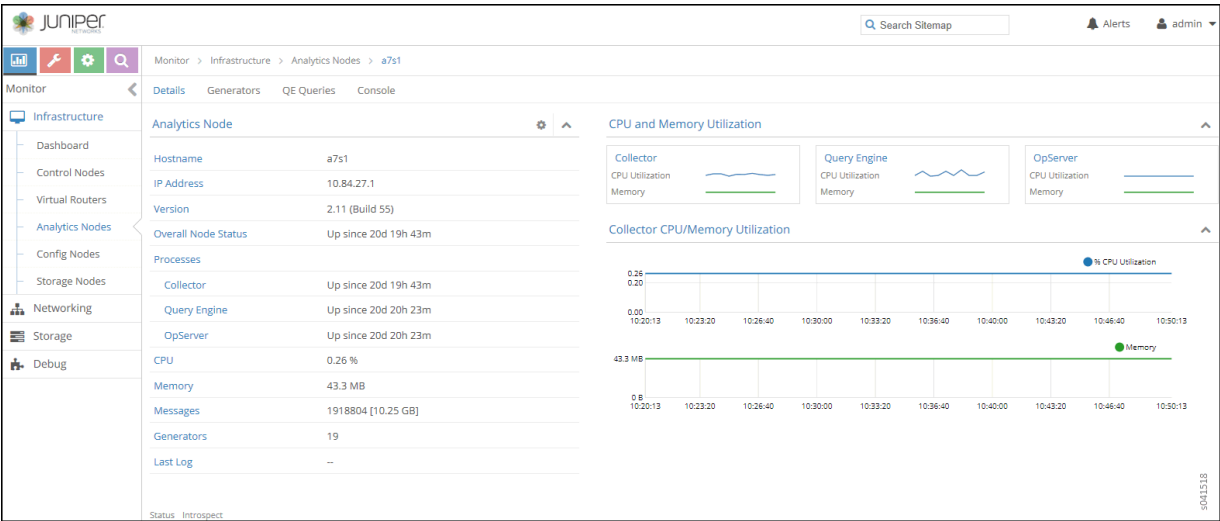


Table 45: Monitor Analytics Individual Node Details Tab Fields

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 45: 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 104 on page 459](#). Click the expansion arrow next to any generator name to reveal more details. See [Table 46 on page 460](#) for descriptions of the fields on the **Peers** tab screen.

Figure 104: Individual Analytics Node—Generators Tab

Name	Status	Messages	Bytes
a7s1:Analytics:contrail-analytics-api:0	Up since 20d 23h 57m, Connected since 20d 23h 16m	476046	1.25 GB
a7s1:Analytics:contrail-analytics-node mgr:0	Up since 20d 23h 56m, Connected since 20d 23h 16m	5	14.32 KB
a7s1:Analytics:contrail-collector:0	Up since 20d 23h 16m, Connected since 20d 23h 16m	1932437	10.25 GB
a7s1:Analytics:contrail-query-engine:0	Up since 20d 23h 57m, Connected since 20d 23h 16m	928348	1.62 GB
a7s1:Analytics:contrail-snmp-collector:0	Up since 20d 23h 57m, Connected since 20d 23h 16m	3	4.5 KB
a7s1:Analytics:contrail-topology:0	Up since 20d 23h 57m, Connected since 20d 23h 16m	3	4.46 KB
a7s1:Compute:Storage-Stats-mgr:0	Up since 20d 23h 15m, Connected since 20d 23h 15m	947488	1.22 GB
a7s1:Compute:contrail-router-agent:0	Up since 20d 23h 57m, Connected since 20d 23h 16m	314603	1.03 GB

Table 46: 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 105 on page 460](#).

See [Table 47 on page 460](#) for descriptions of the fields on the **QE Queries** tab screen.

Figure 105: Individual Analytics Node—QE QueriesTab

Monitor > Infrastructure > Analytics Nodes > b3s10

Search Sitemap

Details

Console

Generators

QE Queries

QE Queries

Q

^

Enqueue Time

Query

Progress

No QE Queries to display

<<

<

0

>

>>

50

Records per page

Table 47: 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 47: 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 106 on page 461](#). See [Table 48 on page 461](#) for descriptions of the fields on the **Console** tab screen.

Figure 106: Analytics Individual Node—Console Tab

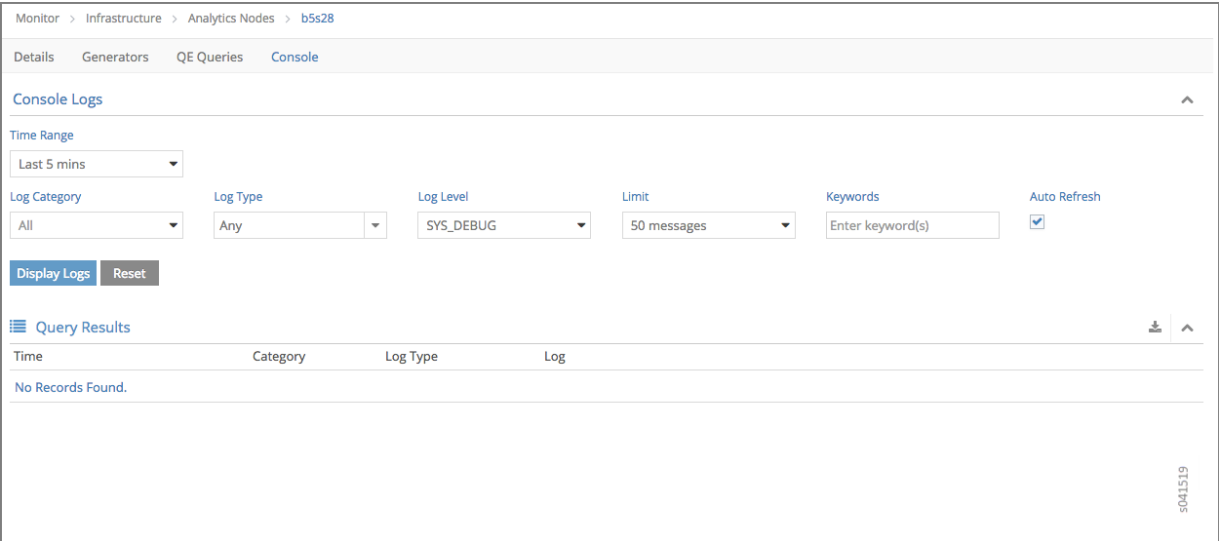


Table 48: Monitor Analytics Individual Node Console Tab Fields

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 48: Monitor Analytics Individual Node Console Tab Fields *(Continued)*

Field	Description
Log Category	<p>Select a log category to display:</p> <ol style="list-style-type: none"> 1. All 2. _default_ 3. XMPP 4. TCP
Log Type	Select a log type to display.
Log Level	<p>Select a log severity level to display:</p> <ol style="list-style-type: none"> 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 48: Monitor Analytics Individual Node Console Tab Fields *(Continued)*

Field	Description
(Limit field)	<p>Select the number of messages to display:</p> <ol style="list-style-type: none"> 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

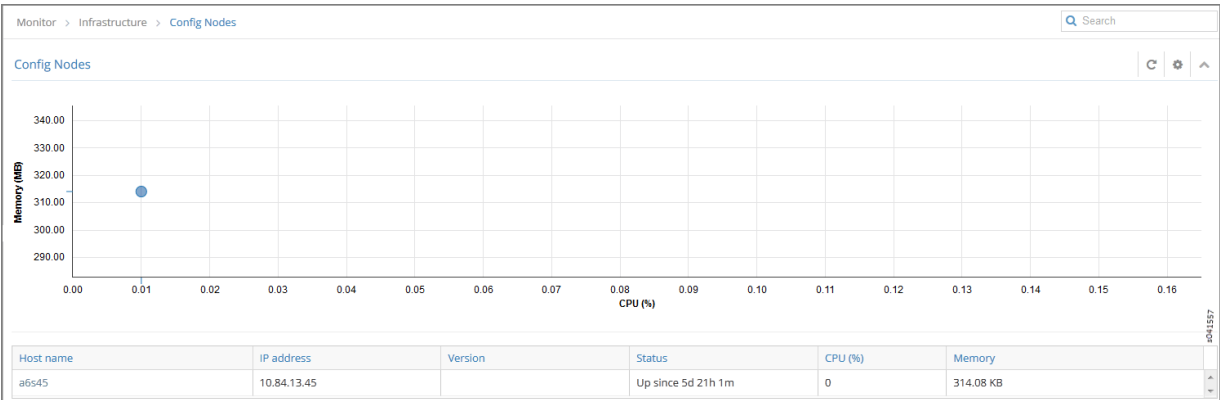
- [Monitor Config Nodes | 464](#)
- [Monitor Individual Config Node Details | 465](#)
- [Monitor Individual Config Node Console | 466](#)

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 107 on page 464](#).

Figure 107: Config Nodes Summary



[Table 49 on page 464](#) describes the fields in the Config Nodes summary.

Table 49: Config Nodes Summary Fields

Field	Description
Host name	The name of this node.

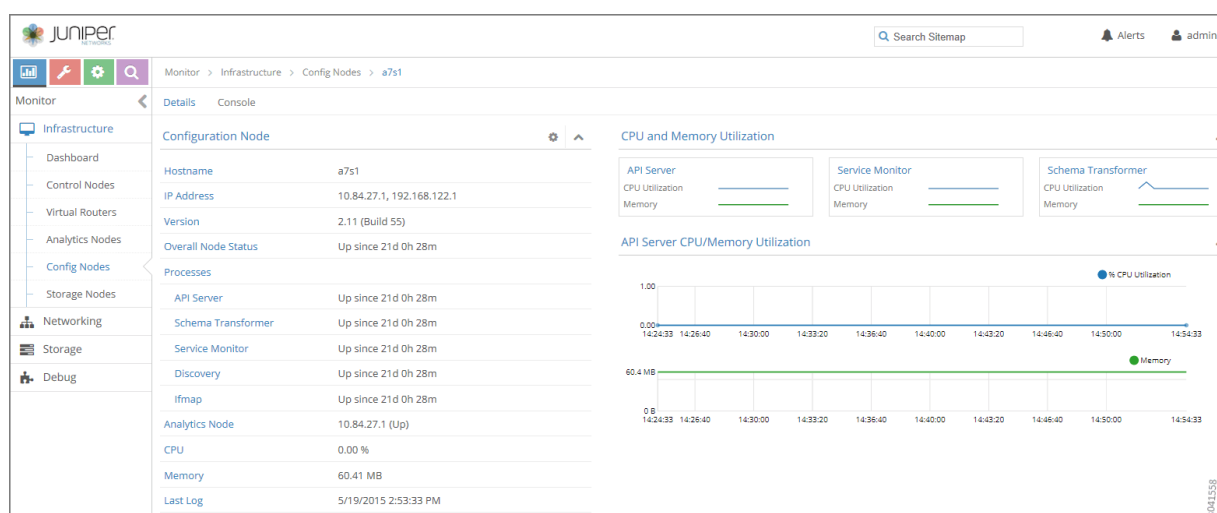
Table 49: Config Nodes Summary Fields *(Continued)*

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 108 on page 465](#).

Figure 108: Individual Config Nodes— Details Tab



[Table 50 on page 466](#) describes the fields on the Details screen.

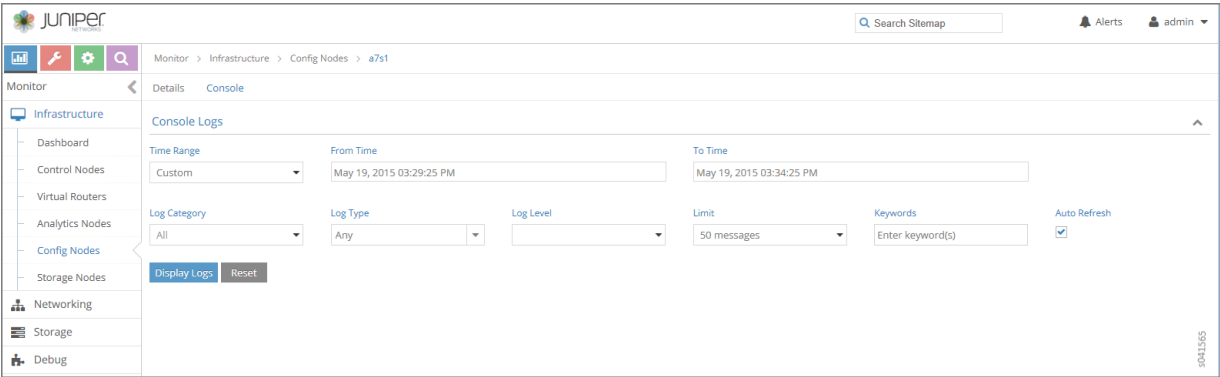
Table 50: Individual Config Nodes— Details Tab Fields

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 109 on page 467](#).

Figure 109: Individual Config Node—Console Tab



See [Table 51 on page 467](#) for descriptions of the fields on the **Console** tab screen.

Table 51: 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: <ol style="list-style-type: none"> 1. All 2. Limit 10 messages 3. Limit 50 messages 4. Limit 100 messages 5. Limit 200 messages 6. Limit 500 messages

Table 51: 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

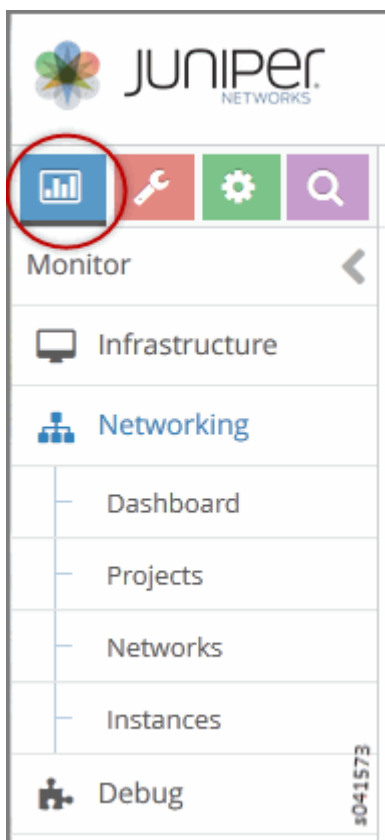
- [Monitor > Networking Menu Options | 468](#)
- [Monitor -> Networking -> Dashboard | 469](#)
- [Monitor > Networking > Projects | 471](#)
- [Monitor Projects Detail | 472](#)
- [Monitor > Networking > Networks | 475](#)

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 110 on page 469](#) shows the menu options available under **Monitor > Networking**.

Figure 110: 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 111 on page 470](#).

Figure 111: Traffic Statistics for Domain Window

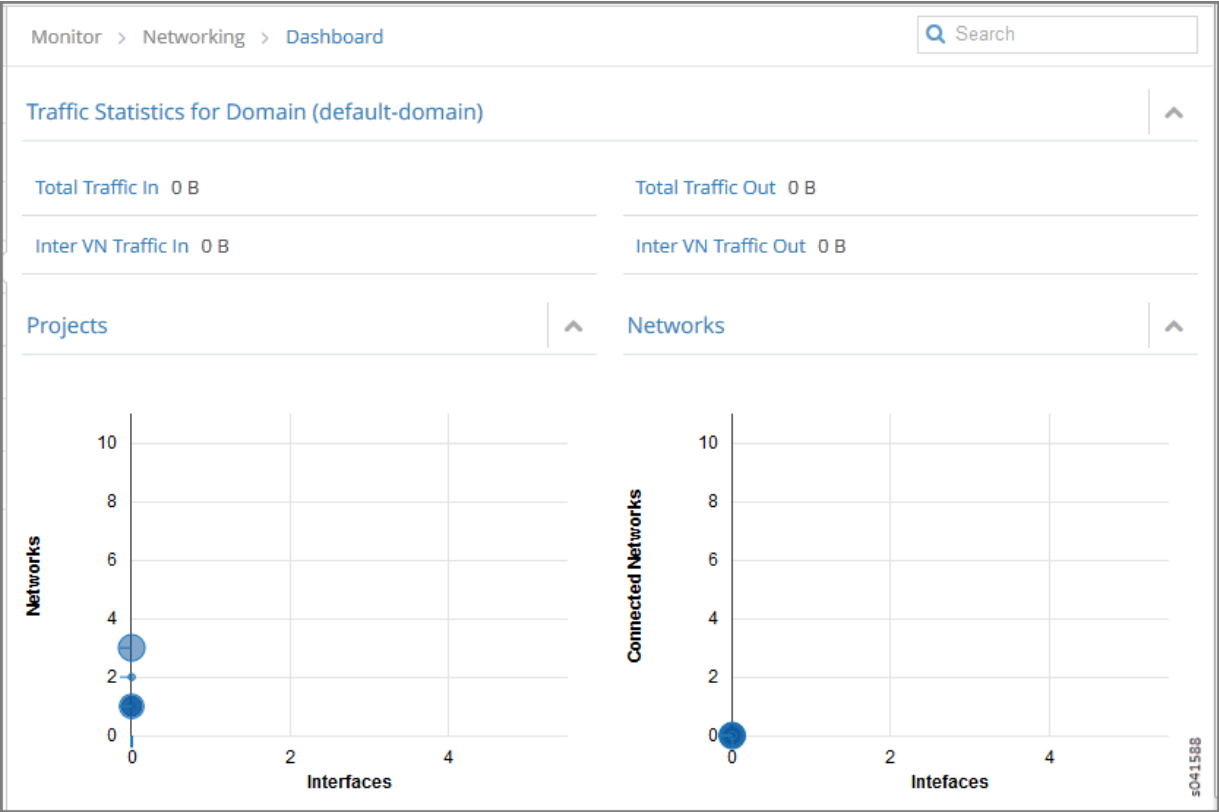


Table 52 on page 470 describes the fields in the Traffic Statistics for Domain window.

Table 52: 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.

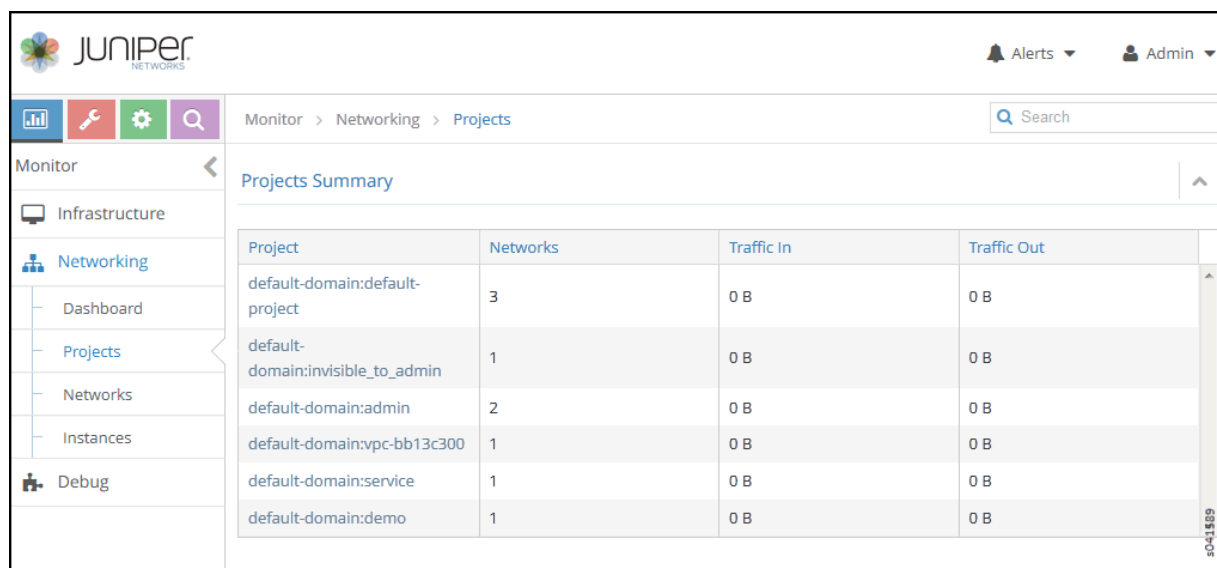
Table 52: Projects Summary Fields *(Continued)*

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 112 on page 471](#).

Figure 112: Monitor > Networking > Projects



Project	Networks	Traffic In	Traffic Out
default-domain:default-project	3	0 B	0 B
default-domain:invisible_to_admin	1	0 B	0 B
default-domain:admin	2	0 B	0 B
default-domain:vpc-bb13c300	1	0 B	0 B
default-domain:service	1	0 B	0 B
default-domain:demo	1	0 B	0 B

See [Table 53 on page 472](#) for descriptions of the fields on this screen.

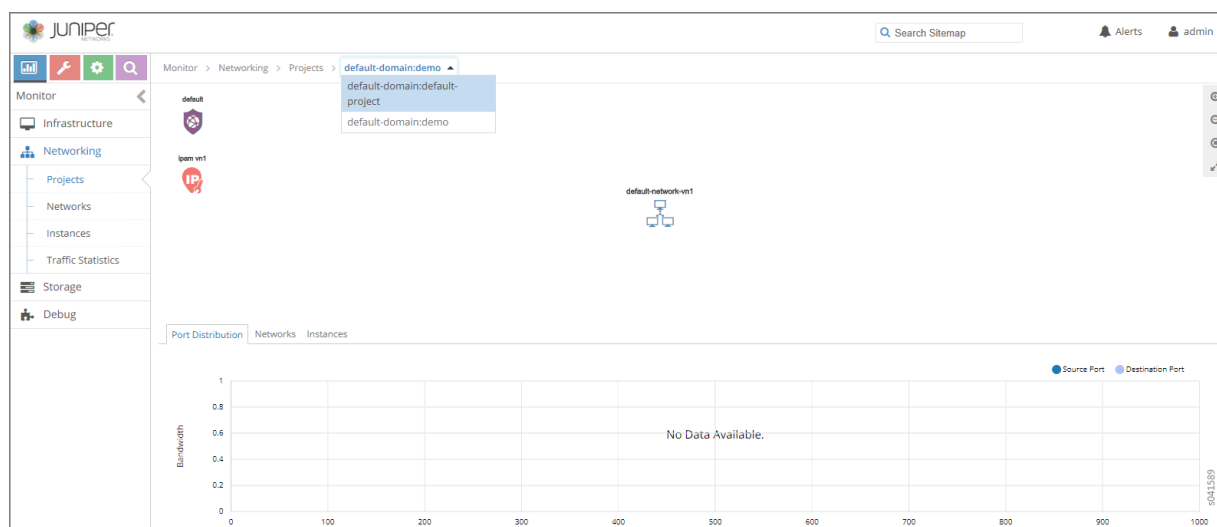
Table 53: Projects Summary Fields

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 113 on page 472](#). Hover over any of the connections to get more details.

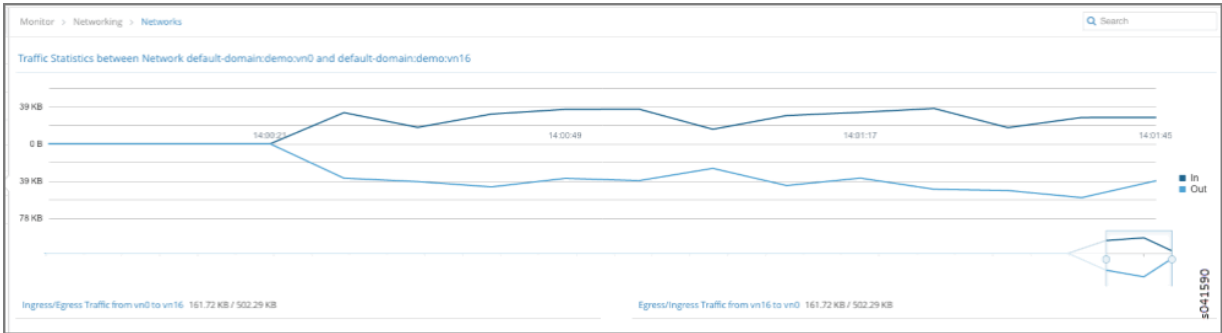
Figure 113: Monitor Projects Connectivity Details



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 114 on page 473](#).

Figure 114: 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 115: Projects Instances Summary

Monitor > Networking > Projects > default-domain:admin

Search

Monitor

Infrastructure

Networking

Dashboard

Projects

Networks

Instances

Debug

Summary

Instances

Instances Summary

	Instance	Virtual Network	Interfaces	vRouter	IP Address	Floating IP	Traffic (In/Out)
	out	default-domain:admin:right	1	hp1	2.2.2.252		129.87 KB / 119.83 KB
	NAT1_1	default-domain:admin:right	1	hp1	2.2.2.253 250.250.1.253 (1 more)		3.69 MB / 1.15 MB
	in	default-domain:admin:left	1	hp1	1.1.1.252		132.75 KB / 122.02 KB

041953

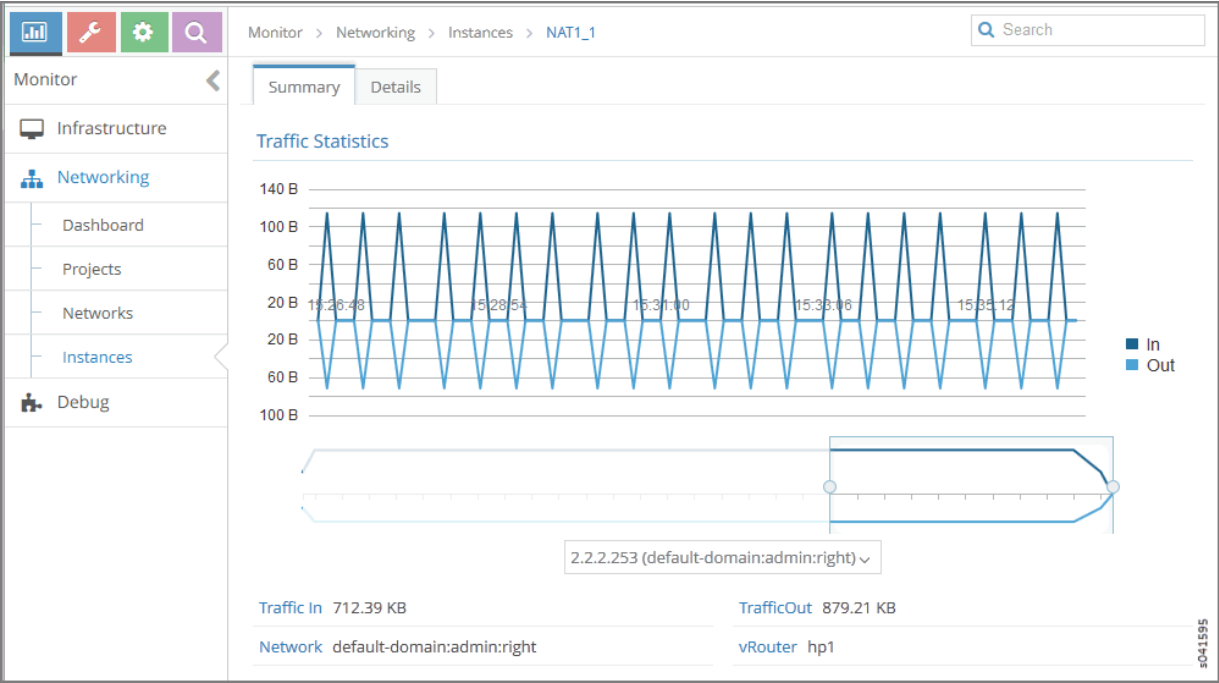
See [Table 3](#) for a description of the fields on this screen.

Table 54: 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 116 on page 475](#).

Figure 116: Instance Traffic Statistics



Monitor > Networking > Networks

Select **Monitor > Networking > Networks** to view a summary of the virtual networks in your system. See [Figure 117 on page 475](#).

Figure 117: Network Summary

Monitor > Networking > Networks

Networks Summary

Network	Instances	Traffic (In/Out) (Last 1 hr)	Throughput (In/Out)
default-domain:default-project__link_local__	0	0 B / 0 B	0 bps / 0 bps
default-domain:default-project:default-virtual-network	0	0 B / 0 B	0 bps / 0 bps
default-domain:default-project:ip-fabric	0	0 B / 0 B	0 bps / 0 bps
default-domain:demo:default-network-vn1	0	0 B / 0 B	0 bps / 0 bps

Ingress Flows 0
Egress Flows 0
ACL Rules 2
Interfaces 0
Total Traffic(In/Out) -/-

Total: 4 records 50 Records

Table 55: 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 118 on page 476 shows the **Summary** tab for an individual network, which displays connectivity details and traffic statistics for the selected network.

Figure 118: Individual Network Connectivity Details—Summary Tab

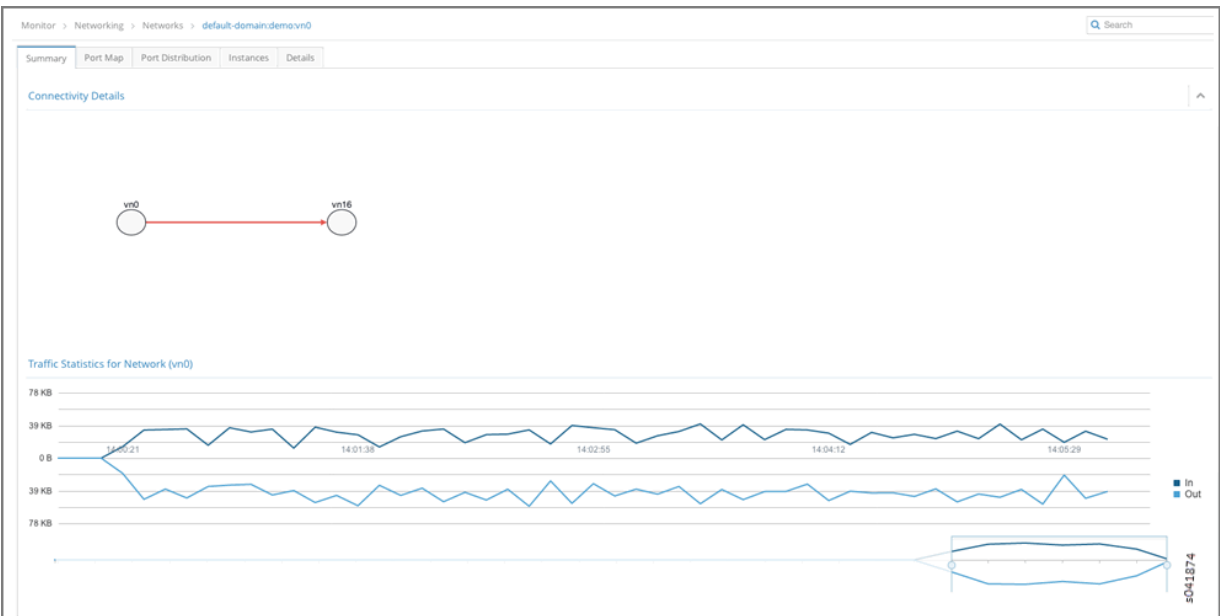


Figure 119 on page 477 shows the **Port Map** tab for an individual network, which displays the relative distribution of traffic for this network by protocol, by port.

Figure 119: Individual Network-- Port Map Tab

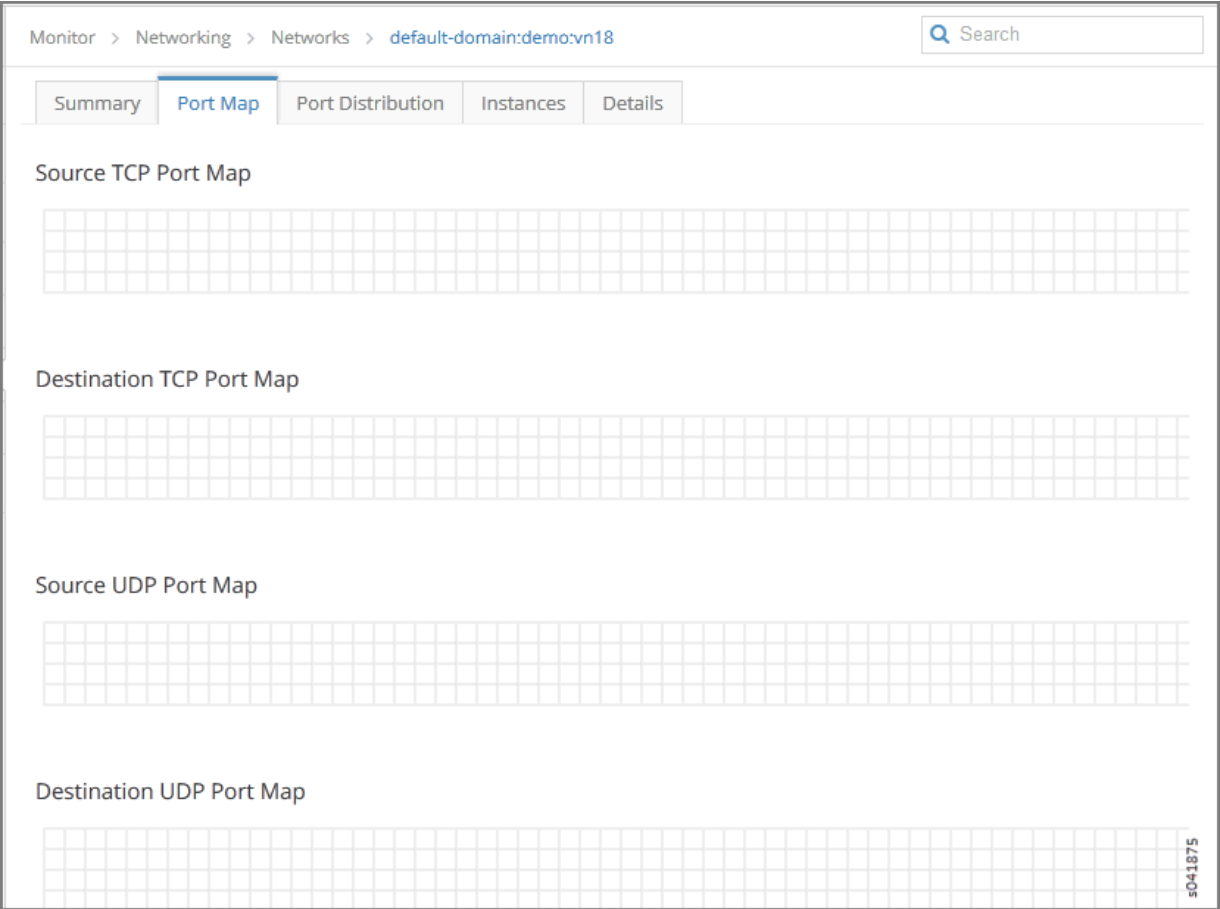


Figure 120 on page 478 shows the **Port Distribution** tab for an individual network, which displays the relative distribution of traffic in and out by source port and destination port.

Figure 120: Individual Network-- Port Distribution Tab

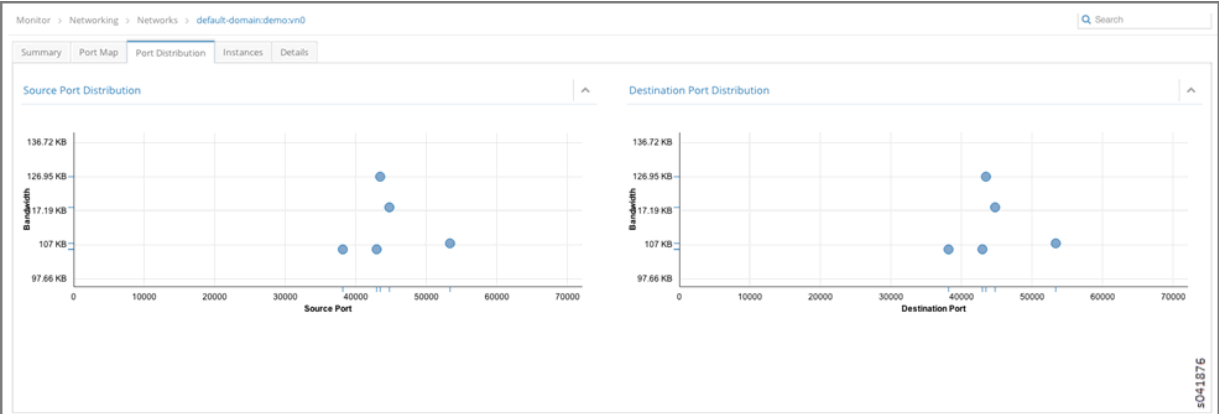


Figure 121 on page 479 shows the **Instances** tab for an individual network, which displays details for each instance associated with this network, including the number of interfaces, the associated vRouter, the instance IP address, and the volume of traffic in and out.

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 121: Individual Network Instances Tab

Monitor > Networking > Networks > default-domain:demo:vn18

Q Search

Summary

Port Map

Port Distribution

Instances

Details

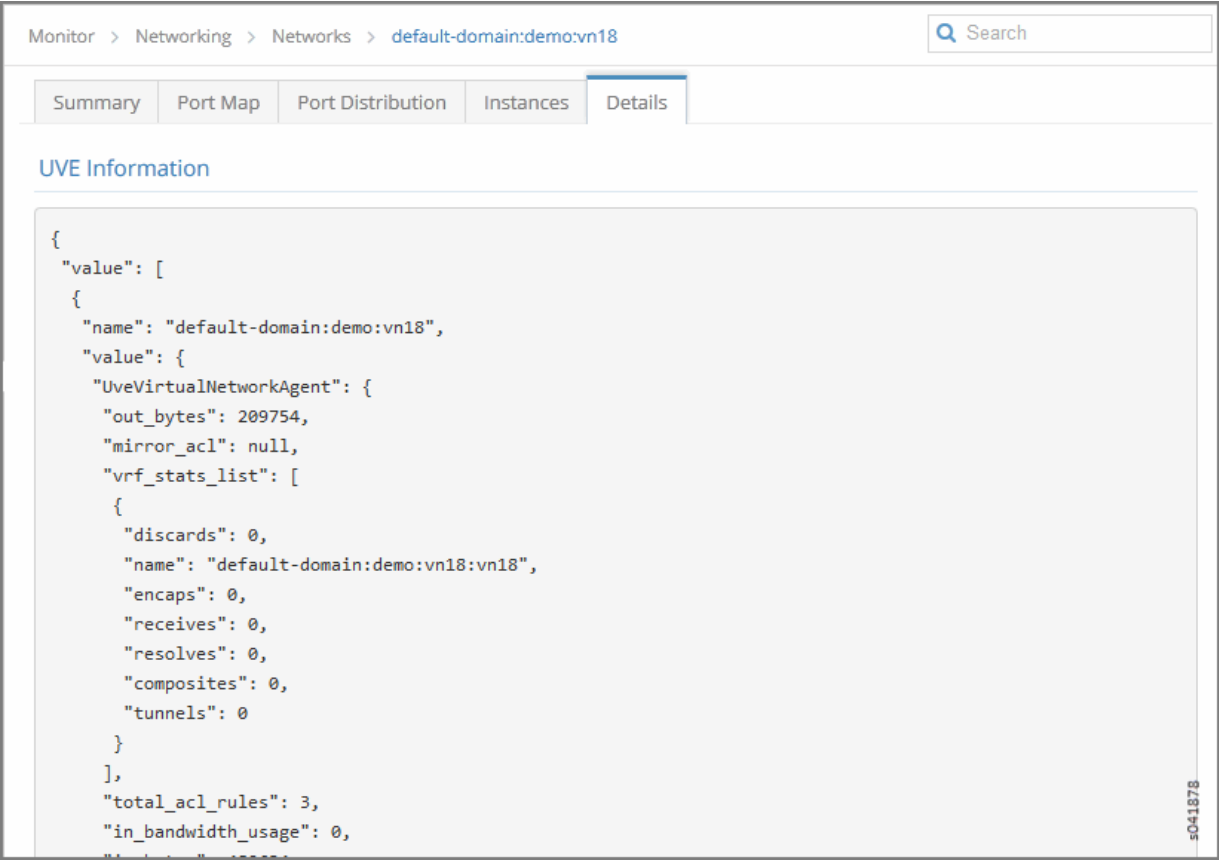
Instances Summary

	Instance	Interfaces	vRouter	IP Address	Floating IP	Traffic (In/Out)	
▶	vn18_vm-b342ca93-9acd-4275-acb8-df7b5843884c	1	b1s29	192.168.18.225		1.13 KB / 712.00 B	
▲	vn18_vm-22a42bf6-fccc-4db3-b5ac-80082bbebfef	1	b1s42	192.168.18.236		1.13 KB / 712.00 B	
<div><div>Interfaces</div><div>IP Address: 192.168.18.236 Label: 17 Mac Address: 02:e9:94:e7:0e:56 Network: default-domain:demo:vn18 Traffic (In/Out): 1.13 KB/712.00 B</div><div>UUID</div><div>22a42bf6-fccc-4db3-b5ac-80082bbebfef</div><div>CPU</div><div>0.01</div><div>Memory</div><div>1.23 GB / 15.63 GB</div><div>(Used/Total)</div></div>							
▶	vn18_vm-f676567a-826f-4e9d-9a81-b4649b7fcde2	1	b1s15	192.168.18.235		1.13 KB / 712.00 B	

5041877

Figure 122 on page 480 shows the **Details** tab for an individual network, which displays the code used to define this network --the User Virtual Environment (UVE) code.

Figure 122: Individual Network Details Tab



Query > Flows

IN THIS SECTION

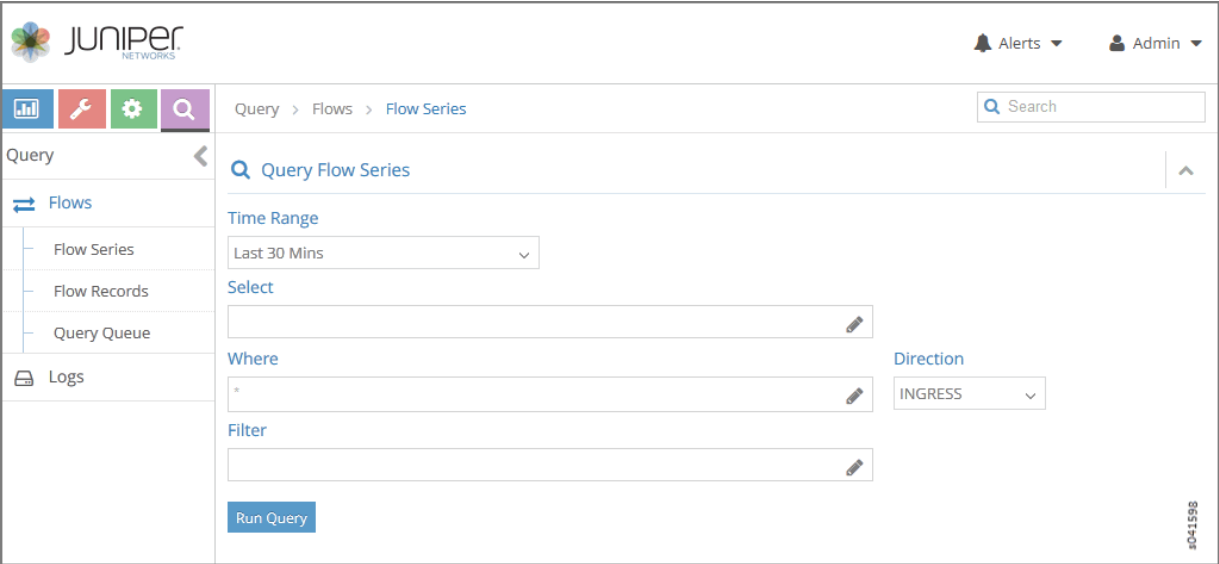
- [Query > Flows > Flow Series | 481](#)
- [Example: Query Flow Series | 484](#)
- [Query > Flow Records | 486](#)
- [Query > Flows > Query Queue | 489](#)

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 123 on page 481](#)

Figure 123: Query Flow Series Window



The query fields available on the screen for the **Flow Series** tab are described in [Table 56 on page 482](#). Enter query data into the fields to create a SQL-like query to display and analyze flows.

Table 56: Query Flow Series Fields

Field	Description
Time Range	<p>Select a range of time to display the flow series:</p> <ul style="list-style-type: none"> • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom <p>Click Custom to enter a specific custom time range in two fields: From Time and To Time.</p>
Select	Click the edit button (pencil icon) to open a Select window (Figure 124 on page 483), 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 125 on page 484), 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 56: 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 124 on page 483](#). 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 124: Flow Series Select

Select ✕

☐ Source VN

☐ Destination VN

☐ Time Granularity

☐ Source IP

☐ Destination IP

☐ Protocol

☐ Source Port

☐ Destination Port

☐ Virtual Router

☐ Bytes

☐ SUM(Bytes)

☐ Packets

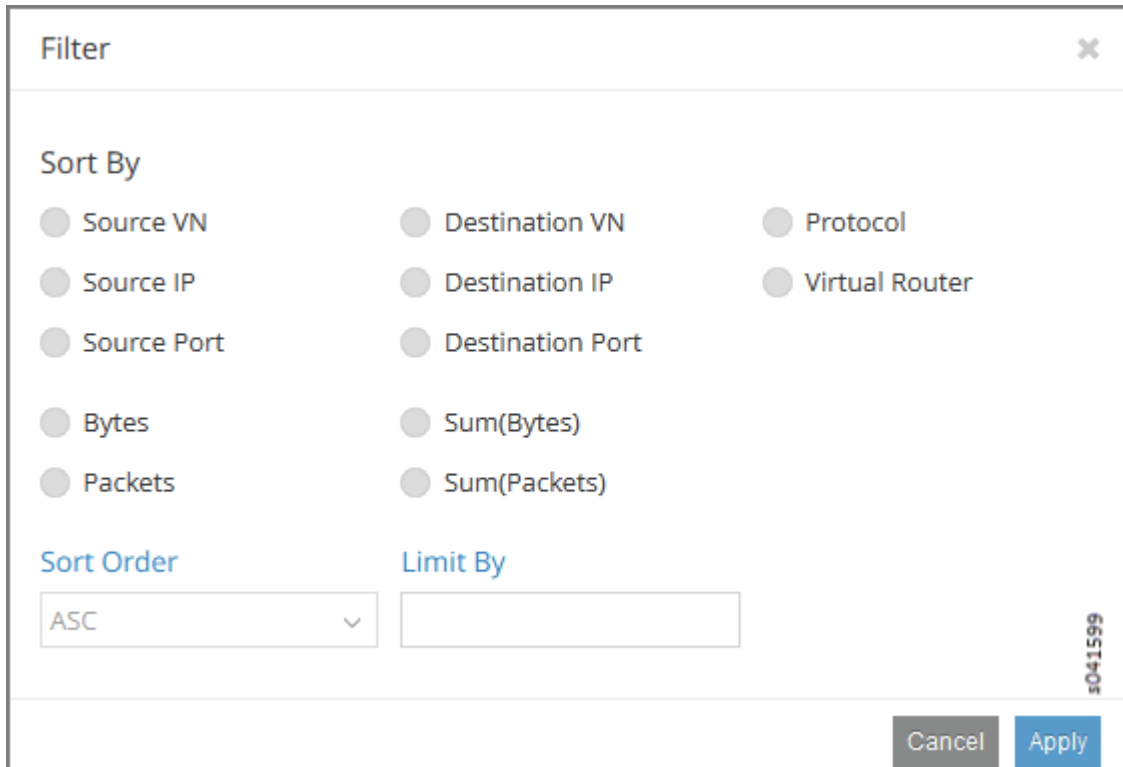
☐ SUM(Packets)

Cancel

Apply

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 125 on page 484](#).

Figure 125: Flow Series Filter



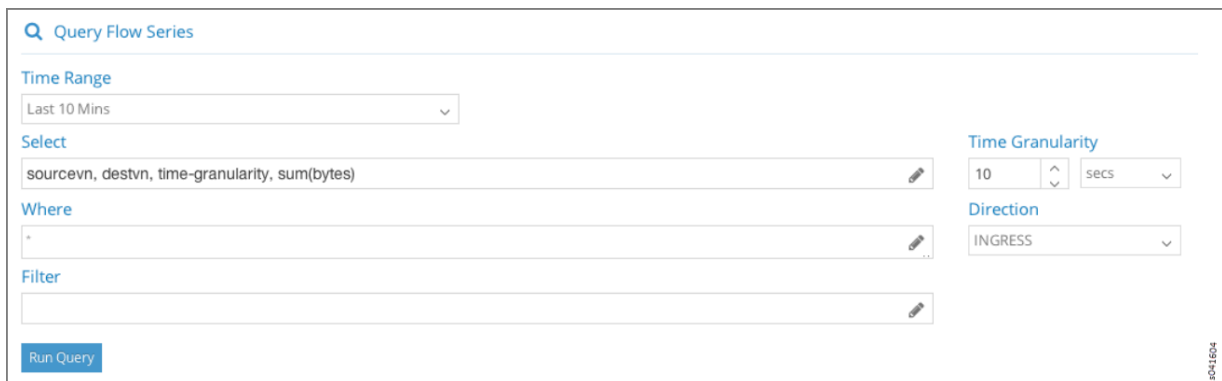
The 'Filter' dialog box contains the following elements:

- Sort By:** A grid of radio buttons for selecting the sort criteria: Source VN, Destination VN, Protocol, Source IP, Destination IP, Virtual Router, Source Port, Destination Port, Bytes, Sum(Bytes), Packets, and Sum(Packets).
- Sort Order:** A dropdown menu currently set to 'ASC'.
- Limit By:** An empty text input field.
- Buttons:** 'Cancel' and 'Apply' buttons at the bottom right.
- Footer:** A vertical text label 's041599' on the right side.

Example: Query Flow Series

The following is an example flow series query that returns the time series of the summation traffic in bytes for all combinations of source VN and destination VN for the last 10 minutes, with the bytes aggregated in 10 second intervals. See [Figure 126 on page 484](#).

Figure 126: Example: Query Flow Series



The 'Query Flow Series' interface includes the following components:

- Search:** A search bar with the text 'Query Flow Series'.
- Time Range:** A dropdown menu set to 'Last 10 Mins'.
- Select:** A text input field containing the query: 'sourcevn, destvn, time-granularity, sum(bytes)'.
- Time Granularity:** A control with a value of '10' and a unit dropdown set to 'secs'.
- Where:** A text input field containing an asterisk '*'.
- Direction:** A dropdown menu set to 'INGRESS'.
- Filter:** An empty text input field.
- Run Query:** A blue button at the bottom left.
- Footer:** A vertical text label 's041594' on the right side.

The query returns tabular time series data, see [Figure 127 on page 485](#), for the following combinations of Source VN and Dest VN:

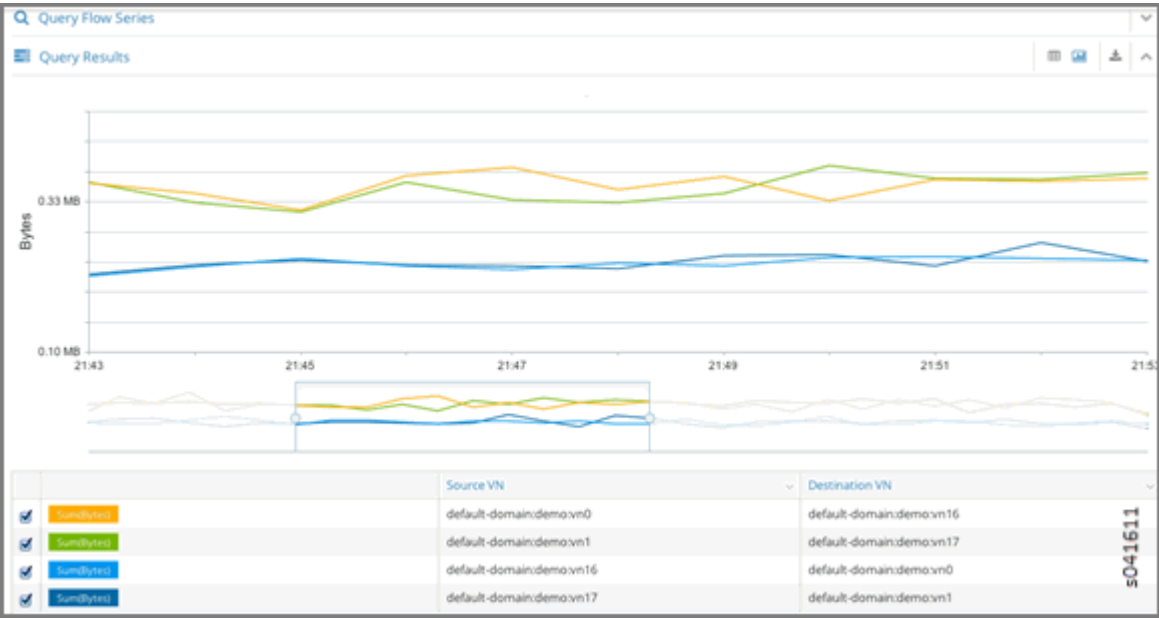
1. Flow Class 1: Source VN = default-domain:demo:front-end, Dest VN=__UNKNOWN__
2. Flow Class 2: Source VN = default-domain:demo:front-end, Dest VN=default-domain:demo:back-end

Figure 127: Query Flow Series Tabular Results

Query Flow Series					
Query Results					
Time	Source VN	Dest. VN	Direction	SUM(Bytes)	
2013-08-05 18:59:30:0:0	default-domain:demo:vn0	default-domain:demo:vn16	INGRESS	421,128	
2013-08-05 18:59:40:0:0	default-domain:demo:vn0	default-domain:demo:vn16	INGRESS	227,000	
2013-08-05 18:59:50:0:0	default-domain:demo:vn0	default-domain:demo:vn16	INGRESS	216,816	
2013-08-05 19:00:00:0:0	default-domain:demo:vn0	default-domain:demo:vn16	INGRESS	387,036	
2013-08-05 18:59:30:0:0	default-domain:demo:vn1	default-domain:demo:vn17	INGRESS	52,944	
2013-08-05 18:59:40:0:0	default-domain:demo:vn1	default-domain:demo:vn17	INGRESS	52,692	
2013-08-05 18:59:50:0:0	default-domain:demo:vn1	default-domain:demo:vn17	INGRESS	58,040	
2013-08-05 19:00:00:0:0	default-domain:demo:vn1	default-domain:demo:vn17	INGRESS	42,480	
2013-08-05 18:59:30:0:0	default-domain:demo:vn16	default-domain:demo:vn0	INGRESS	17,832	
2013-08-05 18:59:40:0:0	default-domain:demo:vn16	default-domain:demo:vn0	INGRESS	27,320	
2013-08-05 18:59:50:0:0	default-domain:demo:vn16	default-domain:demo:vn0	INGRESS	20,792	
2013-08-05 19:00:00:0:0	default-domain:demo:vn16	default-domain:demo:vn0	INGRESS	10,404	

Because **Time Granularity** is selected, the results can also be displayed as graphical charts. Click the graph button on the right side of the tabular results. The results are displayed in a graphical flow chart. See [Figure 128 on page 486](#).

Figure 128: 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 129: Flow Records

Query

Flows

Flow Series

Flow Records

Query Queue

Logs

Query > Flows > Flow Records

Search

Query Flow Records

Time Range

Last 10 Mins

Select

Where

Direction

INGRESS

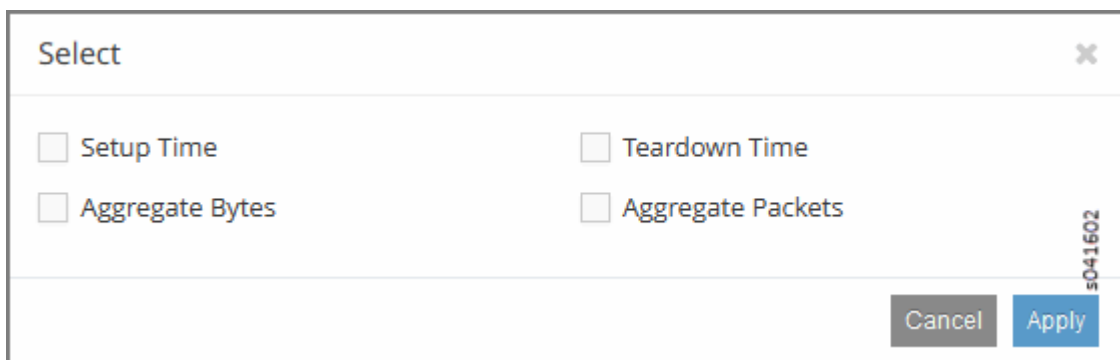
Run Query

The query fields available on the screen for the **Flow Records** tab are described in [Table 57 on page 487](#). Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 57: Query Flow Records Fields

Field	Description
Time Range	<p>Select a range of time for the flow records:</p> <ul style="list-style-type: none"> • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom <p>Click Custom to enter a specified custom time range in two fields: From Time and To Time.</p>
Select	Click the edit button (pencil icon) to open a Select window (Figure 130 on page 488), 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 130 on page 488](#).

Figure 130: Flow Records Select WindowA screenshot of a 'Select' dialog box. The dialog has a title bar with the word 'Select' and a close button (X). Inside, there are four checkboxes arranged in a 2x2 grid: 'Setup Time', 'Teardown Time', 'Aggregate Bytes', and 'Aggregate Packets'. All checkboxes are currently unchecked. At the bottom right, there are two buttons: 'Cancel' (grey) and 'Apply' (blue). A small vertical text '5041602' is visible on the right side of the dialog box.

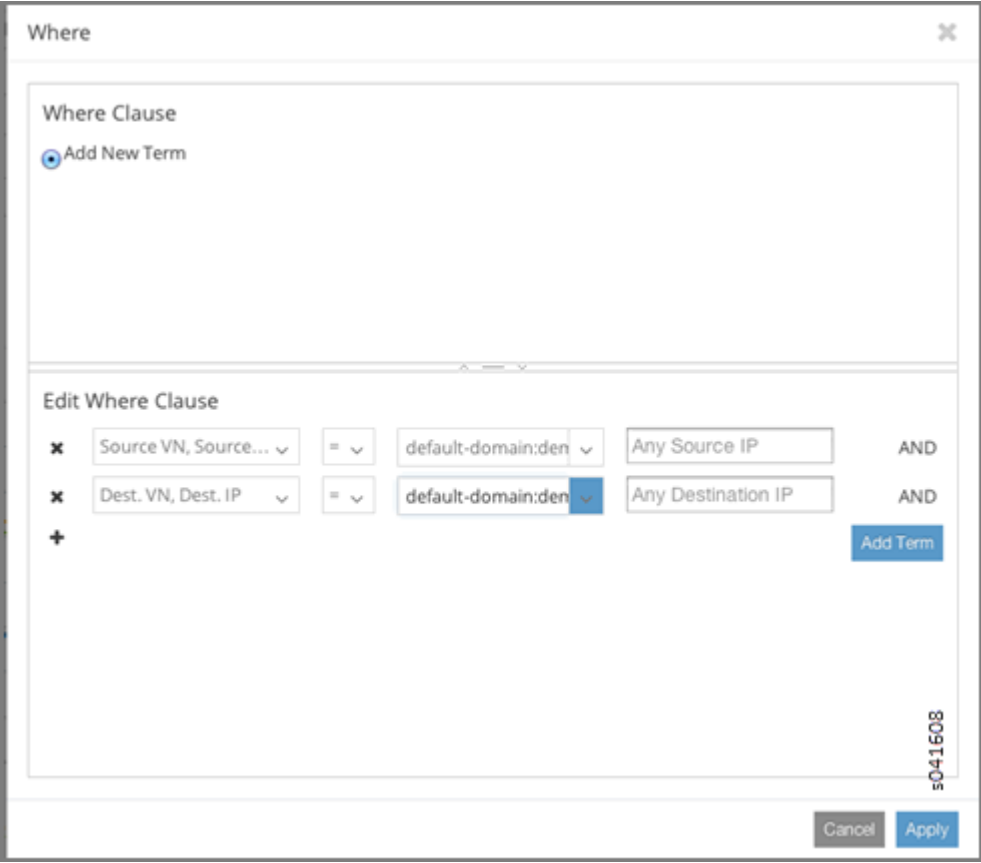
Select	
<input type="checkbox"/> Setup Time	<input type="checkbox"/> Teardown Time
<input type="checkbox"/> Aggregate Bytes	<input type="checkbox"/> Aggregate Packets
<div>Cancel Apply</div>	

You can restrict the query to a particular source VN and destination VN combination using the **Where** section.

The **Where Clause** supports logical AND and logical OR operations, and is modeled as a logical OR of multiple AND terms. For example: ((term1 AND term2 AND term3..) OR (term4 AND term5) OR...).

Each term is a single variable expression such as **Source VN = VN1**.

Figure 131: 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 132 on page 489](#).

Figure 132: Flows Query Queue

Query > Flows > Query Queue							Search Sitemap
Flow Query Queue							
Date	Query	Progress	Records	Status	Time Taken		
2013-10-09 18:07:06	{ "table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": ["flow_class_id", "direction_ing", "sum(bytes)", "T=60"], "dir": 1 }	100%	180	completed	150 secs		
2013-10-09 17:55:48	{ "table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": ["flow_class_id", "direction_ing", "sum(bytes)", "T=60"], "dir": 1 }	100%	180	completed	145 secs		
2013-10-09 17:29:39	{ "table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": ["flow_class_id", "direction_ing", "sum(bytes)", "T=60"], "dir": 1 }	100%	180	completed	170 secs		
2013-10-09 16:57:10	{ "table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": ["flow_class_id", "direction_ing", "sum(bytes)", "T=60"], "dir": 1 }	100%	180	completed	270 secs		
2013-10-09 16:39:48	{ "table": "FlowSeriesTable", "start_time": 1381360140000000, "end_time": 1381361940000000, "select_fields": ["flow_class_id", "direction_ing", "T=60", "sum(bytes)", "dir": 1 }	100%	30	completed	60 secs		
2013-10-09 11:07:29	{ "table": "FlowSeriesTable", "start_time": 1381338420000000, "end_time": 1381342020000000, "select_fields": ["flow_class_id", "direction_ing", "sum(bytes)", "T=60"], "dir": 1 }	100%	7	completed	15 secs		
1 2 3 4 5 6 >							Displaying 1 - 6 of 31 Records

The query fields available on the screen for the **Flow Records** tab are described in [Table 58 on page 490](#). Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 58: 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

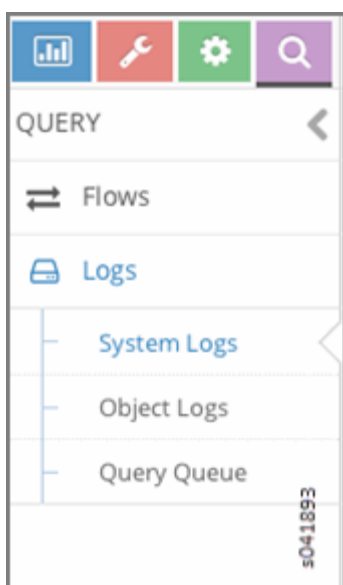
- [Query > Logs Menu Options | 491](#)
- [Query > Logs > System Logs | 491](#)
- [Sample Query for System Logs | 493](#)

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 133 on page 491](#).

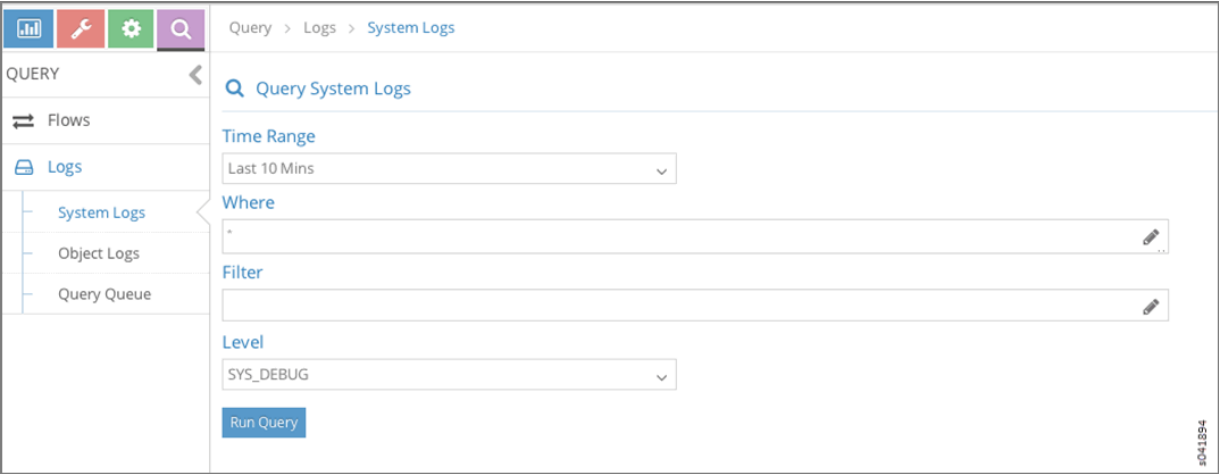
Figure 133: Query > Logs



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 134 on page 492](#).

Figure 134: Query > Logs > System Logs



The query fields available on the **Query System Logs** screen are described in [Table 59 on page 492](#).

Table 59: Query System Logs Fields

Field	Description
Time Range	<p>Select a range of time for which to see the system logs:</p> <ul style="list-style-type: none">• Last 10 Mins• Last 30 Mins• Last 1 Hr• Last 6 Hrs• Last 12 Hrs• Custom <p>If you click Custom, enter a desired time range in two new fields: From Time and To Time.</p>
Where	<p>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.</p>

Table 59: Query System Logs Fields *(Continued)*

Field	Description
Level	<p>Select the message severity level to view:</p> <ul style="list-style-type: none"> • SYS_NOTICE • SYS_EMERG • SYS_ALERT • SYS_CRIT • SYS_ERR • SYS_WARN • SYS_INFO • SYS_DEBUG
Run Query	Click this button to retrieve the system logs that match the query. The logs are listed in a box with columns showing the Time , Source , Module Id , Category , Log Type , and Log message.
Export	This button appears after you click Run Query , allowing you to export the list of system messages to a text/csv file.

Sample Query for System Logs

This section shows a sample system logs query designed to show all **System Logs** from ModuleId = VRouterAgent on Source = b1s16 and filtered by **Level** = SYS_DEBUG.

1. At the **Query System Logs** screen, click in the **Where** field to access the **Where** query screen and enter information defining the location to query in the **Edit Where Clause** section and click **OK**; see [Figure 135 on page 494](#).

Figure 135: Edit Where Clause

Where

Where Clause

Add New Term

Edit Where Clause

×

ModuleId

▼

=

▼

VRouterAgent

▼

AND

×

Source

▼

=

▼

b1s16

▼

AND

+

Add Term

OK

Cancel

2. The information you defined at the Where screen displays on the **Query System Logs**. Enter any more defining information needed; see [Figure 136 on page 495](#). When finished, click **Run Query** to display the results.

Figure 136: Sample Query System Logs

Query System Logs

Time Range

Last 10 Mins

Where

(ModuleId = VRouterAgent AND Source = b1s16)

Filter

Level

SYS_DEBUG

Run Query

#041896

Query > Logs > Object Logs

Object logs allow you to search for logs associated with a particular object, for example, all logs for a specified virtual network. Object logs record information related to modifications made to objects, including creation, deletion, and other modifications; see [Figure 137 on page 495](#).

Figure 137: Query > Logs > Object Logs

Query Object Logs

Time Range

Last 12 Hrs

Object Type

Virtual Network

Object Id

default-domain:demo:vn14

Select

ObjectLog, SystemLog

Where

*

Filter

Run Query

#041897

The query fields available on the **Object Logs** screen are described in [Table 60 on page 496](#).

Table 60: Object Logs Query Fields

Field	Description
Time Range	<p>Select a range of time for which to see the logs:</p> <ul style="list-style-type: none"> • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom <p>If you click Custom, enter a desired time range in two new fields: From Time and To Time.</p>
Object Type	<p>Select the object type for which to show logs:</p> <ul style="list-style-type: none"> • Virtual Network • Virtual Machine • Virtual Router • BGP Peer • Routing Instance • XMPP Connection
Object Id	Select from a list of available identifiers the name of the object you wish to use.
Select	<p>Click the edit button (pencil icon) to open a window where you can select searchable types by clicking a checkbox:</p> <ul style="list-style-type: none"> • ObjectLog • SystemLog

Table 60: 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 | 497](#)

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 138 on page 498](#)

Figure 138: Navigate to Instance

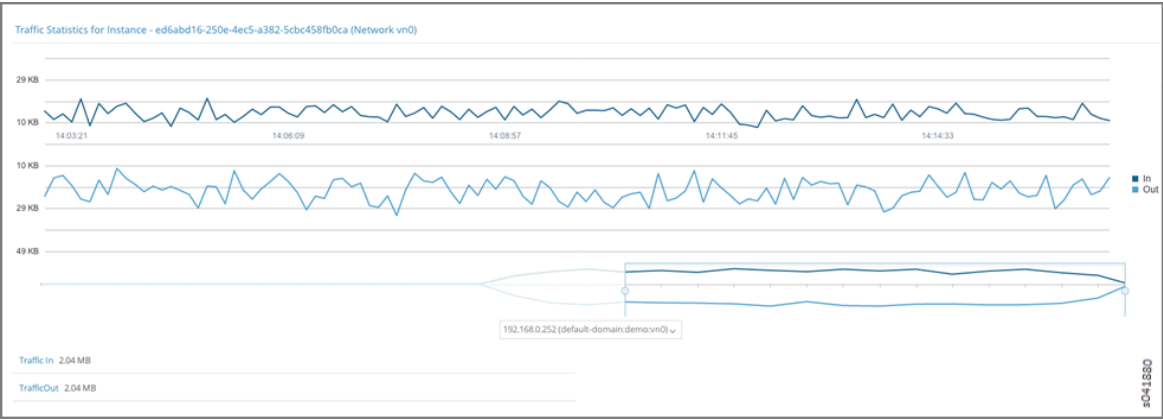
Monitor > Networking > Networks > default-domain:demo:vn0

SummaryPort MapPort DistributionInstancesDetails

Instance	Traffic In	Traffic Out
ed6ab16-250e-4ec5-a382-5bc458fb0ca	1.73 MB	1.74 MB
682b7414-c4ba-45ee-91bc-9c22cd6fc09d	1.72 MB	1.72 MB

2. Click the instance to view **Traffic Statistics for Instance**. see [Figure 139 on page 498](#).

Figure 139: 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 140 on page 498](#) and [Figure 141 on page 499](#).

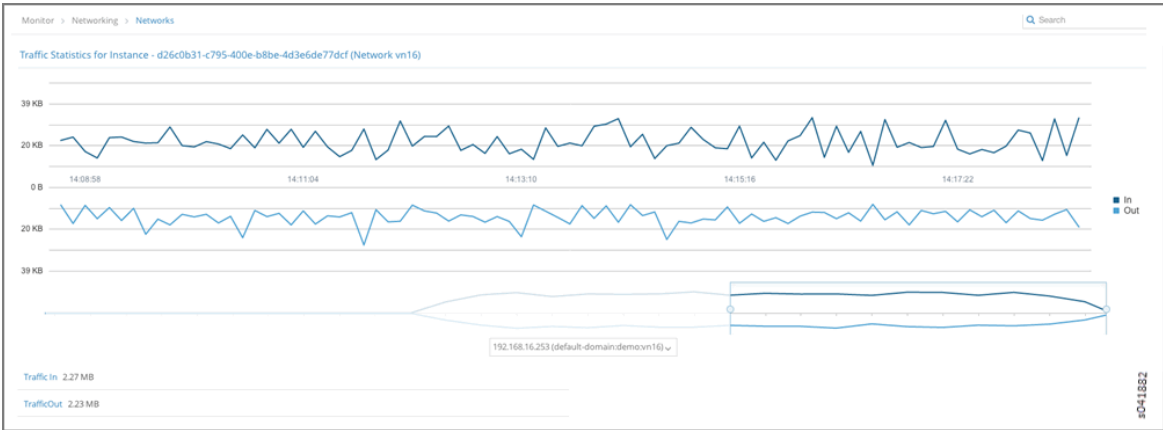
Figure 140: Navigate to Instance

Monitor > Networking > Networks > default-domain:demo:vn16

SummaryPort MapPort DistributionInstancesDetails

Instance	Traffic In	Traffic Out
d26c0b31-c795-400e-b8be-4d3e6de77dcf	2.18 MB	2.13 MB
23045415-b679-4d9a-8f9d-96c162de28be	2.11 MB	2.16 MB

Figure 141: 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 142 on page 499](#).

Figure 142: Navigate to a3s18 Interfaces

Monitor > Infrastructure > Virtual Routers > a3s18						
Details Console Interfaces Networks ACL Flows Routes						
Name	Label	Status	Network	IP Address	Floating IP	Instance
tap1d4e0121-4c	16	Up	default-domain:demo:vn0	192.168.0.252	None	ed6abd16-250e-4ec5-a382-5cbc458fb0ca
tap249de2e1-97	18	Up	default-domain:demo:vn16	192.168.16.252	None	23045415-b679-4d9a-8f9d-96c162de28be
tap5b3b3d63-74	19	Up	default-domain:demo:vn17	192.168.17.252	None	99311eda-261e-47eb-b4a7-8d126d7499bf
tapc740843c-6b	17	Up	default-domain:demo:vn1	192.168.1.252	None	20244ef9-a4ed-4a32-803f-15cf5323572e

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 143 on page 499](#).

Figure 143: Navigate to a3s19 Interfaces

Monitor > Infrastructure > Virtual Routers > a3s19						
Details Console Interfaces Networks ACL Flows Routes						
Name	Label	Status	Network	IP Address	Floating IP	Instance
tap29585b2f-c2	19	Up	default-domain:demo:vn16	192.168.16.253	None	d26c0b31-c795-400e-b8be-4d3e6de77dcf
tapb257d21d-d3	18	Up	default-domain:demo:vn1	192.168.1.253	None	eebce321-7536-46e7-a454-ceff1f13ac695
tapc83e9d87-66	17	Up	default-domain:demo:vn17	192.168.17.253	None	b2425f95-6f7e-4060-9478-81a4a82f5541
tapc5ea97e3-55	16	Up	default-domain:demo:vn0	192.168.0.253	None	682b7414-c4ba-45ee-91bc-9c22cd96c69d

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 144 on page 500](#) and [Figure 145 on page 500](#).

Figure 144: ACL Connectivity a3s18

Monitor > Infrastructure > Virtual Routers > a3s18										
Details Console Interfaces Networks ACL Flows Routes										
UUID	Flows	Action	Protocol	Source Network or Prefix	Source Port	Destination Network or Prefix	Destination Port	Source Policy Rule	ACE Id	
a724928e-3f30-477a-ad...	16	pass	any	default-domain:demo:vn0	any	default-domain:demo:vn16	any		1	
		pass	any	default-domain:demo:vn16	any	default-domain:demo:vn0	any		2	
		pass	any	default-domain:demo:vn0	any	default-domain:demo:vn0	any		3	
b32143a3-0ed0-4ae2-9c...	16	pass	any	default-domain:demo:vn1	any	default-domain:demo:vn17	any		1	
		pass	any	default-domain:demo:vn17	any	default-domain:demo:vn1	any		2	
		pass	any	default-domain:demo:vn1	any	default-domain:demo:vn1	any		3	
b8cf9810-effc-41f8-aa7...	16	pass	any	default-domain:demo:vn0	any	default-domain:demo:vn16	any		1	
		pass	any	default-domain:demo:vn16	any	default-domain:demo:vn0	any		2	
		pass	any	default-domain:demo:vn16	any	default-domain:demo:vn16	any		3	
d1b47291-7a21-4fde-8d...	16	pass	any	default-domain:demo:vn1	any	default-domain:demo:vn17	any		1	
		pass	any	default-domain:demo:vn17	any	default-domain:demo:vn1	any		2	
		pass	any	default-domain:demo:vn17	any	default-domain:demo:vn17	any		3	

Figure 145: ACL Connectivity a3s19

Monitor > Infrastructure > Virtual Routers > a3s19										
Details Console Interfaces Networks ACL Flows Routes										
UUID	Flows	Action	Protocol	Source Network or Prefix	Source Port	Destination Network or Prefix	Destination Port	Source Policy Rule	ACE Id	
a724928e-3f30-477a-ad...	16	pass	any	default-domain:demo:vn0	any	default-domain:demo:vn16	any		1	
		pass	any	default-domain:demo:vn16	any	default-domain:demo:vn0	any		2	
		pass	any	default-domain:demo:vn0	any	default-domain:demo:vn0	any		3	
b32143a3-0ed0-4ae2-9c...	16	pass	any	default-domain:demo:vn1	any	default-domain:demo:vn17	any		1	
		pass	any	default-domain:demo:vn17	any	default-domain:demo:vn1	any		2	
		pass	any	default-domain:demo:vn1	any	default-domain:demo:vn1	any		3	
b8cf9810-effc-41f8-aa7...	16	pass	any	default-domain:demo:vn0	any	default-domain:demo:vn16	any		1	
		pass	any	default-domain:demo:vn16	any	default-domain:demo:vn0	any		2	
		pass	any	default-domain:demo:vn16	any	default-domain:demo:vn16	any		3	
d1b47291-7a21-4fde-8d...	16	pass	any	default-domain:demo:vn1	any	default-domain:demo:vn17	any		1	
		pass	any	default-domain:demo:vn17	any	default-domain:demo:vn1	any		2	
		pass	any	default-domain:demo:vn17	any	default-domain:demo:vn17	any		3	

- Next, verify the routes on the control node for routing instances default-domain:demo:vn0:vn0 and default-domain:demo:vn16:vn16; see [Figure 146 on page 501](#) and [Figure 147 on page 501](#).

Figure 146: Routes default-domain:demo:vn0:vn0

Monitor > Infrastructure > Control Nodes > a3s15 Search

Details Console Peers **Routes**

Routing Instance: default-domain:demo:vn0:vn0 Address Family: All Limit 50 Routes

Peer Source: All Prefix: Prefix Display Routes Reset

Prefix	Address Family	Protocol	Source	Next hop	Label	Local Preference	AS Path
192.168.0.252/32	inet	XMPP	a3s18	10.84.17.4	16	100	-
	inet	BGP	10.84.17.3	10.84.17.4	16	100	AS_PATH: 0
192.168.0.253/32	inet	XMPP	a3s19	10.84.17.5	16	100	-
	inet	BGP	10.84.17.3	10.84.17.5	16	100	AS_PATH: 0
192.168.16.252/32	inet	XMPP	a3s18	10.84.17.4	17	100	-
	inet	BGP	10.84.17.3	10.84.17.4	17	100	AS_PATH: 0
192.168.16.253/32	inet	XMPP	a3s19	10.84.17.5	17	100	-
	inet	BGP	10.84.17.3	10.84.17.5	17	100	AS_PATH: 0
10.84.17.4:1:192.168.0.255,0.0.0.0	inetmcast	XMPP	a3s18	10.84.17.4	0	100	-
10.84.17.4:1:255.255.255.255,0.0.0.0	inetmcast	XMPP	a3s18	10.84.17.4	0	100	-
10.84.17.5:1:192.168.0.255,0.0.0.0	inetmcast	XMPP	a3s19	10.84.17.5	0	100	-
10.84.17.5:1:255.255.255.255,0.0.0.0	inetmcast	XMPP	a3s19	10.84.17.5	0	100	-

1041887

Figure 147: Routes default-domain:demo:vn16:vn16

Monitor > Infrastructure > Control Nodes > a3s15 Search

Details Console Peers **Routes**

Routing Instance: default-domain:demo:vn16:vn16 Address Family: All Limit 50 Routes

Peer Source: All Prefix: Prefix Display Routes Reset

Prefix	Address Family	Protocol	Source	Next hop	Label	Local Preference	AS Path
192.168.0.252/32	inet	XMPP	a3s18	10.84.17.4	16	100	-
	inet	BGP	10.84.17.3	10.84.17.4	16	100	AS_PATH: 0
192.168.0.253/32	inet	XMPP	a3s19	10.84.17.5	16	100	-
	inet	BGP	10.84.17.3	10.84.17.5	16	100	AS_PATH: 0
192.168.16.252/32	inet	XMPP	a3s18	10.84.17.4	17	100	-
	inet	BGP	10.84.17.3	10.84.17.4	17	100	AS_PATH: 0
192.168.16.253/32	inet	XMPP	a3s19	10.84.17.5	17	100	-
	inet	BGP	10.84.17.3	10.84.17.5	17	100	AS_PATH: 0
10.84.17.4:2:192.168.16.255,0.0.0.0	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	-

1041888

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 148 on page 502](#).

Figure 148: Verify Route and Next Hop a3s18

Monitor > Infrastructure > Virtual Routers > a3s18		
Details	Console	Interfaces
Networks	ACL	Flows
Routes		
VRF	default-domain:demo:vn0:vn0	Show Routes
		Unicast Multicast
Prefix	Next ho...	Next hop details
169.254.169.254 / 32	receive	Source: MData Dest VN: default-domain:default-project:link_local
192.168.0.252 / 32	interface	Interface: tap1dae0121-4c Dest VN: default-domain:demo:vn0
	interface	Interface: tap1dae0121-4c Dest VN: default-domain:demo:vn0
	interface	Interface: tap1dae0121-4c Dest VN: default-domain:demo:vn0
192.168.0.253 / 32	tunnel	Dest IP: 10.84.17.5 Dest VN: default-domain:demo:vn0 Label: 16
	tunnel	Dest IP: 10.84.17.5 Dest VN: default-domain:demo:vn0 Label: 16
192.168.0.254 / 32	interface	Interface: pkt0 Dest VN: default-domain:demo:vn0
192.168.16.252 / 32	interface	Interface: tap249de2e1-97 Dest VN: default-domain:demo:vn16
	interface	Interface: tap249de2e1-97 Dest VN: default-domain:demo:vn16
192.168.16.253 / 32	tunnel	Dest IP: 10.84.17.5 Dest VN: default-domain:demo:vn16 Label: 19

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 149 on page 503](#).

Figure 149: Verify Route and Next Hop a3s19

Monitor > Infrastructure > Virtual Routers > a3s19

Details Console Interfaces Networks ACL Flows Routes

VRF default-domain:demo:vn16:vn16 Show Routes Unicast Multicast

Prefix	Next ho...	Next hop details
169.254.169.254 / 32	receive	Source: MData Dest VN: default-domain:default-project:__link_local__
192.168.0.252 / 32	tunnel	Dest IP: 10.84.17.4 Dest VN: default-domain:demo:vn0 Label: 16
	tunnel	Dest IP: 10.84.17.4 Dest VN: default-domain:demo:vn0 Label: 16
192.168.0.253 / 32	interface	Interface: tape5ea97e3-55 Dest VN: default-domain:demo:vn0
	interface	Interface: tape5ea97e3-55 Dest VN: default-domain:demo:vn0
192.168.16.252 / 32	tunnel	Dest IP: 10.84.17.4 Dest VN: default-domain:demo:vn16 Label: 18
	tunnel	Dest IP: 10.84.17.4 Dest VN: default-domain:demo:vn16 Label: 18
192.168.16.253 / 32	interface	Interface: tap29585b2f-c2 Dest VN: default-domain:demo:vn16
	interface	Interface: tap29585b2f-c2 Dest VN: default-domain:demo:vn16
	interface	Interface: tap29585b2f-c2 Dest VN: default-domain:demo:vn16
192.168.16.254 / 32	interface	Interface: pkt0 Dest VN: default-domain:demo:vn16

s041890

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 150 on page 503](#) and [Figure 151 on page 504](#).

Figure 150: Flows for a3s18

Monitor > Infrastructure > Virtual Routers > a3s18

Details Console Interfaces Networks ACL Flows Routes

Search

Active Flows: 64

Protocol	Source Network	Source IP	Source Port	Destination Network	Destination IP	Destination Port	Bytes/Pkts	Setup Time
TCP	vn0	192.168.0.252	43434	vn16	192.168.16.253	9100	1884588/5417	21:00:22.131180 2013-Aug-06
TCP	vn16	192.168.16.253	9100	vn0	192.168.0.252	43434	1969668/5891	21:00:22.131193 2013-Aug-06
TCP	vn16	192.168.16.253	9101	vn0	192.168.0.252	53369	1903500/5805	21:00:22.206222 2013-Aug-06
TCP	vn0	192.168.0.252	53369	vn16	192.168.16.253	9101	1890088/5302	21:00:22.206207 2013-Aug-06
UDP	vn0	192.168.0.252	39522	vn16	192.168.16.252	9200	0/0	21:00:22.382861 2013-Aug-06
UDP	vn0	192.168.0.252	44794	vn16	192.168.16.253	9201	1707392/3144	21:00:24.104277 2013-Aug-06
UDP	vn16	192.168.16.253	9201	vn0	192.168.0.252	44794	1735788/3107	21:00:24.104293 2013-Aug-06
UDP	vn0	192.168.0.252	40561	vn16	192.168.16.253	9200	1693476/3067	21:00:22.037377 2013-Aug-06
UDP	vn16	192.168.16.253	9200	vn0	192.168.0.252	40561	1643324/3061	21:00:22.037387 2013-Aug-06
UDP	vn0	192.168.0.252	39522	vn16	192.168.16.252	9200	1676616/3074	21:00:22.306703 2013-Aug-06
TCP	vn0	192.168.0.252	34236	vn16	192.168.16.252	9100	1891368/5686	21:00:22.395695 2013-Aug-06
TCP	vn0	192.168.0.252	34236	vn16	192.168.16.252	9100	0/0	21:00:22.400371 2013-Aug-06

s041891

Figure 151: Flows for a3s19

Monitor > Infrastructure > Virtual Routers > a3s19

Search

Details Console Interfaces Networks ACL Flows Routes

Active Flows: 64

Protocol	Source Network	Source IP	Source Port	Destination Network	Destination IP	Destination Port	Bytes/Pkts	Setup Time
UDP	vn0	192.168.0.252	44794	vn16	192.168.16.253	9201	1069380/1975	21:00:24.111374 2013-Aug-06
UDP	vn16	192.168.16.253	9201	vn0	192.168.0.252	44794	1100604/1963	21:00:24.111380 2013-Aug-06
UDP	vn0	192.168.0.252	40561	vn16	192.168.16.253	9200	1046756/1877	21:00:22.047747 2013-Aug-06
UDP	vn16	192.168.16.253	9200	vn0	192.168.0.253	47270	1061900/1921	21:00:25.373941 2013-Aug-06
UDP	vn16	192.168.16.253	9200	vn0	192.168.0.252	40561	1010568/1914	21:00:22.047756 2013-Aug-06
TCP	vn16	192.168.16.253	9100	vn0	192.168.0.253	53314	1217772/3649	21:00:23.465564 2013-Aug-06
TCP	vn0	192.168.0.252	43434	vn16	192.168.16.253	9100	1196536/3400	21:00:22.137665 2013-Aug-06
TCP	vn16	192.168.16.253	9100	vn0	192.168.0.252	43434	1239616/3724	21:00:22.137679 2013-Aug-06
UDP	vn16	192.168.16.253	9200	vn0	192.168.0.253	47270	0/0	21:00:25.347868 2013-Aug-06
TCP	vn16	192.168.16.253	9100	vn0	192.168.0.253	53314	0/0	21:00:23.440090 2013-Aug-06
UDP	vn16	192.168.16.253	9201	vn0	192.168.0.253	53930	1088692/1953	21:00:25.443166 2013-Aug-06
TCP	vn16	192.168.16.253	9101	vn0	192.168.0.253	34551	0/0	21:00:23.514246 2013-Aug-06
TCP	vn16	192.168.16.253	9101	vn0	192.168.0.253	34551	1394273/1604	21:00:23.514251 2013-Aug-06