JUNIPER NETWORKS | Engineering Simplicity

Contrail® Networking

Contrail Networking and Security User Guide

Published
2023-11-22

RELEASE
21.4

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

Contrail® Networking Contrail Networking and Security User Guide
21.4

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 | vii

1 Contrail Security

Security Policy Features | 2

Security Policy Features in OpenStack | 22

Policy Generation | 24

Configuring Policy Generation | 25

Host-based Firewalls | 32

- Host-based Firewalls Overview | 33
- Deploying Host-based Firewalls | 33
  - Prerequisites | 34
  - Topology | 34
  - Deployment Instructions | 36

2 Configuring Virtual Networks

Creating a Virtual Network with Juniper Networks Contrail | 46

Creating a Floating IP Address Pool | 49

Support for IPv6 Networks in Contrail | 51

Configuring EVPN and VXLAN | 55

- Configuring the VXLAN Identifier Mode | 57
- Configuring Forwarding | 60
- Configuring the VXLAN Identifier | 61
- Configuring Encapsulation Methods | 62

Support for EVPN Route Type 5 | 65

Support for EVPN Type 6 Selective Multicast Ethernet Tag Route | 66

Support for L3VPN Inter AS Option C | 68

Contrail vRouter Next Hop Configuration | 71

3 Deploying a Multi-Tier Web Application Using Contrail Networking

Example: Deploying a Multi-Tier Web Application | 76

- Multi-Tier Web Application Overview | 76
- Example: Setting Up Virtual Networks for a Simple Tiered Web Application | 77
- Verifying the Multi-Tier Web Application | 80
- Sample Addressing Scheme for Simple Tiered Web Application | 80
- Sample Physical Topology for Simple Tiered Web Application | 82
- Sample Physical Topology Addressing | 82

Sample Network Configuration for Devices for Simple Tiered Web Application | 84

4 Configuring Services

Configuring DNS Servers | 92

- DNS Overview | 92
- Defining Multiple Virtual Domain Name Servers in Contrail | 93
- IPAM and Virtual DNS | 93
- DNS Record Types | 94
- Configuring DNS on the User Interface | 95
- Configuring DNS Using Scripts | 103

Distributed Service Resource Allocation with Containerized Contrail | 105

Support for Broadcast and Multicast | 116

- Subnet Broadcast | 116
- All-Broadcast/Limited-Broadcast and Link-Local Multicast | 117
- Host Broadcast | 118

5 Configuring Service Chaining

Service Chaining | 120

- Service Chaining Basics | 120

Service Chaining Configuration Elements | 122

Service Chaining MX Series Configuration | 124

ECMP Load Balancing in the Service Chain | 126

Service Chain Version 2 with Port Tuple | 127

Service Chain Route Reorigination | 131

Example: Creating an In-Network Service Chain by Using Contrail Command | 154

Hardware and Software Requirements | 155

Overview | 155

Configuration | 155

Example: Creating an In-Network-NAT Service Chain | 163

Prerequisites | 163

Overview | 164

Configuration | 165

Example: Creating a Transparent Service Chain by Using Contrail Command | 172

Prerequisites | 172

Overview | 173

Configuration | 173

Using Static Routes with Services | 181

Static Routes for Service Instances | 181

Configuring Static Routes on a Service Instance | 181

Configuring Static Routes on Service Instance Interfaces | 183

Configuring Static Routes as Host Routes | 184

Configuring Metadata Service | 185

6 Optimizing Contrail Networking

Source Network Address Translation (SNAT) | 188

Overview | 188

SNAT on MX Series Routers Acting as Data Center Gateways | **189**

How to Enable SNAT on an MX Series Router Using Contrail Command | **189**

Neutron APIs for Routers | **190**

Network Namespace | **191**

SNAT and Security Groups | **192**

Using the Web UI to Configure Routers with SNAT | **192**

Using the Web UI to Configure Distributed SNAT | **194**

About This Guide

Use this guide to understand how you can create and orchestrate highly secure virtual networks using Contrail Networking and Security. Contrail Networking provides dynamic end-to-end networking policy and control for any cloud, any workload, and any deployment, from a single user interface. And unlike the conventional firewall configuration of IP addresses and port ranges, Contrail Security implements intent-based policy framework that uses policy constructs such as tags, labels, application policy set, address groups, and service groups.

Contrail Networking product documentation is organized into multiple guides as shown in "About This Guide" on page vii, according to the task you want to perform or the deployment scenario.

Table 1: Contrail Networking Guides

Guide Name	Description
Contrail Networking Installation and Upgrade Guide	Provides step-by-step instructions to install and bring up Contrail and its various components.
Contrail Networking for Container Networking Environments User Guide	Provides information about installing and using Contrail Networking in containerized environments using Kubernetes orchestration.
Contrail Networking Fabric Lifecycle Management Guide	Provides information about Contrail underlay management and data center automation.
Contrail Networking and Security User Guide	Provides information about creating and orchestrating highly secure virtual networks.
Contrail Networking Service Provider Focused Features Guide	Provides information about the features that are used by service providers.
Contrail Networking Monitoring and Troubleshooting Guide	Provides information about Contrail Insights and Contrail analytics.

RELATED DOCUMENTATION

- README Access to Contrail Networking Registry 21xx
- Contrail Networking Release Notes 21xx
- Tungsten Fabric Architecture Guide
- Juniper Networks TechWiki: Contrail Networking

CHAPTER 1

Contrail Security

Security Policy Features | 2

Security Policy Features in OpenStack | 22

Policy Generation | 24

Configuring Policy Generation | 25

Host-based Firewalls | 32

Security Policy Features

IN THIS SECTION

- Overview of Existing Network Policy and Security Groups in Contrail | 2
- Security Policy Enhancements | 3
- Using Tags and Configuration Objects to Enhance Security Policy | 3
- Configuration Objects | 6
- Using the Contrail Web User Interface to Manage Security Policies | 11

Overview of Existing Network Policy and Security Groups in Contrail

Virtual networks are isolated by default in Contrail. Network policy is used to connect the two networks, with security policy being applied to provide more granular allow and deny rules to specific traffic.

In modern cloud environments, workloads are moving from one server to another, one rack to another and so on. Therefore, users must rely less on using IP addresses to identify the endpoints to be protected. Instead users must leverage application attributes to author policies, so that the policies don't need to be updated on account of workload mobility.

You might want to segregate traffic based on the different categories, such as:

- Application name
- Segregating traffic for specific component tiers within the application
- Segregating traffic based on the deployment environment for the application instance
- Segregating traffic based on the specific geographic location where the application is deployed

Additionally, you might need to group workloads based on combinations of tags. These intents are hard to express with existing network policy constructs or Security Group constructs. Besides, existing policy constructs leveraging the network coordinates, must continually be rewritten or updated each time workloads move or their IP addresses change.

Security Policy Enhancements

As the Contrail environment has grown and become more complex, it has become harder to achieve desired security results with the existing network policy and security group constructs. The Contrail network policies have been tied to routing, making it difficult to express security policies for environments such as cross sectioning between categories, or having a multi-tier application supporting development and production environment workloads with no cross environment traffic.

Starting with Contrail Release 4.1, limitations of the current network policy and security group constructs are addressed by supporting decoupling of routing from security policies, multidimension segmentation, and policy portability. This release also enhances user visibility and analytics functions for security.

Contrail Release 4.1 introduces new firewall security policy objects, including the following enhancements:

- Routing and policy decoupling—introducing new firewall policy objects, which decouples policy from routing.
- Multidimension segmentation—segment traffic and add security features, based on multiple dimensions of entities, such as application, tier, deployment, site, usergroup, and so on.
- Policy portability—security policies can be ported to different environments, such as 'from development to production', 'from pci-complaint to production', 'to bare metal environment' and 'to container environment'.
- Visibility and analytics

Using Tags and Configuration Objects to Enhance Security Policy

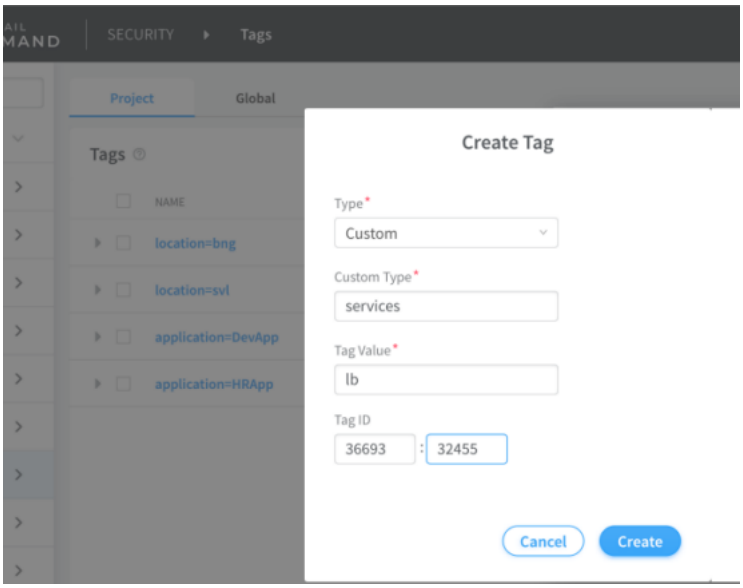
Starting with Contrail Release 4.1, tags and configuration objects are used to create new firewall policy objects that decouple routing and network policies, enabling multidimension segmentation and policy portability.

Multidimension traffic segmentation helps you segment traffic based on dimensions such as application, tier, deployment, site, and usergroup.

You can also port security policies to different environments. Portability of policies are enabled by providing match conditions for tags. Match tags must be added to the policy rule to match tag values of source and destination workloads without mentioning tag values. For example, in order for the `'allow protocol tcp source application-tier=web destination application-tier=application match application and site'` rule to take effect, the application and site values must match.

To create a tag in the Command UI, navigate to **Security** > **Tags**. Tags can be created with a Project scope or a Global scope. For more information on tags, see "Security Policy Features" on page 2. For example, see Figure 1 on page 4.

Figure 1: Create Tag



Starting in Contrail Networking Release 21.3, security policy allows optional user-defined tags, which enables you to define tag IDs along with tag names.

The range of the user-defined tag IDs is 32000 through 64000.

The tag format is: Tag-type ID (upper 16 bit) and Tag value ID (lower 16 bit).

- User-defined tag type range: 0x8000 - 0xFFFF
- User-defined tag value ID range: 0x8000 - 0xFFFF

NOTE: Defining user-defined tag IDs is optional. If you do not define the tag ID, the device will auto generate a tag ID within the range 0x0 – 0x7FFF. For example: `Application=Hrapp 0x0001 000`.

Predefined Tags

You can choose predefined tags based on the environment and deployment requirements.

Predefined tags include:

- application

- application-tier
- deployment
- site
- label (a special tag that allows the user to label objects)

Starting in Contrail Networking Release 21.3, you can create a predefined tag type with user-defined tag value ID.

For example: `Application=FinApp(user def tag value id : 32778) 0x0001 800a`

> **NOTE**: You can only enter decimal values in the Command UI. Range is 32767 through 65535.

Custom Tags

You can also define custom tags for a Kubernetes environment. You can define tags in the UI or upload configurations in JSON format.

Starting in Contrail Networking Release 21.3, custom tags supports user-defined tags.

Example Tag Usage

```
application = HRApp
application-tier = Web
site = USA
```

For user-defined custom tag: `Foo=HrApp(tag value id 32778 tag type id 32778) 0x800a 800a`

Tagging Objects

A user can tag the objects project, VN, VM, and VMI with tags and values to map their security requirements. Tags follow the hierarchy of project, VN, VM and VMI and are inherited in that order. This gives an option for the user to provide default settings for any tags at any level. Policies can specify their security in terms of tagged endpoints, in addition to expressing in terms of ip prefix, network, and address groups endpoints.

Policy Application

Policy application is a new object, implemented by means of the application tag. The user can create a list of policies per application to be applied during the flow acceptance evaluation. Introducing global

scoped policies and project scoped policies. There are global scoped policies, which can be applied globally for all projects, and project scoped policies, which are applied to specific projects.

Configuration Objects

The following are the configuration objects for the new security features.

- firewall-policy
- firewall-rule
- policy-management
- application-policy
- service-group
- address-group
- tag
- global-application-policy

Configuration Object Tag Object

Each configuration object tag object contains:

- tag: one of the defined tag types, stored as string and a 32-bit ID.
- tag type: Contains the type string and ID (the first 16 bits of the tag) and references to the tag resource type

Each value entered by the user creates a unique ID that is set in the tag_id field. The system can have up to 64 million tag values. On average, each tag can have up to 2k values, but there are no restrictions per tag.

Tags and labels can be attached to any object, for example, project, VN, VM, VMI, and policy, and these objects have a tag reference list to support multiple tags.

RBAC controls the users allowed to modify or remove attached tags. Some tags (typically facts) are attached by the system by default or by means of introspection.

Tag APIs

Tag APIs are used to give RBAC per tag in any object (VMI, VM, Project ….).

- REST: `HTTP POST to /set_tag_<tag_type>/<obj_uuid>`
- Python: `set_tag_<tag_type> (object_type, object_uuid, tag_value)`

Configuration also supports the following APIs:

- tag query
- tags (policy)
- tags (application tag)
- object query
- tags (object)
- tags (type, value)

Label

Label is special tag type, used to assign labels for objects. All of the tag constructs are valid, except that tag type is 'label'. One difference from other tags is that an object can have any number of labels. All other tag types are restricted to one tag per object.

Local and Global Tags

Tags can be defined globally or locally under a project; tag objects are children of either config-root or a project. An object can be tagged with a tag in its project or with a globally-scoped tag.

Analytics

When given a tag query with a SQL where clause and select clause, analytics should give out objects. The query can also contain labels, and the labels can have different operators.

Example:

User might want to know: a list of VMIs where `'site == USA and deployment == Production'`

list of VMIs where `'site == USA and deployment == Production has '`

Given tag SQL where clause and select clause, analytics should give out flows.

Control Node

The control node passes the tags, along with route updates, to agents and other control nodes.

Agent

Agent gets attached tags along with configuration objects. Agent also gets route updates containing tags associated with IP route. This process is similar to getting security group IDs along with the route update.

Address-Group Configuration Object

There are multiple ways to add IP address to address-group.

- Manually add IP prefixes to the address-group by means of configuration.
- Label a work load with the address-group's specified label. All ports that are labelled with the same label are considered to be part of that address-group.
- Use introspect workloads, based on certain criteria, to add ip-address to address-group.

Configuration

The address-group object refers to a label object, description, and list of IP prefixes. The label - object is created using the tag APIs.

Agent

Agent gets address-group and label objects referenced in policy configuration. Agent uses this address group for matching policy rules.

Analytics

When given address group label, analytics gets all the objects associated with it. Given address group label, get all the flows associated with it.

Service-Group Configuration Object

Configuration

The service-group contains a list of ports and protocols. The open stack service-group has a list of service objects; the service object contains attributes: id, name, service group id, protocol, source_port, destination_port, icmp_code, icmp_type, timeout, tenant id.

Agent

Agent gets service-group object as it is referred to in a policy rule. Agent uses this service group during policy evaluation.

Application-policy-set Configuration Object

The application-policy-set configuration object can refer to a tag of type application, network-policy objects, and firewall-policy objects. This object can be local (project) or globally scoped.

When an application tag is attached to an application-policy-set object, the policies referred by that object are automatically applied to the ports that have the same application tag.

Any firewall-policies referred by the application-policy-set objects are ordered using sequence numbers. If the same application tag is attached to multiple application-policy-sets, all those sets will apply, but order among those sets is undefined.

One application-policy-set (called default-policy-application-set) is special in that policies referred by it are applied to all interfaces by default, after applying policies referred to other application-policy-sets.

Upon seeing the application tag for any object, the associated policies are sent to agent. Agent will use this information to find out the list of policies to be applied and their sequence during flow evaluation. User can attach application tag to allowed objects (Project, VN, VM or VMI).

Policy-management Configuration Object

Policy-management is a global container object for all policy-related configuration.

Policy-management object contains

- network-policies (NPs)
- firewall-policies (FWPs)
- application-policy-sets
- global-policy objects
- global-policy-apply objects
- NPs - List of contrail networking policy objects
- FWPs - List of new firewall policy objects
- Application-policies - List of Application-policy objects
- Global-policies - List of new firewall policy objects, that are defined for global access
- Global-policy-apply - List of global policies in a sequence, and these policies applied during flow evaluation.
- Network Policies (NP) references are available, as they are today.

Firewall-policy Configuration Object

`Firewall-policy` is a new policy object that contains a list of firewall-rule-objects and audited flag. Firewall-policy can be project or global scoped depending on usage. Includes an audited Boolean flag to indicate that the owner of the policy indicated that the policy is audited. Default is False, and will have to explicitly be set to True after review. Generates a log event for audited with timestamp and user details.

Firewall-rule Configuration Object

Firewall-rule is a new rule object, which contains the following fields. The syntax is to give information about their layout inside the rule.

- <sequence number>
  There is a string object sequence number on the link from firewall-policy to firewall-policy-rule objects. The sequence number decides the order in which the rules are applied.
- [< id >]

  uuid
- [name < name >]

  Unique name selected by user
- [description < description >]
- public
- {permit | deny}
- [protocol {< protocol-name > | any } destination-port { < port range > | any } [source-port { < port range > | any}]] | service-group < name >
- endpoint-1 { [ip < prefix >] | [virtual-network < vnname >] | [address-group < group name >] | [tags T1 == V1 && T2 == V2 … && Tn == Vn && label == label name...] | any}
- { -> | <- | <-> }

  Specifies connection direction. All the rules are connection oriented and this option gives the direction of the connection.
- endpoint-2 { [ip < prefix >] | [virtual-network < vnname >] | [address-group < group name >] | [tags T1 == V1 && T2 == V2 … && Tn == Vn && label == label name...] | any }

Tags at endpoints support an expression of tags. We support only '==' and '&&' operators. User can specify labels also as part the expression. Configuration object contains list of tag names (or global:tag-name in case of global tags) for endpoints.

- [match_tags {T1 .... Tn} | none}]

  List of tag types or none. User can specify either match with list of tags or none. Match with list of tags mean, source and destination tag values should match for the rule to take effect.

- [log| mirror | alert | activate | drop | reject | sdrop]

  complex actions

- { enable | disable }

  A boolean flag to indicate the rule is enabled or disabled. Facilitates selectively turn off the rules, without remove the rule from the policy. Default is True.

- filter

Compilation of Rules

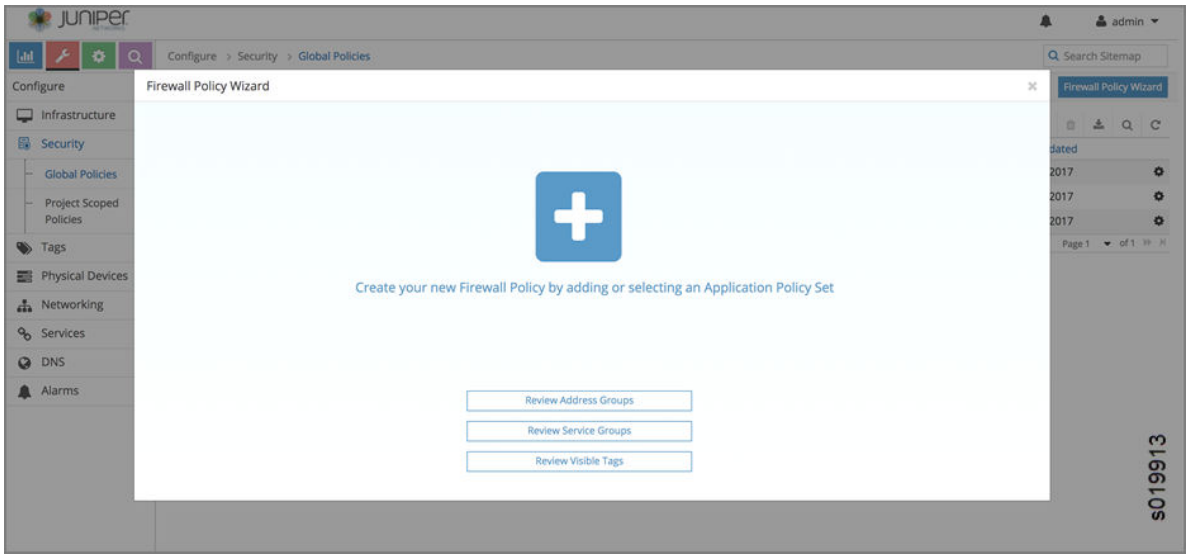
Whenever the API server receives a request to create/update a firewall policy rule object, it analyzes the object data to make sure that all virtual-networks, address-group, tag objects exist. If any of them do not exist, the request will be rejected. In addition, it will actually create a reference to those objects mentioned in the two endpoints. This achieves two purposes. First, we don't allow users to name non-existent objects in the rule and second, the user is not allowed to delete those objects without first removing them from all rules that are referring to them.

Using the Contrail Web User Interface to Manage Security Policies

Adding Security Policies

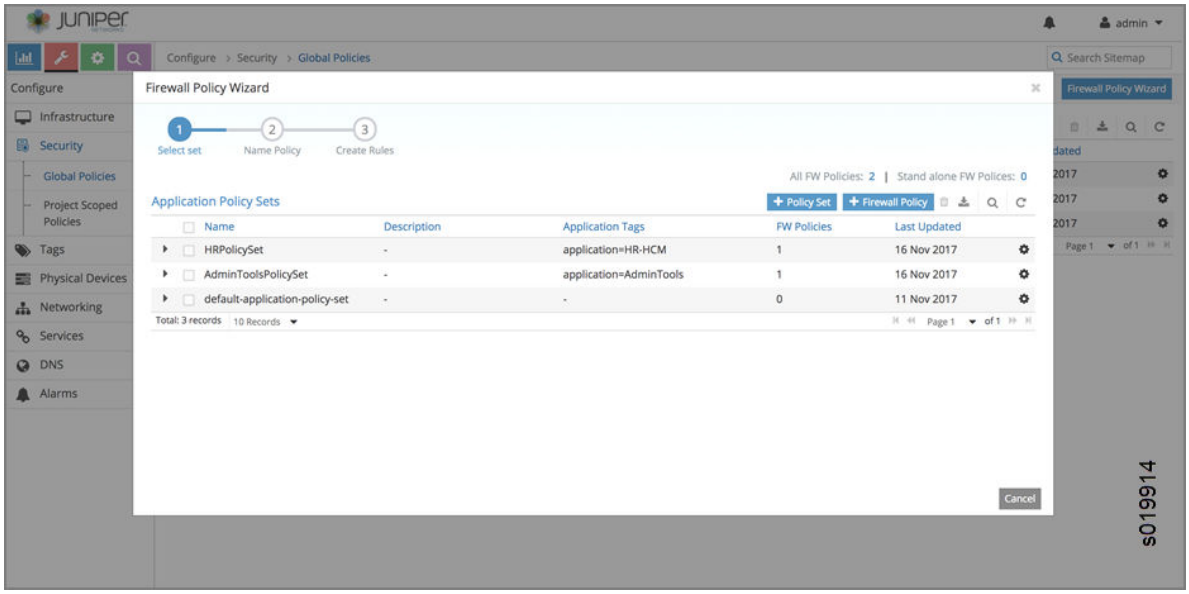
1. To add a security policy, go to **Configure > Security > Global Policies**. Near the upper right, click the button **Firewall Policy Wizard**. The **Firewall Policy Wizard** appears, where you can create your new firewall policy by adding or selecting an application policy set. See Figure 2 on page 12.

Figure 2: Firewall Policy Wizard



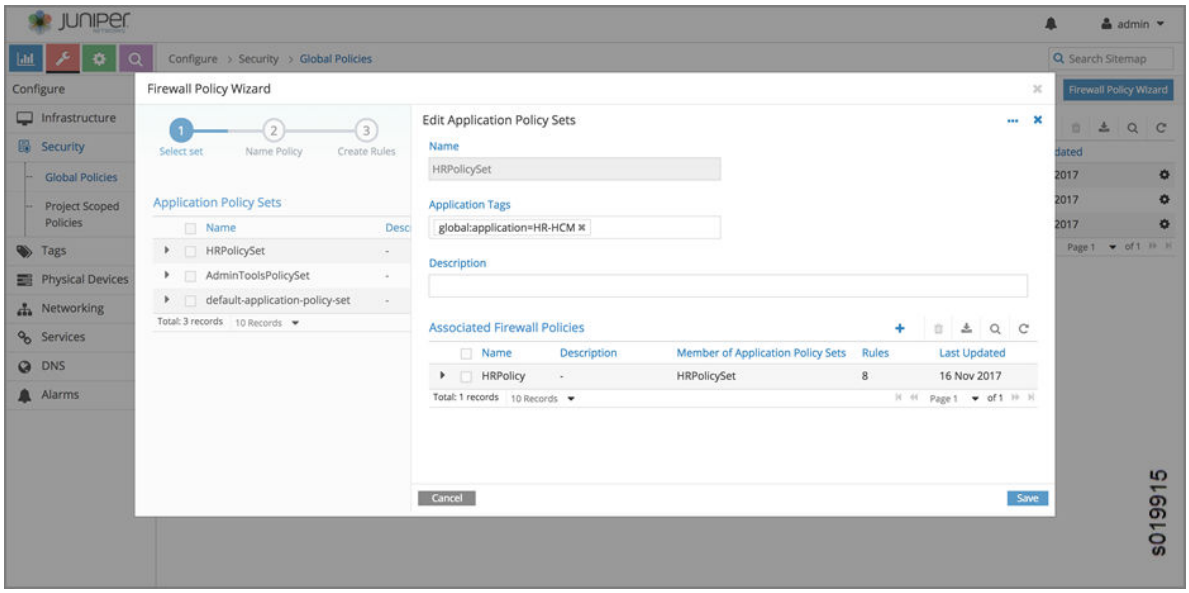
2. Click the large + on the Firewall Policy Wizard screen to view the **Application Policy Sets** window. The existing application policy sets are displayed. See Figure 3 on page 12.

Figure 3: Application Policy Sets



3. To create a new firewall policy, click the application policy set in the list to which the new firewall policy will belong. The **Edit Application Policy Sets** window appears, displaying a field for the description of the selected policy set and listing firewall policies associated with the set. See Figure 4 on page 13, where the **HRPolicySet** has been selected.

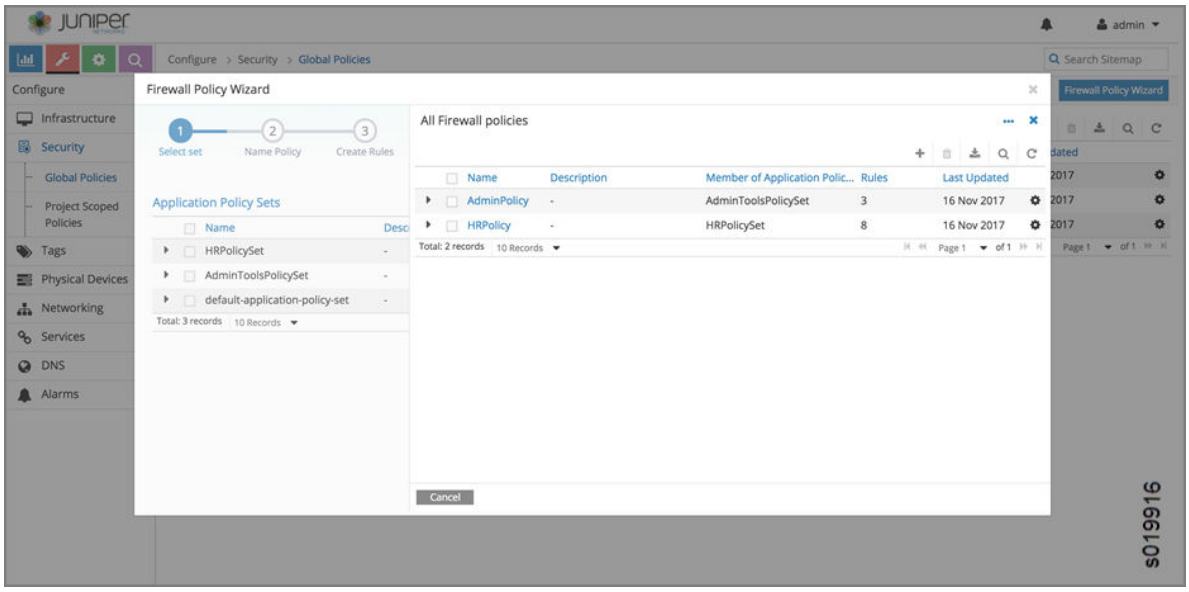
Figure 4: Edit Application Policy Sets



4. To view all firewall policies, click the Application Policy Sets link in the left side.

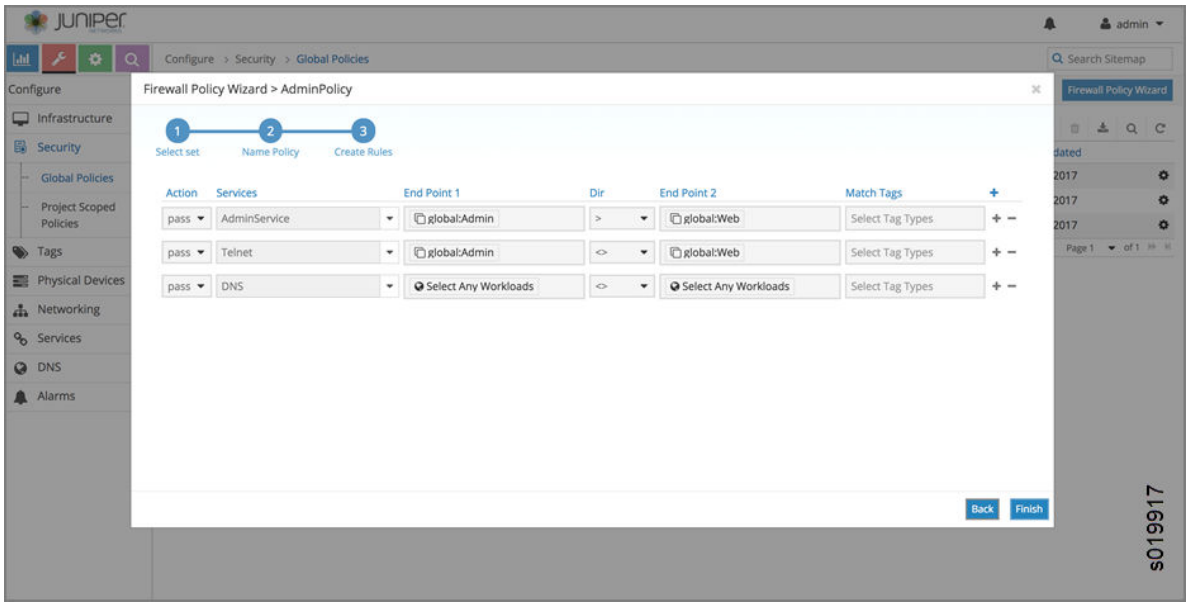
   See Figure 5 on page 13.

Figure 5: All Firewall Policies



5. Select any listed firewall policy to view or edit the rules associated with that policy. See Figure 6 on page 14, where all the rules for the **AdminPolicy** are listed. Use the dropdown menus in each field to add or change policy rules, and use the +, - icons to the right of each rule to add or delete the rule.

Figure 6: Firewall Policy Rules

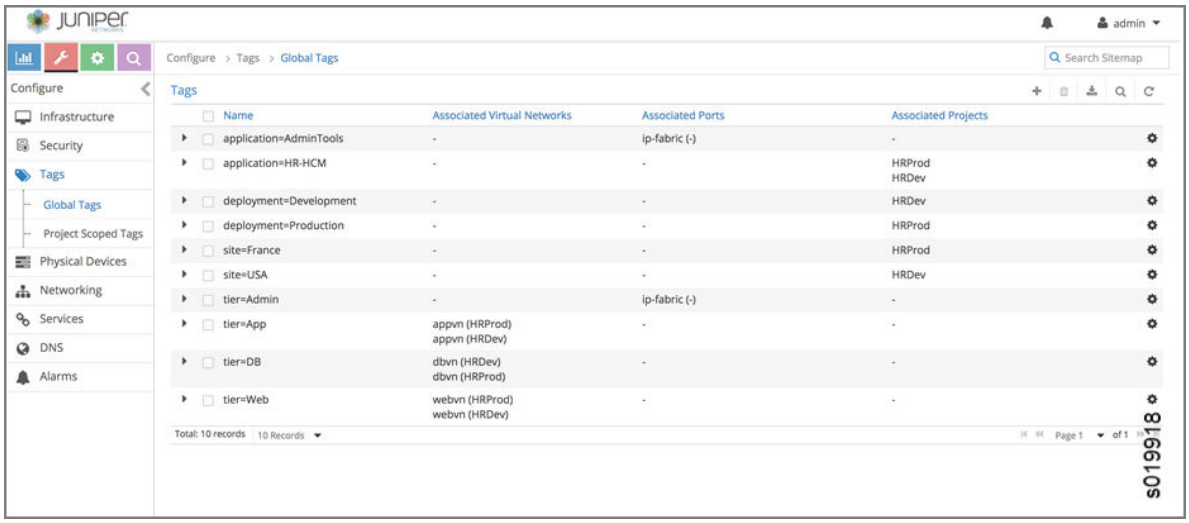


Managing Policy Tags

You can use the Contrail web user interface to create and manage the tags used to provide granularity to security policies. You can have global tags, applicable to the entire system, or project tags, defined for specific uses in specific projects.

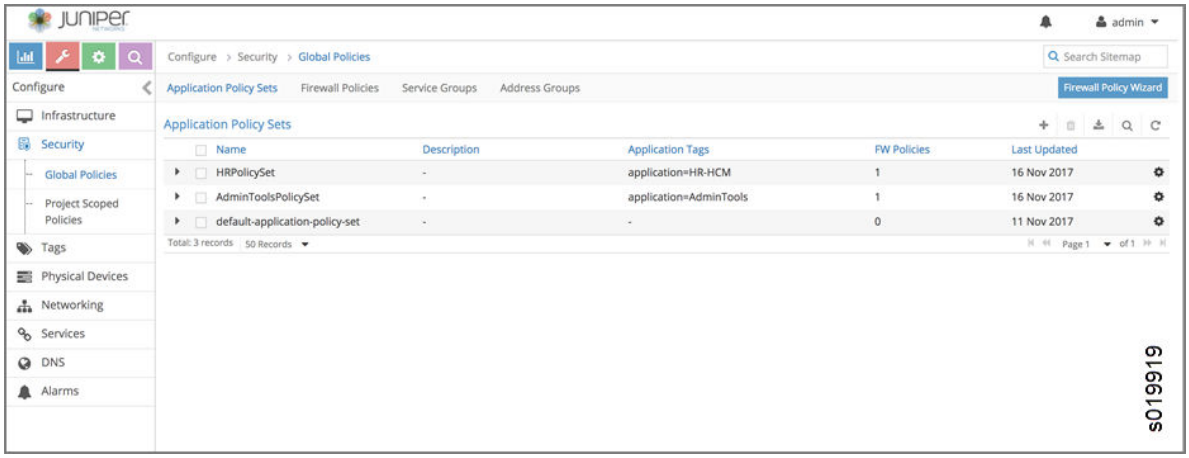
1. To manage policy tags, go to **Configure > Tags > Global Tags**. The **Tags** window appears, listing all of the tags in use in the system, with the associated virtual networks, ports, and projects for each tag. Tags are defined first by type, such as application, deployment, site, tier, and so on. See Figure 7 on page 15.

Figure 7: Tags



2. You can click through any listed tag to see the rules to which the tag is applied. See Figure 8 on page 15, which shows the application tags that are applied to the current application sets. You can also reach this page from **Configure > Security > Global Policies**.

Figure 8: View Application Tags

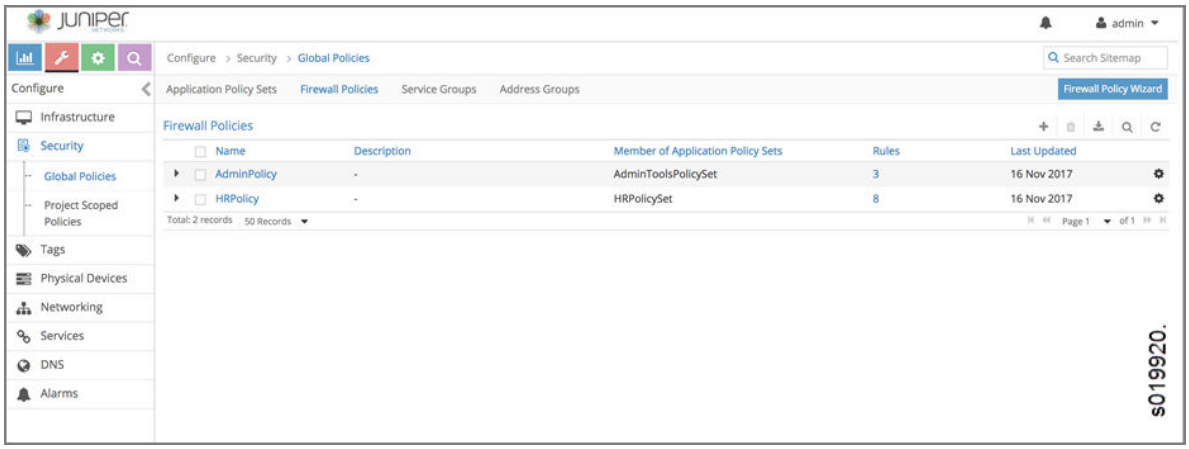


Viewing Global Policies

From **Configure > Security > Global Policies**, in addition to viewing the policies includes in application policy sets, you can also view all firewall policies, all service groups policies, and all address groups policies.

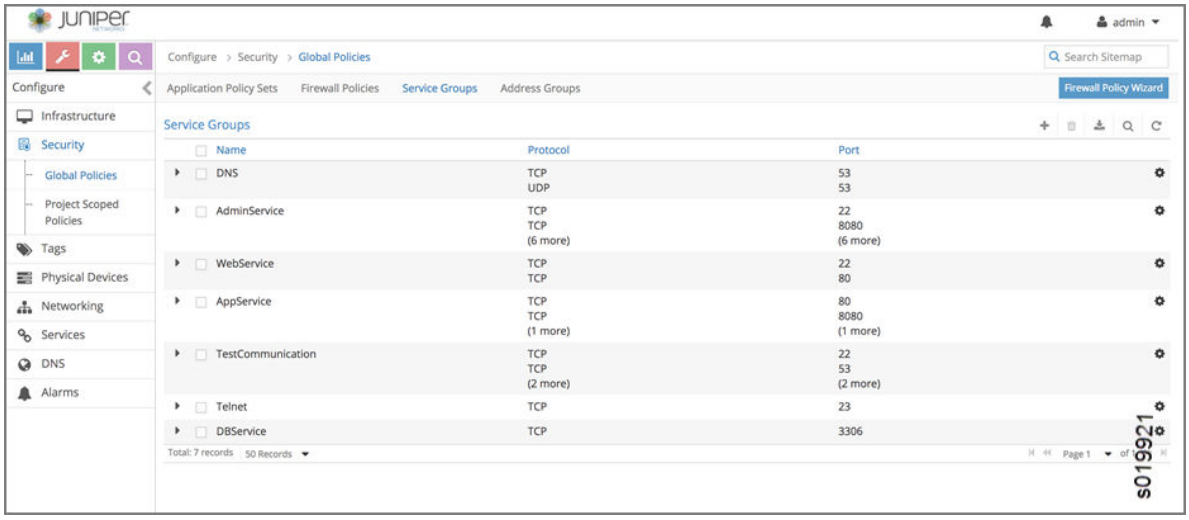
1. To view and manage the global firewall policies, from **Configure > Security > Global Policies**, click the Firewall Policies tab to view the details for system firewall policies, see Figure 9 on page 16

Figure 9: Firewall Policies



2. To view and manage the service groups policies, from **Configure > Security > Global Policies**, click the **Service Groups** tab to view the details for system policies for service groups, see Figure 10 on page 16.

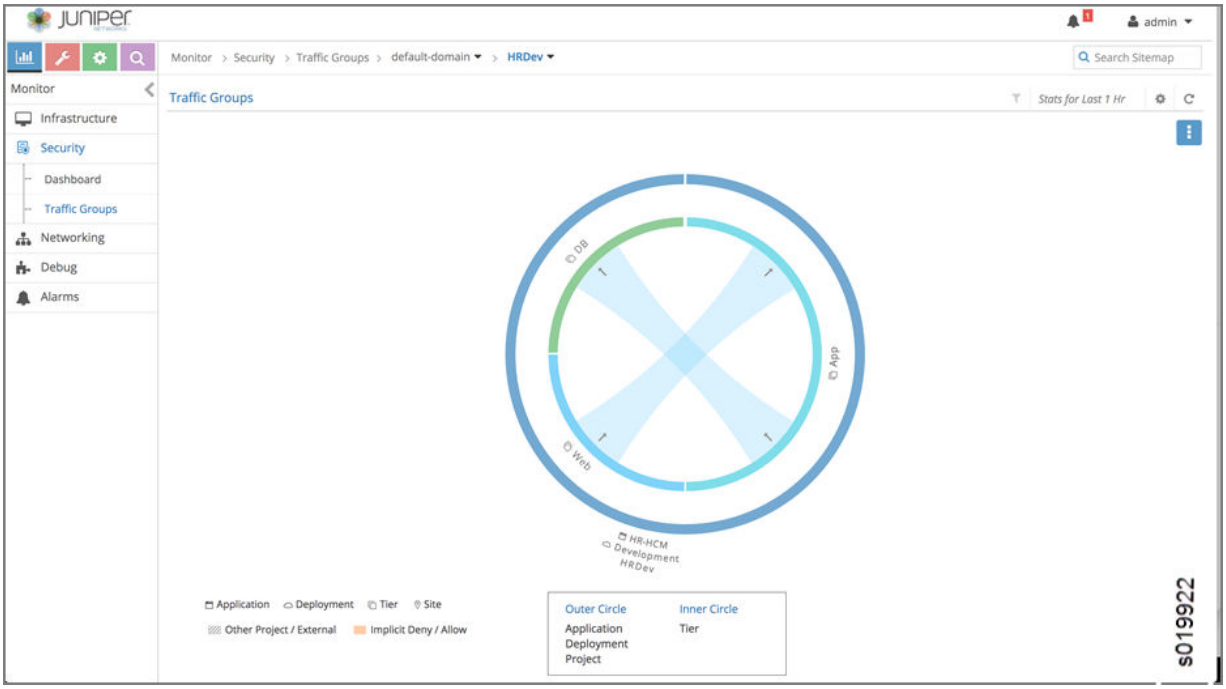
Figure 10: Service Groups



Visualizing Traffic Groups

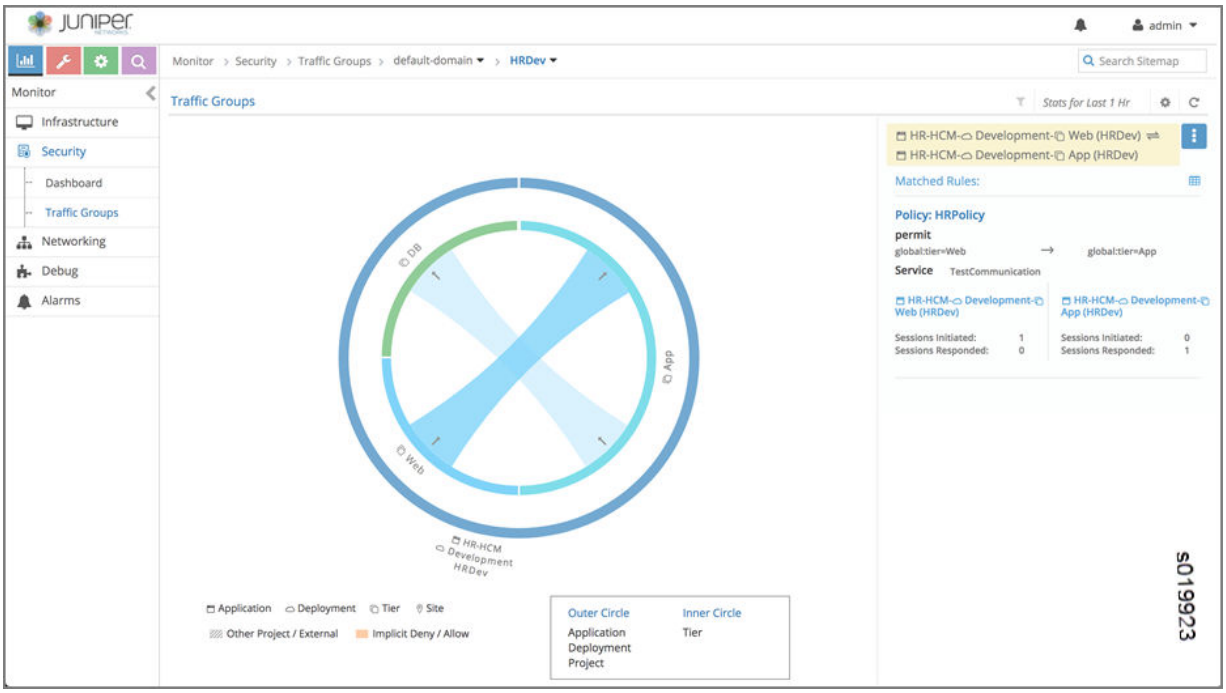
Use **Monitor > Security > Traffic Groups** to explore visual representations of how policies are applied to traffic groups. See Figure 11 on page 17, which is a visual representation of the source and destination traffic for the past one hour of a traffic group named Traffic Groups. The outer circle represents traffic tagged with application, deployment, or project. The inner circle represents traffic tagged with tier. The center of the circle shows the traffic origination and destination.

Figure 11: Traffic Groups



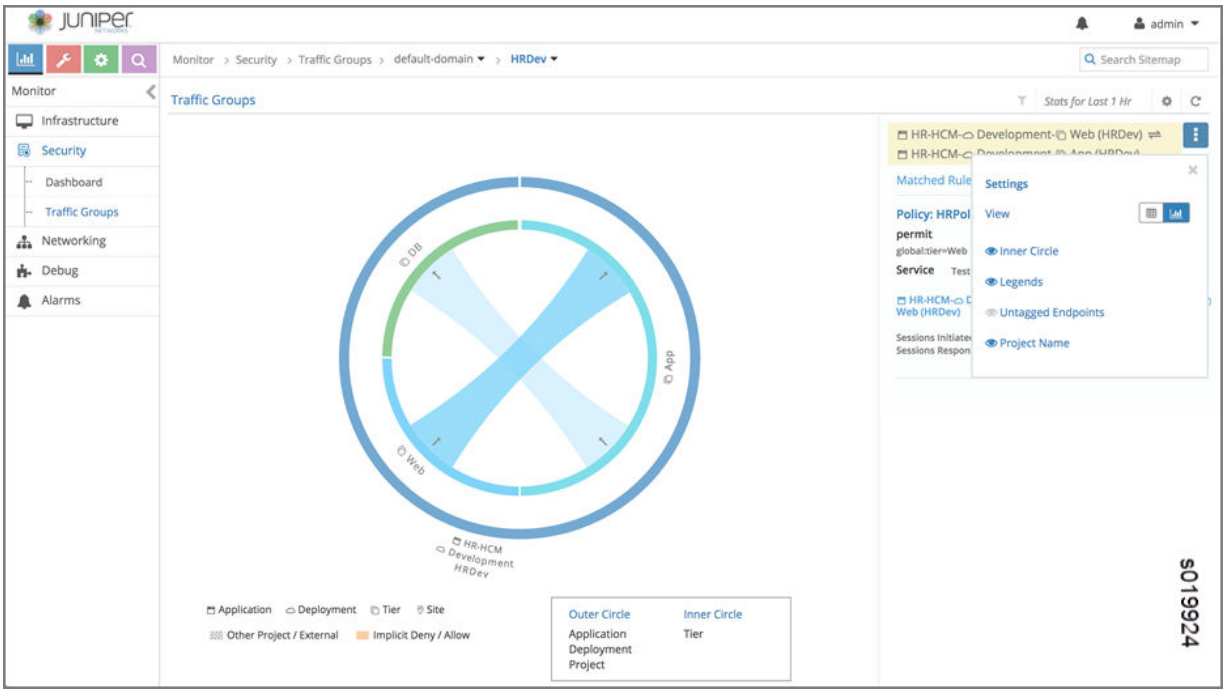
You can click in the right side of the screen to get details of the policy rules that have been matched by the selected traffic. See Figure 12 on page 18.

Figure 12: Traffic Groups, Policy Details



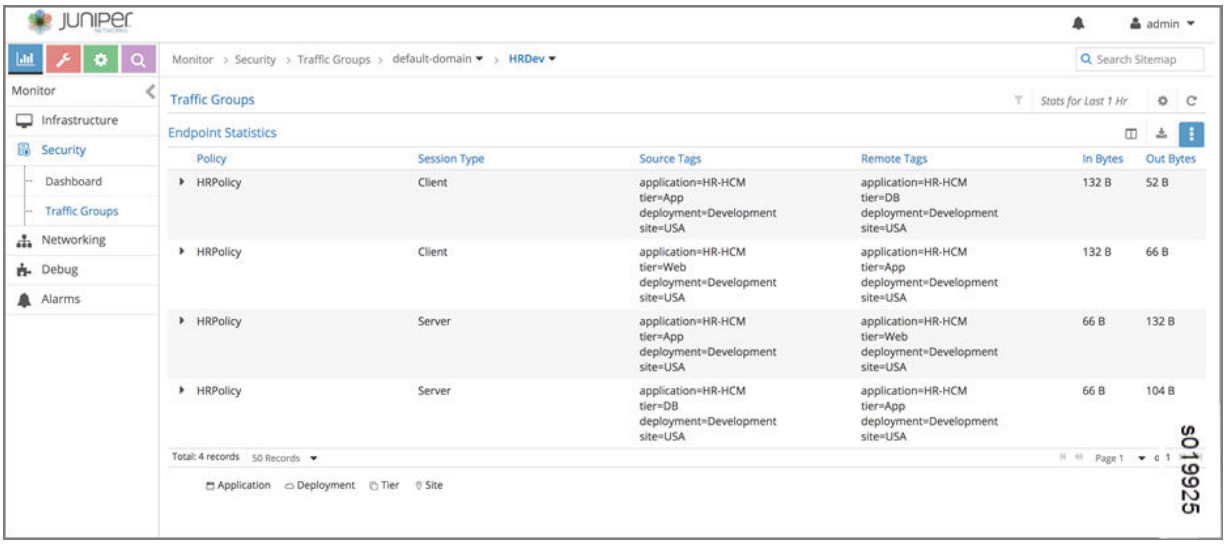
You can click in the right side of the screen to get to the **Settings** window, where you can change the type of view and change which items appear in the visual representation. See Figure 13 on page 19.

Figure 13: Traffic Groups, Settings



You can click on the name of a policy that has been matched to view the endpoint statistics, including source tags and remote tags, of the traffic currently represented in the visual. See Figure 14 on page 19.

Figure 14: Traffic Groups, Endpoint Statistics



You can click deeper through any linked statistic to view more details about that statistic, see Figure 16 on page 20 and Figure 16 on page 20.

Figure 15: Traffic Groups, Details

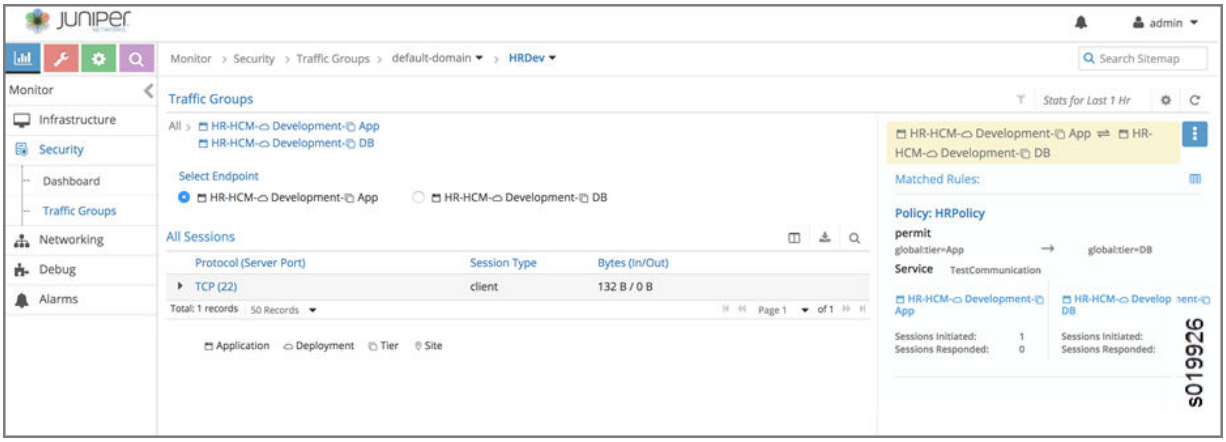
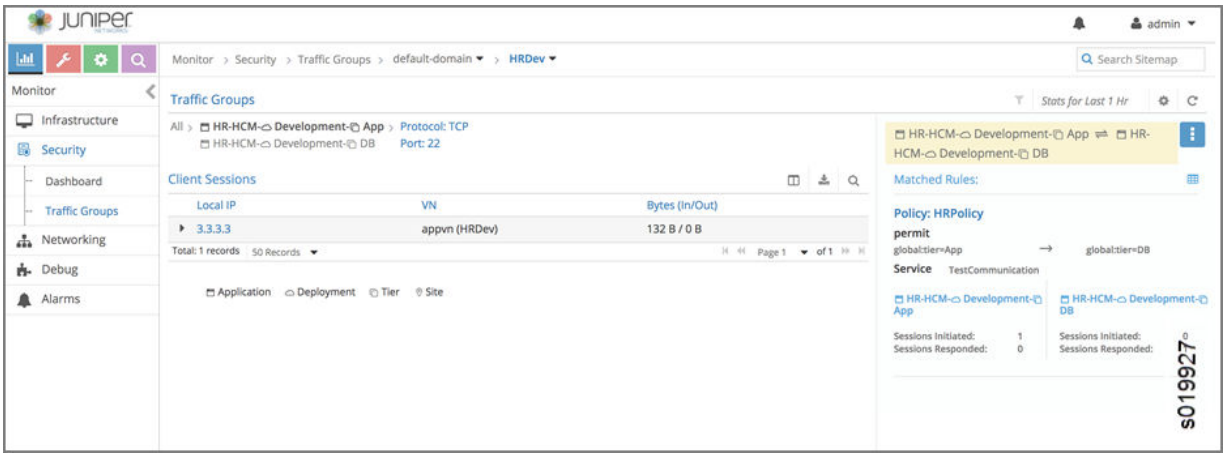
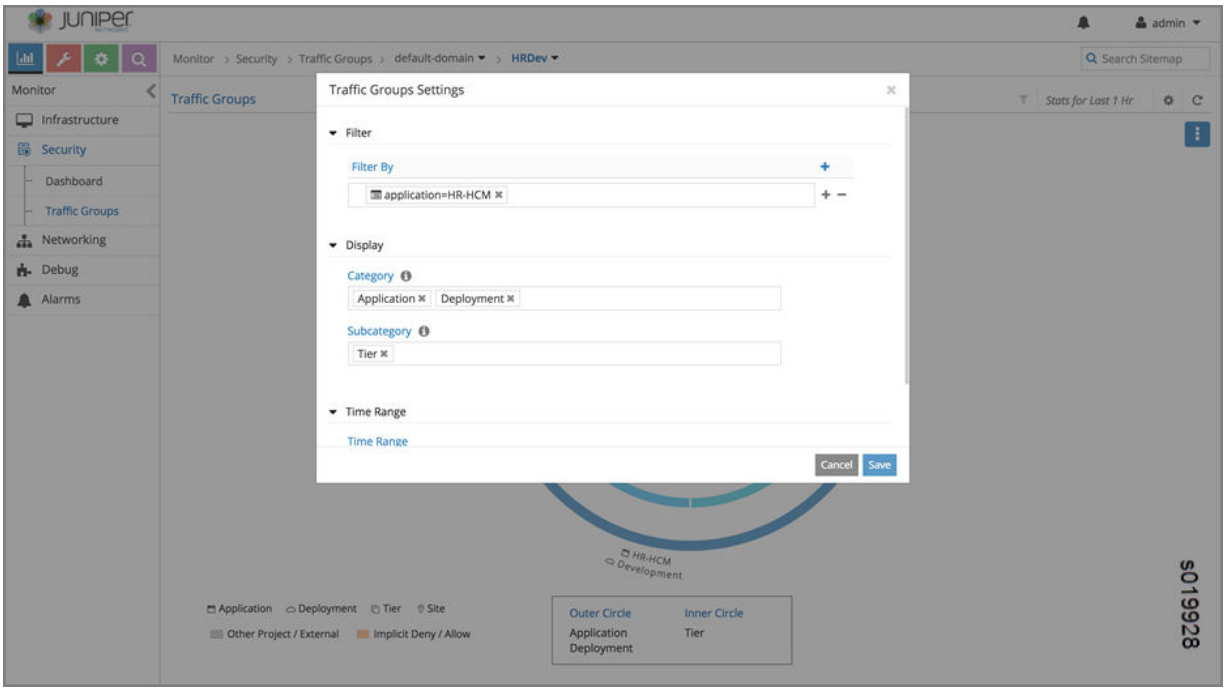


Figure 16: Traffic Groups, Details



You can change the settings of what statistics are displayed in each traffic group at the **Traffic Groups Settings** screen see Figure 17 on page 21.

Figure 17: Traffic Groups Settings



Release History Table

Release	Description
21.3	Starting in Contrail Networking Release 21.3, security policy allows optional user-defined tags, which enables you to define tag IDs along with tag names.
21.3	Starting in Contrail Networking Release 21.3, you can create a predefined tag type with user-defined tag value ID
21.3	Starting in Contrail Networking Release 21.3, custom tags supports user-defined tags.

RELATED DOCUMENTATION

Security Policy Features in OpenStack | 22

Security Policy Features in OpenStack

IN THIS SECTION

- Overview of Existing Network Policy and Security Groups in OpenStack | 22
- Security Policy Enhancements | 23
- Configuration Objects | 23

Overview of Existing Network Policy and Security Groups in OpenStack

Contrail virtual networks are isolated by default. Workloads in a virtual network cannot communicate with workloads in other virtual networks, by default. A neutron router or a Contrail network policy may be used to connect two virtual networks. In addition, Contrail network policy also provides security between two virtual networks by allowing or denying specified traffic.

OpenStack security groups allow access between workloads and instances for specified traffic types and any other types are denied.

A security policy model for any given customer first needs to map to the OpenStack network policy framework and security group constructs.

In modern cloud environments, workloads are moving from one server to another, one rack to another and so on. Therefore, users must rely less on using IP addresses or other network coordinates to identify the endpoints to be protected. Instead users must leverage application attributes to author policies, so that the policies don't need to be updated on account of workload mobility.

You might want to segregate traffic based on the different categories of data origination, such as:

- Protecting the application itself
- Segregating traffic for specific component tiers within the application
- Segregating traffic based on the deployment environment for the application instance
- Segregating traffic based on the specific geographic location where the application is deployed

There are many other possible scenarios where traffic needs to be segregated.

Additionally, you might need to group workloads based on combinations of tags. These intents are hard to express with existing network policy constructs or Security Group constructs. Besides, existing policy constructs leveraging the network coordinates, must continually be rewritten or updated each time workloads move and their associated network coordinates change.

Security Policy Enhancements

As the Contrail environment has grown and become more complex, it has become harder to achieve desired security results with the existing network policy and security group constructs. The Contrail network policies have been tied to routing, making it difficult to express security policies for environments such as cross sectioning between categories, or having a multi-tier application supporting development and production environment workloads with no cross environment traffic.

Starting with Release 5.1, Contrail Networking supports the OpenStack Neutron Firewall version 2 API extension known as Neutron FWaaS (Firewall as a Service). The Neutron API enhancements make the existing FWaaS more granular by giving you the ability to apply the firewall rules at the port level rather than at the router level, and to have different firewall policies with different rules applied to inbound versus outbound connections. Support is extended to various types of Neutron ports, including VM ports and SFC ports as well as router ports. It also provides better grouping mechanisms (firewall groups, address groups and service groups). Finally, the Firewall Group enables firewall policies to be bound to Neutron ports.

Related enhancements to the OpenStack Neutron and Contrail security groups API include:

- Firewall rules support deny, reject, description, and admin status attributes
- A share attribute for firewall rules allow them to be shared between different projects
- Filtering based on the source and destination address prefix and port rather than just the remote destination
- Firewall groups reference firewall rules through a firewall policy, allowing reuse of shareable firewall policies that are referenced by multiple firewall groups

Configuration Objects

The following are the configuration objects for the new security features.

- firewall-policy
- firewall-rule

- policy-management
- application-policy
- service-group
- address-group
- tag
- global-application-policy

For more information on security policies in Contrail, see "Security Policy Features" on page 2.

RELATED DOCUMENTATION

Security Policy Features | 2

Policy Generation

The policy generation feature in Contrail Release 5.1 automates the task of policy creation based on observed traffic flows. Contrail creates and enforces intent-based policies. In many cases, Contrail Security is deployed in brownfield environments, in which inter-applications and intra-application traffic policies are pre-existing. However, in greenfield deployments and in complex environments, where many applications are communicating internally and externally, creating policies one-by-one is a tedious and time consuming task. Sometimes, manually created policies do not perform as per expectations in real-time traffic or sometimes you might create extra policies which are never used by the applications. The policy generation feature simplifies this process of creating policies by automating the generation of policies based on application communication.

The policy generation feature aids in the creation of policies based on observed traffic, without enforcing any new policies. In order to generate policies, workloads VMs or Containers need to be grouped within Contrail objects like virtual networks and Projects. Subsequently, tags must be created and associated with Projects, virtual networks or ports. In the policy generation mode, traffic from selected applications is allowed to pass for a selected period of time. The vRouter observes and forwards all traffic between the selected applications because the implicit rule is to allow all traffic to pass. On the basis of this observation, the vRouter generates a draft policy which is saved in the policy draft mode. You can review the draft policy and edit it as required before enforcing the policy. The policy generation feature significantly reduces the burden of policy creation from scratch.

To use the policy generation feature, the sequence of high level steps are as listed here:

- Create tags.
- Associate tags with projects, virtual networks (VNs) or ports.
- Run traffic.
- Edit the generated policies available in draft mode.
- Commit the (optionally) edited policies to enforce them

RELATED DOCUMENTATION

Configuring Policy Generation | 25

Configuring Policy Generation

To configure policy generation:

1. Create tags.

   In the Command UI, navigate to **Security > Tags**. Tags are key and value pairs. Create tags as appropriate for your environment. Tags can be created with a Project scope or a Global scope.

2. Associate tags.

   Tags created can be associated with either individual virtual machines (VMs) or container ports or with groups of virtual machines or containers at either the VN level or the project level. Tags associated with a VN are inherited by all VMs or containers in that VN. Similarly, tags associated with a project get inherited by all VNs in that project and in turn by all VMs or containers in each VN in that project. To associate tags at any of these levels, perform the following steps:

   - Project

     Navigate to the **Projects** tab, double click on the project and associate the tags.

   - Virtual Network

     Navigate to **Overlay > Virtual Networks** and click **Edit** to add appropriate tags.

   - Ports

     Navigate to **Overlay > ports** and click **Edit** to add appropriate tags.

3. When running policy generation for the first time, you must provision the policy generator module using the following commands:

a. Download the Contrail Security Apps tarball from the Support - Software Downloads site. Untar the **.tgz** file

```
untar contrail-security-apps-*.tgz
```

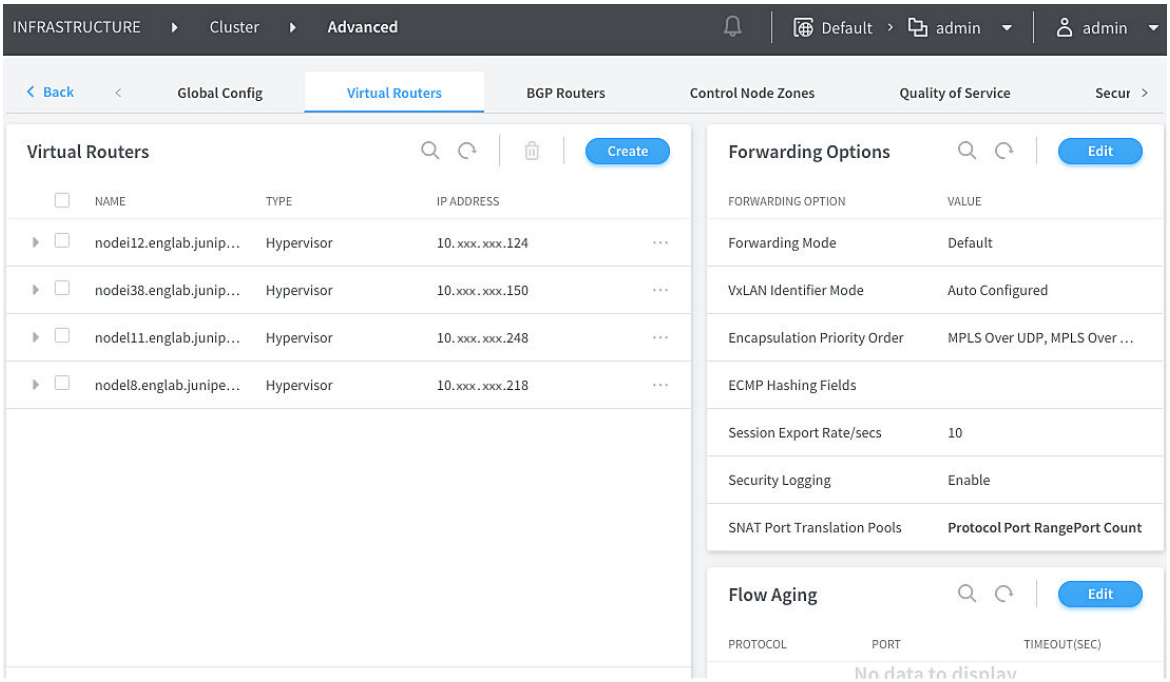
b. `cd contrail-security-apps`

c. `edit` **ansible/inventory/inventory.yml** file and specify the required values. For a sample of the **inventory.yml** file, see "No Link Title" on page 31.

d. `ansible-playbook -i ansible/inventory/inventory.yml ansible/playbooks/deploy_and_run_all.yml`

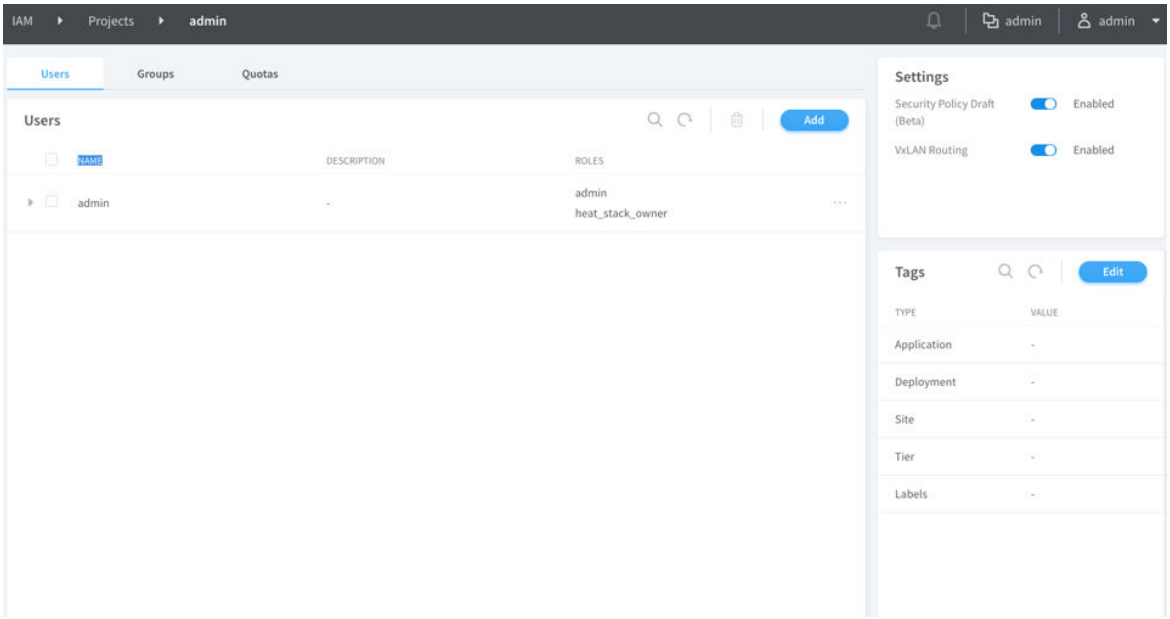
4. Specify the session export rate. Navigate to **INFRASTRUCTURE > Cluster > Advanced Options**.

   Click the **Virtual Routers** tab, click **Edit** under **Forwarding Options** and enter the **Session Export Rate/secs** value.



5. Enable the security policy draft mode, either for the Project scope or for Global scope depending on your requirement.

   For Project scope, navigate to **IAM > Projects**, select and click the project and enable the **Security Policy Draft** mode under **Settings**.

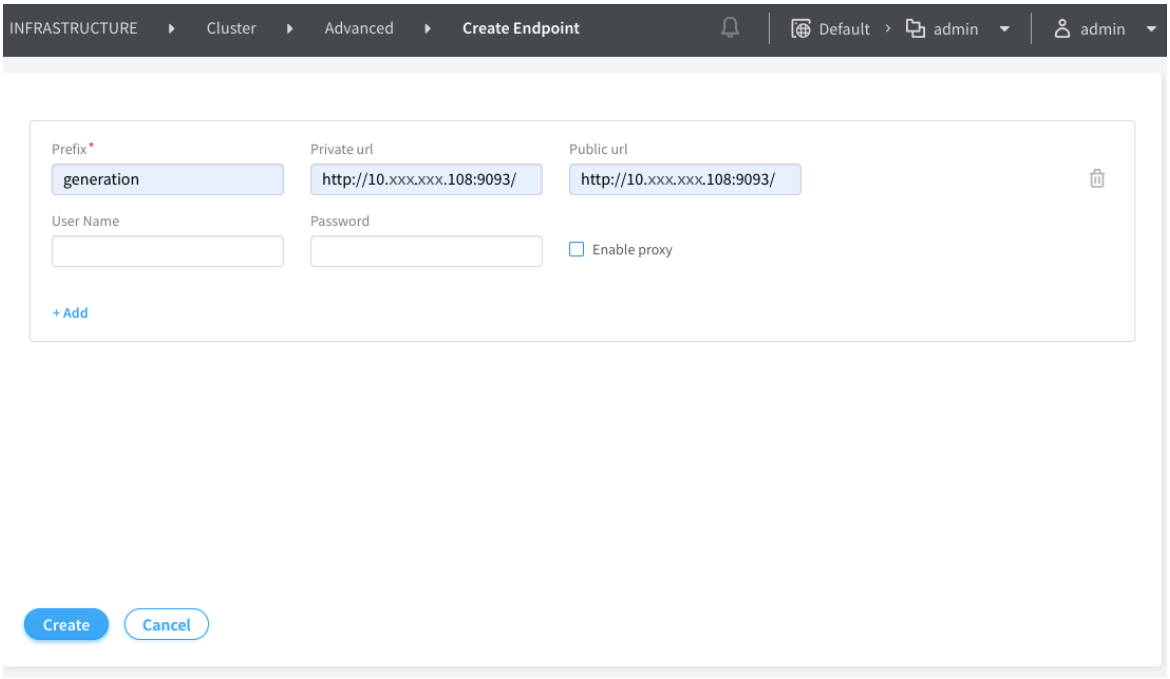


For Global scope, navigate to **INFRASTRUCTURE > Cluster > Advanced Options**, and click the **Global Config** tab. The **Edit System Configuration** page appears. Click **Edit** and enable the **Security Policy Draft** mode. Click **Save.**

6. Enable policy generation endpoint.

   Navigate to the **INFRASTRUCTURE > Cluster > Advanced Options**, click the **Endpoints** tab, and click **Create**. The **Create Endpoint** page appears.

7. Enter **generation** under **Prefix** and enter the required URLs. Click **Create** to save the endpoint.

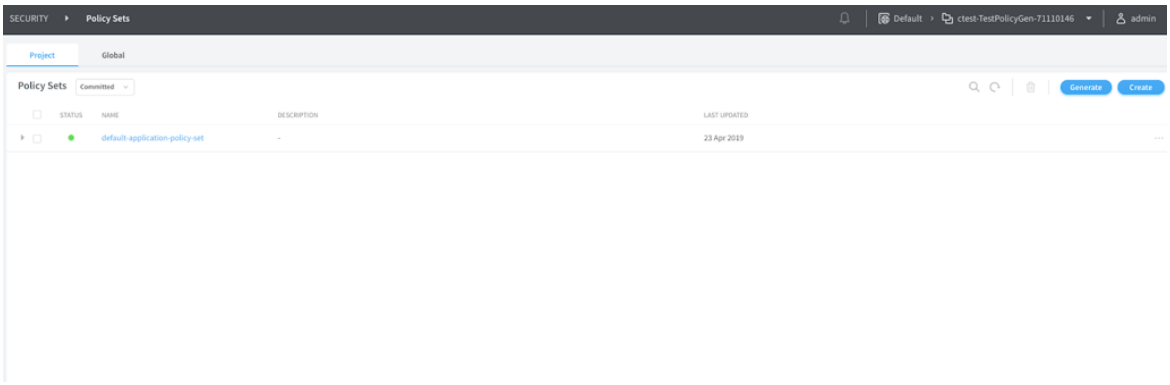


8. Generate traffic between the applications.

9. Generate policies.

   Navigate to **Security > Policy Sets** and click **Generate Policy**. The **Generate Project Policy** page appears.

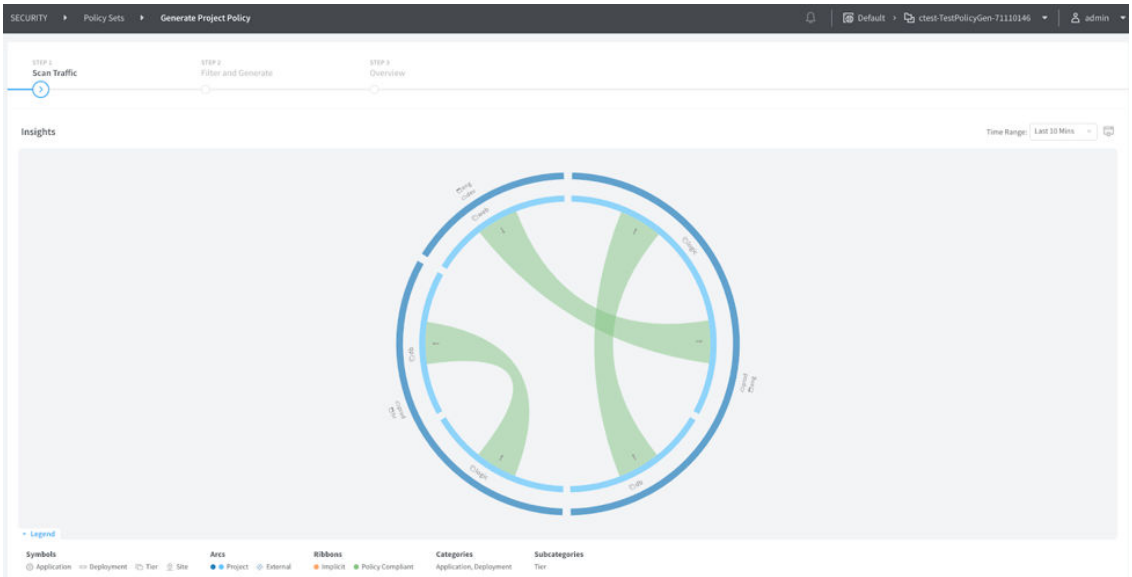
   The **Generate Project Policy** page has three steps.



   a. **Step 1 Scan Traffic**

      Each vRouter scans the traffic it sees between and within applications. The controller analyzes the observed traffic patterns and displays the observed traffic in a graphical visualization. Arcs inside the circular graph represent the different observed flows. Mouse over the arcs to view additional details about the applications involved in that flow, other tags associated with the endpoints, and other flow characteristics.

      The period of traffic considered as an input for policy generation can be customized by editing the **Time Range**. The default time range is 10 minutes. Click **Next** to proceed with policy generation.

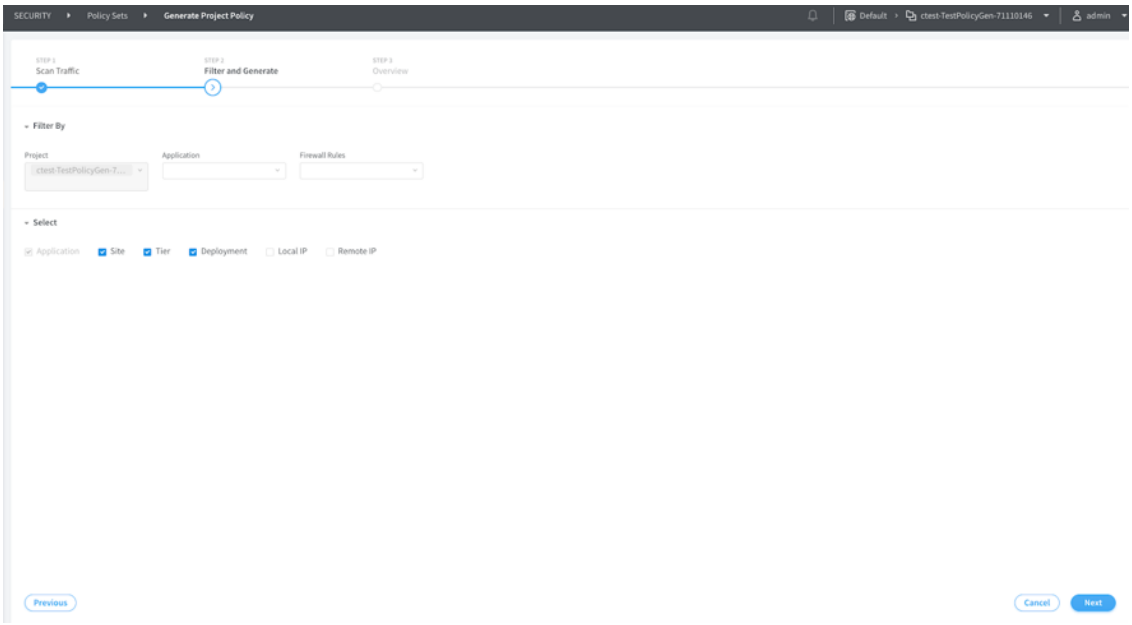


   b. **Step 2 Filter and Generate**

You can filter traffic and generate a draft policy based on the selected filters. By default, the current project is selected and the predefined tags, application, deployment, tier, and site are selected. Selected tags must be associated with the workloads, because not having these tags in flow records creates unknown flows.

If you haven't attached any of the predefined tags to workloads and you don't need the tags to be part of the policies, deselect them. However, it is mandatory to select at least the application tag, else flows designated with unknown are displayed.

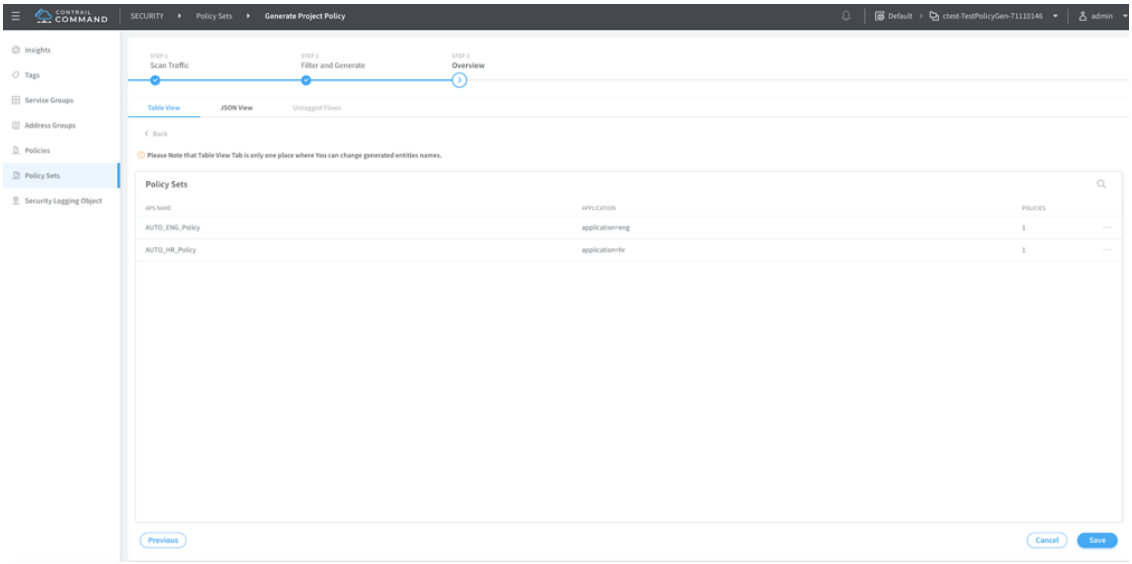
Click **Next**. The default firewall rules allow all traffic.



c. Step 3 Overview

You can view the application policy sets based on the selected input parameters. You can view the application policy sets in a tabular format as well as JSON format. You can also view traffic flows for untagged applications.

Click **Save** to save the application policy set and generate the draft policy.



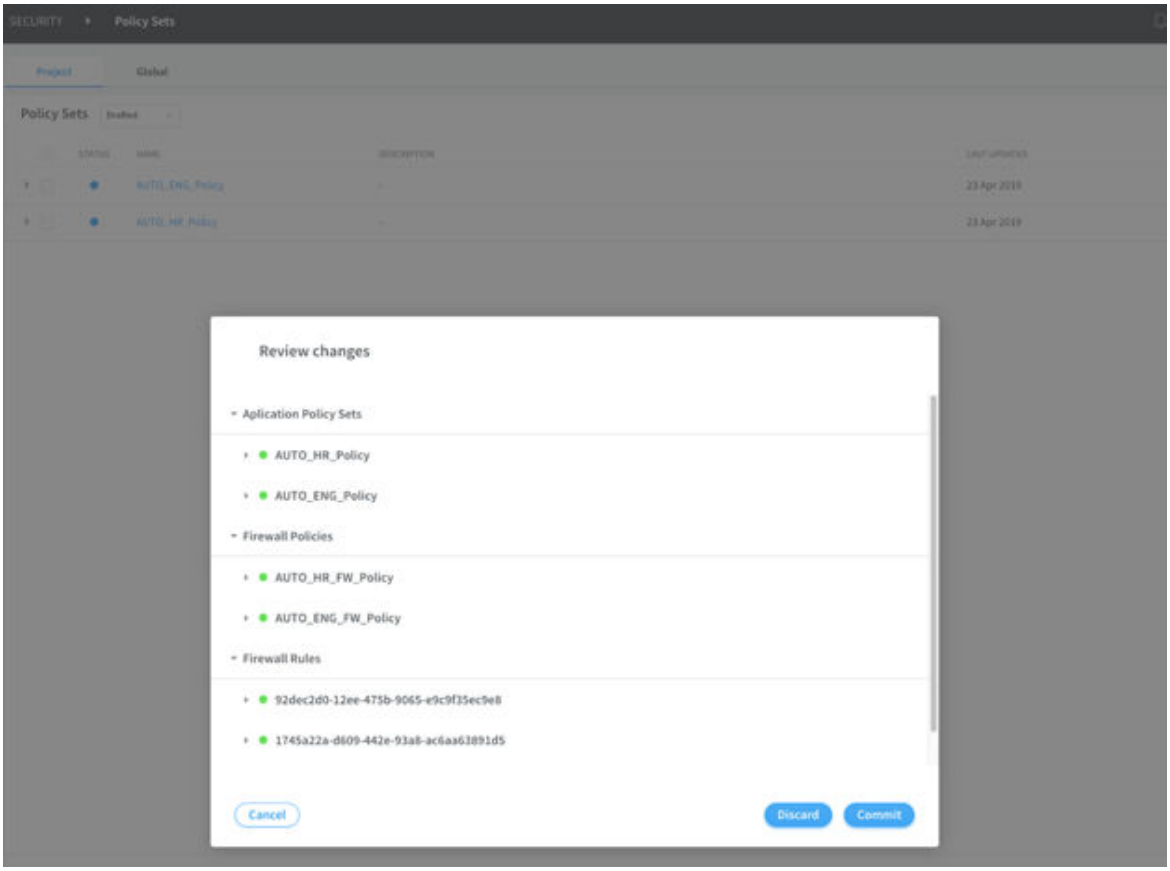
10. Review the draft policy.

    The **Security > Policy Sets** page is displayed with the draft application policy set. Click the draft application policy set to view details about the policies.

    Select the draft application policy set and click **Review**.

11. The **Review Changes** page appears listing the policies in the draft mode. You can review the draft policy, edit it as required and click **Commit** to enforce the application policy set.

    Alternatively, click **Discard** to discard the generated application policy set.



Sample inventory.yml file

```
all:
  hosts:
    localhost:
      ansible_connection: local

  vars:
    PATH_POLICY_GENERATION_CONFIG: '/etc/contrail/securityapps/'
    # Policy generator API port
    rest_api_port: {{ pg_port }}
    # Policy generator API server ip address
    rest_api_ip: {{ pg_ip }}
    # Policy generator logs and log level
    log_file: '/var/log/contrail/policy-generation-server.log'
    log_console_level: logging.DEBUG

    # Registry inventory
    CONTAINER_REGISTRY: {{ pg_registry }}
```

```
CONTRAIL_VERSION: {{ pg_version }}
ansible_user: 'root'

# Config API server
api_server_listen_port: {{ config_port }}
api_server_ip: {{ config_ip }}

# Analytics IP & Port 8081 for remote host, 8181 for localhost
analytics_ip: {{ analytics_ip }}
admin_port: {{ analytics_port }}

# Keystone credentials with user
admin_password: {{ admin_password }}
admin_tenant_name: {{ admin_tenant }}
admin_user: {{ admin_user }}
auth_host: {{ auth_host }}
auth_port: {{ auth_port }}
auth_protocol: {{ auth_protocol }}
auth_version: {{ auth_version }}
auth_type: password
user_domain_name: Default
project_domain_name: Default
region_name: RegionOne
```

RELATED DOCUMENTATION

Policy Generation | 24

Host-based Firewalls

IN THIS SECTION

- Host-based Firewalls Overview | 33
- Deploying Host-based Firewalls | 33

This topic discusses the host-based firewall feature introduced in Contrail Networking Release 2003.

Host-based Firewalls Overview

Contrail Networking Release 2003 provides *beta* support for the host-based firewall feature which enables the creation of next generation firewalls using cSRX devices. Next-generation firewalls provide the ability to filter packets based on applications. They provide deep-packet inspection with intrusion detection and prevention at the application level.

Historically the vRouter has supported creation of regular Layer 4 firewall policies. To create Layer 7 application-level firewall policies, Contrail Networking uses service chaining. However, service chaining works only in cases of inter-virtual network traffic and not for intra-virtual network traffic. The host-based firewall feature offers next-generation firewall functions for traffic originating and ending in the same virtual network as well as in different networks. It uses the bump in the wire mode where the firewall instance does not change the packet format or Layer 2 header but applies Layer 7 policies on the packet.

Additionally, the host-based firewall feature uses tag-based policies to steer traffic. Tags are a simple and intuitive way of applying firewall intents and have the power to span multiple VNs, scale better, and can be attached at a VMI level as opposed to service chains. You can steer traffic towards the host-based firewall instance using tag-based policies. Policies are used to steer only specific traffic since the host-based firewall instance requires a fair amount of compute resources.

Also, host-based firewalls provide next-generation firewall functions closer to the workloads and can integrate with third-party firewall features.

Deploying Host-based Firewalls

IN THIS SECTION

- Prerequisites | 34
- Topology | 34
- Deployment Instructions | 36

Perform the following steps to deploy a host-based firewall. In this example we use Kubernetes as the orchestration platform since Kubernetes provides the flexibility to instantiate host-based firewall instances on selected compute nodes. The high-level list of steps are as follows:

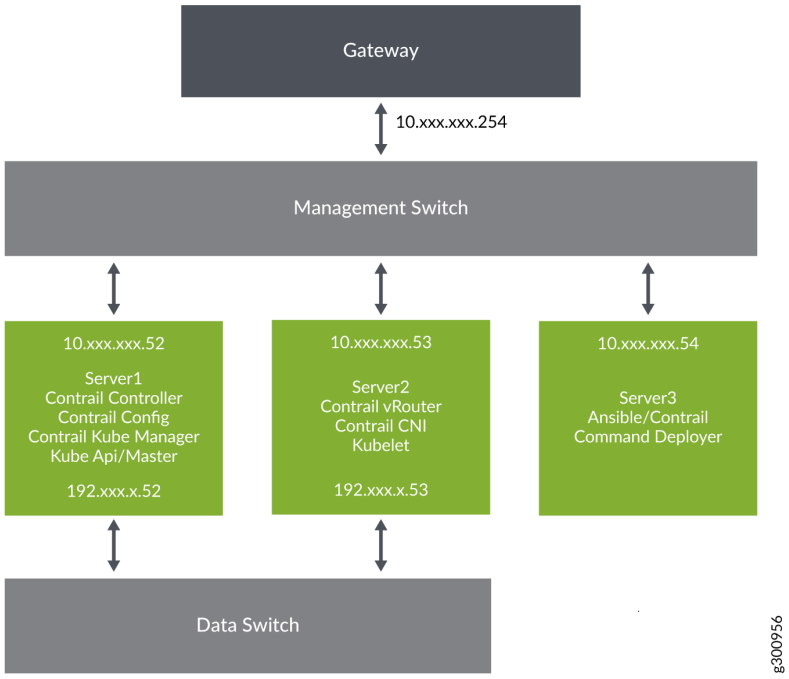
Prerequisites

Install a Contrail-Kubernetes setup by using either contrail-ansible deployer or Contrail Command. See Provisioning of Kubernetes Clusters or Installing Standalone Kubernetes Contrail Cluster using the Contrail Command UI.

Topology

Consider the following sample Contrail-Kubernetes topology and **instances.yml** file.

Figure 18: Sample Contrail-Kubernetes Topology



Sample **instances.yaml** file

```
deployment:
    orchestrator: kubernetes
    deployer: contrail-ansible-deployer
provider_config:
  bms:
```

```
    ssh_pwd: <password>
    ssh_user: username
    ntpserver: <IP NTP server>
    domainsuffix: <domain-suffix>
instances:
  server1:
    provider: bms
    ip: 10.xx.xx.52
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      k8s_master:
      kubemanager:
  server2:
    provider: bms
    ip: 10.xx.xx.53
    roles:
      k8s_node:
      vrouter:
    VROUTER_GATEWAY: 77.xx.x.100
global_configuration:
  CONTAINER_REGISTRY: 50.xx.xx.50:5000
  REGISTRY_PRIVATE_INSECURE: True
contrail_configuration:
  CONTRAIL_VERSION: <contrail-version>
  CLOUD_ORCHESTRATOR: kubernetes
  CONTROLLER_NODES: 10.xx.xx.52
  CONTROL_NODES: 77.xx.xx.20
  KUBERNETES_API_NODES: 77.xx.xx.20
  KUBERNETES_API_SERVER: 77.xx.xx.20
  CONTAINER_REGISTRY: 50.xx.xx.50:5000
  REGISTRY_PRIVATE_INSECURE: True
  VROUTER_GATEWAY: 77.xx.x.100
```

Deployment Instructions

IN THIS SECTION

- Procedure | 36

Procedure

Step-by-Step Procedure

To deploy a host-based firewall.

1. Create a namespace in Kubernetes. The namespace creates an equivalent project in Contrail.

   #### Step-by-Step Procedure

   a. Create a namespace.

   ```
   kubectl create namespace hbf
   ```

   b. Enable isolation on the namespace.

   ```
   kubectl annotate namespace hbf "opencontrail.org/isolation"="true"
   ```

   c. Verify namespace creation.

   ```
   kubectl get ns  hbf
           NAME   STATUS   AGE
           hbf    Active   2d5h
   ```

2. Label the compute nodes for the host-based firewall function.

   #### Step-by-Step Procedure

a. Get the list of compute nodes.

```
kubectl get nodes
NAME      STATUS     ROLES    AGE   VERSION
Server2   Ready      <none>   16d   v1.12.9
server1   NotReady   master   16d   v1.12.9
```

b. Select the nodes for the host-based firewall function and label them.

```
kubectl label node server type=hbf
```

Where *server* is the Kubernetes node name and *hbf* is the label.

3. Create a Kubernetes secret object in the namespace created earlier to pull the cSRX image.

```
kubectl create secret docker-registry hbf --docker-server=hub.juniper.net/security --docker-
username=testuser --docker-password=testpassword -n hbf
```

4. Create a *hbs* object in the previously created namespace.

Step-by-Step Procedure

a. Create a python file with the following content and use the following command on the config_api Docker container.

```
docker exec -it config_api_1 bash
     (config-api)[root@server /]$ python hbs.py
```

b.

```
cat hbs.py
    ----------------

from vnc_api import *
from vnc_api.vnc_api import VncApi
from vnc_api.vnc_api import Project
from vnc_api.vnc_api import HostBasedService
from vnc_api.exceptions import NoIdError
from vnc_api.gen.resource_xsd import QuotaType
```

```
hbs_name = ‘hbs’  # any other user defined name can be given
project_name = ‘k8s-hbf’  # k8s is the default cluster name
# user can change according to their
admin_user = 'admin'
admin_password = '<admin-password>'
admin_project = 'admin'
api_node_ip = '10.xx.xx.52'  # change according to your topology
api_node_port = '8082'  # config api port
default_domain = 'default-domain'

if __name__ == "__main__":
    api = VncApi(
        username=admin_user,
        password=admin_password,
        tenant_name=admin_project,
        api_server_host=api_node_ip,
        api_server_port=api_node_port)
'''Creates a project using vnc apis if it doesn’t exist already and enable
hbf, if it’s not enabled already'''

try:
    project = api.project_read(fq_name=[default_domain, project_name])
except NoIdError:
    project = Project(name=project_name)
    puuid = api.project_create(project)
    project = api.project_read(fq_name=[default_domain, project_name])
project.set_quota(QuotaType(host_based_service=1))
api.project_update(project)

try:
    hbs = api.host_based_service_read(fq_name=project.fq_name + [hbs_name])
    hbs_created = False
except NoIdError:
    hbs = HostBasedService(hbs_name, parent_obj=project)
    hbs_created = True

if hbs_created:
    api.host_based_service_create(hbs)
else:
    api.host_based_service_update(hbs)
```

5. Create a daemonset for the host-based firewall instances. By default, host-based firewall instances run on all compute nodes. You can choose to run host-based firewall instances on specific compute nodes only by labeling them as shown in "2.b" on page 37. The host-based firewall instance has three interfaces. The traffic flows in to the left interface and firewall functions are performed on the packets and traffic flows out of the right interface. The management interface is the default pod network.

   Step-by-Step Procedure

   a. Generate a **ds.yaml** file as shown in the following example to create a daemonset with the cSRX container image. The left and right interfaces are automatically created and link to the *hbs* object with 'left' and 'right' so that traffic flows marked for the host-based firewall are steered through the cSRX device. Note that, Kubernetes objects names and values can be changed as per your requirement.

```
cat ds.yaml
      -----------
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  annotations:
    opencontrail.org/network: '{"domain":"default-domain", "project":"k8s-hbf",
      "name":"__hbs-hbf-left__"}'
  name: left
  namespace: hbf
spec:
  config: '{"cniVersion":"0.3.0", "type": "contrail-k8s-cni" }'
---
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  annotations:
    opencontrail.org/network: '{"domain":"default-domain", "project":"k8s-hbf",
      "name":"__hbs-hbf-right__"}'
  name: right
  namespace: hbf
spec:
  config: '{"cniVersion":"0.3.0", "type": "contrail-k8s-cni" }'
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
```

```
  labels:
    type: hbf
  name: hbf
  namespace: hbf
spec:
  selector:
    matchLabels:
      type: hbf
  template:
    metadata:
      annotations:
        k8s.v1.cni.cncf.io/networks: '[{"name":"left"}, {"name":"right"}]'
      labels:
        type: hbf
    spec:
      containers:
      - env:
        - name: CSRX_FORWARD_MODE
          value: wire
        image: hub.juniper.net/security/csrx:19.2R1.8
        name: csrx
        securityContext:
          privileged: true
        stdin: true
        tty: false
      imagePullSecrets:
      - name: hbf
      nodeSelector:
        type: hbf
      restartPolicy: Always
```

b. Create the Kubernetes objects which will in turn create a cSRX pod with the left and right interfaces on each compute node for each namespace.

```
kubectl create -f ds.yaml
```

c. Verify the objects, daemonset, network attachment definitions, and the cSRX pods.

```
kubectl get ds -n  hbf

NAME   DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
```

```
 hbf   1         1         1       1            1           type=hbf        34h

kubectl get network-attachment-definitions -n hbf
NAME     AGE
left     34h
right    34h

[root@nodeg12 ~]# kubectl get pods -n hbf
NAME         READY   STATUS    RESTARTS   AGE
csrx-75ncs   1/1     Running   0          34h
csrx-9qxl7   1/1     Running   0          34h
```

d. Configure the cSRX pods of the daemonset on each compute node with the following configuration.

```
set interfaces ge-0/0/0 unit 0
set interfaces ge-0/0/1 unit 0
set security policies default-policy permit-all
set security zones security-zone trust interfaces ge-0/0/0.0
set security zones security-zone untrust interfaces ge-0/0/1.0
commit
```

6. Create a network policy between left and right interfaces using the vnc API or through Contrail Command.

   Network policies are necessary only for inter-virtual network traffic and not for intra-virtual network traffic.

```
docker exec -it config_api_1 bash
     (config-api)[root@server /]$ python nwp.py



cat nwp.py
==========
from vnc_api import *
from vnc_api import vnc_api
from vnc_api.vnc_api import VncApi
from vnc_api.vnc_api import Project
from vnc_api.exceptions import NoIdError
```

```
policy_name = 'allow-left-right'.  # any policy name

if __name__ == "__main__":
    vn_left = api.virtual_network_read(fq_name= \
              [default_domain, project_name, 'k8s-left-pod-network'])
    vn_right = api.virtual_network_read(fq_name= \
              [default_domain, project_name, 'k8s-right-pod-network'])

    policy_obj = vnc_api.NetworkPolicy(policy_name, network_policy_entries= \
                  vnc_api.PolicyEntriesType([vnc_api.PolicyRuleType(direction='<>', \

action_list=vnc_api.ActionListType(simple_action='pass'), \
                                           protocol='any', \
                                           src_addresses=[vnc_api.AddressType(
                                           virtual_network=vn_left.get_fq_name_str())], \
                                           src_ports=[vnc_api.PortType(-1, -1)], \
                                           dst_addresses=[vnc_api.AddressType(
                                           virtual_network=vn_right.get_fq_name_str())], \
                                           dst_ports=[vnc_api.PortType(-1, -1)])]), \
                  parent_obj=project)

    api.network_policy_create(policy_obj)
    vn_left.add_network_policy(policy_obj, \

vnc_api.VirtualNetworkPolicyType(sequence=vnc_api.SequenceType(0, 0)))
    vn_right.add_network_policy(policy_obj, \

vnc_api.VirtualNetworkPolicyType(sequence=vnc_api.SequenceType(0, 0)))
    api.virtual_network_update(vn_left)
    api.virtual_network_update(vn_right)
```

7. Create a firewall policy and enable host-based firewall for the firewall rules.

   Create tags, application policy sets (APS), as well as create Firewall Policy and Firewall Rule under project scoped rules. Enable host-based firewall on the firewall rules and set host_based_service = True.

```
cat add_hbs_fr.py
------------------

from vnc_api import *
from vnc_api import vnc_api
```

```
from vnc_api.vnc_api import VncApi
from vnc_api.vnc_api import Project
from vnc_api.vnc_api import HostBasedService
from vnc_api.vnc_api import FirewallRule, FirewallServiceType, PortType,
FirewallRuleEndpointType
from vnc_api.exceptions import NoIdError
from vnc_api.vnc_api import ActionListType


if __name__ == "__main__":
    try :
        rule = api.firewall_rule_read(fq_name=fr_fq_name)
        #fr_fq_name = [default_domain,project_name,<firewall rule>
    except NoIdError:
        rule = api.firewall_rule_create(rule_obj)
        rule = api.firewall_rule_read(rule_obj)
    rule.set_action_list(
         ActionListType(host_based_service=True,simple_action="pass"))
    api.firewall_rule_update(rule)
    fwp.add_firewall_rule(rule)
```

8. When the traffic goes through the host-based firewall, the cSRX on the corresponding compute nodes creates the host-based firewall flow and respective sessions.

```
vrouter-agent)[root@nodec61 /]$ flow -l --match 1.xx.xx.251
Flow table(size 161218560, entries 629760)
Listing flows matching ([1.xx.xx.251]:*)
    Index                Source:Port/Destination:Port                      Proto(V)
-----------------------------------------------------------------------------------------
---------------------
   320632<=>327760       1.xx.xx.251:407                                      1 (8->12)
                         2.xx.xx.251:0
(Gen: 4, K(nh):95, Action:F, Flags:, QOS:-1, S(nh):95,  Stats:21/2058,SPort 49640, TTL 0,
HbsLeft, Sinfo 12.0.0.0)

   327760<=>320632       2.xx.xx.251:407                                      1 (8->12)
                         1.xx.xx.251:0
(Gen: 3, K(nh):95, Action:F, Flags:, QOS:-1, S(nh):24,  Stats:20/1680, SPort 61330, TTL 0,
HbsRight, Sinfo 77.xx.xx.21)
```

```
On csrx
========
root@csrx-7vhn6> show security flow session
Session ID: 76342, Policy name: default-policy-logical-system-00/2, Timeout: 4, Valid
In: 2.xx.xx.251/407 --> 1.xx.xx.251/128;icmp, Conn Tag: 0x0, If: ge-0/0/1.0, Pkts: 1, Bytes:
98,
Out: 1.xx.xx.251/128 --> 2.xx.xx.251/407;icmp, Conn Tag: 0x0, If: ge-0/0/0.0, Pkts: 0, Bytes:
0,
Total sessions: 1
```

Release History Table

Release	Description
2003	Contrail Networking Release 2003 provides *beta* support for the host-based firewall feature which enables the creation of next generation firewalls using cSRX devices.

CHAPTER 2

Configuring Virtual Networks

Creating a Virtual Network with Juniper Networks Contrail | 46

Creating a Floating IP Address Pool | 49

Support for IPv6 Networks in Contrail | 51

Configuring EVPN and VXLAN | 55

Support for EVPN Route Type 5 | 65

Support for EVPN Type 6 Selective Multicast Ethernet Tag Route | 66

Support for L3VPN Inter AS Option C | 68

Contrail vRouter Next Hop Configuration | 71

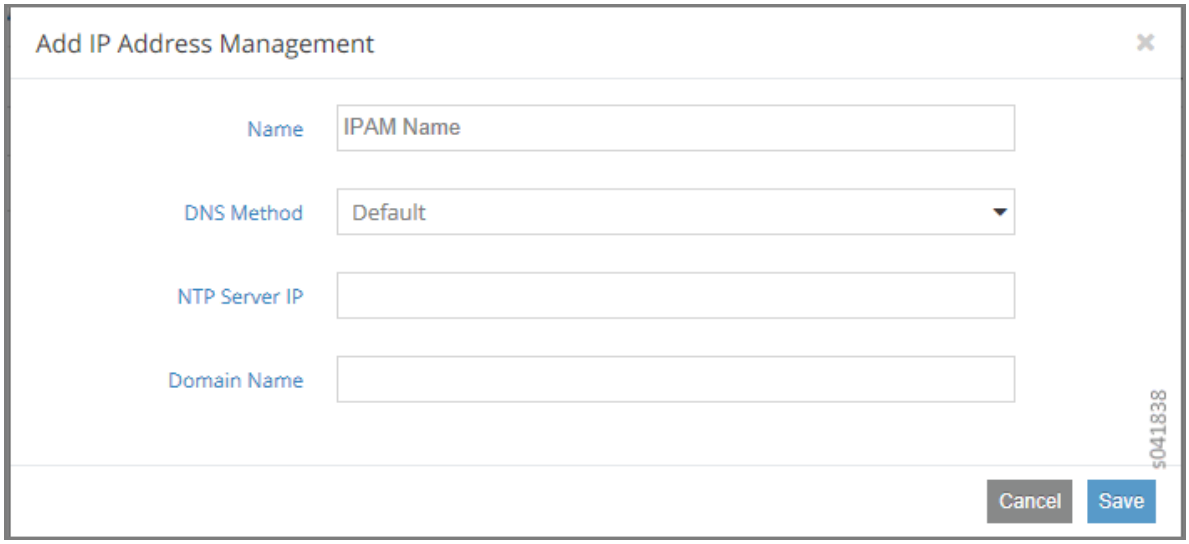
Creating a Virtual Network with Juniper Networks Contrail

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 19 on page 46.

Figure 19: Add IP Address Management



2. Complete the fields in **Add IP Address Management**: The fields are described in Table 2 on page 46.

Table 2: Add IP Address Management Fields

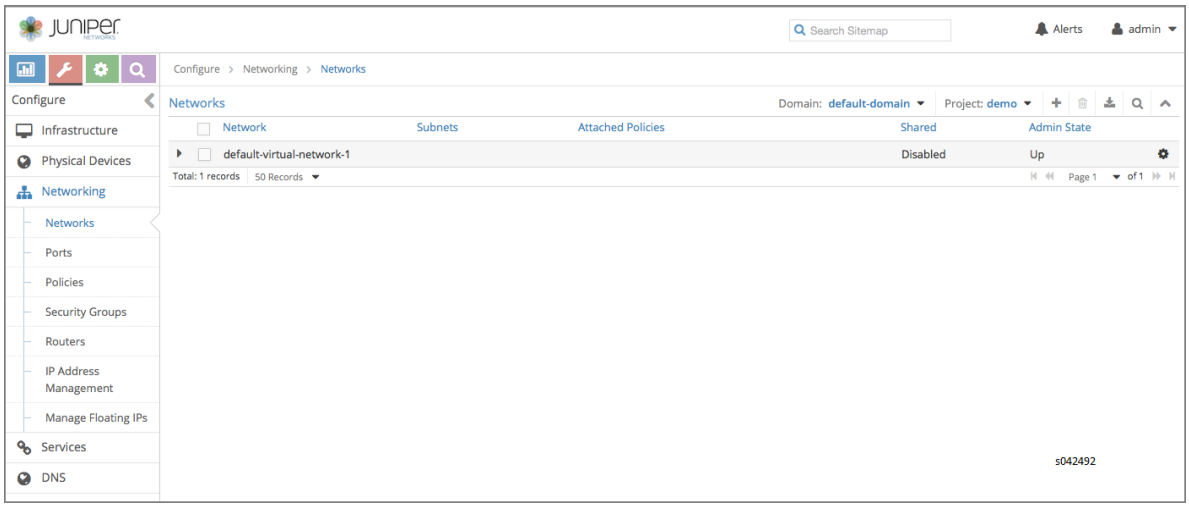
Field	Description
Name	Enter a name for the IPAM you are creating.
DNS Method	Select from a list the domain name server method for this IPAM: **Default, Virtual DNS, Tenant**, or **None**.

Table 2: Add IP Address Management Fields *(Continued)*

Field	Description
NTP Server IP	Enter the IP address of an NTP server to be used for this IPAM.
Domain Name	Enter a domain name to be used for this IPAM.

3. Select **Configure > Networking > Networks** to access the **Configure Networks** page; see Figure 20 on page 47.

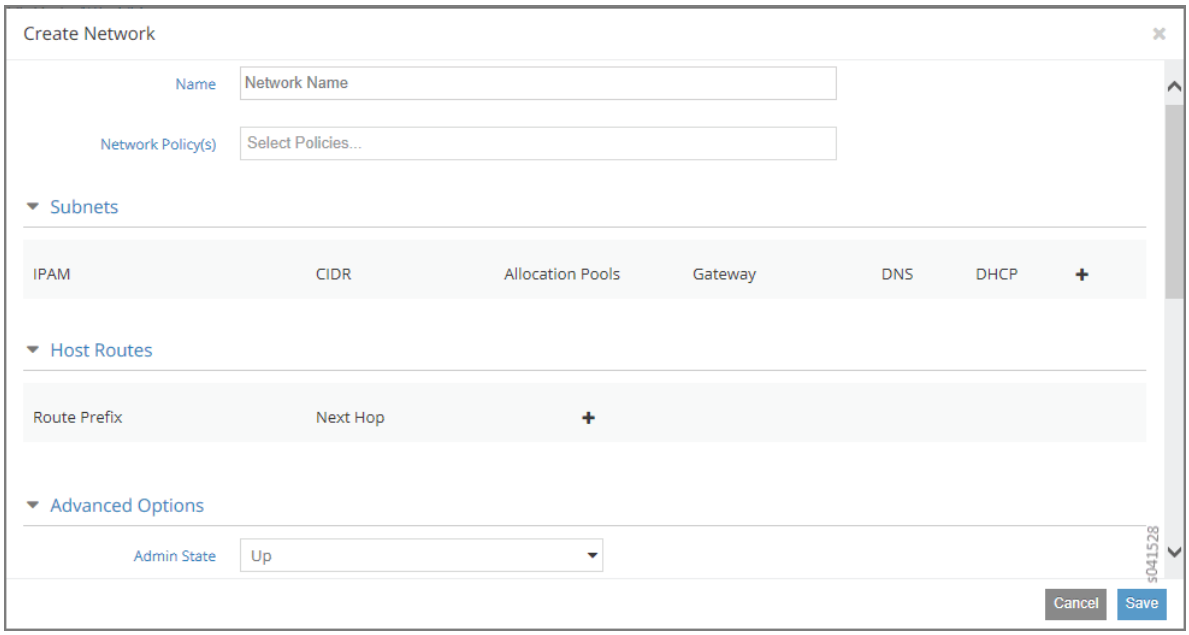
Figure 20: Configure Networks



4. Verify that your project is displayed as active in the upper-right field, then click the

icon. The **Create Network** window is displayed. See Figure 21 on page 48. Use the scroll bar to access all sections of this window.

Figure 21: Create Network



5. Complete the fields in the **Create Network** window with values that identify the network name, network policy, and IP options as needed. See field descriptions in Table 3 on page 48.

Table 3: 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.

Table 3: Create Network Fields *(Continued)*

Field	Description
Advanced Options	Use this area to add or remove advanced options, including identifying the Admin State as Up or Down, to identify the network as Shared or External, to add DNS servers, or to define a VxLAN Identifier.
Floating IP Pools	Use this area to identify and manage the floating IP address pools for this virtual network. Click the + icon to open fields where you can enter the Pool Name and Projects. Click the + icon to add the information, or click the - icon to remove the information.
Route Target	Move the scroll bar down to access this area, then specify one or more route targets for this virtual network. Click the + icon to open fields where you can enter route target identifiers. Click the + icon to add the information, or click the - icon to remove the information.

6. To save your network, click the **Save** button, or click **Cancel** to discard your work and start over.

RELATED DOCUMENTATION

Creating an Image for a Project in OpenStack Contrail

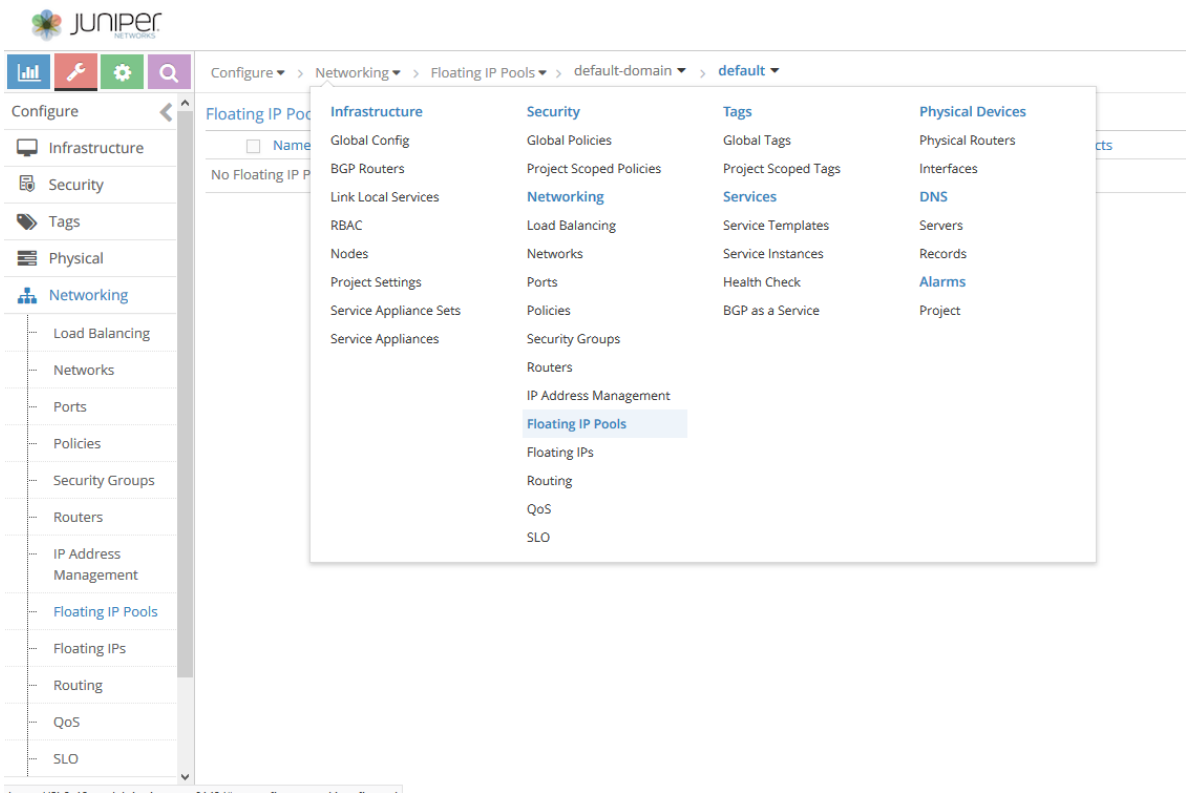
Creating a Floating IP Address Pool

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:

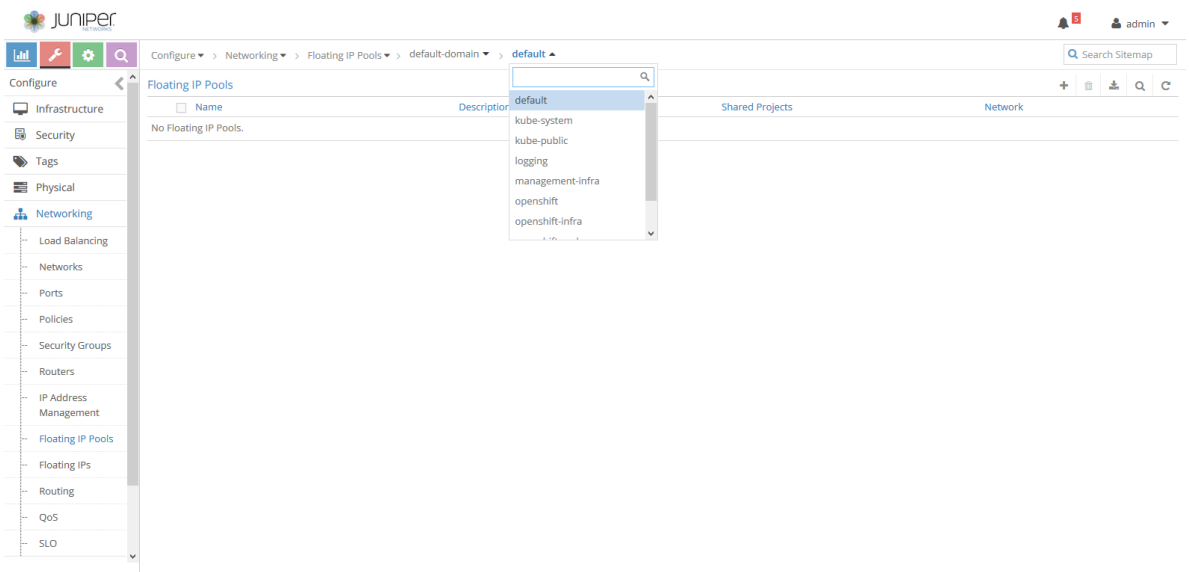
1. Select **Configure > Networking > Floating IP Pools**.

Figure 22: Floating IP Pools Selection



2. Select the network you want to associate with a floating IP pool.

Figure 23: Network Selection

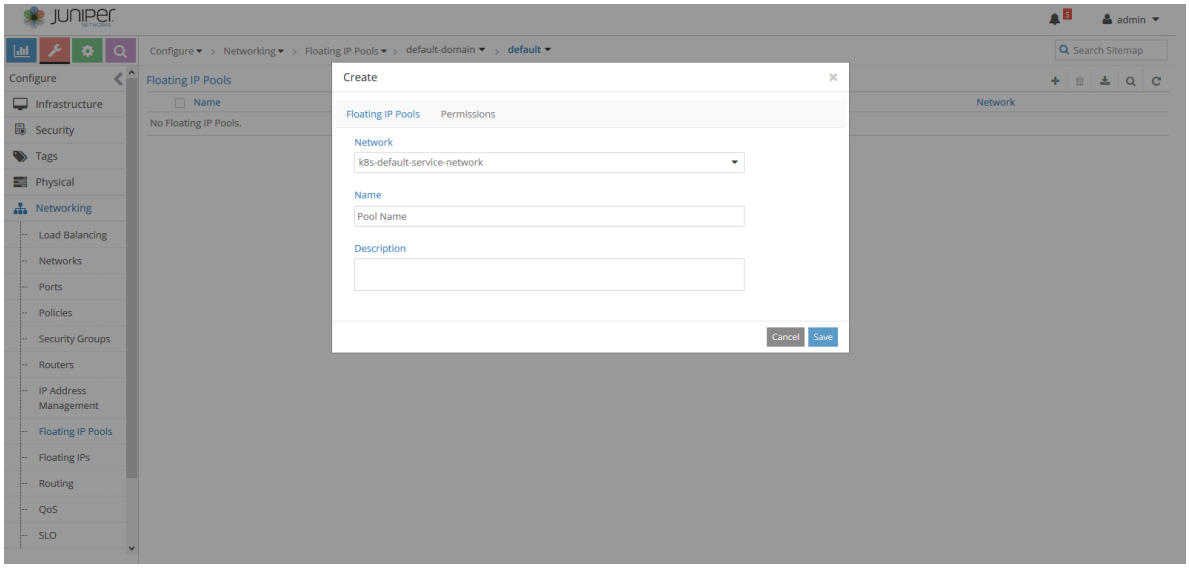


3. Click the add icon (**+**) to create a floating IP pool.

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 24: 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.

Support for IPv6 Networks in Contrail

IN THIS SECTION

- Overview: IPv6 Networks in Contrail | 52
- Creating IPv6 Virtual Networks in Contrail | 53
- Adding IPv6 Peers | 54

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

Overview: IPv6 Networks in Contrail

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

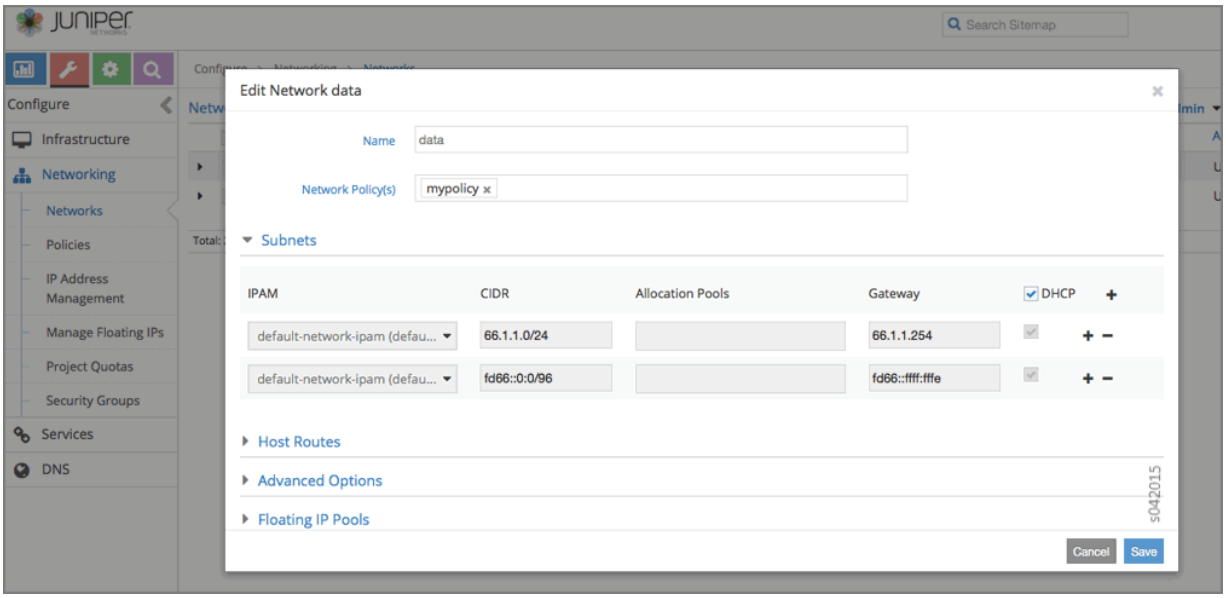
- Virtual machines with IPv6 and IPv4 interfaces
- Virtual machines with IPv6-only interfaces
- DHCPv6 and neighbor discovery
- Policy and Security groups
- IPv6 flow set up, tear down, and aging
- Flow set up and tear down based on TCP state machine
- Protocol-based flow aging
- Fat flow
- Allowed address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

NOT present is support for the following:

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

Creating IPv6 Virtual Networks in Contrail

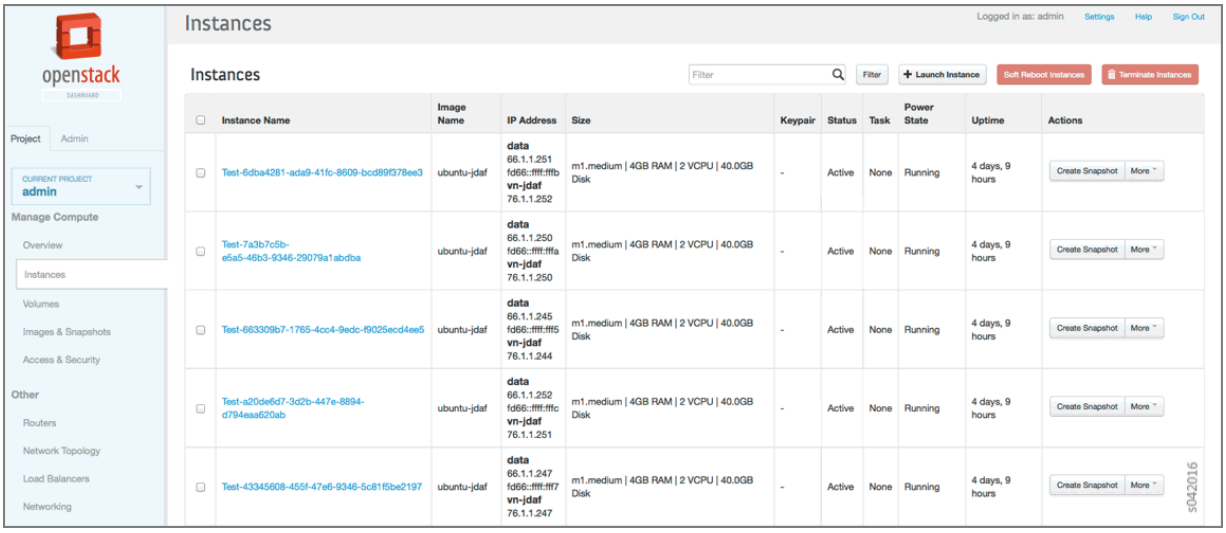
You can create an IPv6 virtual network from the Contrail user interface in the same way you create an IPv4 virtual network. When you create a new virtual network by selecting **Configure > Networking > Networks**, the Edit fields accept IPv6 addresses, as shown in the following image.



Address Assignments

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

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



Enabling DHCPv6 In Virtual Machines

To allow IPv6 address assignment using DHCPv6, the virtual machine network interface configuration must be updated appropriately.

For example, to enable DHCPv6 for Ubuntu-based virtual machines, add the following line in the **/etc/network/interfaces** file:

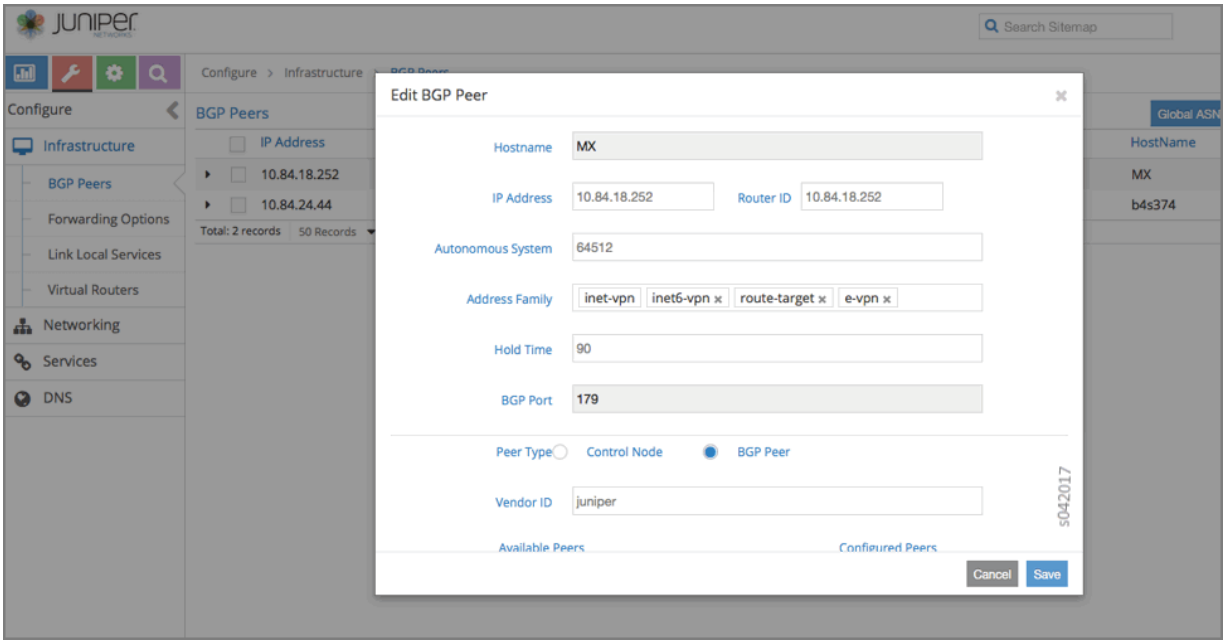
```
iface eht0 inet6 dhcp
```

Also, `dhclient -6` can be run from within the virtual machine to get IPv6 addresses using DHCPv6.

Adding IPv6 Peers

The procedure to add an IPv6 BGP peer in Contrail is similar to adding an IPv4 peer. Select **Configure > Infrastructure > BGP Peers**, include `inet6-vpn` in the Address Family list to allow advertisement of IPv6 addresses.

A sample is shown in the following.



NOTE: Additional configuration is required on the peer router to allow inet6-vpn peering.

Configuring EVPN and VXLAN

IN THIS SECTION

- Configuring the VXLAN Identifier Mode | **57**
- Configuring Forwarding | **60**
- Configuring the VXLAN Identifier | **61**
- Configuring Encapsulation Methods | **62**

Contrail supports Ethernet VPNs (EVPN) and Virtual Extensible Local Area Networks (VXLAN).

EVPN is a flexible solution that uses Layer 2 overlays to interconnect multiple edges (virtual machines) within a data center. Traditionally, the data center is built as a flat Layer 2 network with issues such as flooding, limitations in redundancy and provisioning, and high volumes of MAC address learning, which

cause churn during node failures. EVPNs are designed to address these issues without disturbing flat MAC connectivity.

In EVPNs, MAC address learning is driven by the control plane, rather than by the data plane, which helps control learned MAC addresses across virtual forwarders, thus avoiding flooding. The forwarders advertise locally learned MAC addresses to the controllers. The controllers use MP-BGP to communicate with peers. The peering of controllers using BGP for EVPN results in better and faster convergence.

With EVPN, MAC learning is confined to the virtual networks to which the virtual machine belongs, thus isolating traffic between multiple virtual networks. In this manner, virtual networks can share the same MAC addresses without any traffic crossover.

Unicast in EVPNs

Unicast forwarding is based on MAC addresses where traffic can terminate on a local endpoint or is encapsulated to reach the remote endpoint. Encapsulation can be MPLS/UDP, MPLS/GRE, or VXLAN.

BUM Traffic in EVPN

Multicast and broadcast traffic is flooded in a virtual network. The replication tree is built by the control plane, based on the advertisements of end nodes (virtual machines) sent by forwarders. Each virtual network has one distribution tree, a method that avoids maintaining multicast states at fabric nodes, so the nodes are unaffected by multicast. The replication happens at the edge forwarders. Per-group subscription is not provided. Broadcast, unknown unicast, and multicast (BUM) traffic is handled the same way, and gets flooded in the virtual network to which the virtual machine belongs.

VXLAN

VXLAN is an overlay technology that encapsulates MAC frames into a UDP header at Layer 2. Communication is established between two virtual tunnel endpoints (VTEPs). VTEPs encapsulate the virtual machine traffic into a VXLAN header, as well as strip off the encapsulation. Virtual machines can only communicate with each other when they belong to the same VXLAN segment. A 24-bit virtual network identifier (VNID) uniquely identifies the VXLAN segment. This enables having the same MAC frames across multiple VXLAN segments without traffic crossover. Multicast in VXLAN is implemented as Layer 3 multicast, in which endpoints subscribe to groups.

Design Details of EVPN and VXLAN

In Contrail Release 1.03 and later, EVPN is enabled by default. The supported forwarding modes include:

- Fallback bridging—IPv4 traffic lookup is performed using the IP FIB. All non-IPv4 traffic is directed to a MAC FIB.
- Layer 2-only— All traffic is forwarded using a MAC FIB lookup.

You can configure the forwarding mode individually on each virtual network.

EVPN is used to share MAC addresses across different control planes in both forwarding models. The result of a MAC address lookup is a next hop, which, similar to IP forwarding, points to a local virtual machine or a tunnel to reach the virtual machine on a remote server. The tunnel encapsulation methods supported for EVPN are MPLSoGRE, MPLSoUDP, and VXLAN. The encapsulation method selected is based on a user-configured priority.

In VXLAN, the VNID is assigned uniquely for every virtual network carried in the VXLAN header. The VNID uniquely identifies a virtual network. When the VXLAN header is received from the fabric at a remote server, the VNID lookup provides the VRF of the virtual machine. This VRF is used for the MAC lookup from the inner header, which then provides the destination virtual machine.

Non-IP multicast traffic uses the same multicast tree as for IP multicast (255.255.255.255). The multicast is matched against the all-broadcast prefix in the bridging table (FF:FF:FF:FF:FF:FF). VXLAN is not supported for IP/non-IP multicast traffic.

The following table summarizes the traffic and encapsulation types supported for EVPN.

		Encapsulation		
		MPLS-GRE	**MPLS-UDP**	**VXLAN**
Traffic Type	**IP unicast**	Yes	Yes	No
	IP-BUM	Yes	Yes	No
	non IP unicast	Yes	Yes	Yes
	non IP-BUM	Yes	Yes	No

Configuring the VXLAN Identifier Mode

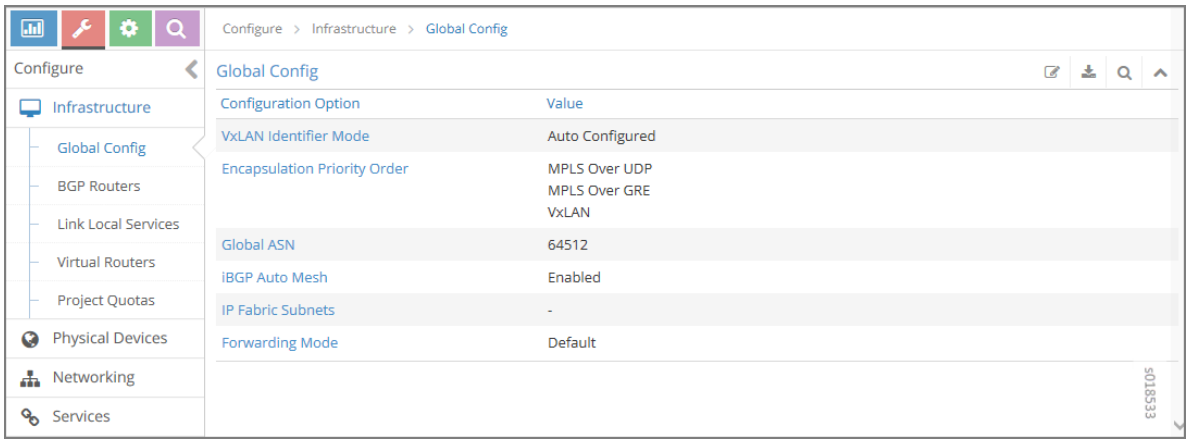
You can configure the global VXLAN identifier mode to select an auto-generated VNID or a user-generated VXLAN ID, either through the Contrail Web UI or by modifying a python file.

To configure the global VXLAN identifier mode:

1. From the Contrail Web UI, select **Configure > Infrastucture > Global Config**.

   The Global Config options and values are displayed in the Global Config window.

Figure 25: Global Config Window for VXLAN ID

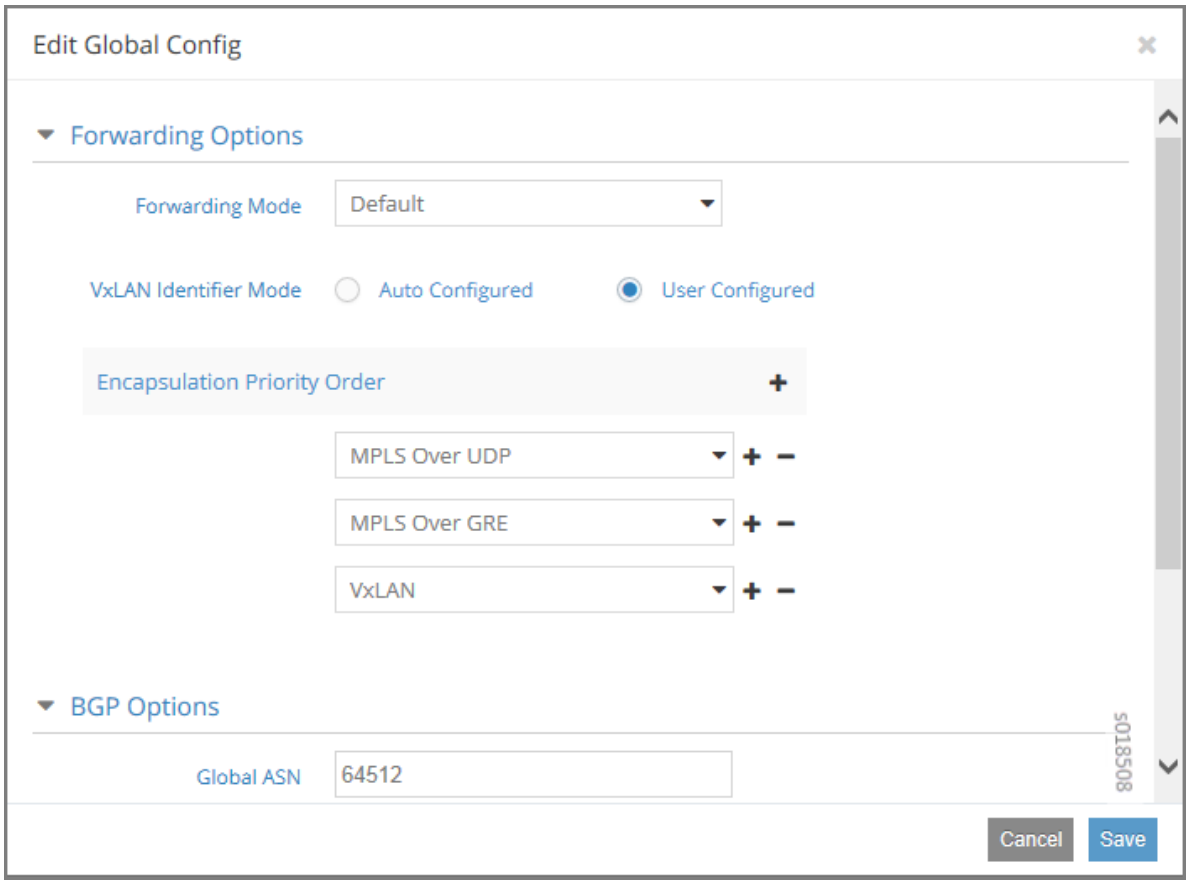


2. Click the edit icon

.

The Edit Global Config window is displayed as shown in Figure 26 on page 59.

Figure 26: 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:

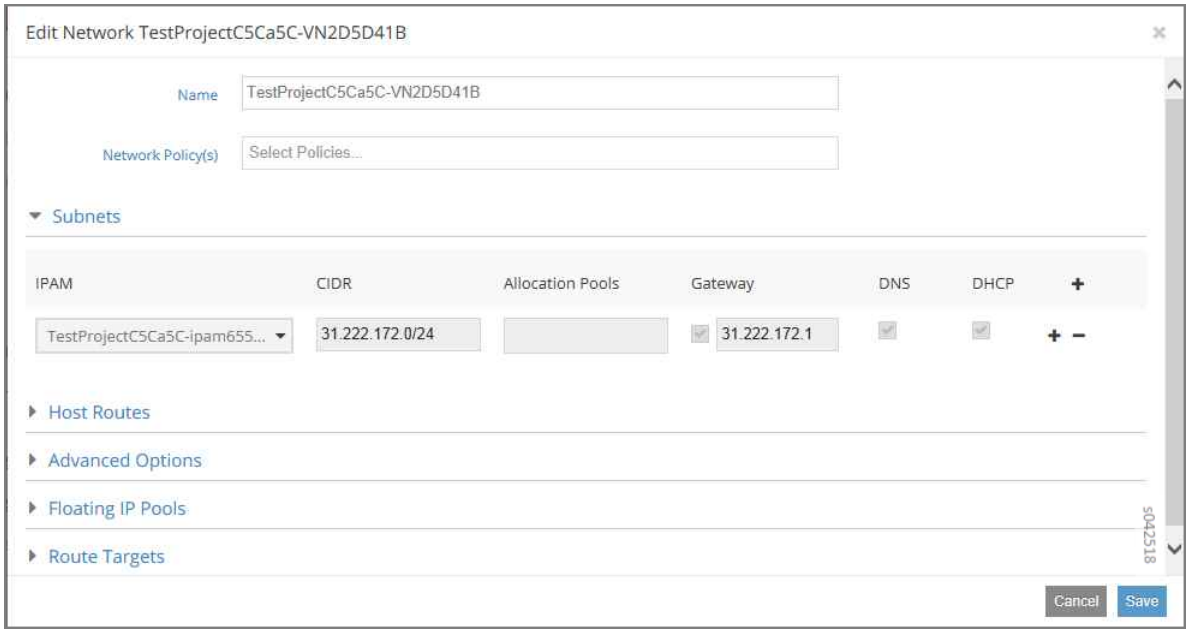
1. From the Contrail Web UI, select **Configure > Networking > Networks**.
2. Select the virtual network that you want to change the forwarding mode for.
3. Click the gear icon

   and select **Edit**.

   The Edit Network window is displayed as shown in Figure 27 on page 60.

Figure 27: Edit Network Window



Under the Advanced Options select the forwarding mode from the following choices:

- Select **Default** to enable the default forwarding mode.
- Select **L2 and L3** to enable IP and MAC FIB (fallback bridging).
- Select **L2 Only** to enable only MAC FIB.
- Select **L3 Only** to enable only IP.

NOTE: The full list of forwarding modes are only displayed if you change entries in the **/usr/src/contrail/contrail-web-core/config/config.global.js** file. For example:

1. To make the **L2** selection available locate the following:

```
config.network = {};
config.network.L2_enable = false;
```

2. Change the entry to the following:

```
config.network = {};
config.network.L2_enable = true;
```

3. To make the other selections available, modify the corresponding entries.
4. Save the file and quit the editor.
5. Restart the Contrail Web user interface process (webui).

Alternatively, you can use the following python provisioning command to change the forwarding mode:

```
python provisioning_forwarding_mode --project_fq_name 'defaultdomain: admin' --vn_name vn1 --forwarding_mode < l2_l3| l2 >
```

Options:

l2_l3 = Enable IP FIB and MAC FIB (fallback bridging)

l2 = Enable MAC FIB only (Layer 2 only)

Configuring the VXLAN Identifier

The VXLAN identifier can be set only if the VXLAN network identifier mode has been set to User Configured. You can then set the VXLAN ID by either using the Contrail Web UI or by using Python commands.

To configure the global VXLAN identifier:

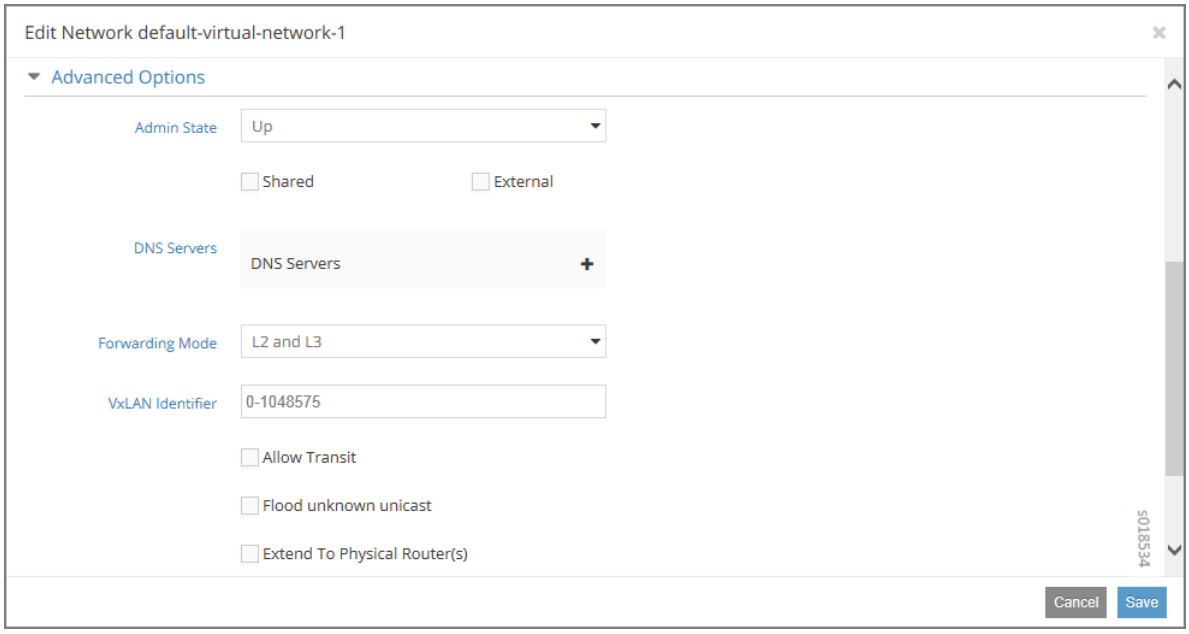
1. From the Contrail Web UI, select **Configure > Networking > Networks**.

2. Select the virtual network that you want to change the forwarding mode for.
3. Click the gear icon

   and select **Edit**.

   The Edit Network window is displayed. Select the **Advanced Options** as shown in Figure 28 on page 62.

Figure 28: Edit Network Window for VXLAN Identifier



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

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

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

Configuring Encapsulation Methods

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

VXLAN. MPLS over UDP is different from MPLS over GRE only in the method of tunnel header encapsulation.

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

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

To configure the global VXLAN identifier mode:

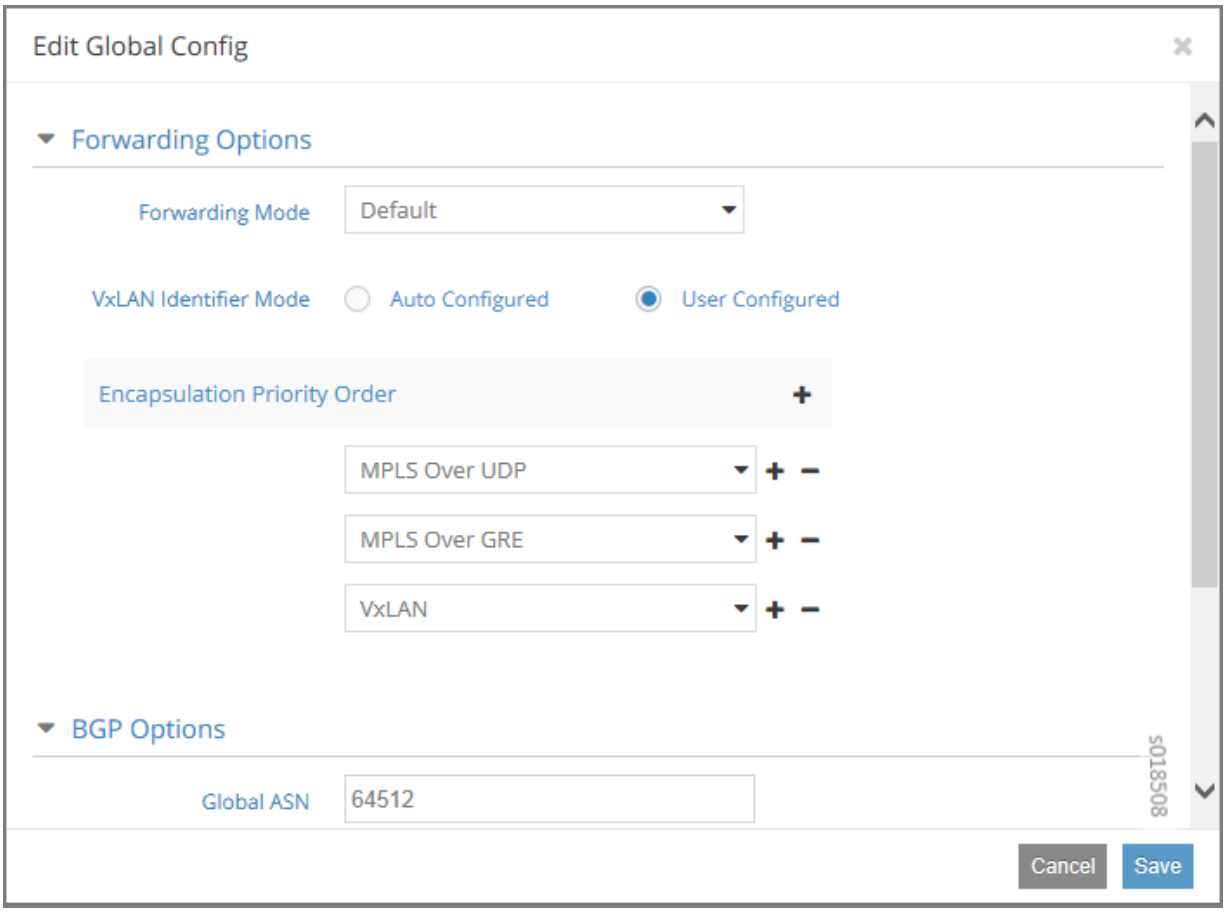
- From the Contrail Web UI, select **Configure > Infrastucture > Global Config**.
- The Global Config options are displayed.
- Click the edit icon

  .

The Edit Global Config window is displayed as shown in Figure 29 on page 64.

Figure 29: Edit Global Config Window for Encapsulation Priority Order



Under Encapsulation Priority Order select one of the following:

- **MPLS over UDP**
- **MPLS over GRE**
- **VxLAN**

Click the + plus symbol to the right of the first priority to add a second priority or third priority.

Use the following procedure to change the default encapsulation method to VXLAN by editing the **encap.py** file.

NOTE: VXLAN is *only* supported for EVPN unicast. It is not supported for IP traffic or multicast traffic. VXLAN priority and presence in the **encap.py** file or configured in the Web UI is ignored for traffic not supported by VXLAN.

To set the priority of encapsulation methods to VXLAN:

1. Modify the **encap.py** file found in the **/opt/contrail/utils/** directory.

   The default encapsulation line is:

   ```
   encap_obj=EncapsulationPrioritiesType(encapsulation=['MPLSoUDP','M PLSoGRE'])
   ```

   Modify the line to:

   ```
   encap_obj=EncapsulationPrioritiesType(encapsulation=['VXLAN', 'MPLSoUDP','MPLSoGRE'])
   ```

2. After the status is modified, execute the following script:

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

   The configuration is applied globally for all virtual networks.

Support for EVPN Route Type 5

Contrail Release 5.0.1 and later supports EVPN Route Type 5 messages as defined in the IETF specification *IP Prefix Advertisement in EVPN*. EVPN Route Type 5 is an extension of EVPN Route Type 2, which carries MAC addresses along with their associated IP addresses. EVPN Route Type 5 facilitates in inter-subnet routing.

Type 5 network layer reachability information (NLRI) contains information in the following format:

```
+---------------------------------------+
|      RD   (8 octets)                  |
+---------------------------------------+
|Ethernet Segment Identifier (10 octets)|
+---------------------------------------+
|  Ethernet Tag ID (4 octets)           |
+---------------------------------------+
|  IP Prefix Length (1 octet)           |
+---------------------------------------+
|  IP Prefix (4 or 16 octets)           |
+---------------------------------------+
|  GW IP Address (4 or 16 octets)       |
+---------------------------------------+
|  MPLS Label (3 octets)                |
+---------------------------------------+
```

When Type-5 EVPN prefix is received from a BGP peer, it is first installed into bgp.evpn.0 like all other routes. From here, based on matching route targets, the route gets replicated into all *.evpn.0 tables as

applicable. From there, the routes are advertised over Extensible messaging and presence protocol (XMPP) to all interested agents.

NOTE: In Release 5.0.1, policy based route-leaking among different L3VRFs is not supported. Hence, service chaining for Type 5 L3VRFs is also not supported.

Support for EVPN Type 6 Selective Multicast Ethernet Tag Route

IN THIS SECTION

- Configuring EVPN Type-6 SMET Routes | **67**

Contrail Release 5.1 and later supports EVPN Type 6 selective multicast Ethernet tag (SMET) route to selectively send or receive traffic based on the presence or absence of active receivers on a compute node. The EVPN Type-6 SMET route helps build and use multicast trees selectively on a per <*, G> basis.

The length of the route update is NLRI (Network Layer Reachability Info) size, which is calculated as follows:

NLRI = Min.size (23 bytes) + 2 * IP_Address_Size + 1 (router_type) where,

The minimum size for the SelectiveMulticastRoute (TYPE-6) = RD (16 bytes) + Ethernet_Tag (4 bytes) + 3 = 23 bytes.

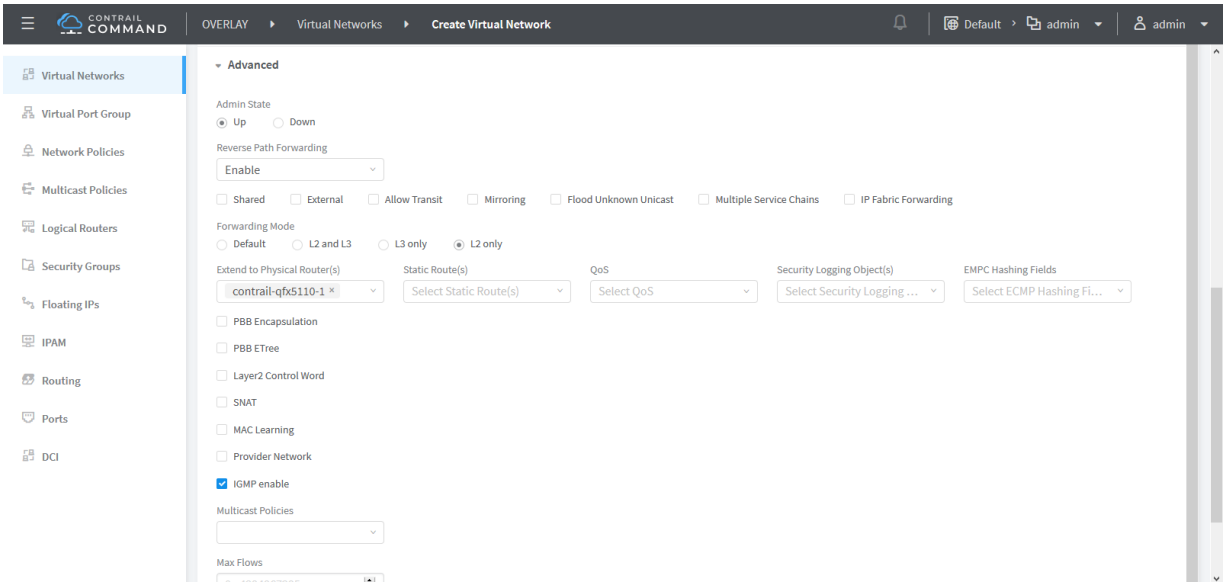
Based on number of prefixes carried, the NLRI length varies. The length of the update when transported through BGP is (NLRI Size * 8).

Currently, all broadcast, unknown unicast, multicast (BUM) traffic is carried over the inclusive multicast ethernet tag (IMET) routes. This results in flooding of all compute nodes irrespective of whether an active receiver is present or not on each of those compute-nodes.

Configuring EVPN Type-6 SMET Routes

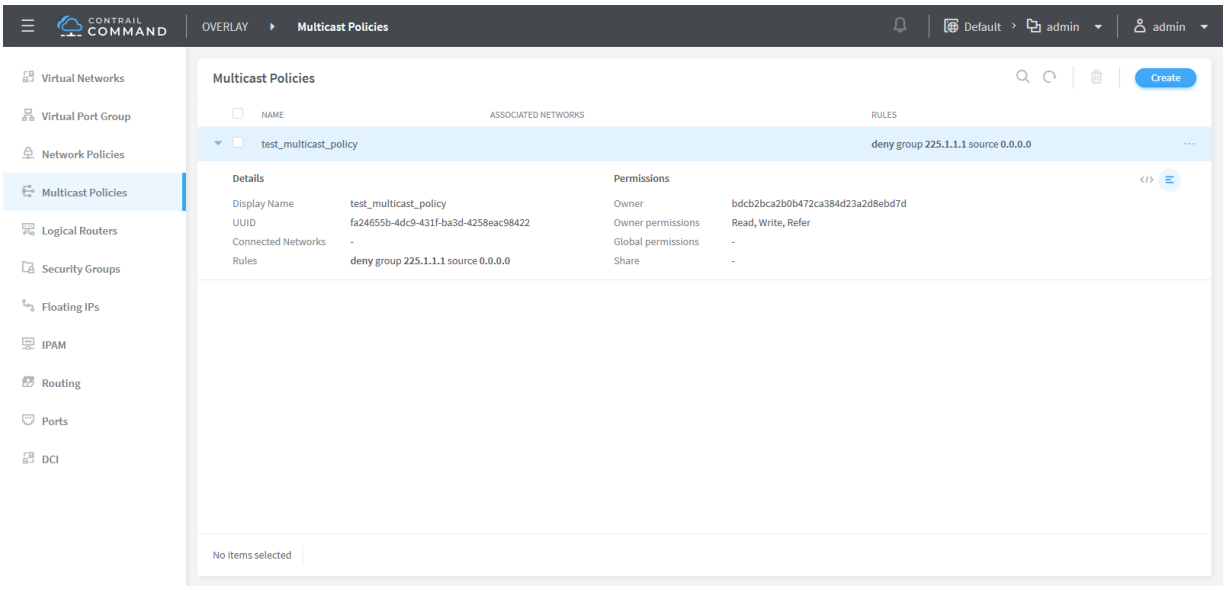
EVPN Type-6 SMET routes capability attaches a specific BGP community attribute `Ethernet Multicast flags Community` (`MF`) to the IMET routes. This community is advertised by default in Contrail Release 5.1 and later. You must enable IGMP on the network as shown in Figure 30 on page 67 as well as on the QFX device to which the multicast source is connected. You can configure IGMP at the global system configuration-level, at virtual network-level, or at VMI-level. Configuring `ERB-UCAST-Gateway` role enables IGMP snooping on the QFX device.

Figure 30: Configure IGMP



You can allow or deny multicast traffic by attaching a policy at the virtual network-level as shown in Figure 31 on page 68.

Figure 31: Define Multicast Policy



In Contrail Release 5.1 and later, the receivers are always inside the contrail cluster and sender is always outside the cluster. This feature is supported only with <*, G> /igmpv2. The SMET feature is supported only on QFX10000 and QFX5110 devices running Junos OS Release 18.4R1 and later.

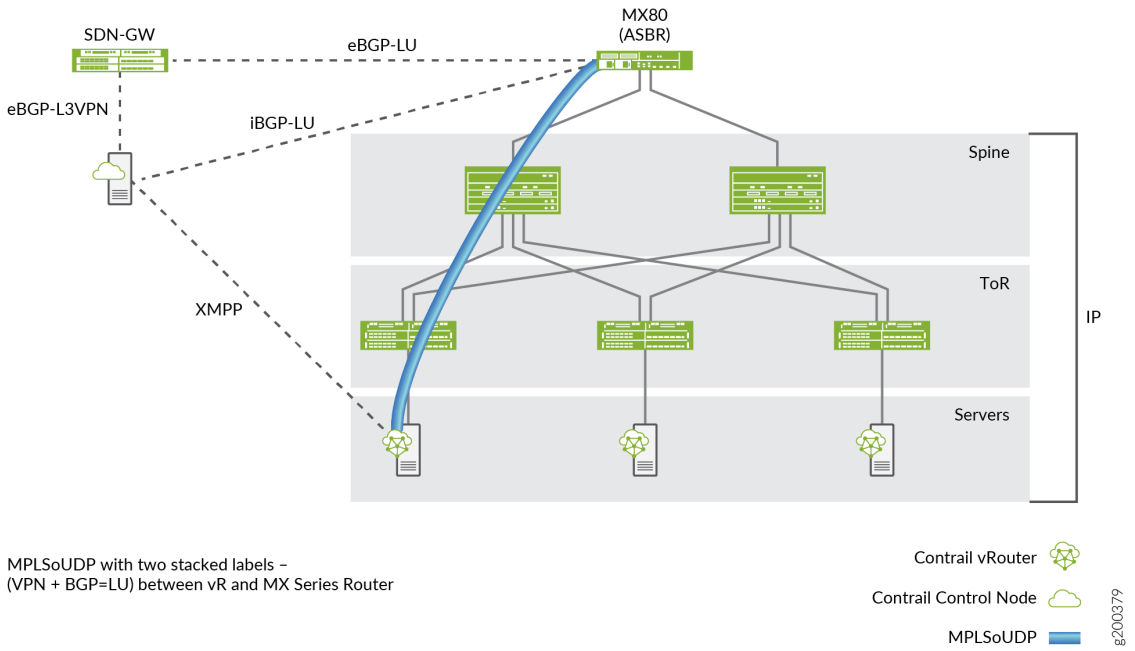
Support for L3VPN Inter AS Option C

IN THIS SECTION

- Configuring Inter AS Option C | **69**

Contrail Release 5.1 and later supports L3VPN inter AS Option C, which is used to interconnect multi-AS backbones as described in RFC 4364. Inter-AS option C uses BGP as the label distribution protocol. Labeled IPv4 routes are redistributed by eBGP between the neighboring autonomous systems. A new address family inet-labeled is added to maintain labeled unicast routes. This table is used for resolving L3VPN routes which are encapsulated using MPLS. Figure 32 on page 69 shows the connectivity and the roles of different components in this architecture.

Figure 32: L3VPN Inter AS Option C Architecture



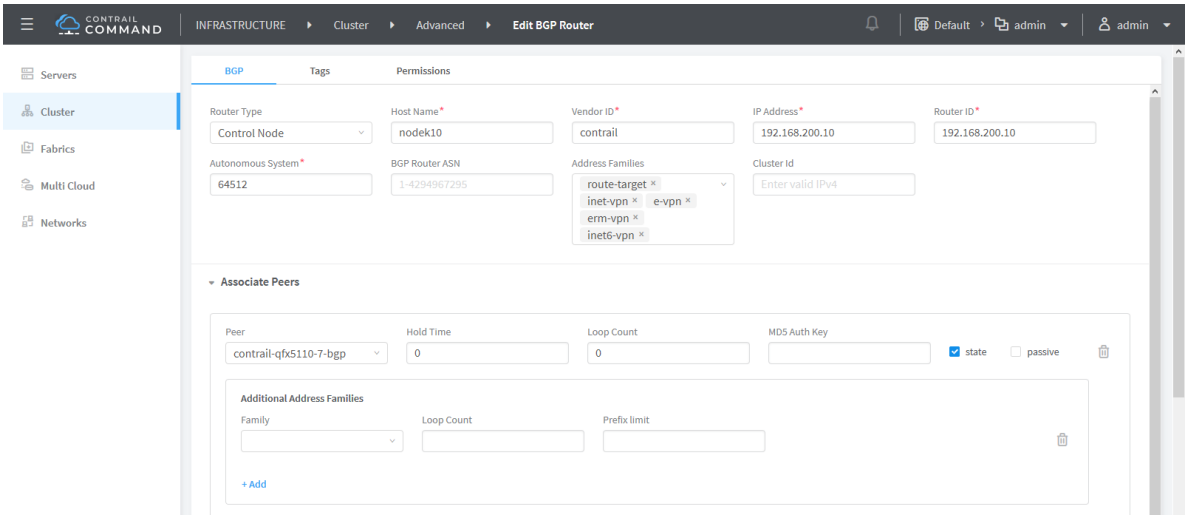
The controller maintains an eBGP session with the SDN-GW router and an iBGP session with the ASBR router. The controller exchanges labeled routes with the vRouters over XMPP. The vRouter uses MPLSoUDP or MPLSoGRE to reach the ASBR and encapsulates two labels within it - the inner VPN label and outer BGP-LU label. For the opposite direction, the vRouter advertises a labeled unicast route for its vhost address with a label 3 (implicit null), so the traffic from the ASBR delivered through the fabric to the vRouter over a UDP/GRE tunnel contains only the VPN label.

Configuring Inter AS Option C

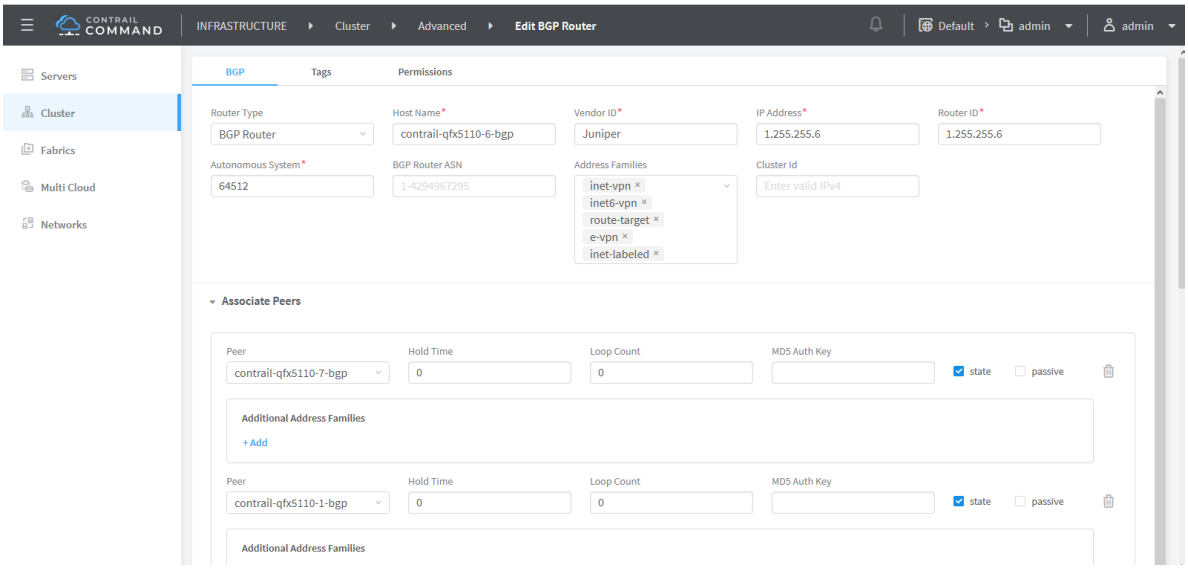
This section describes how to configure L3VPN Inter AS Option C from Contrail Command UI.

1. Navigate to Infrastructure > Cluster > Advanced.
2. Click **BGP Routers** tab and click the **Edit** icon against the node that you want to configure.

   The Edit BGP Router page is displayed.
3. Configure eBGP session for the SDN gateway. Make sure you include `inet-labeled` to the **Address Family** field.



4. Configure the iBGP session with the ASBR router. Address family must be set to `inet-labeled`.

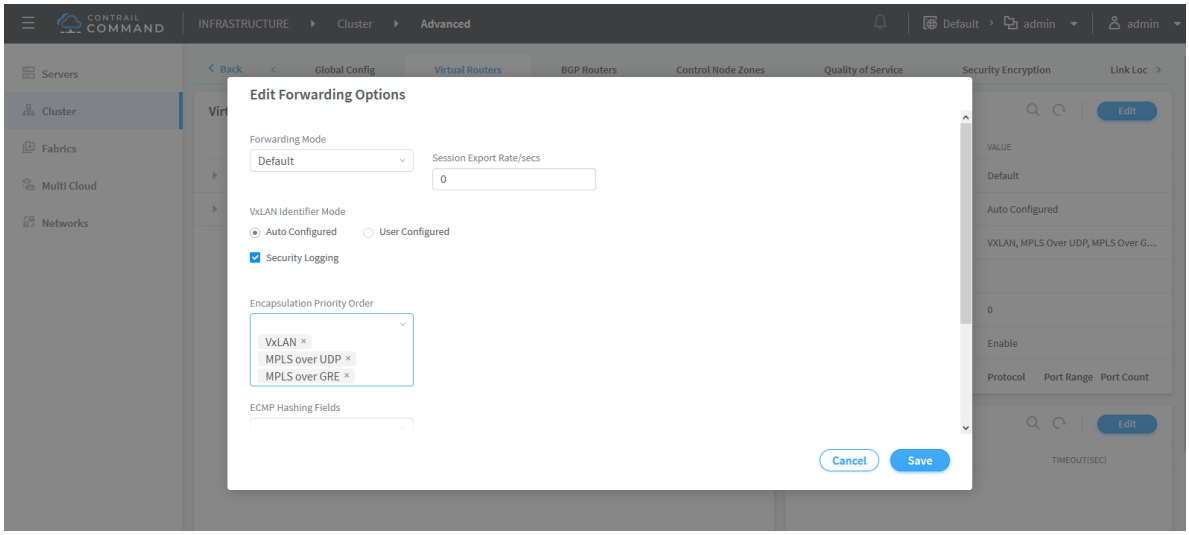


5. Navigate to **Infrastructure > Cluster > Advanced**.

6. Click the **Virtual Routers** tab and on the Forwarding Options pane, click Edit to modify the encapsulation settings.

   Contrail supports three encapsulation types. They are VXLAN, MPLS Over UDP, and MPLS Over GRE.

7. Set the Encapsulation priority order.



8. Click **Save**.

   Changes made to Forwarding Options settings are saved.

Contrail vRouter Next Hop Configuration

IN THIS SECTION

- Benefits of Increasing Next Hop Limit | 73

In Contrail Networking Release 1911, the next hop value in the vRouter is increased to 32 bits. By default, the vRouter can now create 512K next hops and supports up to 1 million next hops. Also, in release 1911 you can assign a high watermark limit in vRouter agent configuration file. If the number of next hops or Multiprotocol Label Switching (MPLS) labels exceed the watermark limit, the vRouter agent generates alarms. These alarms are generated based on the usage of next hops and MPLS labels against the watermark limit and maximum limit of next hops and MPLS labels.

In releases prior to release 1911, Contrail supported 16 bits next hop value in the vRouter. As the next hop value was assigned a 16 bit value, the vRouter could create a maximum of 65,536 next hops. The vRouter agent did not generate alarms when the number of next hops increased. When the number of next hops exceeded the maximum limit, the vRouter agent failed to perform another next hop, which led to loss of traffic.

In the vRouter agent configuration file, `contrail-vrouter-agent.conf`, you can configure a high watermark limit according to your requirement. The watermark limit specifies the maximum percentage of next hops or MPLS labels that you can use. The vRouter agent generates alarms when the next hop usage or the MPLS labels usage exceeds the watermark limit. For example, the default watermark limit is set to 80 (80% of the maximum next hops or MPLS labels vRouter can create). If the maximum number of next hops possible on the compute node is 100, an alarm is raised after 80 next hops are created. If the maximum number of MPLS labels that can be created on the compute node is 50, the alarm is raised after 40 MPLS labels are created.

NOTE: The low watermark limit is calculated to be 95% of the high watermark limit.

To configure vRouter object watermark limit in a cluster at the time of provisioning, you must assign a value to `VROUTER_AGENT__DEFAULT__vr_object_high_watermark` parameter either in the `roles: vrouter:` section or in the `contrail_configuration` section of the **instances.yml** file. You must assign a watermark limit in the range of 50–95 to the `VROUTER_AGENT__DEFAULT__vr_object_high_watermark` parameter.

For example, to configure watermark limit to 60%, you must assign a value 60 to the `VROUTER_AGENT__DEFAULT__vr_object_high_watermark` parameter under the following sections:

```
roles:
       vrouter:
          VROUTER_AGENT__DEFAULT__vr_object_high_watermark: 60
```

```
contrail_configuration:
            VROUTER_AGENT__DEFAULT__vr_object_high_watermark: 60
```

NOTE: If you assign a value to `VROUTER_AGENT__DEFAULT__vr_object_high_watermark` in the `contrail_configuration` section, the watermark limit for all vRouters that are configured using **instances.yml** file will be the same. To assign a different watermark limit to a vRouter, you have to assign the watermark limit to the `VROUTER_AGENT__DEFAULT__vr_object_high_watermark` parameter under the `roles: vrouter:` section of a vRouter.

To change the watermark limit later, you must modify the `vr_object_high_watermark` parameter present in the `[DEFAULT]` section of the `entrypoint.sh` file. After you assign a watermark value to the `vr_object_high_watermark` parameter in the `entrypoint.sh` file, the `contrail-vrouter-agent.conf` configuration file is now updated with the `vr_object_high_watermark` parameter, which denotes the watermark limit.

For example, to configure watermark limit to 75%, you must assign a value 75 to the `vr_object_high_watermark` parameter under the `[DEFAULT]` section:

```
[DEFAULT]
     vr_object_high_watermark
```

Based on the next hops or MPLS labels usage, the vRouter agent generates system defined alarms with various severity. See Table 4 on page 73.

Table 4: Alarms Generated by vRouter Agent

Next Hop and MPLS Label Usage Against the Watermark Limit and Maximum Limit	Severity Level of Alarm
Next hop or MPLS labels usage exceeds the high watermark limit	Major alarm is generated.
Next hop or MPLS labels usage equals 100% of the maximum limit	Critical alarm is generated, and high watermark alarm is also present.
Next hop or MPLS labels usage reduces to 95% of the maximum limit	Critical alarm is cleared, and high watermark alarm is present.
Next hop or MPLS labels usage reduces to 95% of the high watermark limit	High watermark alarm is cleared.

Benefits of Increasing Next Hop Limit

- Increase in next hop limit allows Contrail to scale more next hops than in earlier releases.
- The alarms generated by vRouter agent enables you to monitor the usage and availability of next hops and MPLS labels.

Release History Table

Release	Description
1911	In Contrail Networking Release 1911, the next hop value in the vRouter is increased to 32 bits. By default, the vRouter can now create 512K next hops and supports up to 1 million next hops. Also, in release 1911 you can assign a high watermark limit in vRouter agent configuration file. If the number of next hops or Multiprotocol Label Switching (MPLS) labels exceed the watermark limit, the vRouter agent generates alarms.

CHAPTER 3

Deploying a Multi-Tier Web Application Using Contrail Networking

Example: Deploying a Multi-Tier Web Application | 76

Sample Network Configuration for Devices for Simple Tiered Web Application | 84

Example: Deploying a Multi-Tier Web Application

IN THIS SECTION

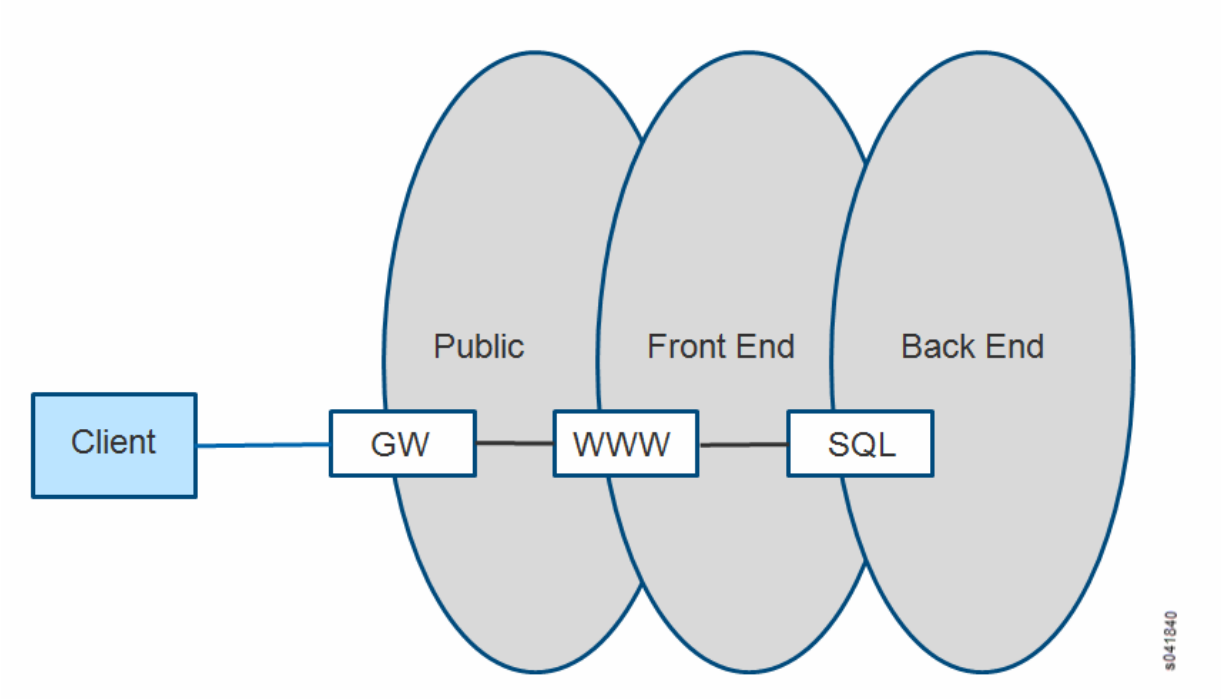
- Multi-Tier Web Application Overview | 76
- Example: Setting Up Virtual Networks for a Simple Tiered Web Application | 77
- Verifying the Multi-Tier Web Application | 80
- Sample Addressing Scheme for Simple Tiered Web Application | 80
- Sample Physical Topology for Simple Tiered Web Application | 82
- Sample Physical Topology Addressing | 82

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 33 on page 77.

Figure 33: 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*.

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.

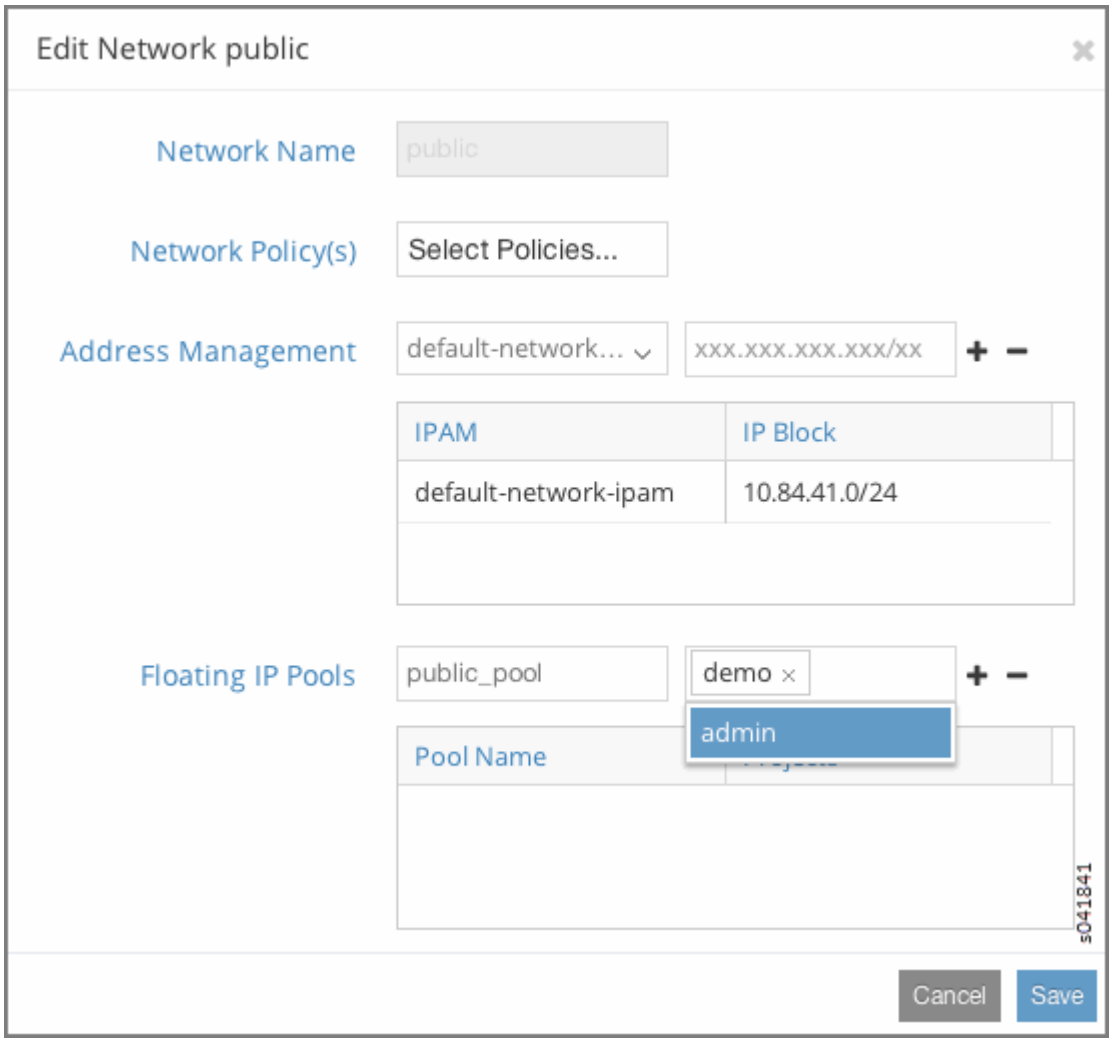
c. A network named **backend** with IP address **192.168.2.0/24**

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* or "Creating a Virtual Network with Juniper Networks Contrail" on page 46.

3. Create a floating IP pool named **public_pool** for the **public** network within the **demo** project; see Figure 34 on page 79.

Figure 34: Create Floating IP Pool



4. Allocate the floating IP pool **public_pool** to the **demo** project; see Figure 35 on page 79.

Figure 35: Allocate Floating IP

5. Verify that the floating IP pool has been allocated; see **Configure > Networking > Allocate Floating IPs.**
6. Create a policy that allows any host to talk to any host using any IP address, protocol, and port, and apply this policy between the **frontend** network and the **backend** network.

   This now allows communication between the web servers in the front-end network and the database servers in the back-end network.
7. Launch the virtual machine instances that represent the web server and the database server.

   > **NOTE**: Your installation might not include the virtual machines needed for the web server and the database server. Contact your account team if you need to download the VMs for this setup.

   On the **Instances** tab for this project, select **Launch Instance** and for each instance that you launch, complete the fields to make the following associations:

   - Web server VM: select **frontend** network and the policy created to allow communication between **frontend** and **backend** networks. Apply the floating IP address pool to the web server.
   - Database server VM: select **backend** network and the policy created to allow communication between **frontend** and **backend** networks.

Verifying the Multi-Tier Web Application

Verify your web setup.

- To demonstrate this web application setup, go to the client machine, open a browser, and navigate to the address in the **public** network that is assigned to the web server in the **frontend** network.

  The result will display the Contrail interface with various data populated, verifying that the web server is communicating with the database server in the **backend** network and retrieving data.

  The client machine only has access to the public IP address. Attempts to browse to any of the addresses assigned to the **frontend** network or to the **backend** network should fail.

Sample Addressing Scheme for Simple Tiered Web Application

Use the information in Table 5 on page 81 as a guide for addressing devices in the simple tiered web example.

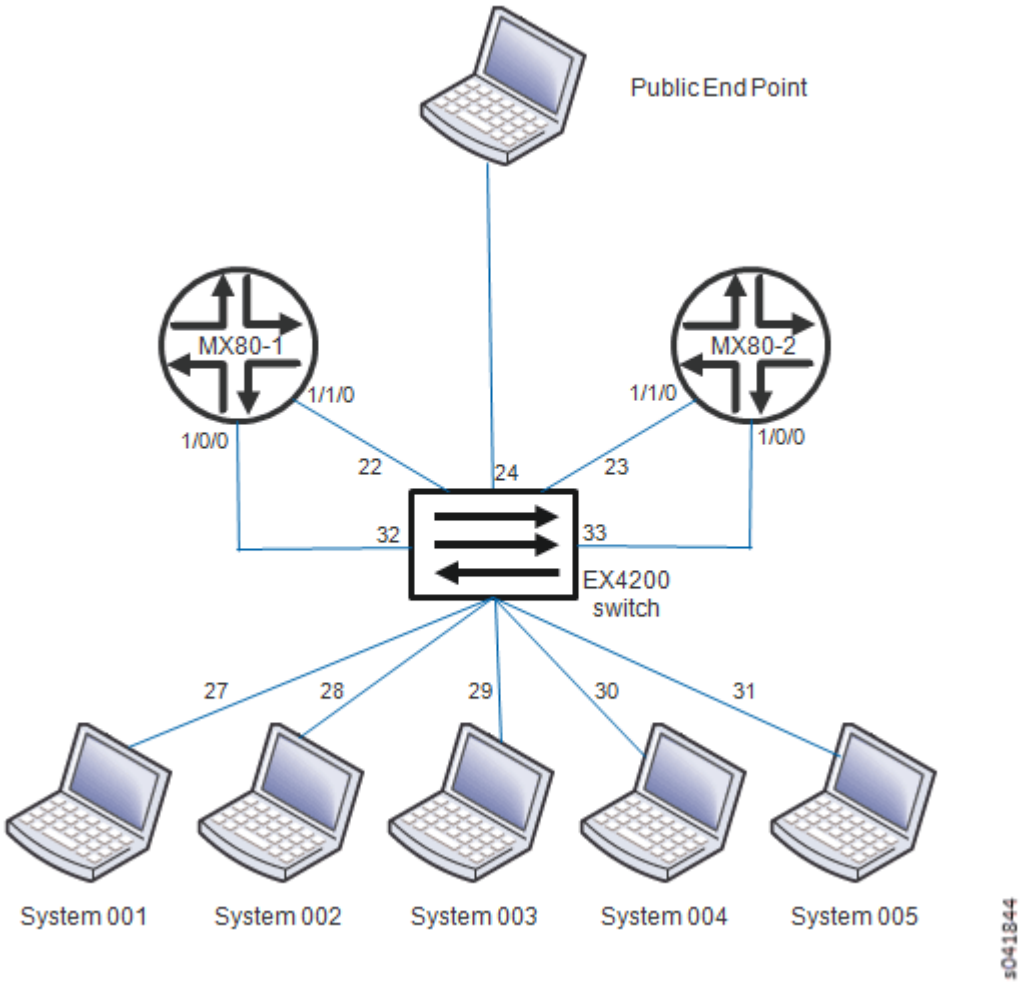
Table 5: Sample Addressing Scheme for Example

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 36 on page 82 provides a guideline diagram for the physical topology for the simple tiered web application example.

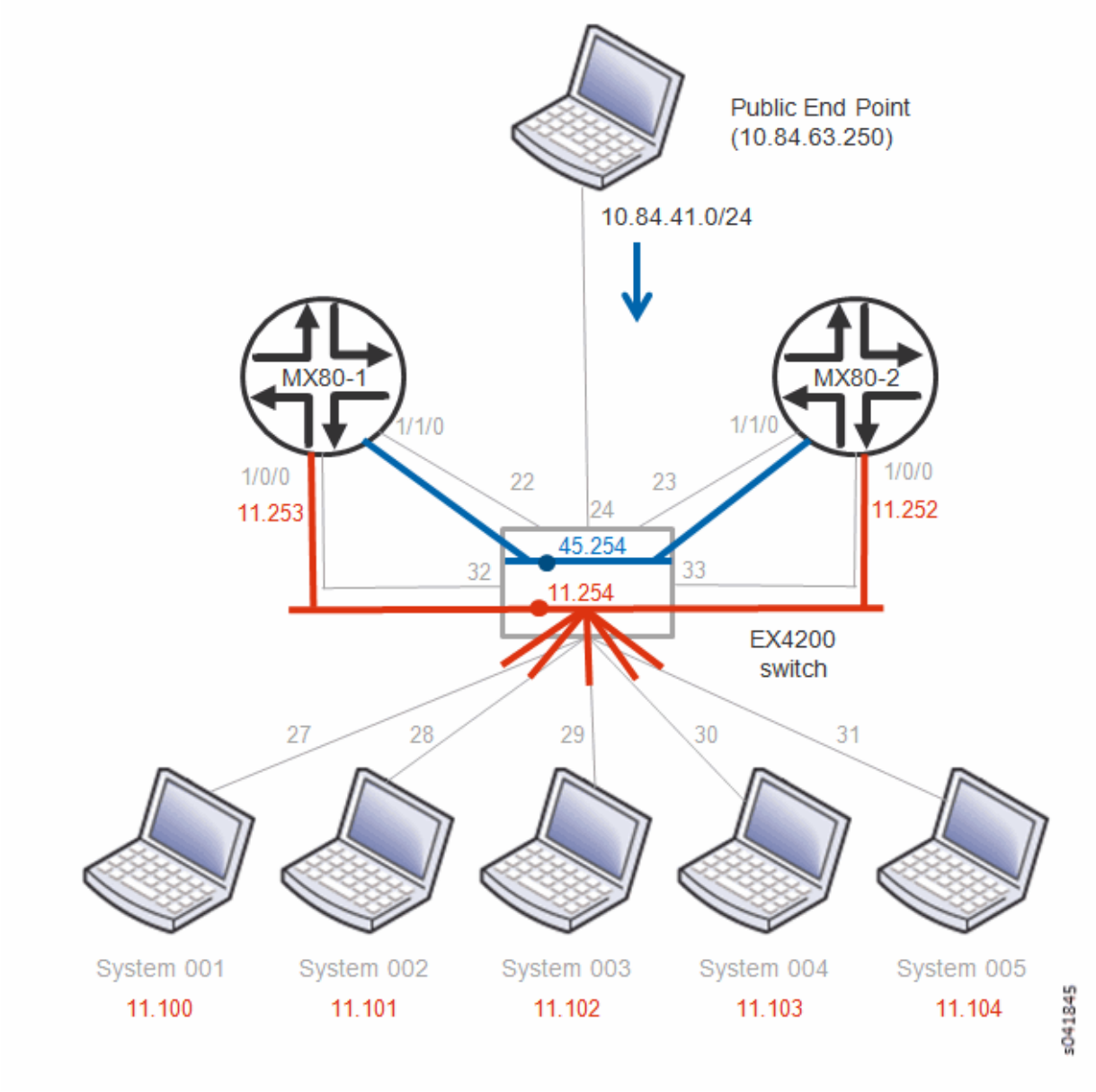
Figure 36: Sample Physical Topology for Simple Tiered Web Application



Sample Physical Topology Addressing

Figure 37 on page 83 provides a guideline diagram for addressing the physical topology for the simple tiered web application example.

Figure 37: Sample Physical Topology Addressing



SEE ALSO

Sample Network Configuration for Devices for Simple Tiered Web Application | 84

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 76. Configurations are shown for Juniper Networks devices: two MX80s and one EX4200.

MX80-1 Configuration

```
version 12.2R1.3;
system {
    root-authentication {
        encrypted-password "xxxxxxxxxx"; ## SECRET-DATA
    }
    services {
        ssh {
            root-login allow;
        }
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
    }
}
chassis {
    fpc 1 {
        pic 0 {
            tunnel-services;
        }
    }
}
interfaces {
    ge-1/0/0 {
        unit 0 {
            family inet {
```

```
                address 10.84.11.253/24;
            }
        }
    }
    ge-1/1/0 {
        description "IP Fabric interface";
        unit 0 {
            family inet {
                address 10.84.45.1/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 127.0.0.1/32;
            }
        }
    }
}
routing-options {
    static {
        route 0.0.0.0/0 next-hop 10.84.45.254;
    }
    route-distinguisher-id 10.84.11.253;
    autonomous-system 64512;
    dynamic-tunnels {
        setup1 {
            source-address 10.84.11.253;
            gre;
            destination-networks {
                10.84.11.0/24;
            }
        }
    }
}
protocols {
    bgp {
        group mx {
            type internal;
            local-address 10.84.11.253;
            family inet-vpn {
                unicast;
```

```
            }
            neighbor 10.84.11.252;
        }
        group contrail-controller {
            type internal;
            local-address 10.84.11.253;
            family inet-vpn {
                unicast;
            }
            neighbor 10.84.11.101;
            neighbor 10.84.11.102;
        }

    }
}
routing-instances {
    customer-public {
        instance-type vrf;
        interface ge-1/1/0.0;
        vrf-target target:64512:10000;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 10.84.45.254;
            }
        }
    }
}
```

MX80-2 Configuration

```
version 12.2R1.3;
system {
    root-authentication {
        encrypted-password "xxxxxxxxx"; ## SECRET-DATA
    }
    services {
        ssh {
            root-login allow;
        }
    }
    syslog {
        user * {
```

```
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
    }
}
chassis {
    fpc 1 {
        pic 0 {
            tunnel-services;
        }
    }
}
interfaces {
    ge-1/0/0 {
        unit 0 {
            family inet {
                address 10.84.11.252/24;
            }
        }
    }
    ge-1/1/0 {
        description "IP Fabric interface";
        unit 0 {
            family inet {
                address 10.84.45.2/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 127.0.0.1/32;
            }
        }
    }
}
routing-options {
    static {
        route 0.0.0.0/0 next-hop 10.84.45.254;
    }
```

```
    route-distinguisher-id 10.84.11.252;
    autonomous-system 64512;
    dynamic-tunnels {
        setup1 {
            source-address 10.84.11.252;
            gre;
            destination-networks {
                10.84.11.0/24;
            }
        }
    }
}
protocols {
    bgp {
        group mx {
            type internal;
            local-address 10.84.11.252;
            family inet-vpn {
                unicast;
            }
            neighbor 10.84.11.253;
        }
        group contrail-controller {
            type internal;
            local-address 10.84.11.252;
            family inet-vpn {
                unicast;
            }
            neighbor 10.84.11.101;
            neighbor 10.84.11.102;
        }

    }

}
routing-instances {
    customer-public {
        instance-type vrf;
        interface ge-1/1/0.0;
        vrf-target target:64512:10000;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 10.84.45.254;
```

```
            }
        }
    }
}
```

EX4200 Configuration

```
system {
    host-name EX4200;
    time-zone America/Los_Angeles;
    root-authentication {
        encrypted-password "xxxxxxxxxxxx"; ## SECRET-DATA
    }
    login {
        class read {
            permissions [ clear interface view view-configuration ];
        }
        user admin {
            uid 2000;
            class super-user;
            authentication {
                encrypted-password "xxxxxxxxxxx"; ## SECRET-DATA
            }
        }
        user user1 {
            uid 2002;
            class read;
            authentication {
                encrypted-password "xxxxxxxxxxxxx"; ## 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;
        }
    }
}
```

4

CHAPTER

Configuring Services

Configuring DNS Servers | 92

Distributed Service Resource Allocation with Containerized Contrail | 105

Support for Broadcast and Multicast | 116

Configuring DNS Servers

IN THIS SECTION

- DNS Overview | 92
- Defining Multiple Virtual Domain Name Servers in Contrail | 93
- IPAM and Virtual DNS | 93
- DNS Record Types | 94
- Configuring DNS on the User Interface | 95
- Configuring DNS Using Scripts | 103

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:

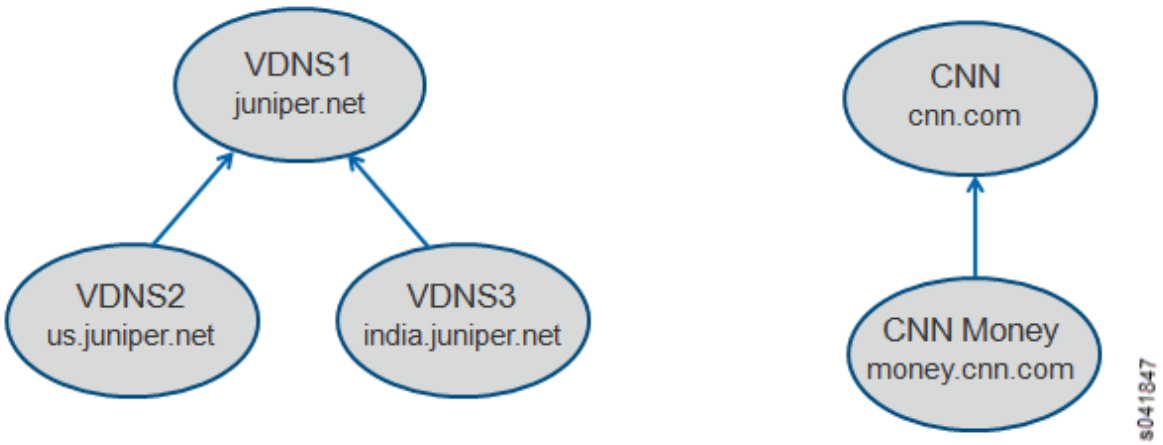
- It should be possible to configure multiple domain name servers to provide name resolution service for the virtual machines spawned in the system.
- It should be possible to configure the domain name servers to form DNS server hierarchies required by each tenant.
    - The hierarchies can be independent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

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 38 on page 93 shows examples of virtual DNS servers defined in **default-domain**, providing the name service for the DNS domains indicated.

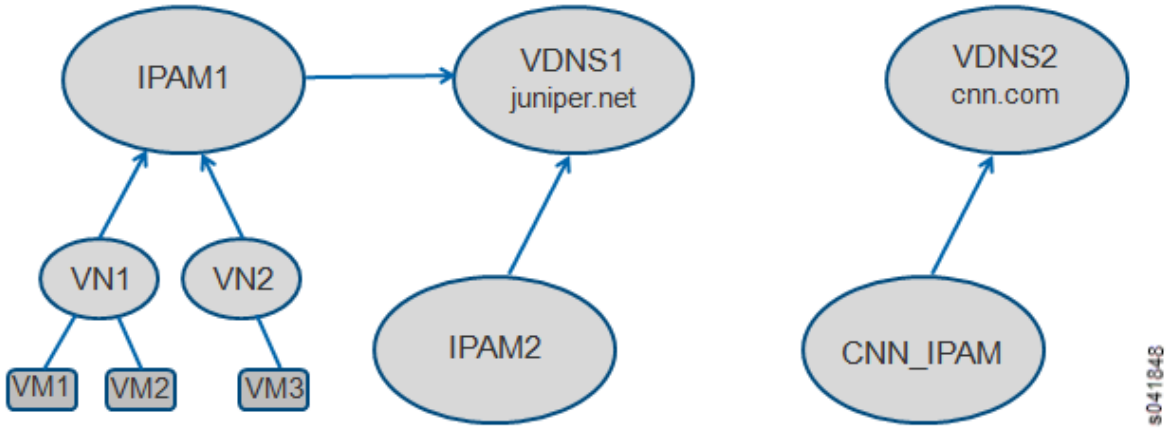
Figure 38: DNS Servers Examples



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 39 on page 94

Figure 39: 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 6 on page 94 for descriptions of the record types.

Table 6: DNS Record Types Supported

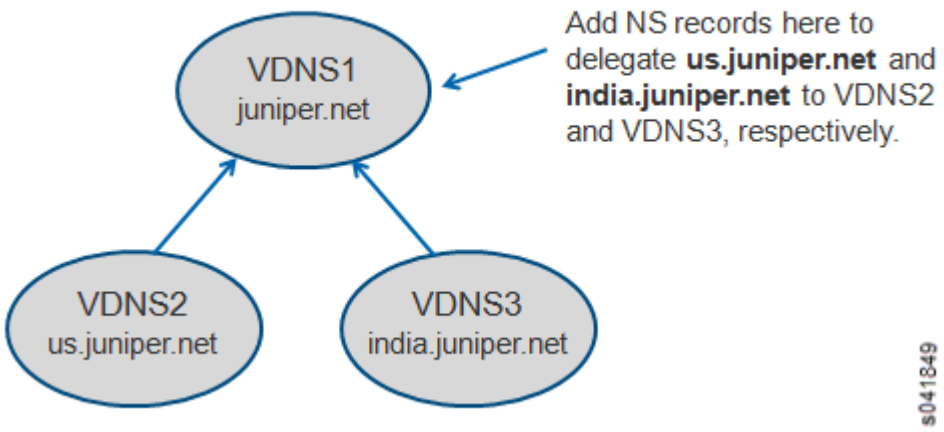
DNS Record Type	Description
A	Used for mapping hostnames to IPv4 addresses. Name refers to the name of the virtual machine, and data is the IPv4 address of the virtual machine.
CNAME	Provides an alias to a name. Name refers to the name of the virtual machine, and data is the new name (alias) for the virtual machine.
PTR	A pointer to a record, it provides reverse mapping from an IP address to a name. Name refers to the IP address, and data is the name for the virtual machine. The address in the PTR record should be part of a subnet configured for a VN within one of the IPAMs referring to this virtual DNS server.

Table 6: DNS Record Types Supported *(Continued)*

DNS Record Type	Description
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 40 on page 95 shows an example usage for the DNS record type of **NS**.

Figure 40: Example Usage for NS Record Type



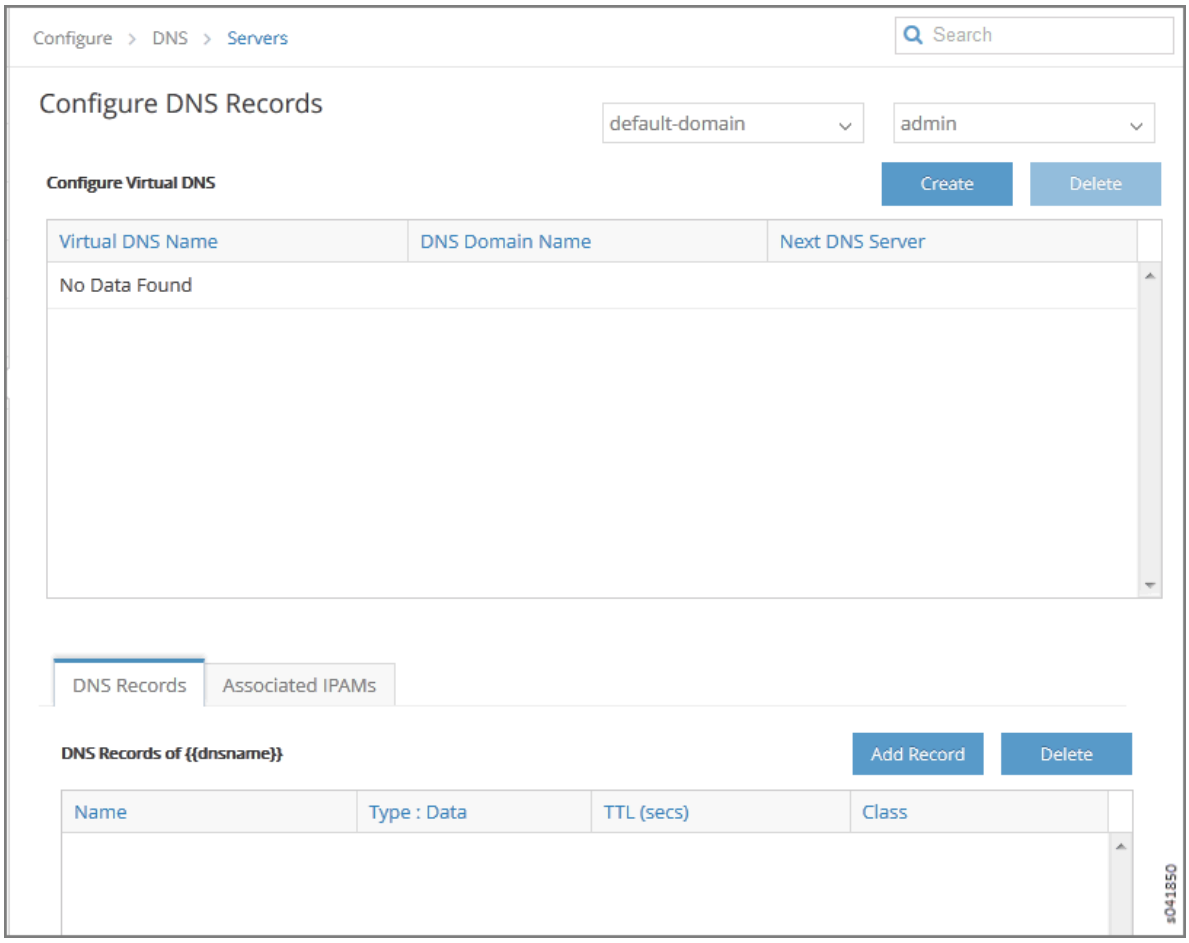
Configuring DNS on the User Interface

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 41 on page 96.

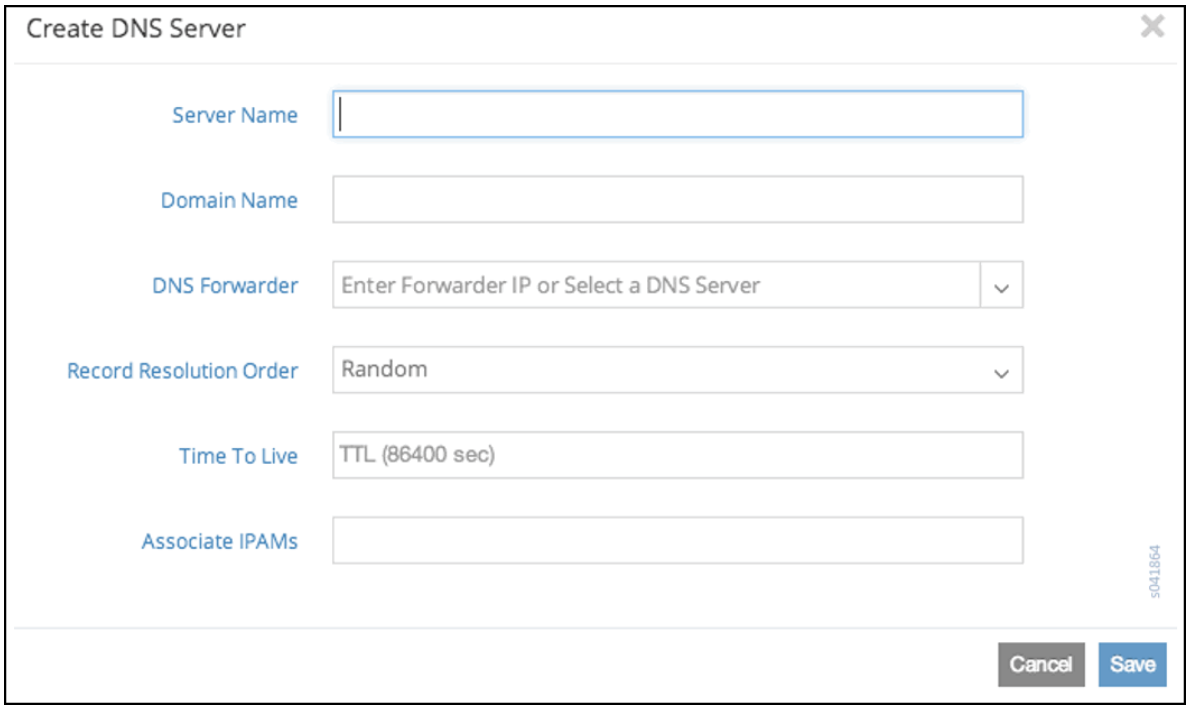
Figure 41: Configure DNS Records



2. To add a new DNS server, click the **Create** button.

   Enter DNS server information in the **Add DNS** window; see Figure 42 on page 97

Figure 42: Add DNS



Complete the fields for the new server; see Table 7 on page 97.

Table 7: 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**.

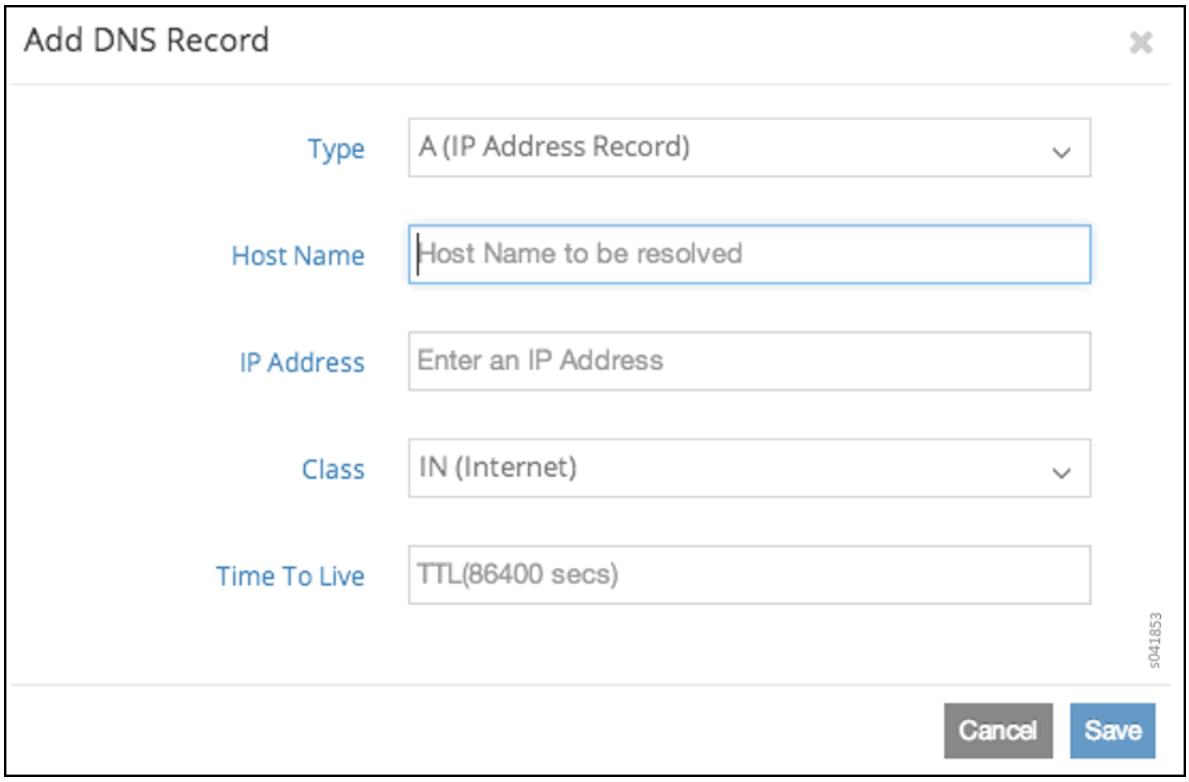
Table 7: Add DNS Fields *(Continued)*

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 43 on page 99.

Figure 43: Add DNS Record



4. Complete the fields for the new record; see Table 8 on page 99.

Table 8: 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.

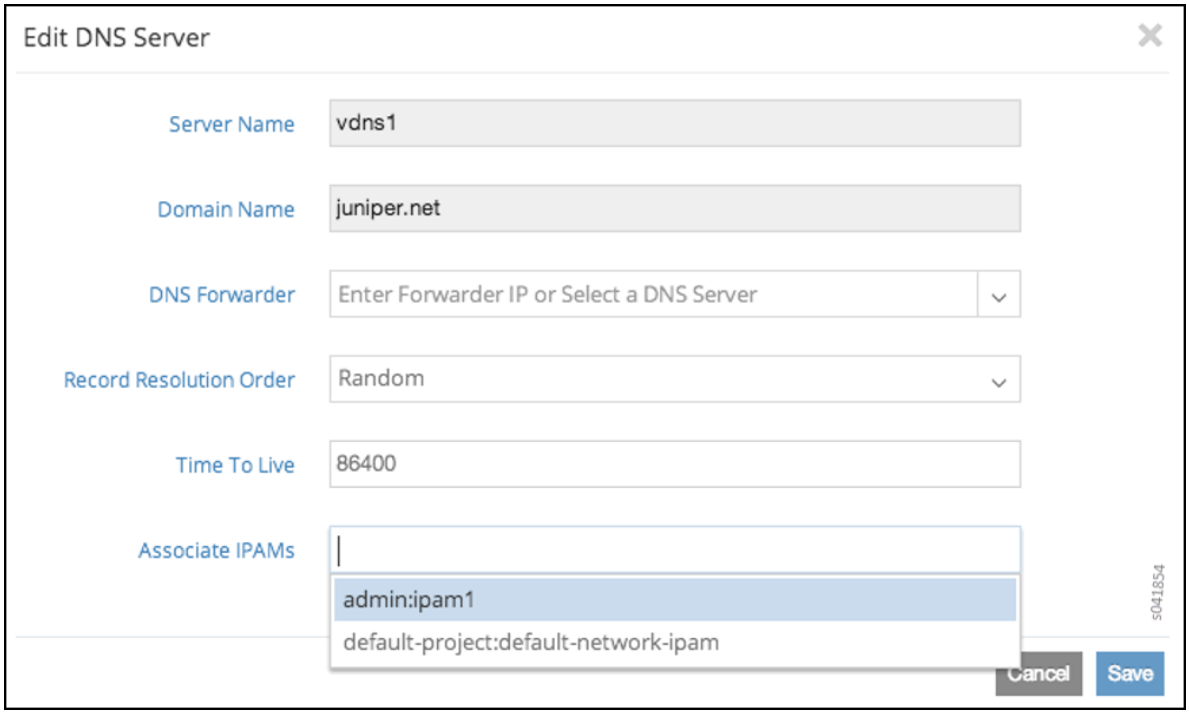
Table 8: Add DNS Record Fields *(Continued)*

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 44 on page 100.

Figure 44: Associate IPAMs to DNS



Complete the IPAM associations, using the field descriptions in Table 9 on page 100.

Table 9: 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 9: 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.

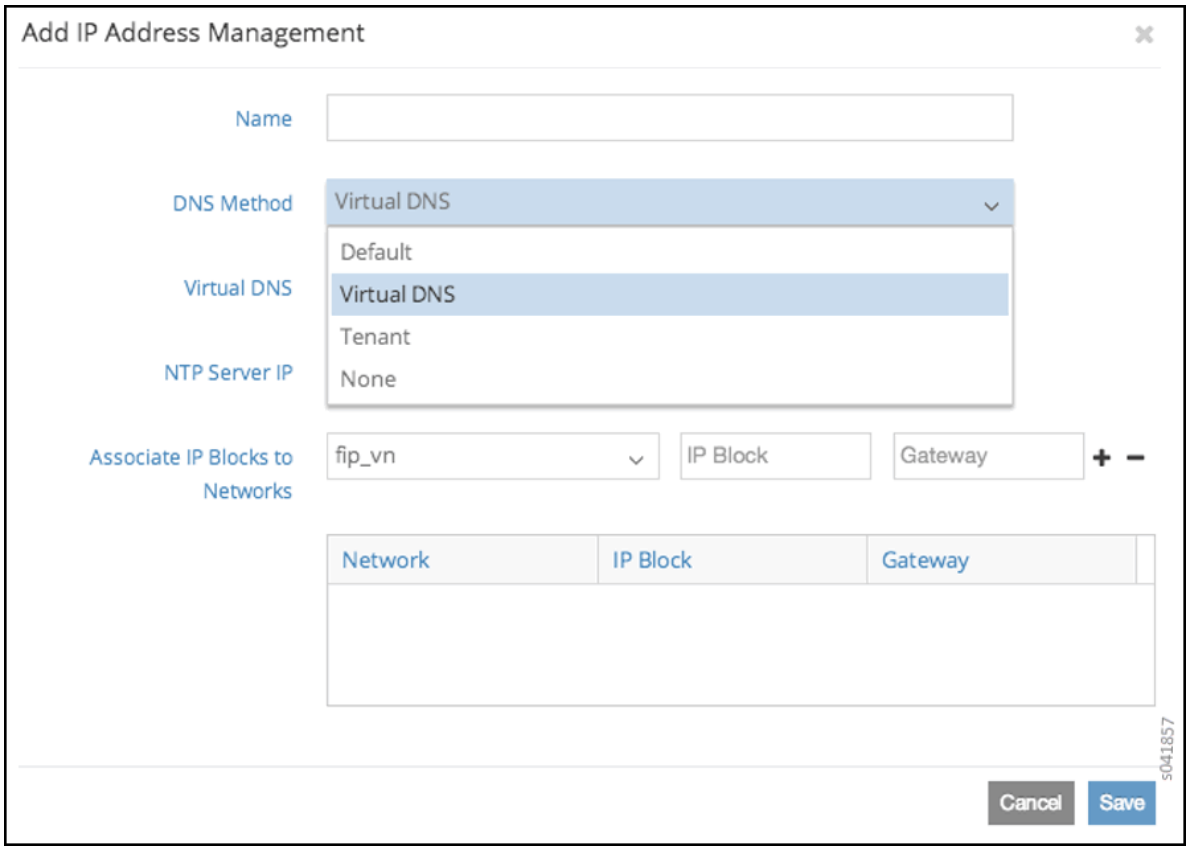
6. Use the **IP Address Management** page (**Configure > Networking > IP Address Management**); see Figure 45 on page 101) to configure the DNS mode for any DNS server and to associate an IPAM to DNS servers of any mode or to tenants' IP addresses.

Figure 45: 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 46 on page 102.

Figure 46: DNS Server



8. In the first field, select the **DNS Method** from a list (**None, Default DNS, Tenant DNS, Virtual DNS**; see Table 10 on page 102.

Table 10: 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 10: 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 11 on page 104.

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 11: DNS Scripts

Action	Script
Add a virtual DNS server	Script: `add_virtual_dns.py` Sample usage: `python add_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --name vdns1 --domain_name default-domain --dns_domain juniper.net --dyn_updates --record_order random --ttl 1200 --next_vdns default-domain:vdns2`
Delete a virtual DNS server	Script: `del_virtual_dns_record.py` Sample usage: `python del_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --fq_name default-domain:vdns1`
Add a DNS record	Script: `add_virtual_dns_record.py` Sample usage: `python add_virtual_dns_record.py --api_server_ip 10.204.216.21 --api_server_port 8082 --name rec1 --vdns_fqname default-domain:vdns1 --rec_name one --rec_type A --rec_class IN --rec_data 1.2.3.4 --rec_ttl 2400`
Delete a DNS record	Script: `del_virtual_dns_record.py` Sample usage: `python del_virtual_dns_record.py --api_server_ip 10.204.216.21 --api_server_port 8082 --fq_name default-domain:vdns1:rec1`
Associate a virtual DNS server with an IPAM	Script: `associate_virtual_dns.py` Sample usage: `python associate_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --ipam_fqname default-domain:demo:ipam1 --vdns_fqname default-domain:vdns1`
Disassociate a virtual DNS server with an IPAM	Script: `disassociate_virtual_dns.py` Sample usage: `python disassociate_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --ipam_fqname default-domain:demo:ipam1 --vdns_fqname default-domain:vdns1`

Distributed Service Resource Allocation with Containerized Contrail

IN THIS SECTION

- Replacement of Centralized Discovery Service | 105
- New Distributed Resource Allocation Manager | 106
- Changes in Configuration Files | 106

Starting with Contrail Release 4.0, the existing centralized Contrail discovery service is replaced with a distributed method of allocating service resources.

Replacement of Centralized Discovery Service

In Contrail releases prior to Release 4.0, the Contrail discovery service is a centralized service resource allocation module with high availability, used primarily to automatically load-balance service resources in the system.

In the previous centralized discovery method, new service resources are registered (published) directly to the Contrail discovery module and allocated to the requester (subscriber) of the service resource, without disrupting the running state of the subscribers.

The centralized discovery method requires using a database to:

- synchronize across Contrail discovery nodes.
- maintain the list of publishers, subscribers, and the health of published services across reloads.
- provide a centralized view of the service allocation and health of the services.

This centralized discovery method resulted in unnecessary system churn when services were falsely marked as down, due to periodic health updates of services made to the database nodes, resulting in reallocation of healthy services.

Starting with Contrail 4.0, the Contrail discovery services centralized resource allocation manager has been removed. Its replacement is a distributed resource allocation list of service nodes, maintained in each module of the system.

New Distributed Resource Allocation Manager

Starting with Contrail Release 4.0, service resources are managed with a distributed allocation manager, with the following features:

- Each system module is provisioned with a list of service nodes (publishers).
- Each system module randomizes the list of service nodes and uses the resources. The randomized list is expected to be fairly load-balanced.
- When currently-used services are down, the system module detects the down immediately and reacts with no downtime by selecting another service from the list. This is distinctly different from the previous model, in which the module would need to contact the discovery service to check for available services, resulting in a finite time loss for allocation, distribution, and application of a new set of services.
- When service nodes are added or deleted, the system administrator updates the configuration file of all daemons using the service type of the service node added or deleted, sending a SIGHUP to the respective daemons.
- Each daemon randomizes the service list independently and reallocates the resources.

Deprecation of IF-MAP

In Contrail 4.0, the Interface for Metadata Access Points (IF-MAP) methodology has been deprecated. Contrail 4.0 uses CONFIGDB sections in configuration files instead of IF-MAP sections.

Changes in Configuration Files

Table 12 on page 107 lists configuration files in the Contrail system that have changes to enable the distributed service resource allocation system, starting with Contrail 4.0. In general, the changes include removing (deprecating) discovery server sections and subsections, and adding parameters needed to identify service resources in all modules.

Each daemon randomizes the published service list and uses the resources. Additionally, each daemon provides a SIGHUP handler to manage the addition or deletion of publishers.

Table 12: Contrail 4.0 Changes in Configuration Files

Configuration File	Configuration Parameter	Changes
contrail-vrouter-agent.conf	[DISCOVERY]	Section deprecated
	[CONTROL-NODE].servers	Provisioned list of control-node [role=control] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:5269 10.1.1.12:5269
	[DNS].servers	Provisioned list of DNS [role=control] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:53 10.1.1.2:5
	[DEFAULT].collectors	Provisioned list of Collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
contrail-control.conf	[DISCOVERY]	Section deprecated
	[DEFAULT].collectors	Provisioned list of Collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
	[CONFIGDB].rabbitmq_server_list	Provisioned list of config-node [role=cfgm] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:5672 10.1.1.2:5672

Table 12: Contrail 4.0 Changes in Configuration Files *(Continued)*

Configuration File	Configuration Parameter	Changes
	`[CONFIGDB].rabbitmq_user`	guest (default string)
	`[CONFIGDB].rabbitmq_password`	guest (default string)
	`[CONFIGDB].config_db_server_list`	Provisioned list of Config DB [role=database] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:9042 10.1.1.2:9042 NOTE: Docker uses 9041 as port
	`[CONFIGDB].certs_store`	Deprecated
	`[CONFIGDB].password`	Deprecated
	`[CONFIGDB].server_url`	Deprecated
	`[CONFIGDB].user`	Deprecated
	`[CONFIGDB].stale_entries_cleanup_timeout`	Deprecated
	`[CONFIGDB].end_of_rib_timeout`	Deprecated
`contrail-dns.conf`		
	`[DISCOVERY]`	Deprecated

Table 12: Contrail 4.0 Changes in Configuration Files *(Continued)*

Configuration File	Configuration Parameter	Changes
	`[DEFAULT].collectors`	Provisioned list of Collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
	`[CONFIGDB].rabbitmq_server_list`	Provisioned list of config-node [role=cfgm] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:5672 10.1.1.2:5672
	`[CONFIGDB].rabbitmq_user`	guest (default string)
	`[CONFIGDB].rabbitmq_password`	guest (default string)
	`[CONFIGDB].config_db_server_list`	Provisioned list of Config DB [role=database] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:9042 10.1.1.2:9042 NOTE: Dockers use 9041 as port
	`[CONFIGDB].certs_store`	Deprecated
	`[CONFIGDB].password`	Deprecated
	`[CONFIGDB].server_url`	Deprecated
	`[CONFIGDB].user`	Deprecated

Table 12: Contrail 4.0 Changes in Configuration Files *(Continued)*

Configuration File	Configuration Parameter	Changes
	`[CONFIGDB].stale_entries_cleanup_timeout`	Deprecated
	`[CONFIGDB].end_of_rib_timeout`	Deprecated
`contrail-collector.conf`	`[DISCOVERY]`	Deprecated
	`[API_SERVER].api_server_list`	Provisioned list of api-servers [role=config] in the format: ip-address:port Example: 10.1.1.1:8082 10.1.1.2:8082
`contrail-alarm-gen.conf`	`[DISCOVERY]`	Deprecated
	`[DEFAULTS].collectors`	Provisioned list of Collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
	`[API_SERVER].api_server_list`	Provisioned list of api-servers [role=config] in the format: ip-address:port Example: 10.1.1.1:8082 10.1.1.2:8082
	`[REDIS].redis_uve_list`	Provisioned list of redis instances [role=collector] Example: 192.168.0.29:6379 192.168.0.30:6379

Table 12: Contrail 4.0 Changes in Configuration Files *(Continued)*

Configuration File	Configuration Parameter	Changes
contrail-analytics-api.conf	[DISCOVERY]	Section deprecated
	[DEFAULTS].collectors	Provisioned list of collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
	[REDIS].redis_uve_list	Provisioned list of redis instances [role=collector] Example: 192.168.0.29:6379 192.168.0.30:6379
contrail-api.conf	[DISCOVERY]	Section deprecated
	[DEFAULTS].collectors	Provisioned list of collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
contrail-schema.conf	[DISCOVERY]	Section deprecated
	[DEFAULTS].collectors	Provisioned list of Collector [role=collector] service providers in ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
contrail-svc-monitor.conf	[DISCOVERY]	Section deprecated

Table 12: Contrail 4.0 Changes in Configuration Files *(Continued)*

Configuration File	Configuration Parameter	Changes
	`[DEFAULTS].collectors`	Provisioned list of Collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
`contrail-device-manager.conf`	`[DISCOVERY]`	Section deprecated
	`[DEFAULTS].collectors`	Provisioned list of Collector [role=collector] service providers in ip-address:port ip-address2:port format Example: 10.1.1.1:8086 10.1.1.2:8086
`contrail-analytics-nodemgr.conf`	`[DISCOVERY]`	Section deprecated
	`[COLLECTOR].server_list`	Provisioned list of Collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
`contrail-config-nodemgr.conf`	`[DISCOVERY]`	Section deprecated
	`[COLLECTOR].server_list`	Provisioned list of Collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
`contrail-control-nodemgr.conf`	`[DISCOVERY]`	Section deprecated

Table 12: Contrail 4.0 Changes in Configuration Files *(Continued)*

Configuration File	Configuration Parameter	Changes
	`[COLLECTOR].server_list`	Provisioned list of Collector [role=collector] service providers in ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
`contrail-database-nodemgr.conf`	`[DISCOVERY]`	Section deprecated
	`[COLLECTOR].server_list`	Provisioned list of Collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
`contrail-vrouter-nodemgr.conf`	`[DISCOVERY]`	Section deprecated
	`[COLLECTOR].server_list`	Provisioned list of Collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
`contrail-query-engine.conf`	`[DISCOVERY]`	Section deprecated
	`[COLLECTOR].server_list`	Provisioned list of Collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
`contrail-snmp-collector.conf`	`[DISCOVERY]`	Section deprecated

Table 12: Contrail 4.0 Changes in Configuration Files *(Continued)*

Configuration File	Configuration Parameter	Changes
	`[DEFAULTS].collectors`	Provisioned list of Collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
	`[API_SERVER].api_server_list`	Provisioned list of api-servers [role=config] in the format: ip-address:port Example: 10.1.1.1:8082 10.1.1.2:8082
`contrail-topology.conf`	`[DISCOVERY]`	Section deprecated
	`[DEFAULTS].collectors`	Provisioned list of Collector [role=collector] service providers in the format: ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086
	`[API_SERVER].api_server_list`	Provisioned list of api-servers [role=config] in ip-address:port Example: 10.1.1.1:8082 10.1.1.2:8082
Contrail Web UI		
`config.global.js`	`config.discovery.server`	Discovery subsection deprecated
	`config.discovery.port`	Discovery subsection deprecated

Table 12: Contrail 4.0 Changes in Configuration Files *(Continued)*

Configuration File	Configuration Parameter	Changes
	`config.cnfg.server_ip`	Provisioned list of Config [role=cfgm] service providers as list of ip-address Example: ['10.1.1.1 10.1.1.2']
	`config.cnfg.server_port`	Server port as a string Example: '8082'
	`config.analytics.server_ip`	Provisioned list of Collector [role=collector] service providers as a list of ip-address Example: ['10.1.1.1 10.1.1.2']
	`config.analytics.server_port`	Server port as a string Example: '8081'
	`config.dns.server_ip`	Provisioned list of Controller [role=control] service providers as a list of ip-address Example: ['10.1.1.1 10.1.1.2']
	`config.dns.server_port`	Server port as a string Example: '8092'

Support for Broadcast and Multicast

IN THIS SECTION

- Subnet Broadcast | 116
- All-Broadcast/Limited-Broadcast and Link-Local Multicast | 117
- Host Broadcast | 118

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.

5

CHAPTER

Configuring Service Chaining

Service Chaining | 120

Service Chaining MX Series Configuration | 124

ECMP Load Balancing in the Service Chain | 126

Service Chain Version 2 with Port Tuple | 127

Service Chain Route Reorigination | 131

Example: Creating an In-Network Service Chain by Using Contrail Command | 154

Example: Creating an In-Network-NAT Service Chain | 163

Example: Creating a Transparent Service Chain by Using Contrail Command | 172

Using Static Routes with Services | 181

Configuring Metadata Service | 185

Service Chaining

IN THIS SECTION

- Service Chaining Basics | 120
- Service Chaining Configuration Elements | 122

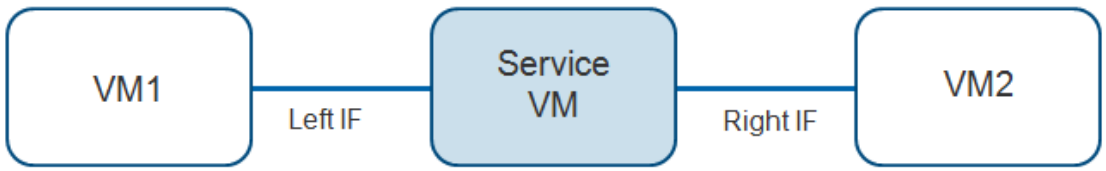
Contrail Controller supports chaining of various Layer 2 through Layer 7 services such as firewall, NAT, IDP, and so on.

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.

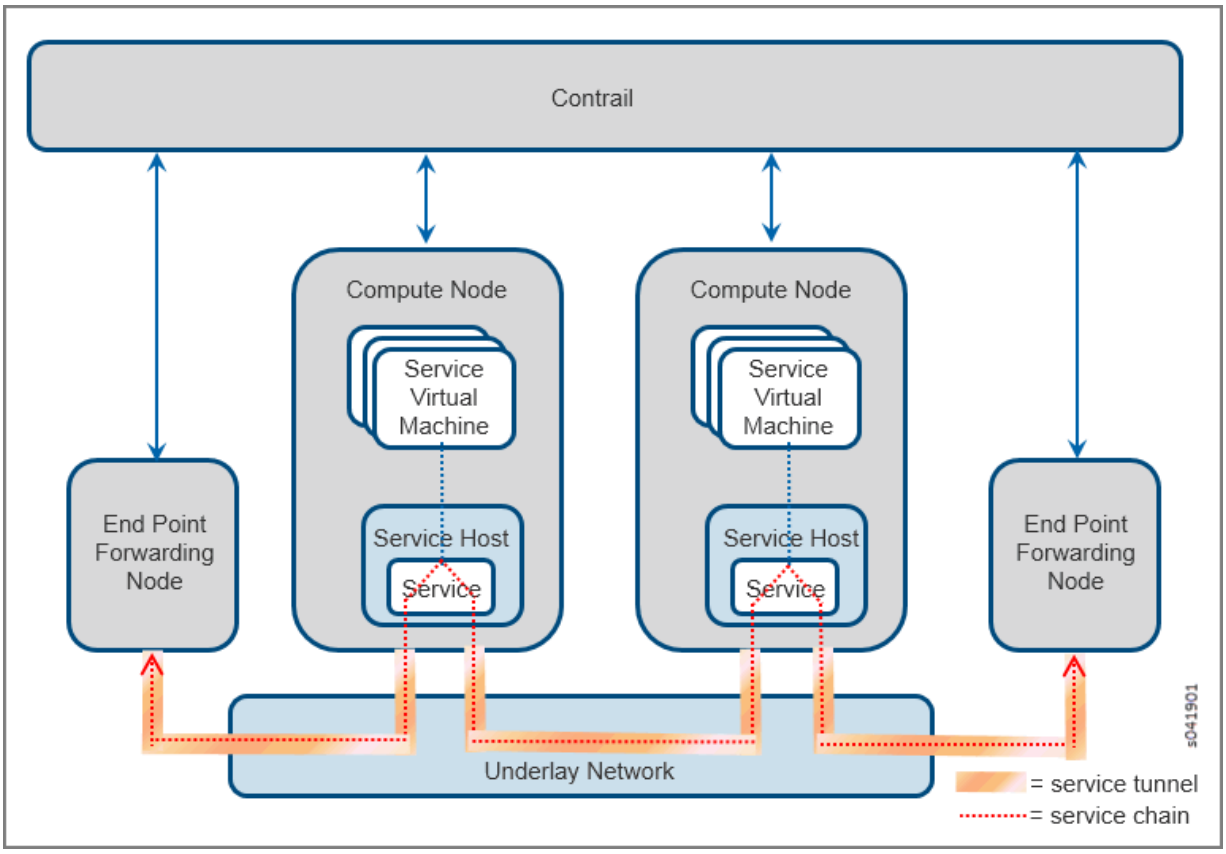
Figure 47 on page 120 shows the basic service chain schema, with a single service. The service VM spawns the service, using the convention of left interface (left IF) and right interface (right IF). Multiple services can also be chained together.

Figure 47: 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 48 on page 121 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 48: 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 | 154

Example: Creating an In-Network-NAT Service Chain | 163

Example: Creating a Transparent Service Chain by Using Contrail Command | 172

ECMP Load Balancing in the Service Chain | 126

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;
        }
}
```

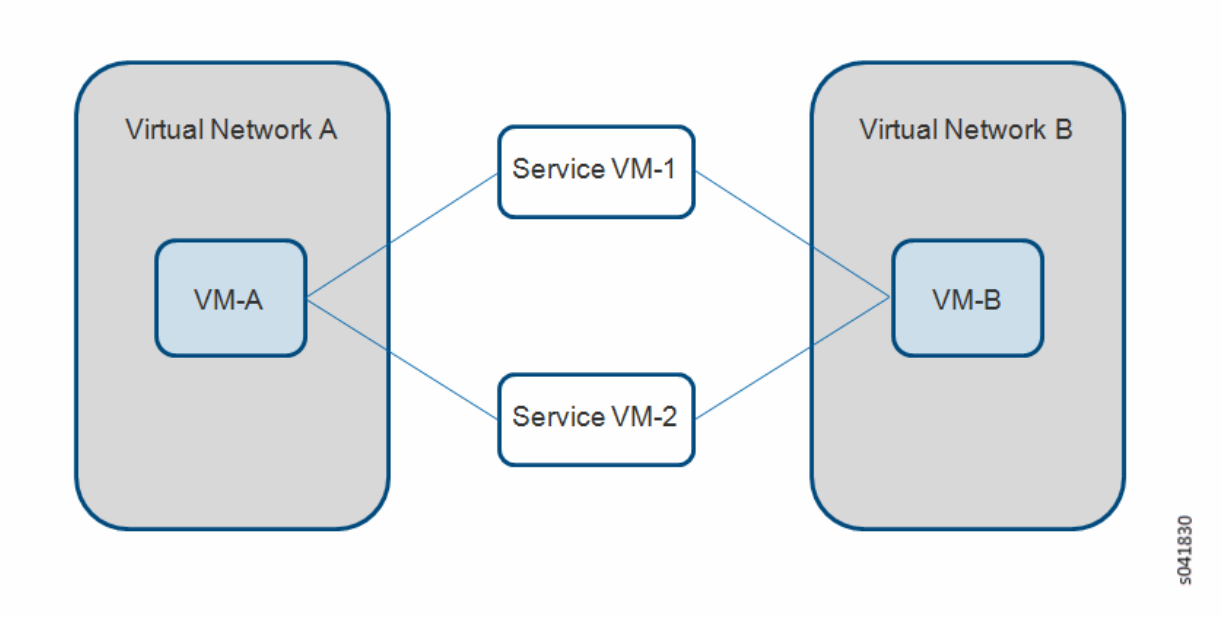
4. The final step is to add `target:100:10000` to the public virtual network and `target:100:20000` to the enterprise virtual network, using the Contrail Juniper Networks interface.

A full MX Series router configuration for Contrail can be seen in "Sample Network Configuration for Devices for Simple Tiered Web Application" on page 84.

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 49 on page 126, 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 49: Load Balancing a Service Chain



The following are the major features of load balancing in the service chain:

- Load balancing can be configured at every level of the service chain.
- Load balancing is supported in routed and bridged service chain modes.
- Load balancing can be used to achieve high availability—if a service VM goes down, the traffic passing through that service VM can be distributed through another service VM.
- A load balanced traffic stream always follows the same path through the chain of service VM.

RELATED DOCUMENTATION

Service Chaining | 120

Customized Hash Field Selection for ECMP Load Balancing

Service Chain Version 2 with Port Tuple

IN THIS SECTION

- Overview of Port Tuple | 127
- Service Chain Version 2 Sample Workflow | 128
- Service Chain with Health Check | 130

Starting with Contrail 3.0, the user can create a port-tuple object for binding service instances to ports.

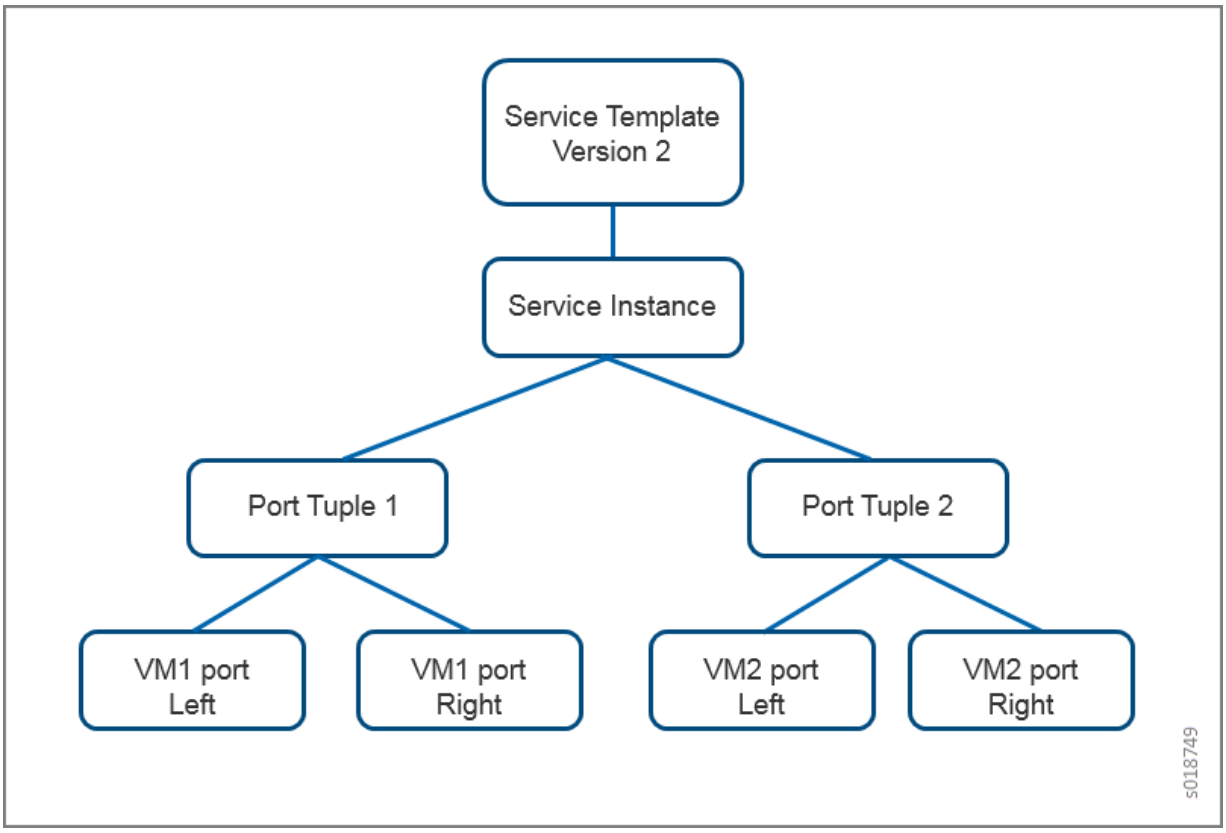
Overview of Port Tuple

In previous versions of Contrail, when a service instance is created for a virtual machine (VM)-based service, the service monitor creates one or more VM objects and creates a port for each VM object. Each VM object is a placeholder for binding a service instance to a port. The VM object also acts as a placeholder for the instance ID when using equal-cost multipath (ECMP).

Using the VM object as a placeholder doesn't add value beyond binding information between the service instance object and the port objects. By using a port-tuple object, the service instance can be linked directly to the port objects, eliminating the need to create a VM object. This simplifies the implementation of service instance objects, and also allows integration with Heat templates.

With a port-tuple object, the user can create ports and pass the port information when creating a service instance. The ports can be created as part of a VM launch from Nova or without using a VM launch. The ports are linked to a port-tuple object that is a child of a service instance. This functionality can also be leveraged in Heat stacks. See Figure 50 on page 128.

Figure 50: Port Tuple Overview



Service Chain Version 2 Sample Workflow

With Contrail service templates Version 2, the user can create ports and bind them to a VM-based or container-based service instance, by means of a port-tuple object. All objects created with the Version 2 service template are visible to the Contrail Heat engine, and are 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 `left` and `right` are used for forwarding.

6. Link the port tuple to a service instance.
7. Launch the service instance. This creates the necessary objects in the Contrail database.

> **NOTE**: Port-tuple is *not* supported on transparent service instance, whether active/active, active/standby, or scale-out.

Service Chain with Equal-Cost Multipath in Active-Active Mode

Equal-cost multipath (ECMP) can be used to distribute traffic across VMs. To support ECMP in the service chain, create multiple port tuples within the same service instance. The labels should be the same for the VM ports in each port tuple. For example, if port tuple 1 uses the labels `left` and `right`, then port tuple 2 in the same service instance should also use the labels `left` and `right` for its ports.

When there are multiple port tuples, the default mode of operation is `active-active`.

Service Chain Active-Standby Mode with Allowed Address Pair

To support `active-standby` mode, you must configure an allowed address pair on the interfaces. The `active-standby` is used as the high availability mode in the allowed address pair. The allowed address pair is configured as part of the service instance for a particular VM port label. For example, if the allowed address pair is configured in a service instance for the port with the label `left`, then all of the port-tuple VM ports with the label `left` will use the allowed address pair high availability mode.

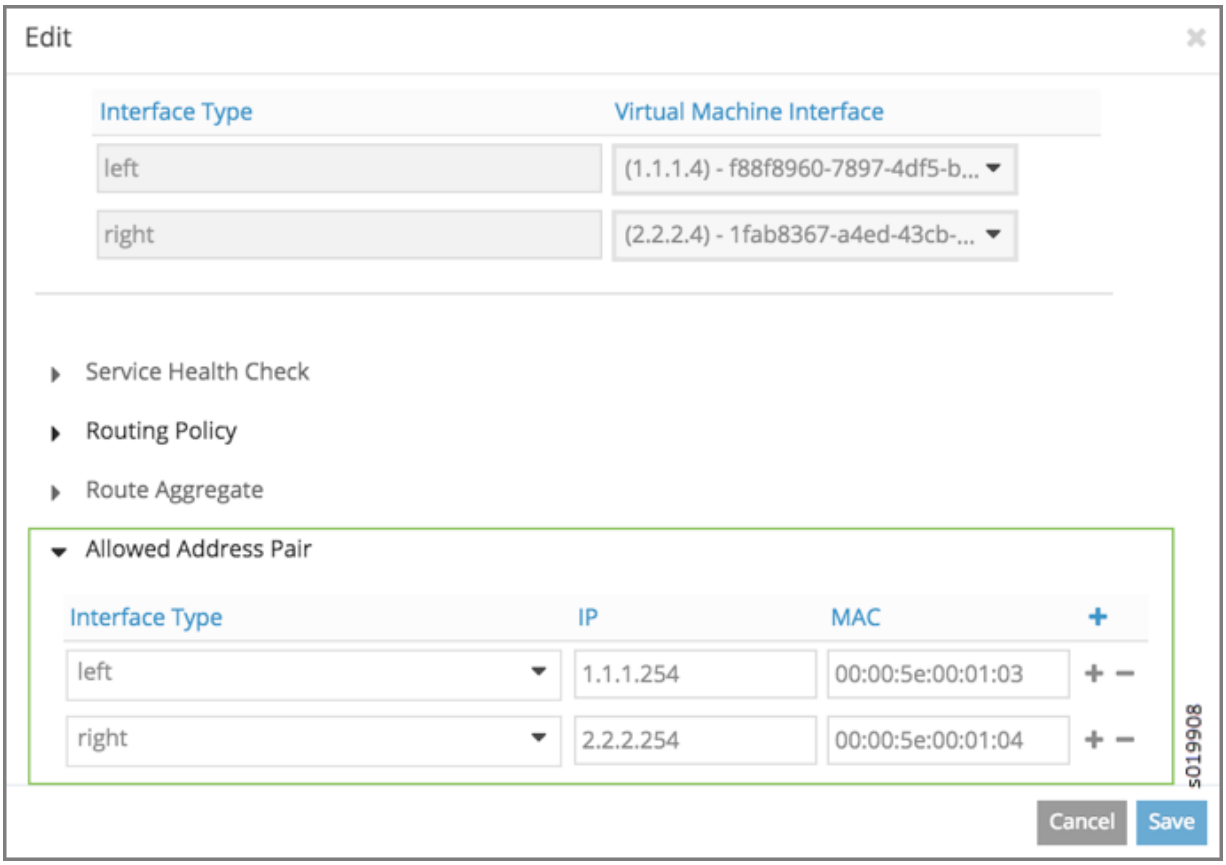
Allowed Address Pair

An allowed address pair extension is an OpenStack feature supported by Contrail.

By default, there is no way to specify additional MAC/IP address pairs that are allowed to pass through a port in Neutron, because ports are locked down to their MAC address and the fixed IPs associated with their port for anti-spoofing reasons. This locking can sometimes prevent protocols such as VRRP from providing a high availability failover strategy. Using the allowed address pair extension enables additional IP/MAC pairs to be allowed through ports in Neutron.

In Contrail, you can configure allowed address pairs in the service instance configuration, using **Configure > Services > Service Instances > Allowed Address Pair**, see Figure 51 on page 130.

Figure 51: Edit Service Instance, Allowed Address Pair



For more information about OpenStack allowed address pairs, see https://docs.openstack.org/dragonflow/latest/specs/allowed_address_pairs.html.

Service Chain with Static Route Table

The service chain Version 2 also supports static route tables. A static route table is configured similar to how the allowed address pair is configured, except with using the label `right`. The route table will be attached to the correct VM ports of the port tuples, based on the configuration of the port with the label `right`.

Service Chain with Health Check

Service chain Version 2 also allows service instance health check configuration on a per interface label. This is used to monitor the health of the service.

For more information about the service instance health check, see *Service Instance Health Checks*.

RELATED DOCUMENTATION

Service Instance Health Checks

Service Chain Route Reorigination

IN THIS SECTION

- Overview: Service Chaining in Contrail | **131**
- Route Aggregation | **133**
- Routing Policy | **141**
- Control for Route Reorigination | **152**

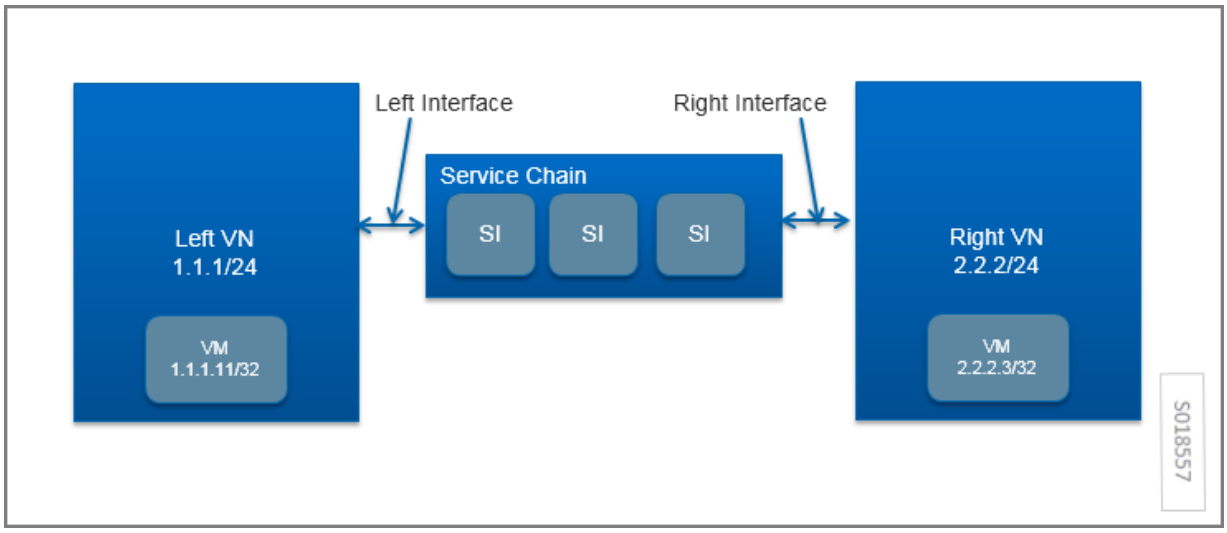
Overview: Service Chaining in Contrail

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 52 on page 132, 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 52: 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*.

Route Aggregation

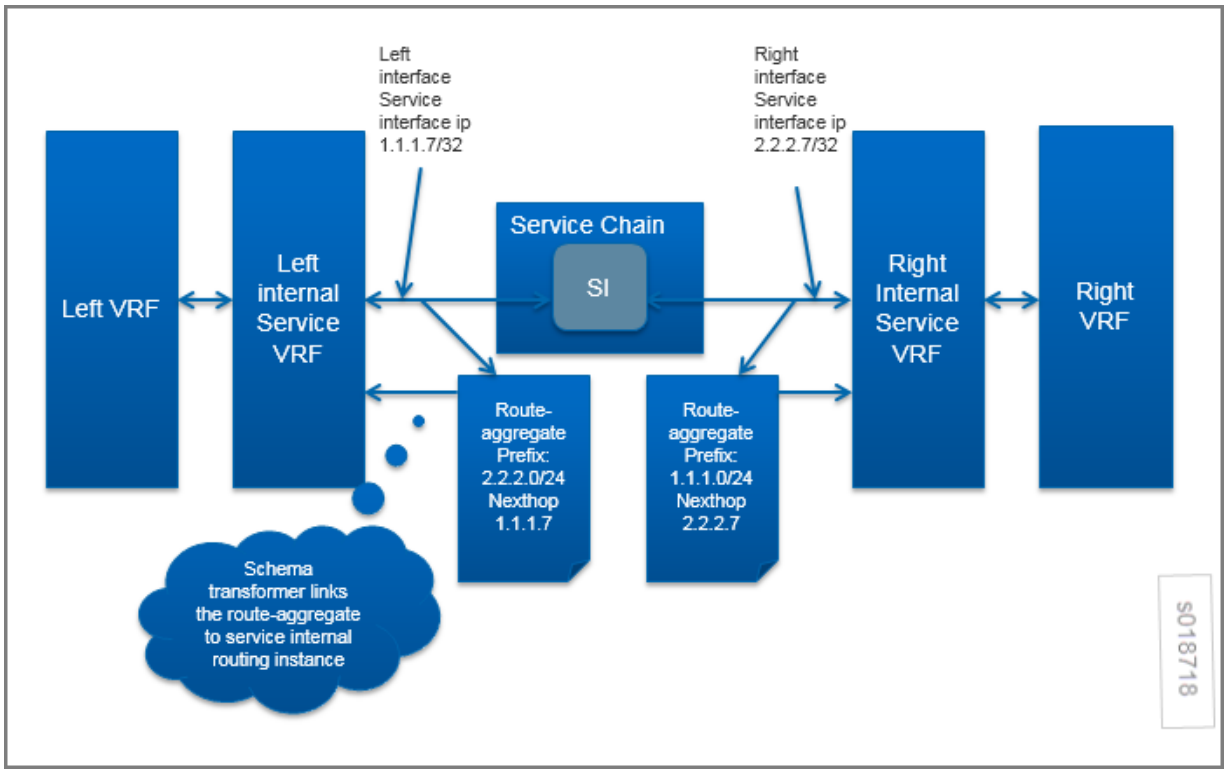
The route aggregation configuration object contains a list of prefixes to aggregate. The next-hop field in the route aggregate object contains the address of the route whose next hop is stitched as a next hop of the aggregate route.

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 53 on page 134.

Figure 53: 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.

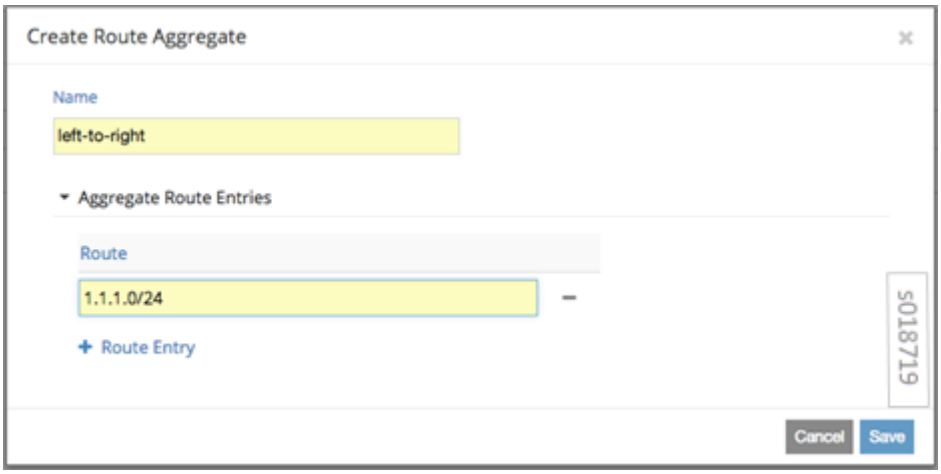
```
<xsd:element name="route-aggregate-routing-instance"/>
<!--#IFMAP-SEMANTICS-IDL
        Link('route-aggregate-routing-instance',
                  'route-aggregate', 'routing-instance', ['ref']) -->
```

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 54 on page 137.

Figure 54: Create Route Aggregate



Example VNC Script to Create a Route Aggregate Object

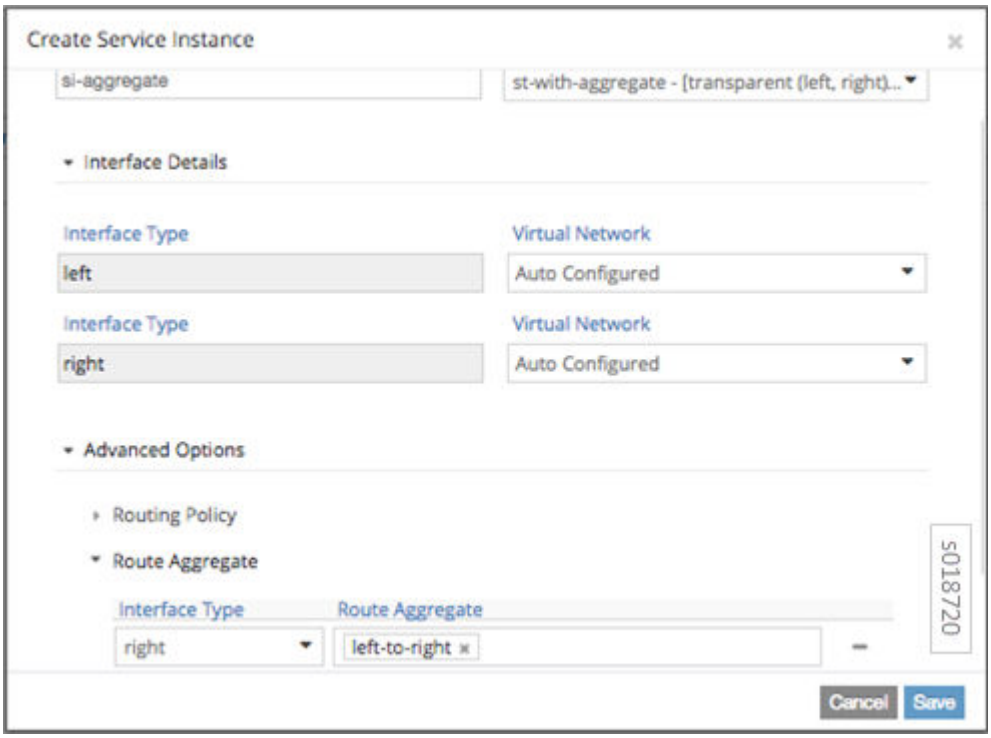
You can use a VNC script to create a route aggregate object, as in the following example:

```
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

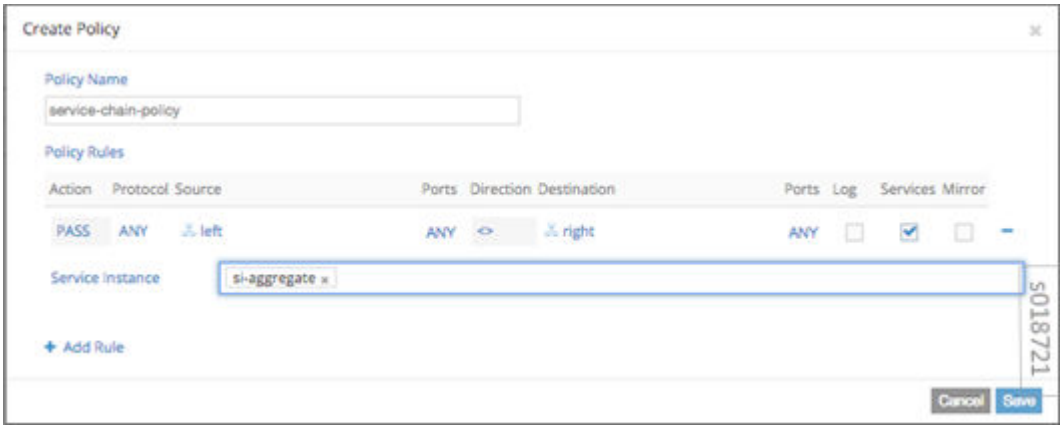
Create a service instance with the route aggregate object linked to the aggregate left network subnet prefix in the right virtual network. See the example in Figure 55 on page 138.

Figure 55: Create Service Instance

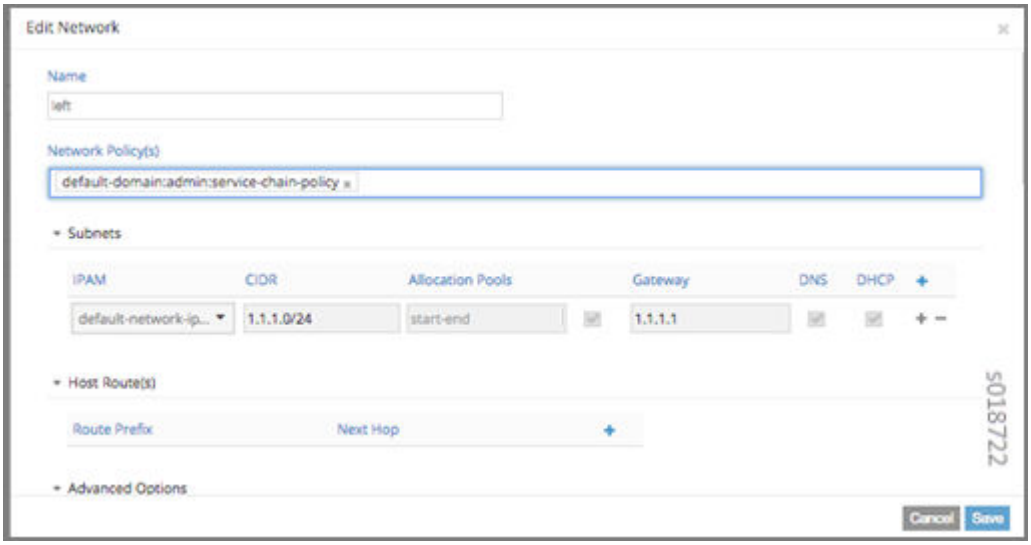


Create a Virtual Network and Network Policy

Create a left and right virtual network with the subnets 1.1.1/24 and 2.2.2/24, respectively. Create a network policy to apply a service chain between the left VN and the right VN. See the following example.



Attach the network policy to create the service chain between the left and right VNs. See the following example.



Validate the Route Aggregate Object in the API Server

Validate the route aggregate object in the API server configuration database. Verify the routing instance reference and the service instance reference for the aggregate object. The `aggregate_route_nexthop` field in the route aggregate object is initialized by the schema transformer to the service chain address. See the following example.

Validate the Route Aggregate Object in the Control Node

Validate the instance configurations of the route aggregate by checking the control node introspect for the service instance internal routing instance. For example:

http://*<control-node>*:8083/Snh_ShowBgpInstanceConfigReq?search_string=default- domain:admin:right:service-ace7ae00-56e3-42d1-96ec-7fe77088d97f-default- domain_admin_si-aggregate

See the following example.

To check the state of the route aggregate object on the control node, point your browser to:

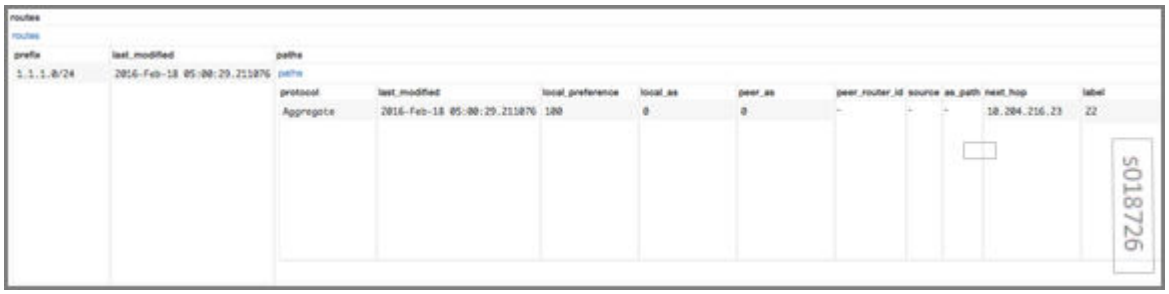
http://<*control-node*>:8083/Snh_ShowRouteAggregateReq

See the following example.

You can also check the route table for the aggregate route in the right VN BGP table. For example:

http://<*control-node*>:8083/Snh_ShowRouteReq?x=default-domain:admin:right:right.inet.0

See the following example.



Routing Policy

Contrail uses routing policy infrastructure to manipulate the route and path attribute dynamically. Contrail also supports attaching the import routing policy on the service instances.

The routing policy contains list terms. A term can be a terminal rule, meaning that upon a match on the specified term, no further terms are evaluated and the route is dropped or accepted, based on the action in that term.

If the term is not a terminal rule, subsequent terms are evaluated for the given route.

The list terms are structured as in the following example.

```
Policy {
     Term-1
```

```
     Term-2
}
```

The matches and actions of the policy term lists operate similarly to the Junos language match and actions operations.

Each term is represented as in the following:

```
from {
    match-condition-1
    match-condition-2
    ..
    ..
}
then {
    action
    update-action-1
    update-action-2
    ..
    ..
}
```

The term should not contain an `any` match condition, for example, an empty `from` should not be present.

If an `any` match condition is present, all routes are considered as matching the term.

However, the `then` condition can be empty or the action can be unspecified.

Applying Routing Policy

The routing policy evaluation has the following key points:

- If the term of a routing policy consists of multiple match conditions, a route must satisfy all match conditions to apply the action specified in the term.
- If a term in the policy does not specify a match condition, all routes are evaluated against the match.
- If a match occurs but the policy does not specify an accept, reject, or next term action, one of the following occurs:
  - The next term, if present, is evaluated.
  - If no other terms are present, the next policy is evaluated.
  - If no other policies are present, the route is accepted. The default routing policy action is “accept”.

- If a match does not occur with a term in a policy, and subsequent terms in the same policy exist, the next term is evaluated.
- If a match does not occur with any terms in a policy, and subsequent policies exist, the next policy is evaluated.
- If a match does not occur by the end of a policy or all policies, the route is accepted.

A routing policy can consist of multiple terms. Each term consists of match conditions and actions to apply to matching routes.

Each route is evaluated against the policy as follows:

1. The route is evaluated against the first term. If it matches, the specified action is taken. If the action is to accept or reject the route, that action is taken and the evaluation of the route ends. If the next term action is specified or if no action is specified, or if the route does not match, the evaluation continues as described above to subsequent terms.
2. Upon hitting the last non-terminal term of the given routing policy, the route is evaluated against the next policy, if present, in the same manner as described in step 1.

Match Condition: From

The match condition `from` contains a list of match conditions to be satisfied for applying the action specified in the term. It is possible that the term doesn't have any match condition. This indicates that all routes match this term and action is applied according to the action specified in the term.

The following table describes the match conditions supported by Contrail.

Match Condition	User Input	Description
Prefix	List of prefixes to match	Each prefix in the list is represented as prefix and match type, where the prefix match type can be: • `exact` • `orlonger` • `longer` Example: 1.1.0.0/16 `orlonger` A route matches this condition if its prefix matches any of the prefixes in the list.

(Continued)

Match Condition	User Input	Description
Community	Community string to match	Represented as either a well-known community string with `no export` or `no reoriginate`, or a string representation of a community (64512:11).
Protocol	Array of path source or path protocol to match	BGP | XMPP | StaticRoute | ServiceChain | Aggregate. A path is considered as matching this condition if the path protocol is one of protocols in the list.

Routing Policy Action and Update Action

The policy action contains two parts, action and update action.

The following table describes `action` as supported by Contrail.

Action	Terminal?	Description
Reject	Yes	Reject the route that matches this term. No more terms are evaluated after hitting this term.
Accept	Yes	Accept the route that matches this term. No more terms are evaluated after hitting this term. The route is updated using the update specified in the policy action.
Next Term	No	This is the default action taken upon matching the policy term. The route is updated according to the update specified in the policy action. Next terms present in the routing policy are processed on the route. If there are no more terms in the policy, the next routing policy is processed, if present.

The update action section specifies the route modification to be performed on the matching route.

The following table describes `update action` as supported by Contrail.

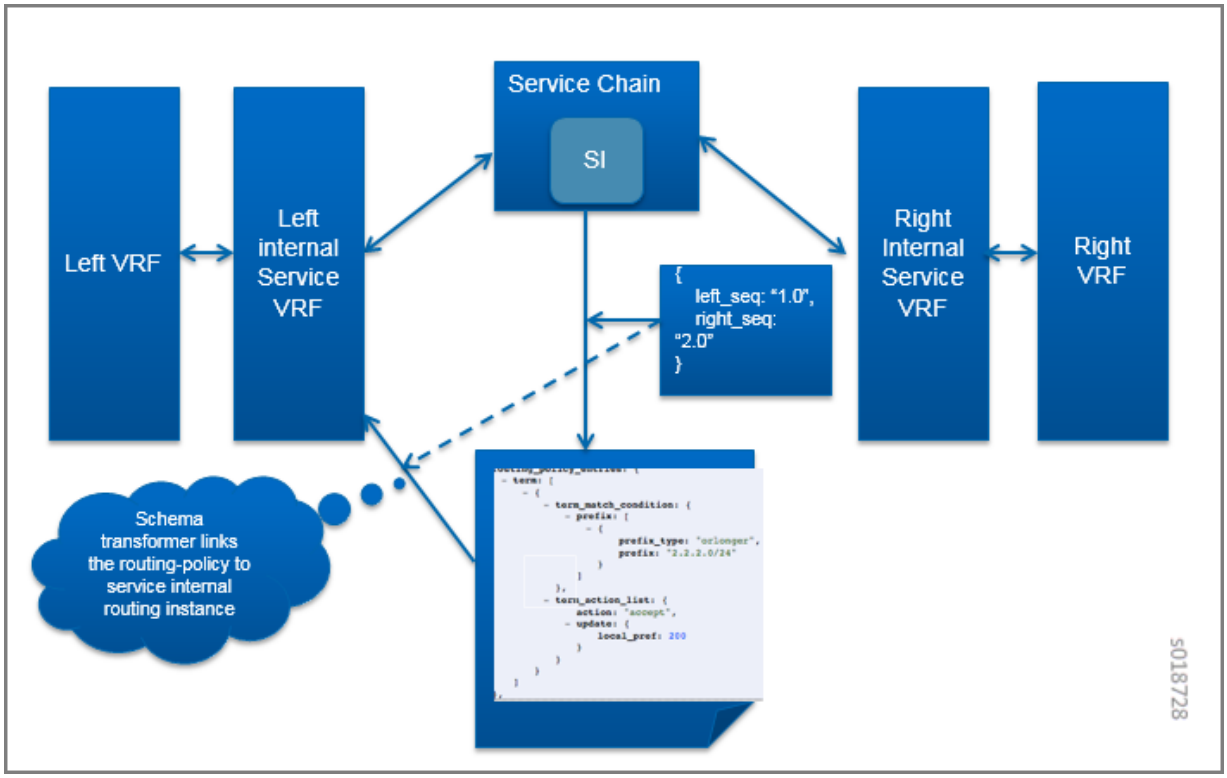
Update Action	User Input	Description
Community	List of community	As part of the policy update, the following actions can be taken for community: • Add a list of community to the existing community. • Set a list of community. • Remove a list of community (if present) from the existing community.
MED	Update the MED of the BgpPath	Unsigned integer representing the MED
local-pref	Update the local-pref of the BgpPath	Unsigned integer representing local-pref

Routing Policy Configuration

Routing policy is configured on the service instance. Multiple routing policies can be attached to a single service instance interface.

When the policy is applied on the left interface, the policy is evaluated for all the routes that are reoriginated in the left VN for routes belonging to the right VN. Similarly, the routing policy attached to the right interface influences the route reorigination in the right VN, for routes belonging to the left VN.

The following figure illustrates a routing policy configuration.



The policy sequence number specified in the routing policy link data determines the order in which the routing policy is evaluated. The routing policy link data on the service instance also specifies whether the policy needs to be applied to the left service interface, to the right service interface, or to both interfaces.

It is possible to attach the same routing policy to both the left and right interfaces for a service instance, in a different order of policy evaluation. Consequently, the routing policy link data contains the sequence number for policy evaluation separately for the left and right interfaces.

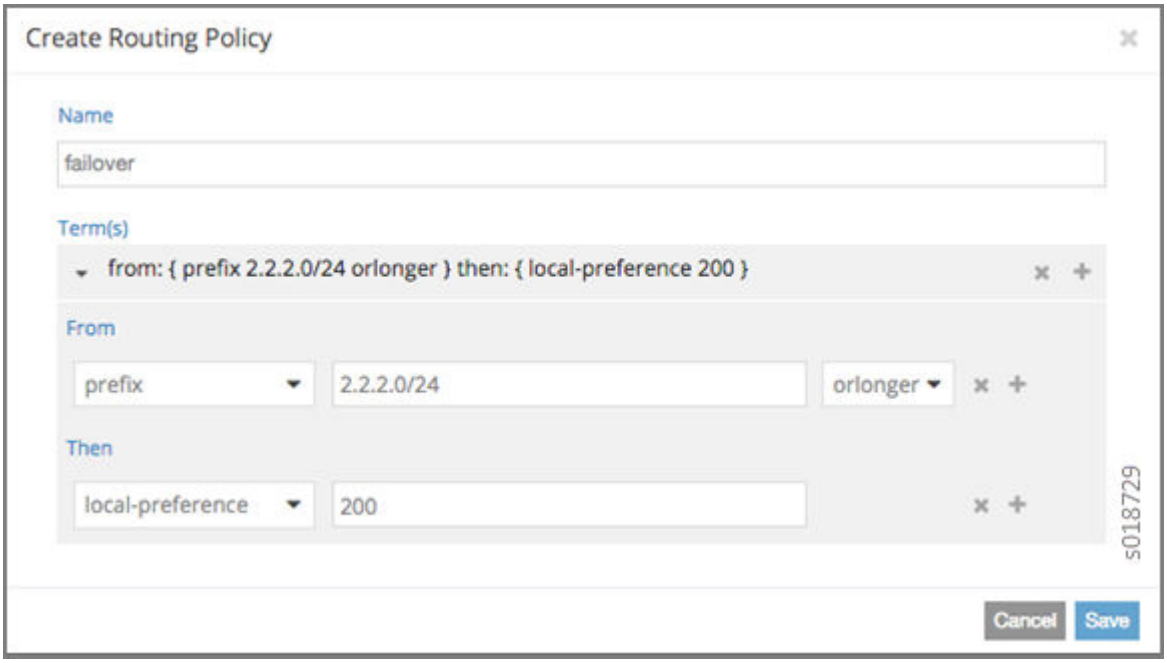
The schema transformer links the routing policy object to the internal routing instance created for the service instance. The transformer also copies the routing policy link data to ensure the same policy order.

Configuring and Troubleshooting Routing Policy

This section shows how to create a routing policy for service chains and how to validate the policy.

Create Routing Policy

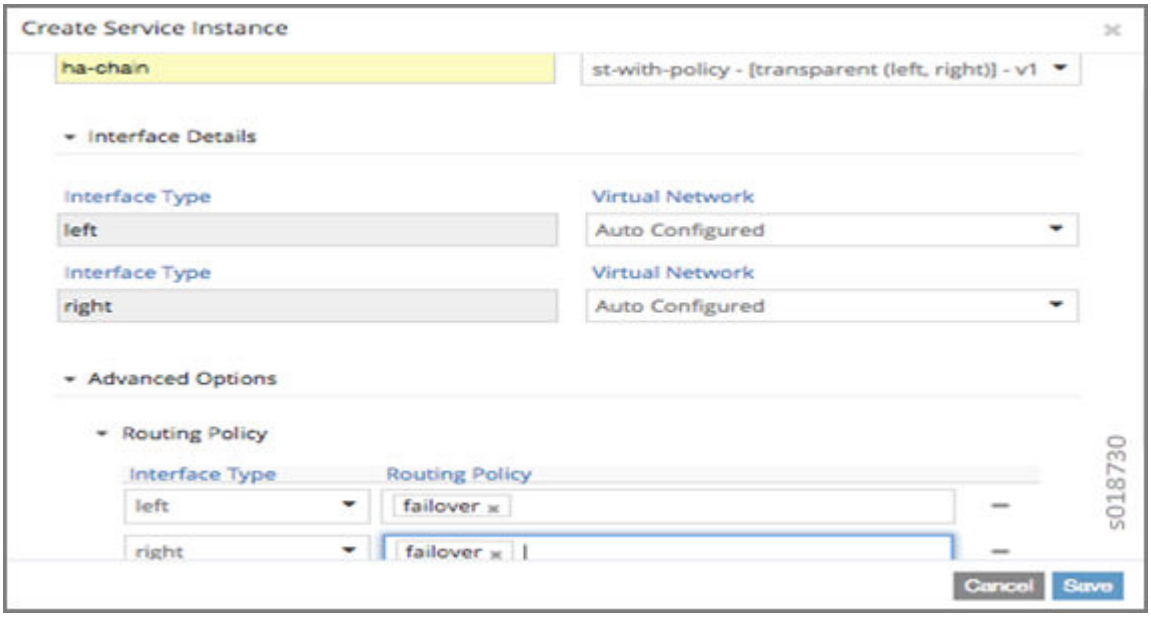
First, create the routing policy, **Configure > Networking > Routing > Create >Routing Policy**. See the following example.



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.

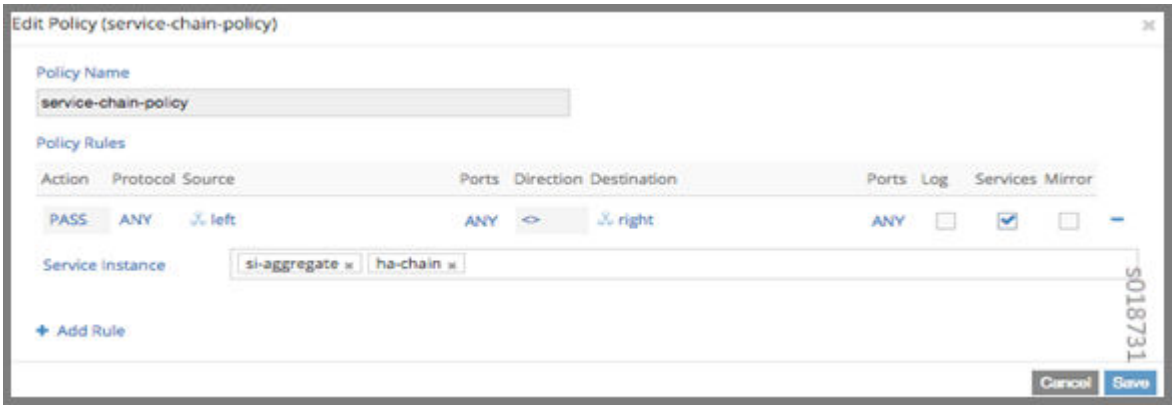
Configure Service Instance

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.



Configure the Network Policy for the Service Chain

At **Edit Policy**, create a policy for the service chain, see the following example.



Using a VNC Script to Create Routing Policy

The following example shows use of a VNC API script to create a routing policy.

```
from vnc_api.vnc_api import *
vnc_lib = VncApi("admin", "<password>", "admin")
project=vnc_lib.project_read(fq_name=["default-domain", "admin"])
routing_policy=RoutingPolicy(name="vnc_3", parent_obj=project)
policy_term=PolicyTermType()
policy_statement=PolicyStatementType()
```

```
match_condition=TermMatchConditionType(protocol=["bgp"], community="22:33")
prefix_match=PrefixMatchType(prefix="1.1.1.0/24", prefix_type="orlonger")
match_condition.set_prefix([prefix_match])

term_action=TermActionListType(action="accept")
action_update=ActionUpdateType(local_pref=101, med=10)
add_community=ActionCommunityType()
comm_list=CommunityListType(["11:22"])
add_community.set_add(comm_list)
action_update.set_community(add_community)
term_action.set_update(action_update)

policy_term.set_term_action_list(term_action)
policy_term.set_term_match_condition(match_condition)

policy_statement.add_term(policy_term)
routing_policy.set_routing_policy_entries(policy_statement)
vnc_lib.routing_policy_create(routing_policy)
```

Verify Routing Policy in API Server

You can verify the service instance references and the routing instance references for the routing policy by looking in the API server configuration database. See the following example.

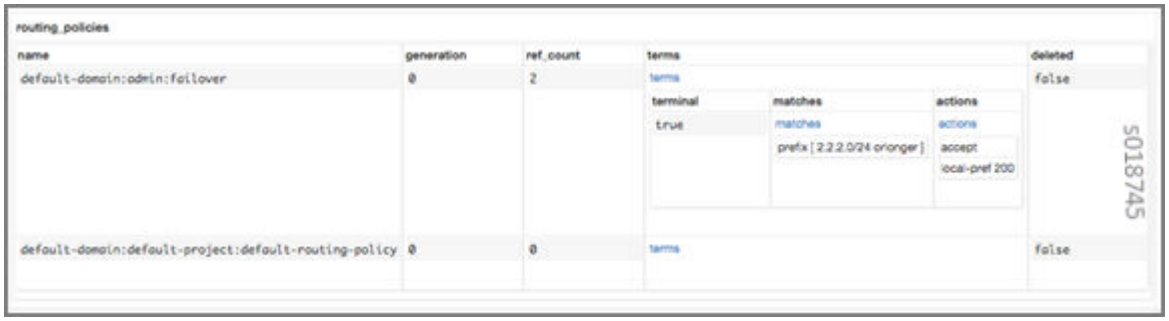
Verify Routing Policy in the Control Node

You can verify the routing policy in the control node.

Point your browser to:

`http://<control-node>:8083/Snh_ShowRoutingPolicyReq?search_string=failover`

See the following example.



Verify Routing Policy Configuration in the Control Node

You can verify the routing policy configuration in the control node.

Point your browser to:

`http://<control-node>:8083/Snh_ShowBgpRoutingPolicyConfigReq?search_string=failover`

See the following example.

Verify Routing Policy Configuration on the Routing Instance

You can verify the routing policy configuration on the internal routing instance.

Point your browser to:

`http://<control-node>:8083/Snh_ShowBgpInstanceConfigReq?search_string=<name-of-internal-vrf>`

See the following example.

You can also verify the routing policy on the routing instance operational object.

Point your browser to:

`http://<control-node>:8083/Snh_ShowRoutingInstanceReq?x=<name-of-internal-vrf>`

See the following example.

Control for Route Reorigination

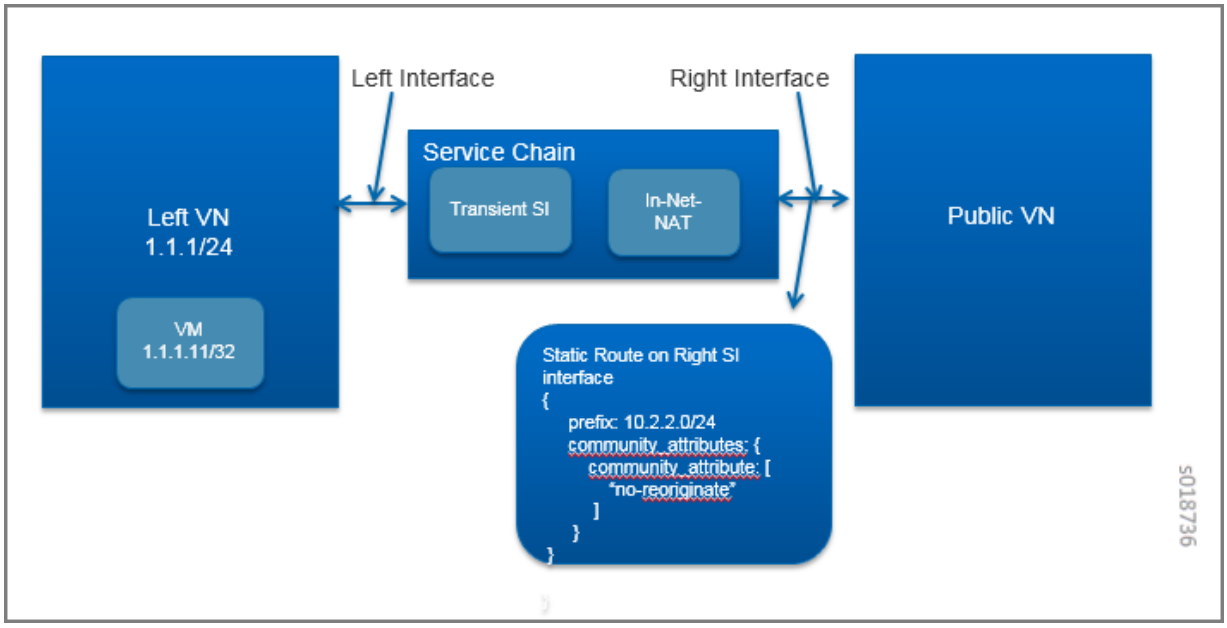
The ability to prevent reorigination of interface static routes is typically required when routes are configured on an interface that belongs to a service VM.

As an example, the following image shows a service chain that has multiple service instances, with an `in-net-nat` service instance as the last service VM, also with the right VN as the public VN.

The last service instance performs NAT by using a NAT pool. The right interface of the service VM must be configured with an interface static route for the NAT pool so that the destination in the right VN knows how to reach addresses in the NAT pool. However, the NAT pool prefix should not be reoriginated into the left VN.

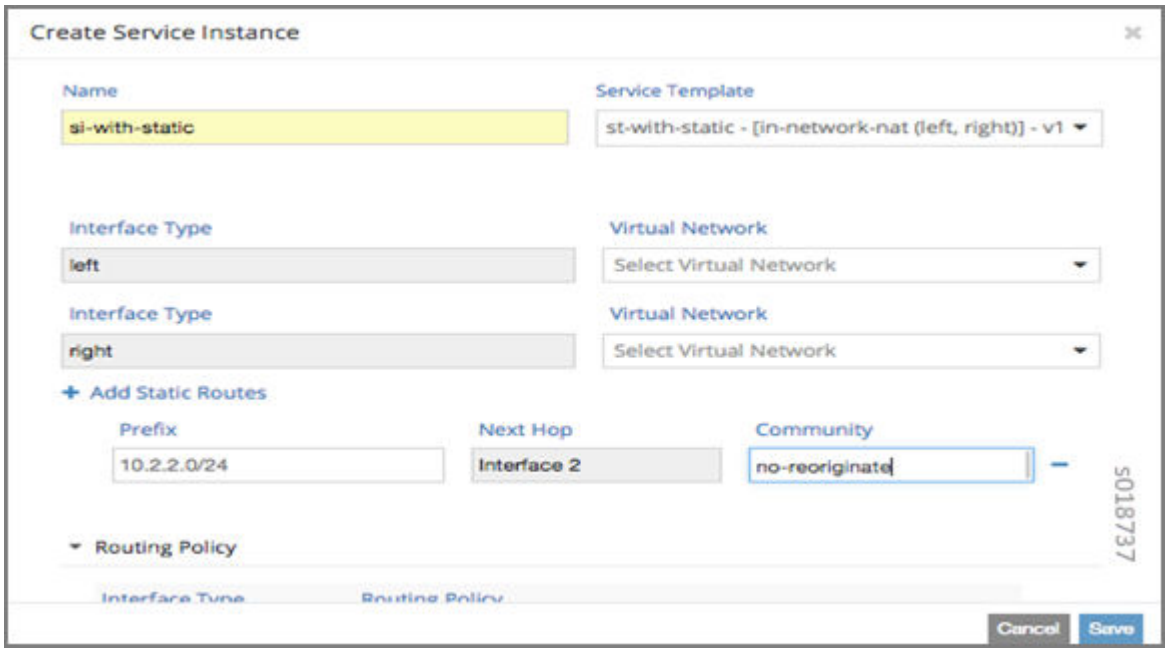
To prevent route reorigination, the interface static route is tagged with a well-known BGP community called `no-reoriginate`.

When the control node is reoriginating the route, it skips the routes that are tagged with the BGP community.



Configuring and Troubleshooting Reorigination Control

The community attribute on the static routes for the interface static route of the service instance is specified during creation of the service instance. See the following example.



Use the following example to verify that the service instance configuration object in the API server has the correct community set for the static route. See the following example.

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

IN THIS SECTION

- Hardware and Software Requirements | 155
- Overview | 155
- Configuration | 155

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 120.

Configuration

IN THIS SECTION

Create Virtual Network | 156

- Create Virtual Machine | **157**
- Configure Service Template | **158**
- Add Service Instance | **159**
- Create Service Policy | **160**
- Attach Service Policy | **161**
- Launch Virtual Machine | **162**

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 156 through "7" on page 157 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 157 through "11" on page 158 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 157.
2. Launch the right virtual machine in right virtual network. See "Create Virtual Machine" on page 157.
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**.

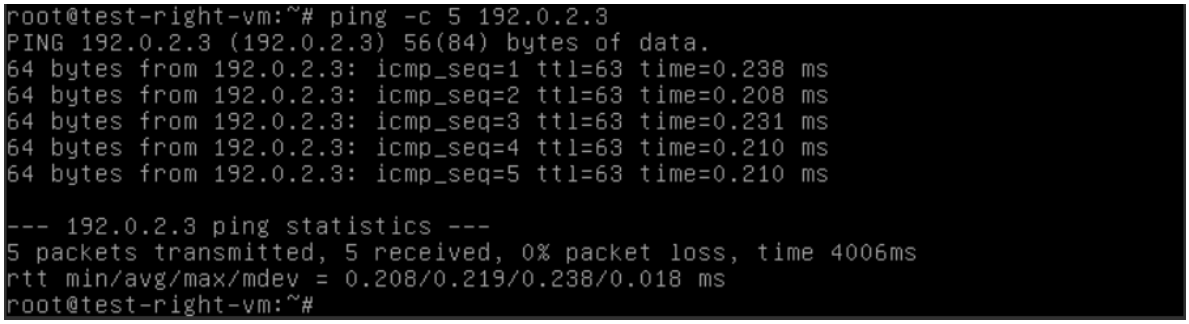
      The Console page is displayed.

   c. Log in using `root` as user name and `c0ntrail123` as password.

   d. Ping the left virtual machine IP address (**190.0.2.3**) from the Console.

      See Figure 56 on page 162 for a sample output.

Figure 56: Ping test-left-VM



RELATED DOCUMENTATION

Service Chaining | 120

Example: Creating an In-Network-NAT Service Chain | 163

Example: Creating a Transparent Service Chain by Using Contrail Command | 172

Use Case: Configuring Fat Flows from Contrail Command

Example: Creating an In-Network-NAT Service Chain

IN THIS SECTION

- Prerequisites | 163
- Overview | 164
- Configuration | 165

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 120.

Configuration

IN THIS SECTION

- Create Virtual Network | 165
- Create Virtual Machine | 166
- Configure Service Template | 167
- Add Service Instance | 168
- Create Service Policy | 169
- Attach Service Policy | 170
- Launch Virtual Machine | 171

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 165 through "6" on page 166 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 166 through "11" on page 167 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 166.
2. Launch the right virtual machine in right virtual network. See "Create Virtual Machine" on page 166.
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 | **120**

Example: Creating an In-Network Service Chain by Using Contrail Command | **154**

Example: Creating a Transparent Service Chain by Using Contrail Command | **172**

Example: Creating a Transparent Service Chain by Using Contrail Command

IN THIS SECTION

- Prerequisites | 172
- Overview | 173
- Configuration | 173

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 120.

Configuration

IN THIS SECTION

- Create Primary Virtual Networks | **174**
- Create Secondary Virtual Network | **175**
- Create Service Virtual Machine | **175**
- Create Virtual Machine | **176**
- Configure Service Template | **177**
- Add Service Instance | **178**
- Create Service Policy | **179**
- Attach Service Policy | **180**

- Launch Virtual Machine | 181

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 174 through "6" on page 174 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 174.

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 176 through "11" on page 177 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 175.
   - **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 180 through "4" on page 180 to attach the service policy to **test-right-VN**.

Launch Virtual Machine

RELATED DOCUMENTATION

Service Chaining | 120

Example: Creating an In-Network-NAT Service Chain | 163

Example: Creating an In-Network Service Chain by Using Contrail Command | 154

Using Static Routes with Services

IN THIS SECTION

- Static Routes for Service Instances | 181
- Configuring Static Routes on a Service Instance | 181
- Configuring Static Routes on Service Instance Interfaces | 183
- Configuring Static Routes as Host Routes | 184

Static Routes for Service Instances

Static routes can be configured in a virtual network to direct traffic to a service virtual machine.

The following figure shows a virtual network with subnet 10.1.1.0/24. All of the traffic from a virtual machine that is directed to subnet 11.1.1.0/24 can be configured to be routed by means of a service machine, by using the static route 11.1.1.252 configured on the service virtual machine interface.

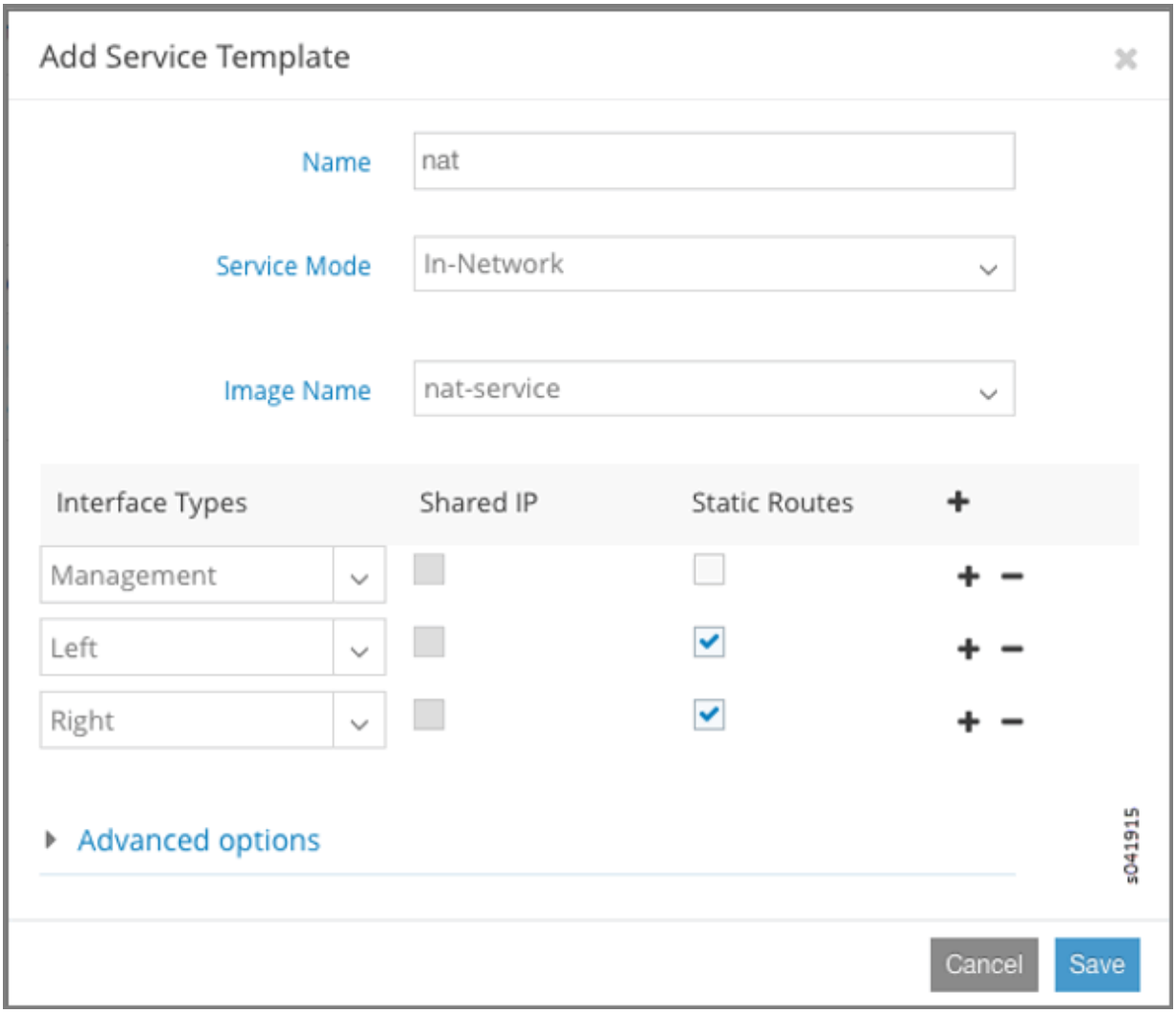
Configuring Static Routes on a Service Instance

To configure static routes on a service instance, first enable the static route option in the service template to be used for the service instance.

To enable the static route option in a service template:

1. Go to **Configure > Services > Service Templates** and click **Create**.
2. At **Add Service Template**, complete the fields for **Name**, **Service Mode**, and **Image Name**.
3. Select the **Interface Types** to use for the template, then for each interface type that might have a static route configured, click the check box under the **Static Routes** column to enable the static route option for that interface.

   The following figure shows a service template in which the left and right interfaces of service instances have the static routes option enabled. Now a user can configure a static route on a corresponding interface on a service instance that is based on the service template shown.



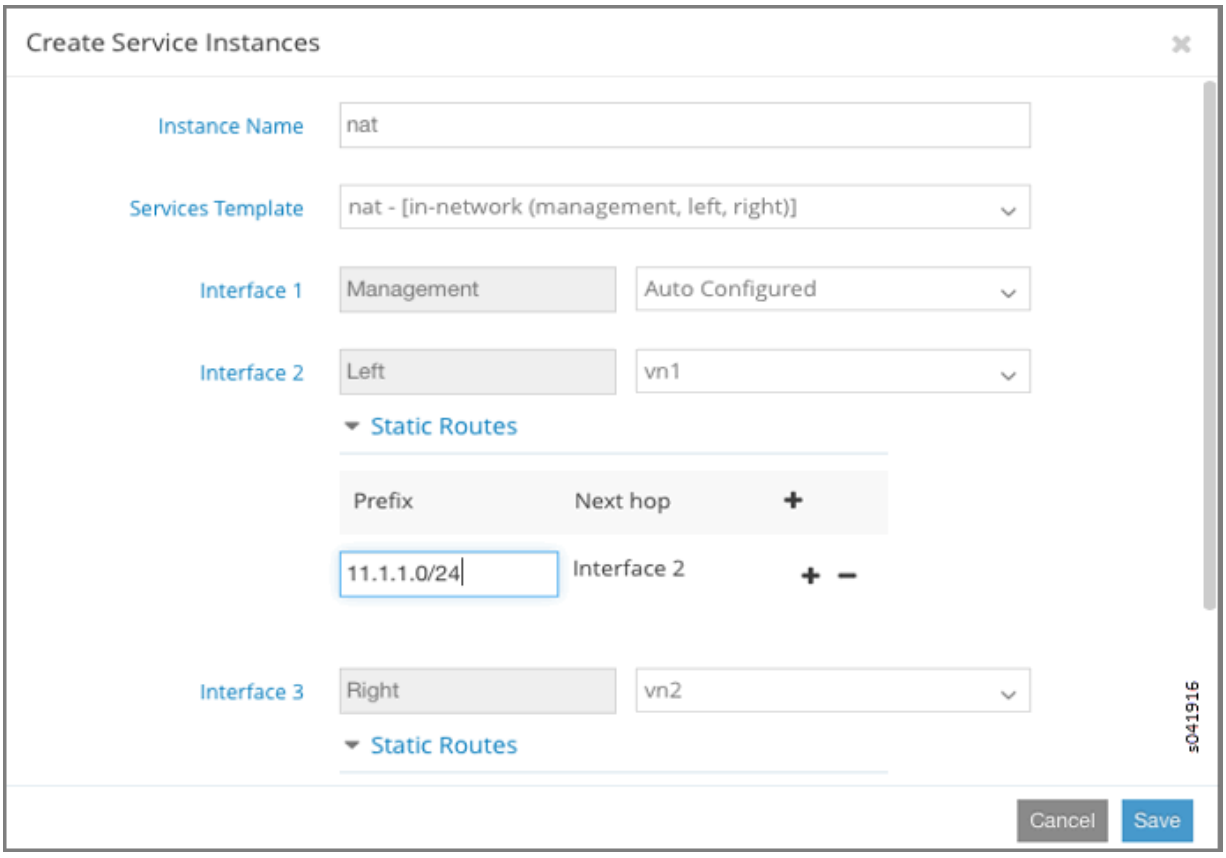
Configuring Static Routes on Service Instance Interfaces

To configure static routes on a service instance interface:

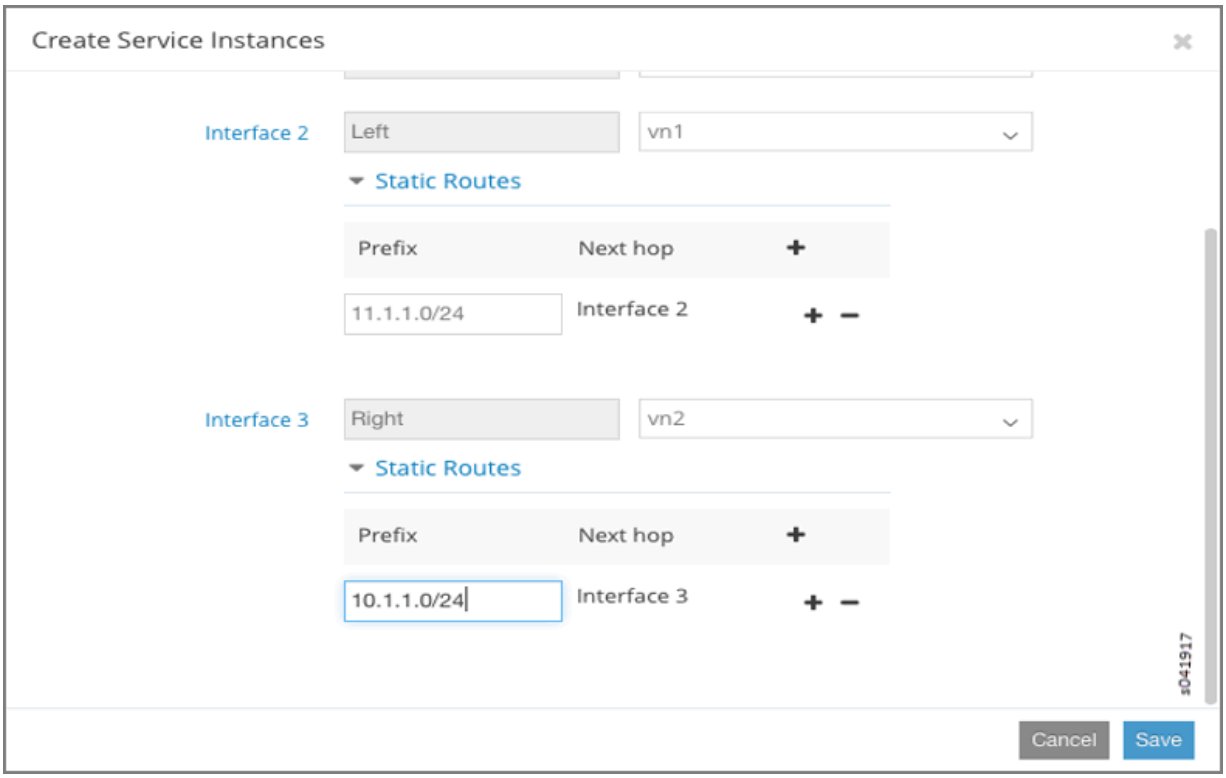
1. Go to **Configure > Services > Service Instances** and click **Create**.
2. At **Create Service Instances**, complete the fields for **Instance Name** and **Services Template**.
3. Select the virtual network for each of the interfaces
4. Click the **Static Routes** dropdown menu under each interface field for which the static routes option is enabled to open the **Static Routes** menu and configure the static routes in the fields provided.

> **NOTE**: If the **Auto Configured** option is selected, traffic destined to the static route subnet is load balanced across service instances.

The following figure shows a configuration to apply a service instance between VN1 (10.1.1.0/24) and VN2 (11.1.1.0/24). The left interface of the service instance is configured with VN1 and the right interface is configured to be VN2 (11.1.1.0/24). The static route 11.1.1.0/24 is configured on the left interface, so that all traffic from VN1 that is destined to VN2 reaches the left interface of the service instance.



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



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

Configuring Static Routes as Host Routes

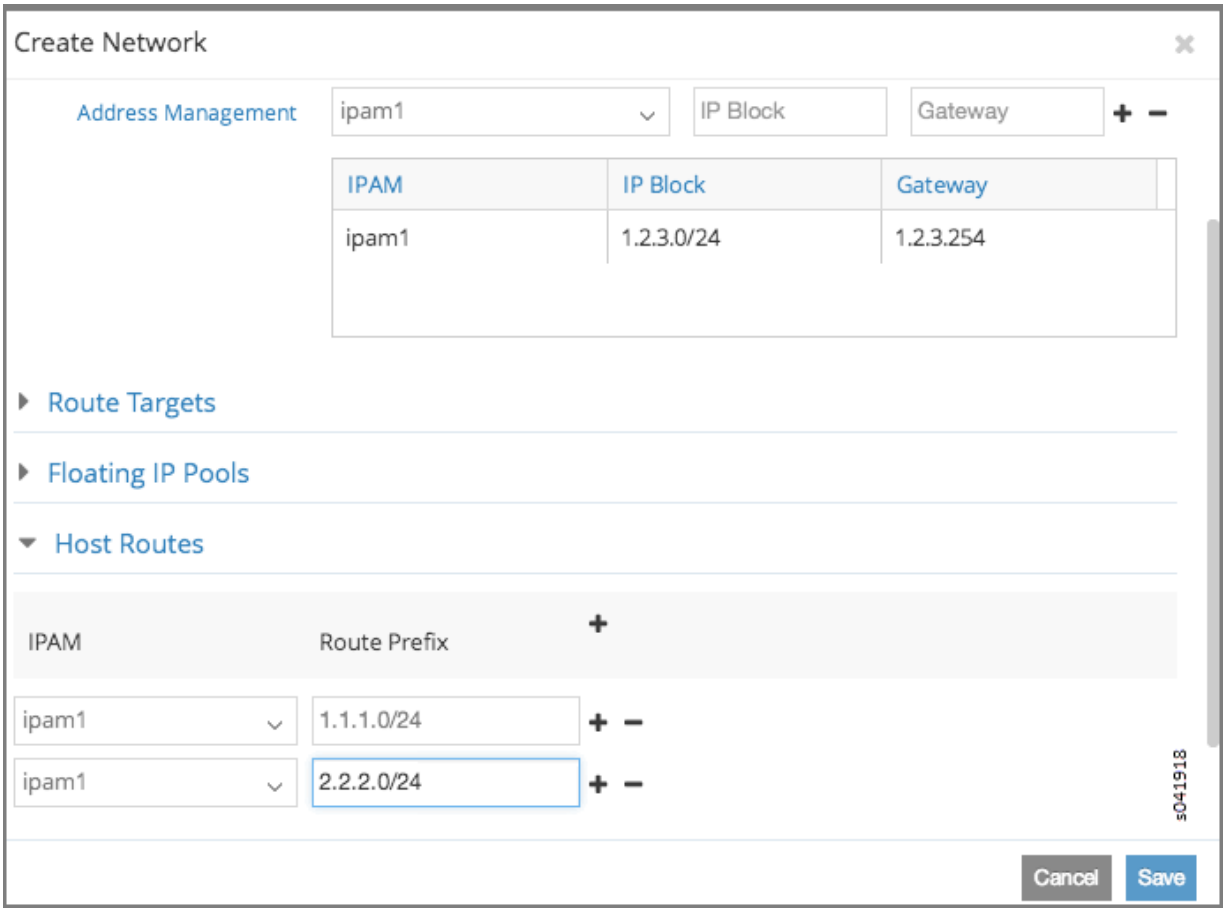
You can also use static routes for host routes for a virtual machine, by using the classless static routes option in the DHCP server response that is sent to the virtual machine.

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

To configure static routes as host routes:

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

   An example is shown in the following figure.



Configuring Metadata Service

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

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

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

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

- `linklocal-service-name = metadata`
- `linklocal-service-ip = 169.254.169.254`

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

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

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

6
CHAPTER

Optimizing Contrail Networking

Source Network Address Translation (SNAT) | 188

Source Network Address Translation (SNAT)

IN THIS SECTION

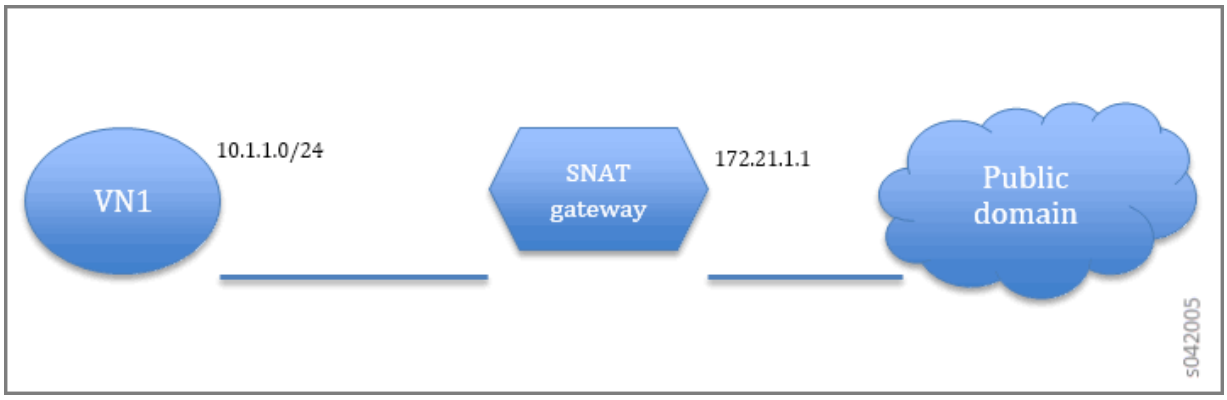
- Overview | 188
- SNAT on MX Series Routers Acting as Data Center Gateways | 189
- How to Enable SNAT on an MX Series Router Using Contrail Command | 189
- Neutron APIs for Routers | 190
- Network Namespace | 191
- SNAT and Security Groups | 192
- Using the Web UI to Configure Routers with SNAT | 192
- Using the Web UI to Configure Distributed SNAT | 194

Overview

Source Network Address Translation (source-nat or SNAT) allows traffic from a private network to go out to the internet. Virtual machines launched on a private network can get to the internet by going through a gateway capable of performing SNAT. The gateway has one arm on the public network and as part of SNAT, it replaces the source IP of the originating packet with its own public side IP. As part of SNAT, the source port is also updated so that multiple VMs can reach the public network through a single gateway public IP.

The following diagram shows a virtual network with the private subnet of 10.1.1.0/24. The default route for the virtual network points to the SNAT gateway. The gateway replaces the source-ip from 10.1.1.0/24 and uses its public address 172.21.1.1 for outgoing packets. To maintain unique NAT sessions the source port of the traffic also needs to be replaced.

Figure 57: Virtual Network With a Private Subnet



SNAT on MX Series Routers Acting as Data Center Gateways

Starting in Contrail Networking Release 2011.L1, you can enable SNAT on MX Series routers using MS-MPC line cards when the MX Series router is functioning in the *DC-Gateway* fabric role. See Contrail Networking Supported Hardware Platforms and Associated Roles And Node Profiles for a list of MX Series routers that support the *DC-Gateway* or any other fabric role.

When SNAT is enabled on the MX Series router, it can be used to translate source IP addresses from physical interfaces on bare metal servers and from virtual interfaces on virtual machines. SNAT can only translate the IP addresses of source traffic leaving the fabric; it cannot be used to translate IP addresses for traffic entering the fabric.

For additional information on SNAT on MX Series routers, see Network Address Translation Overview.

How to Enable SNAT on an MX Series Router Using Contrail Command

To enable SNAT on an MX Series Router from Contrail Command:

1. Ensure that a fabric using an MX Series router with one or more MS-MPC line cards is configured into the *DC-Gateway* fabric role in your fabric.

   See In Focus: How to Onboard a Fabric and Create an Overlay to setup a fabric.

   See Assign a Role to a Device to change the routing role of a device in a fabric.

2. Click **Infrastructure** > **Fabrics** > *fabric-name* to navigate to the devices in your fabric. Mouse over the *mx-router-name* of the router configured as a DC gateway in your fabric that will perform SNAT. Click the ellipsis (...) button—located as the last option on the far right for the router—and select **Edit**.

The **Fabric Device** page opens.

3. From the **Fabric Device** page, open **Netconf Settings**.

   In the **Junos Service Interface** field, add the services interface name—for instance, *ms-1/0/0*—from the MX Series router.

4. (BMS interfaces that require SNAT only) Create a Virtual Port Group (VPG) that maps VLANs to physical interfaces on bare metal servers (BMSs). See Configuring Virtual Port Groups.

   The VPG will be used later in the process to identify traffic that requires IP address translation using SNAT.

   This step is needed to identify source IP addresses on BMS hosts only. You can skip this step when you are using SNAT to translate source IP addresses from virtual machine interfaces.

5. Create a public logical router for SNAT. See Create Logical Routers.

   The logical router is configured in the **Overlay** > **Logical Routers** > **Edit Logical Router** menu. From this menu, include the following configuration parameters:

   - **connected networks** field: add the virtual networks that were created to carry traffic.

     The traffic in these virtual networks will be translated using SNAT.

   - **Public Logical Router** checkbox: Select the checkbox.

     The **SNAT POOL** drop-down menu appears. Select *snat_pool*.

   - **Extend to Physical Router** field: add the MX Series router in the fabric where source-based IP address translation is performed.

6. To monitor SNAT after completing the configuration, log onto the MX Series router and enter the following JUNOS commands:

   - **show configuration** to verify NAT configuration in JUNOS.
   - **show services nat pool** to verify translation.
   - Monitor system messages.

   For additional information on using and monitoring NAT in Junos, see the Network Address Translation User Guide.

Neutron APIs for Routers

OpenStack supports SNAT gateway implementation through its Neutron APIs for routers. The SNAT flag can be enabled or disabled on the external gateway of the router. The default is True (enabled).

The Tungsten Fabric plugin supports the Neutron APIs for routers and creates the relevant service-template and service-instance objects in the API server. The service scheduler in Tungsten Fabric instantiates the gateway on a randomly-selected virtual router. Tungsten Fabric uses network namespace to support this feature.

Example Configuration: SNAT for Contrail

The SNAT feature is enabled on Tungsten Fabric through Neutron API calls.

The following configuration example shows how to create a test network and a public network, allowing the test network to reach the public domain through the SNAT gateway.

1. Create the public network and set the router external flag.

```
neutron net-create public

neutron subnet-create public 172.21.1.0/24

neutron net-update public -- --router:external=True
```

2. Create the test network.

```
neutron net-create test

neutron subnet-create --name test-subnet test 10.1.1.0/24
```

3. Create the router with one interface in test.

```
neutron router-create r1

neutron router-interface-add r1 test-subnet
```

4. Set the external gateway for the router.

```
neutron router-gateway-set r1 public
```

Network Namespace

Setting the external gateway is the trigger for Tungsten Fabric to set up the Linux network namespace for SNAT.

The network namespace can be cleared by issuing the following Neutron command:

```
neutron router-gateway-clear r1
```

SNAT and Security Groups

When a logical router is enabled to support SNAT, the default security group is automatically applied to the left SNAT interface. This automatic application of the default security group allows the virtual machine to send and receive traffic without additional user configuration when the default security group is used by interconnected virtual machines. Additional configuration is required to send and receive traffic, however, when your virtual machine is connected to virtual machines that are not using the default security group.

If you are connecting your virtual machine to a virtual machine that is not using the default security group, you must make one of the following configuration updates to allow your virtual machine to pass traffic:

- update the default security group to add rules that allow the VM traffic.
- update the rules to the VM security group to allow traffic from the default security group.
- apply the same security group to the VM and the SNAT left interface.

For information on configuring security groups in environments using Contrail Networking, see Using Security Groups with Virtual Machines Instances.

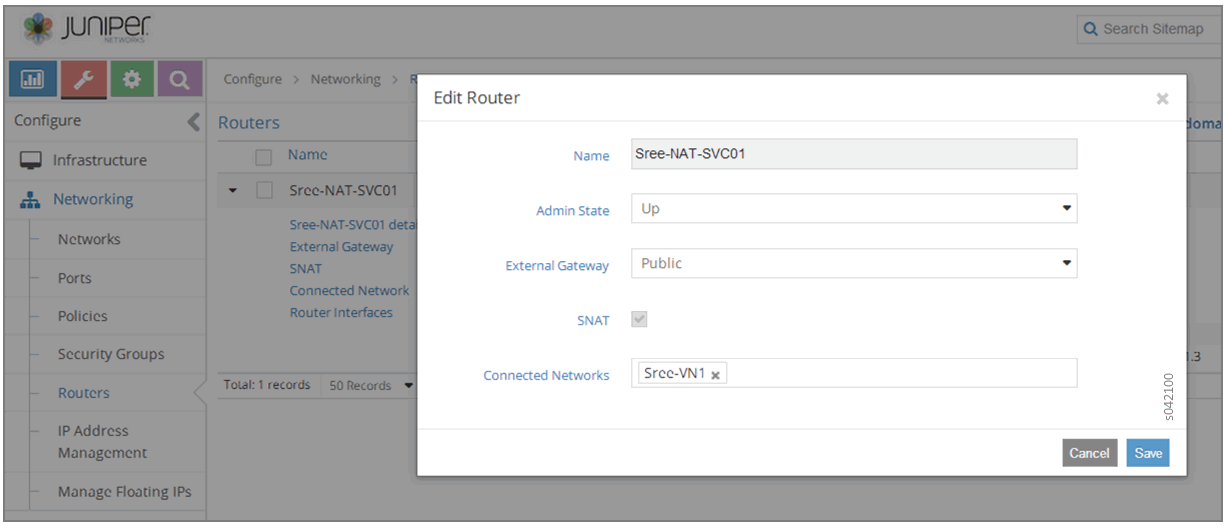
Using the Web UI to Configure Routers with SNAT

You can use the Contrail user interface to configure routers for SNAT and to check the SNAT status of routers.

To enable SNAT for a router, go to **Configure > Networking > Routers**. In the list of routers, select the router for which SNAT should be enabled. Click the Edit cog to reveal the **Edit Routers** window. Click the check box for SNAT to enable SNAT on the router.

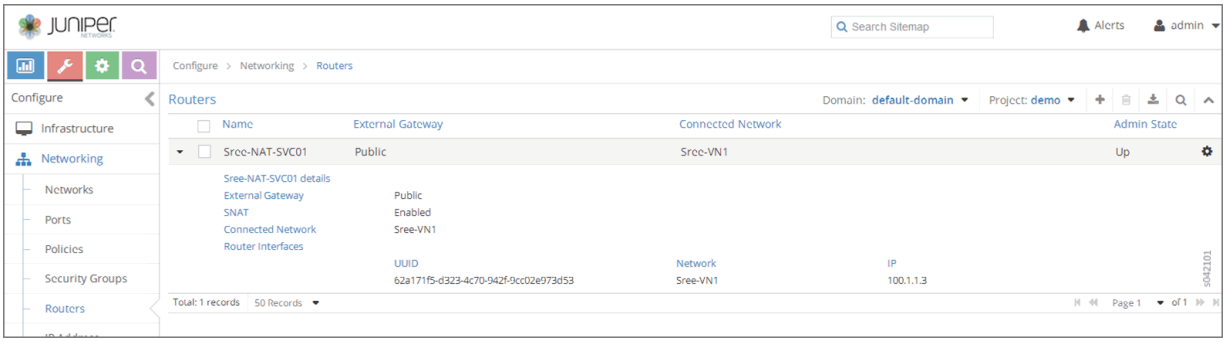
The following shows a router for which SNAT has been **Enabled**.

Figure 58: Edit Router Window to Enable SNAT



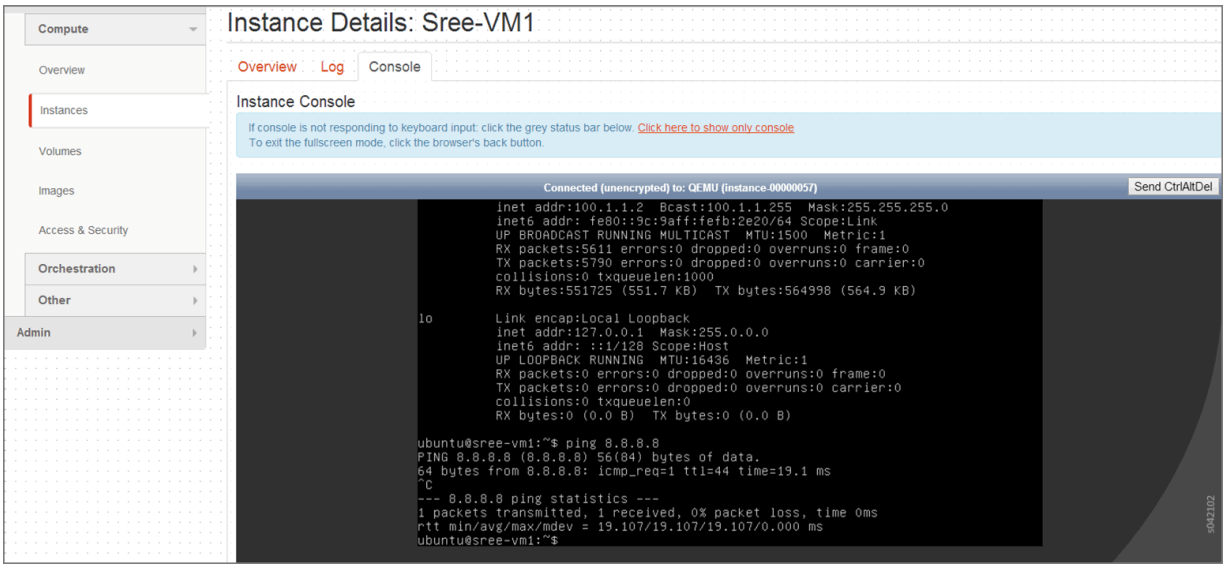
When a router has been **Enabled** for SNAT, the configuration can be seen by selecting **Configure > Networking > Routers**. In the list of routers, click open the router of interest. In the list of features for that router, the status of SNAT is listed. The following shows a router that has been opened in the list. The status of the router shows that SNAT is **Enabled**.

Figure 59: Router Status for SNAT



You can view the real time status of a router with SNAT by viewing the instance console, as in the following.

Figure 60: Instance Details Window



Using the Web UI to Configure Distributed SNAT

The distributed SNAT feature allows virtual machines to communicate with the IP fabric network using the existing forwarding infrastructure for compute node connectivity. This functionality is achieved through port address translation of virtual machine traffic using the IP address of the compute node as the public address.

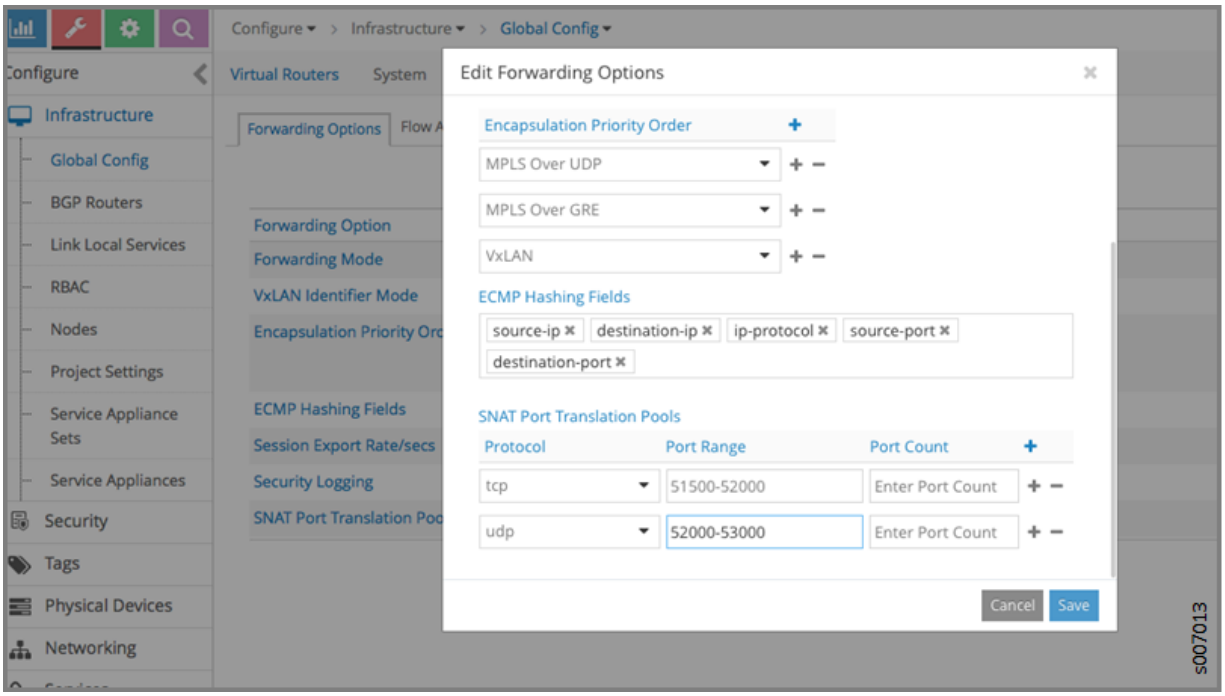
The following distributed SNAT use case is supported:

- Virtual networks with distributed SNAT enabled can communicate with the IP fabric network. The session must be initiated from a virtual machine. Sessions initiated from the external network are not supported.

Distributed SNAT is supported only for TCP and UDP, and you can configure discrete port ranges for both protocols.

A pool of ports is used for distributed SNAT. To create a pool of ports, go to **Configure > Infrastructure > Global Config**. The following shows an example of a port range used for port address translation.

Figure 61: Edit Forwarding Options Window



To use distributed SNAT, you must enable SNAT on the virtual network. To enable SNAT on the virtual network, go to **Configure > Networking > Networks**. The following shows a virtual network for which SNAT has been enabled under Advanced Options.

Figure 62: Create Window