

Contrail™

Contrail Service Provider Focused Features Guide

Published
2020-12-02

Release
5.1

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

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

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

Contrail™ Contrail Service Provider Focused Features Guide

5.1

Copyright © 2020 Juniper Networks, Inc. All rights reserved.

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

YEAR 2000 NOTICE

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

END USER LICENSE AGREEMENT

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

Table of Contents

About the Documentation | vii

Documentation and Release Notes | vii

Documentation Conventions | vii

Documentation Feedback | x

Requesting Technical Support | x

Self-Help Online Tools and Resources | xi

Creating a Service Request with JTAC | xi

1

Data Plane Optimization

Configuring the Data Plane Development Kit (DPDK) Integrated with Contrail vRouter | 13

DPDK Support in Contrail | 13

Preparing the Environment File for Provisioning a Cluster Node with DPDK | 13

Creating a Flavor for DPDK | 15

Configuring Single Root I/O Virtualization (SR-IOV) | 16

Overview: Configuring SR-IOV | 16

Enabling ASPM in BIOS | 16

Configuring SR-IOV Using the Ansible Deployer | 17

Configuring SR-IOV Using Helm | 18

Launching SR-IOV Virtual Machines | 21

Using the Contrail UI to Enable and Launch an SR-IOV Virtual Machine | 21

Using the CLI to Enable and Launch SR-IOV Virtual Machines | 22

2

Advanced Network Topologies

Configuring Virtual Networks for Hub-and-Spoke Topology | 25

Route Targets for Virtual Networks in Hub-and-Spoke Topology | 25

Example: Hub-and-Spoke Topology | 26

Troubleshooting Hub-and-Spoke Topology | 27

Remote Compute | 31

Remote Compute Overview | 31

Remote Compute Features | 32

Remote Compute Operations | 32

Subcluster Properties | 33

Provisioning a Remote Compute Cluster | 33

Advanced Service Chain Configuration

Customized Hash Field Selection for ECMP Load Balancing | 42

Overview: Custom Hash Feature | 42

Using ECMP Hash Fields Selection | 44

Configuring ECMP Hash Fields Over Service Chains | 44

Routing Policy | 45

Applying Routing Policy | 46

Match Condition: From | 47

Routing Policy Action and Update Action | 48

Routing Policy Configuration | 49

Configuring and Troubleshooting Routing Policy | 50

Create Routing Policy | 51

Configure Service Instance | 52

Configure the Network Policy for the Service Chain | 52

Using a VNC Script to Create Routing Policy | 52

Verify Routing Policy in API Server | 54

Verify Routing Policy in the Control Node | 55

Verify Routing Policy Configuration in the Control Node | 55

Verify Routing Policy Configuration on the Routing Instance | 56

Control for Route Reorigination | 56

Configuring and Troubleshooting Reorigination Control | 58

Creating a Routing Policy With Extended Communities in Contrail Command | 60

Service Instance Health Checks | 64

Health Check Object | 65

Health Check Overview | 65

Health Check Object Configuration | 65

Creating a Health Check with the Contrail User Interface | 67

Using the Health Check | 68

- Health Check Process | 68

- Bidirectional Forwarding and Detection Health Check over Virtual Machine Interfaces | 69

- Bidirectional Forwarding and Detection Health Check for BGPaaS | 69

- Health Check of Transparent Service Chain | 70

- Service Instance Fate Sharing | 70

ECMP Support in Service Chain | 71

- Service Chain with Equal-Cost Multipath in Active-Active Mode | 71

- Service Chain with Health Check | 71

Route Reflector Support in Contrail Control Node | 72

- Benefits of RRs in Contrail | 73

Configuring Route Reflectors from Contrail Command | 73

BGP as a Service | 76

- Understanding BGP as a Service | 76

- Contrail BGPaaS Features | 76

- BGPaaS Use Cases | 78

- Configuring BGPaaS using VNC API | 80

- Configuring BGPaaS from Contrail Web UI | 81

- Configuring BGPaaS from Contrail Command | 82

Fat Flows | 85

- Understanding Fat Flow | 85

- Configuring Fat Flow from Contrail Command | 86

- Limitations of Fat Flow | 98

Use Case: Configuring Fat Flows from Contrail Command | 98

Overview | 99

Ignore Address - Source, Destination | 100

Ignore Address - None | 101

Prerequisites | 101

Getting Started | 102

Configuration | 103

Create Virtual Network | 104

Create Virtual Machine | 105

Create Service Template | 108

Add Service Instance | 109

Configure Fat Flow | 111

Create Service Policy | 113

Attach Service Policy | 114

Launch Virtual Machine | 114

Understanding Flow Sampling | 116

Flow Sampling | 117

Flow Handling | 118

Flow Aging | 118

TCP State-Based Flow Handling and Aging | 119

TCP State-Based Flow Handling | 119

Protocol-Based Flow Aging | 119

About the Documentation

IN THIS SECTION

- Documentation and Release Notes | vii
- Documentation Conventions | vii
- Documentation Feedback | x
- Requesting Technical Support | x

Use this guide to understand the features that would be used by service providers. This guide also provides information about advanced service chain configuration in Contrail.

Documentation and Release Notes

To obtain the most current version of all Juniper Networks[®] technical documentation, see the product documentation page on the Juniper Networks website at <https://www.juniper.net/documentation/>.

If the information in the latest release notes differs from the information in the documentation, follow the product Release Notes.

Juniper Networks Books publishes books by Juniper Networks engineers and subject matter experts. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration. The current list can be viewed at <https://www.juniper.net/books>.

Documentation Conventions

[Table 1 on page viii](#) defines notice icons used in this guide.

Table 1: Notice Icons

Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.
	Tip	Indicates helpful information.
	Best practice	Alerts you to a recommended use or implementation.

Table 2 on page viii defines the text and syntax conventions used in this guide.

Table 2: Text and Syntax Conventions

Convention	Description	Examples
Bold text like this	Represents text that you type.	To enter configuration mode, type the configure command: user@host> configure
Fixed-width text like this	Represents output that appears on the terminal screen.	user@host> show chassis alarms No alarms currently active
<i>Italic text like this</i>	<ul style="list-style-type: none"> Introduces or emphasizes important new terms. Identifies guide names. Identifies RFC and Internet draft titles. 	<ul style="list-style-type: none"> A policy <i>term</i> is a named structure that defines match conditions and actions. <i>Junos OS CLI User Guide</i> RFC 1997, <i>BGP Communities Attribute</i>

Table 2: Text and Syntax Conventions (*continued*)

Convention	Description	Examples
<i>Italic text like this</i>	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name: [edit] root@# set system domain-name <i>domain-name</i>
Text like this	Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components.	<ul style="list-style-type: none"> To configure a stub area, include the stub statement at the [edit protocols ospf area area-id] hierarchy level. The console port is labeled CONSOLE.
< > (angle brackets)	Encloses optional keywords or variables.	stub <default-metric <i>metric</i> >;
(pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	broadcast multicast (<i>string1</i> <i>string2</i> <i>string3</i>)
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	rsvp { # Required for dynamic MPLS only
[] (square brackets)	Encloses a variable for which you can substitute one or more values.	community name members [<i>community-ids</i>]
Indentation and braces ({ })	Identifies a level in the configuration hierarchy.	[edit] routing-options { static { route default { nexthop <i>address</i> ; retain; } } }
; (semicolon)	Identifies a leaf statement at a configuration hierarchy level.	

GUI Conventions

Table 2: Text and Syntax Conventions (*continued*)

Convention	Description	Examples
Bold text like this	Represents graphical user interface (GUI) items you click or select.	<ul style="list-style-type: none"> In the Logical Interfaces box, select All Interfaces. To cancel the configuration, click Cancel.
> (bold right angle bracket)	Separates levels in a hierarchy of menu selections.	In the configuration editor hierarchy, select Protocols>Ospf .

Documentation Feedback

We encourage you to provide feedback so that we can improve our documentation. You can use either of the following methods:

- Online feedback system—Click TechLibrary Feedback, on the lower right of any page on the [Juniper Networks TechLibrary](#) site, and do one of the following:



- Click the thumbs-up icon if the information on the page was helpful to you.
- Click the thumbs-down icon if the information on the page was not helpful to you or if you have suggestions for improvement, and use the pop-up form to provide feedback.
- E-mail—Send your comments to techpubs-comments@juniper.net. Include the document or topic name, URL or page number, and software version (if applicable).

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active Juniper Care or Partner Support Services support contract, or are

covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at <https://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf>.
- Product warranties—For product warranty information, visit <https://www.juniper.net/support/warranty/>.
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: <https://www.juniper.net/customers/support/>
- Search for known bugs: <https://prsearch.juniper.net/>
- Find product documentation: <https://www.juniper.net/documentation/>
- Find solutions and answer questions using our Knowledge Base: <https://kb.juniper.net/>
- Download the latest versions of software and review release notes: <https://www.juniper.net/customers/csc/software/>
- Search technical bulletins for relevant hardware and software notifications: <https://kb.juniper.net/InfoCenter/>
- Join and participate in the Juniper Networks Community Forum: <https://www.juniper.net/company/communities/>
- Create a service request online: <https://myjuniper.juniper.net>

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

Creating a Service Request with JTAC

You can create a service request with JTAC on the Web or by telephone.

- Visit <https://myjuniper.juniper.net>.
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see <https://support.juniper.net/support/requesting-support/>.

1

CHAPTER

Data Plane Optimization

Configuring the Data Plane Development Kit (DPDK) Integrated with Contrail vRouter | **13**

Configuring Single Root I/O Virtualization (SR-IOV) | **16**

Configuring the Data Plane Development Kit (DPDK) Integrated with Contrail vRouter

IN THIS SECTION

- [DPDK Support in Contrail | 13](#)
- [Preparing the Environment File for Provisioning a Cluster Node with DPDK | 13](#)
- [Creating a Flavor for DPDK | 15](#)

DPDK Support in Contrail

Contrail supports the Data Plane Development Kit (DPDK). DPDK is an open source set of libraries and drivers for fast packet processing. DPDK enables fast packet processing by allowing network interface cards (NICs) to send direct memory access (DMA) packets directly into an application's address space, allowing the application to poll for packets, and thereby avoiding the overhead of interrupts from the NIC.

Integrating with DPDK allows a Contrail vRouter to process more packets per second than is possible when running as a kernel module.

When using DPDK with Contrail Release 5.0, the DPDK configuration should be defined in **instances.yaml** for ansible based provision, or in **host.yaml** for helm-based provision. The AGENT_MODE configuration specifies whether the hypervisor is provisioned in the DPDK mode of operation.

When a Contrail compute node is provisioned with DPDK, the corresponding yaml file specifies the number of CPU cores to use for forwarding packets, the number of huge pages to allocate for DPDK, and the uio driver to use for DPDK.

Preparing the Environment File for Provisioning a Cluster Node with DPDK

The environment file is used at provisioning to specify all of the options necessary for the installation of a Contrail cluster, including whether any node should be configured to use DPDK.

Each node to be configured with the DPDK vRouter must be listed in the provisioning file with a dictionary entry, along with the percentage of memory for DPDK huge pages and the CPUs to be used.

The following are descriptions of the required entries for the server configuration. :

- **HUGE_PAGES**—Specify the percentage of host memory to be reserved for the DPDK huge pages. The reserved memory will be used by the vRouter and the Quick Emulator (QEMU) for allocating memory resources for the virtual machines (VMs) spawned on that host.

NOTE: The percentage allocated to **HUGE_PAGES** should not be too high, because the host Linux kernel also requires memory.

- **CPU_CORE_MASK**—Specify a CPU affinity mask with which vRouter will run. vRouter will use only the CPUs specified for its threads of execution. These CPU cores will be constantly polling for packets, and they will be displayed as 100% busy in the output of “top”.

The supported format is hexadecimal (for example, 0xf).

- **DPDK_UIO_DRIVER**—Specify the UIO driver to be used with DPDK.

The supported values include:

- **vfio-pci**—Specify that the vfio module in the Linux kernel should be used instead of uio. The vfio module protects memory access using the IOMMU when a SR-IOV virtual function is used as the physical interface of vrouter.
- **uio_pci_generic**—Specify that the UIO driver built into the Linux kernel should be used. This option does not support the use of SR-IOV VFs. This is the default option if **DPDK_UIO_DRIVER** is not specified.
- **mlnx** – For Mellanox ConnectX-4 and Mellanox ConnectX-5 NICs. This is available starting from Contrail release 5.0.1.

NOTE: For RHEL and Intel x710 Fortville-based NIC, use **vfio-pci** instead of the default **uio_pci_generic**.

Use the standard Ansible or helm-based provision procedure. Upon completion, your cluster, with specified nodes using the DPDK vRouter implementation, is ready to use.

Sample configuration in instances.yml for ansible-based provision

```
Bms1:
  provider: bms
  ip: ip-address
  roles:
    vrouter:
```

```
AGENT_MODE: dpdk
CPU_CORE_MASK: "0xff"
DPDK_UIO_DRIVER: uio_pci_generic
HUGE_PAGES: 32000
```

Sample configuration in host.yml for helm-based provision

```
...
AGENT_MODE: dpdk
CPU_CORE_MASK: "0xff"
DPDK_UIO_DRIVER: uio_pci_generic
HUGE_PAGES: 32000
```

Creating a Flavor for DPDK

To launch a VM in a DPDK-enabled vRouter hypervisor, the flavor for the VM should be set to use huge pages. The use of huge pages is a requirement for using a DPDK vRouter.

Use the following command to add the flavor, where **m1.large** is the name of the flavor. When a VM is created using this flavor, OpenStack ensures that the VM will only be spawned on a compute node that has huge pages enabled.

```
# openstack flavor set m1.large --property hw:mem_page_size=large
```

Huge pages are enabled for compute nodes where vRouter is provisioned with DPDK.

If a VM is spawned with a flavor that does not have huge pages enabled, the VM should not be created on a compute node on which vRouter is provisioned with DPDK.

You can use OpenStack availability zones or host aggregates to exclude the hosts where vRouter is provisioned with DPDK.

NOTE: Note: By default, 2MB huge pages are provisioned. If 1GB huge page is required, it must be done by the Administrator.

RELATED DOCUMENTATION

Configuring Single Root I/O Virtualization (SR-IOV)

IN THIS SECTION

- [Overview: Configuring SR-IOV | 16](#)
- [Enabling ASPM in BIOS | 16](#)
- [Configuring SR-IOV Using the Ansible Deployer | 17](#)
- [Configuring SR-IOV Using Helm | 18](#)
- [Launching SR-IOV Virtual Machines | 21](#)

Overview: Configuring SR-IOV

Contrail Release 3.0 through Release 4.0 supports single root I/O virtualization (SR-IOV) on Ubuntu systems only. With Release 4.1, Contrail supports SR-IOV on Red Hat Enterprise Linux (RHEL) operating systems as well.

SR-IOV is an interface extension of the PCI Express (PCIe) specification. SR-IOV allows a device, such as a network adapter to have separate access to its resources among various hardware functions.

As an example, the Data Plane Development Kit (DPDK) library has drivers that run in user space for several network interface cards (NICs). However, if the application runs inside a virtual machine (VM), it does not see the physical NIC unless SR-IOV is enabled on the NIC.

This topic shows how to configure SR-IOV with your Contrail system.

Enabling ASPM in BIOS

To use SR-IOV, it must have Active State Power Management (ASPM) enabled for PCI Express (PCIe) devices. Enable ASPM in the system BIOS.

NOTE: The BIOS of your system might need to be upgraded to a version that can enable ASPM.

Configuring SR-IOV Using the Ansible Deployer

You must perform the following tasks to enable SR-IOV on a system.

1. Enable the Intel Input/Output Memory Management Unit (IOMMU) on Linux.
2. Enable the required number of Virtual Functions (VFs) on the selected NIC.
3. Configure the names of the physical networks whose VMs can interface with the VFs.
4. Reboot Nova compute.

```
service nova-compute restart
```

5. Configure a Nova Scheduler filter based on the new PCI configuration, as in the following example:

```
/etc/nova/nova.conf
[default]
scheduler_default_filters = PciPassthroughFilter
scheduler_available_filters = nova.scheduler.filters.all_filters
scheduler_available_filters =
nova.scheduler.filters.pci_passthrough_filter.PciPassthroughFilter
```

6. Restart Nova Scheduler.

```
service nova-scheduler restart
```

The above tasks are handled by the Ansible Deployer playbook. The cluster members and its configuration parameters are specified in the **instances.yaml** file located in the config directory within the ansible-deployer repository.

The compute instances that are going to be in SR-IOV mode should have an SR-IOV configuration. The **instance.yaml** snippet below shows a sample instance definition.

```
instances:
  bms1:
```

```

    provider: bms
    ip: ip-address
    roles:
      openstack:
bms2:
  provider: bms
  ip: ip-address
  roles:
    config_database:
    config:
    control:
    analytics_database:
    analytics:
    webui:
bms3:
  provider: bms
  ip: ip-address
  roles:
    openstack_compute:
    vrouter:
      SRIOV: true
      SRIOV_VF: 3
      SRIOV_PHYSICAL_INTERFACE: eno1
      SRIOV_PHYS_NET: physnet1

```

Configuring SR-IOV Using Helm

You must perform the following tasks to enable SR-IOV on a system.

1. Enable the Intel Input/Output Memory Management Unit (IOMMU) on Linux.
2. Enable the required number of Virtual Functions (VFs) on the selected NIC.
3. Configure the names of the physical networks whose VMs can interface with the VFs.
4. Reboot Nova compute.

```
service nova-compute restart
```

5. Configure a Nova Scheduler filter based on the new PCI configuration, as in the following example:

```
/etc/nova/nova.conf
[default]
scheduler_default_filters = PciPassthroughFilter
scheduler_available_filters = nova.scheduler.filters.all_filters
scheduler_available_filters =
nova.scheduler.filters.pci_passthrough_filter.PciPassthroughFilter
```

6. Restart Nova Scheduler.

```
service nova-scheduler restart
```

The above tasks are handled by the Helm charts. The cluster members and its configuration parameters are specified in the **multinode-inventory** file located in the config directory within the openstack-helm-infra repository.

For Helm, the configuration and SR-IOV environment-specific parameters must be updated in three different places:

- The compute instance must be set as contrail-vrouter-sriov.

For example, the following is a snippet from the **tools/gate/devel/multinode-inventory.yaml** file in the openstack-helm-infra repository.

```
all:
  children:
    primary:
      hosts:
        node1:
          ansible_port: 22
          ansible_host: host-ip-address
          ansible_user: ubuntu
          ansible_ssh_private_key_file: /home/ubuntu/.ssh/insecure.pem
          ansible_ssh_extra_args: -o StrictHostKeyChecking=no
  nodes:
    children:
      openstack-compute:
        children:
          contrail-vrouter-sriov: #compute instance set to contrail-vrouter-sriov

          hosts:
            node7:
              ansible_port: 22
              ansible_host: host-ip-address
              ansible_user: ubuntu
```

```

ansible_ssh_private_key_file: /home/ubuntu/.ssh/insecure.pem
ansible_ssh_extra_args: -o StrictHostKeyChecking=no

```

- Contrail-vrouter-sriov must be labeled appropriately.

For example, the following is a snippet from the **tools/gate/devel/multinode-vars.yaml** in the **openstack-helm-infra** repository.

```

nodes:
  labels:
    primary:
      - name: openstack-helm-node-class
        value: primary

      all:
        - name: openstack-helm-node-class
          value: general
    contrail-controller:
      - name: opencontrail.org/controller
        value: enabled
    openstack-compute:
      - name: openstack-compute-node
        value: enabled
    contrail-vrouter-dpdk:
      - name: opencontrail.org/vrouter-dpdk
        value: enabled
    contrail-vrouter-sriov: # label as contrail-vrouter-sriov
      - name: vrouter-sriov
        value: enabled

```

- SR-IOV config parameters must be updated in the **contrail-vrouter/values.yaml** file.

For example, the following is a snippet from the **contrail-vrouter/values.yaml** file in the **contrail-helm-deployer** repository.

```

contrail_env_vrouter_kernel:
  AGENT_MODE: kernel

contrail_env_vrouter_sriov:
  SRIOV: true
  per_compute_info:
    node_name: k8snode1
    SRIOV_VF: 10

```

```
SRIOV_PHYSICAL_INTERFACE: enp129s0f1
SRIOV_PHYS_NET: physnet1
```

Launching SR-IOV Virtual Machines

IN THIS SECTION

- Using the Contrail UI to Enable and Launch an SR-IOV Virtual Machine | 21
- Using the CLI to Enable and Launch SR-IOV Virtual Machines | 22

After ensuring that SR-IOV features are enabled on your system, use one of the following procedures to create a virtual network from which to launch an SR-IOV VM, either by using the Contrail UI or the CLI. Both methods are included.

Using the Contrail UI to Enable and Launch an SR-IOV Virtual Machine

To use the Contrail UI to enable and launch an SR-IOV VM:

1. At **Configure > Networking > Networks**, create a virtual network with SR-IOV enabled. Ensure the virtual network is created with a subnet attached. In the Advanced section, select the **Provider Network** check box, and specify the physical network already enabled for SR-IOV (in `testbed.py` or `nova.conf`) and its VLAN ID. See [Figure 1 on page 21](#).

Figure 1: Edit Network

Provider Network

☒

Physical Network

VLAN

2. On the virtual network, create a Neutron port (**Configure > Networking > Ports**), and in the **Port Binding** section, define a **Key** value of SR-IOV and a **Value** of direct. See [Figure 2 on page 22](#).

Figure 2: Create Port

Create

ECMP Hashing Fields

Select ECMP Hashing Fields

Device Owner

None

Port Binding(s)

Key	Value
SR-IOV (vnic_type:direct)	direct

☐ Disable Policy

☐ Sub Interface

☐ Mirroring

S018551

Cancel Save

- Using the UUID of the Neutron port you created, use the **nova boot** command to launch the VM from that port.

```
nova boot --flavor m1.large --image <image name> --nic port-id=<uuid of above port> <vm name>
```

Using the CLI to Enable and Launch SR-IOV Virtual Machines

To use CLI to enable and launch an SR-IOV VM:

- Create a virtual network with SR-IOV enabled. Specify the physical network already enabled for SR-IOV (in **testbed.py** or **nova.conf**) and its VLAN ID.

The following example creates **vn1** with a VLAN ID of 100 and is part of **physnet1**:

```
neutron net-create --provider:physical_network=physnet1 --provider:segmentation_id=100 vn1
```

- Create a subnet in vn1.

```
neutron subnet-create vn1 a.b.c.0/24
```

3. On the virtual network, create a Neutron port on the subnet, with a binding type of direct.

```
neutron port-create --fixed-ip subnet_id=<subnet uuid>,ip_address=<IP address from above subnet>
--name <name of port> <vn uuid> --binding:vnic_type direct
```

4. Using the UUID of the Neutron port created, use the **nova boot** command to launch the VM from that port.

```
nova boot --flavor m1.large --image <image name> --nic port-id=<uuid of above port> <vm name>
```

5. Log in to the VM and verify that the Ethernet controller is VF by using the **lspci** command to list the PCI buses.

The VF that gets configured with the VLAN can be observed using the **ip link** command.

RELATED DOCUMENTATION

| [Configuring the Data Plane Development Kit \(DPDK\) Integrated with Contrail vRouter](#) | 13

2

CHAPTER

Advanced Network Topologies

Configuring Virtual Networks for Hub-and-Spoke Topology | **25**

Remote Compute | **31**

Configuring Virtual Networks for Hub-and-Spoke Topology

IN THIS SECTION

- [Route Targets for Virtual Networks in Hub-and-Spoke Topology | 25](#)
- [Example: Hub-and-Spoke Topology | 26](#)
- [Troubleshooting Hub-and-Spoke Topology | 27](#)

As of Contrail Release 3.0, hub-and-spoke topology can be used to ensure that virtual machines (VMs) don't communicate with each other directly; their communication is only allowed indirectly by means of a designated hub virtual network.

Route Targets for Virtual Networks in Hub-and-Spoke Topology

Hub-and-spoke topology can be used to ensure that virtual machines (VMs) don't communicate with each other directly; their communication is only allowed indirectly by means of a designated hub virtual network (VN). The VMs are configured in spoke VNs.

This is useful for enabling VMs in a spoke VN to communicate by means of a policy or firewall, where the firewall exists in a hub site.

hub-and-spoke topology is implemented using two route targets (**hub-rt** and **spoke-rt**), as follows:

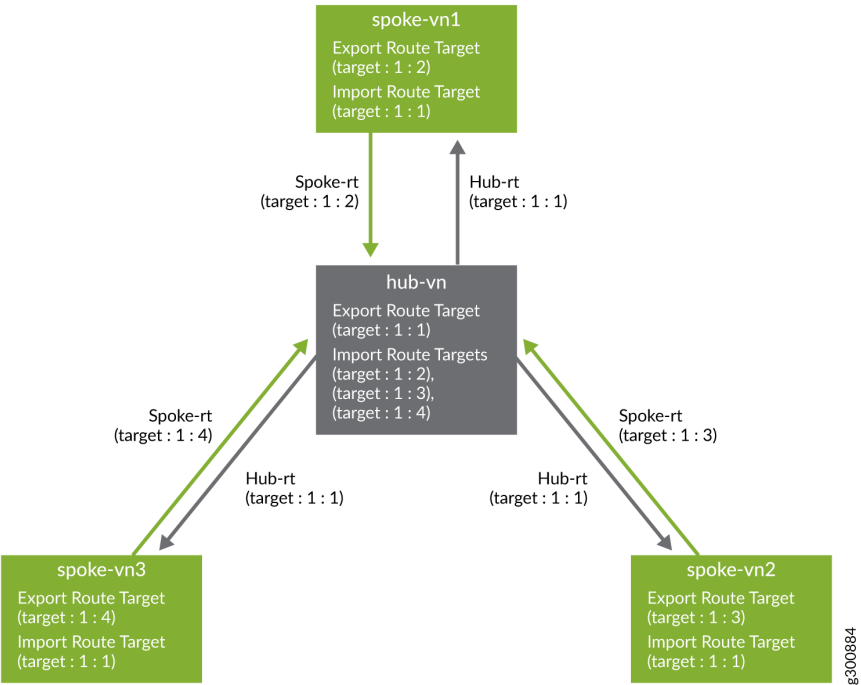
- Hub route target (**hub-rt**):
 - The hub VN *exports* all routes tagged with **hub-rt**.
 - The spoke VN *imports* routes tagged with **hub-rt**, ensuring that the spoke VN has only routes exported by the hub VN.
 - To attract spoke traffic, the hub VN readvertises the spoke routes or advertises the default route.
- Spoke route target (**spoke-rt**):
 - All spoke VNs export routes with route target **spoke-rt**.
 - The hub VN imports all spoke routes, ensuring that hub VN has all spoke routes.

NOTE: The hub VN or VRF can reside in an external gateway, such as an MX Series router, while the spoke VN resides in the Contrail controller.

Example: Hub-and-Spoke Topology

In the example shown in [Figure 3 on page 26](#), the **hub-vn** is configured as a hub virtual network, and the three **spoke-vns** are configured as spoke virtual networks. The hub and spokes each use a unique export route target. The **hub-vn** exports its **hub-rt (target:1:1)** routes to the spokes, and each **spoke-vn** imports them. Each **spoke-vn** exports its **spoke-rt (target:1:2, target:1:3, target:1:4)** routes to the hub, and the **hub-vn** imports them.

Figure 3: Hub-and-Spoke Topology



Troubleshooting Hub-and-Spoke Topology

The following examples provide methods to help you troubleshoot hub-and-spoke configurations.

Example: Validating the Configuration on the Virtual Network

The following example uses the api-server HTTP get request to validate the configuration on the virtual network.

Hub VN configuration:

`curl -u admin:<password> http://<host ip>/virtual-network/<hub-vn-uuid> | python -m json.tool`

```
{
  "virtual-network": {
    "display_name": "hub-vn",
    "fq_name": [
      "default-domain",
      "admin",
      "hub-vn"
    ],
    "export_route_target_list": {
      "route_target": [
        "target:1:2"
      ]
    },
    "import_route_target_list": {
      "route_target": [
        "target:1:1"
      ]
    },
  }
}
```

Spoke VN configuration:

`curl -u admin:<password> http://<host ip>:8095/virtual-network/<spoke-vn-uuid> | python -m json.tool`

```
{
  {
    "virtual-network": {
      "display_name": "spoke-vn1",
      "fq_name": [
        "default-domain",
```

```

        "admin",
        "spoke-vn1"
    ],
    "export_route_target_list": {
        "route_target": [
            "target:1:1"
        ]
    },
    "import_route_target_list": {
        "route_target": [
            "target:1:2"
        ]
    },
}
}

```

Example: Validate the Configuration on the Routing Instance

The following example uses **api-server HTTP get** request to validate the configuration on the routing instance.

Spoke VRF configuration (with a system-created VRF by schema transformer):

```

user@node:/opt/contrail/utls# curl -u admin:<password> http://<host
ip>:8095/routing-instance/<spoke-vrf-uuid>| python -m json.tool

```

```

{
  "routing-instance": {
    "display_name": "spoke-vn1",
    "fq_name": [
      "default-domain",
      "admin",
      "spoke-vn1",
      "spoke-vn1"
    ],
    "route_target_refs": [
      {
        "attr": {
          "import_export": "export"
        },
        "href": "http://<host
ip>:8095/route-target/446a3bbe-f263-4b58-a537-8333878dd7c3",
        "to": [

```

```

        "target:1:1"
    ],
    "uuid": "446a3bbe-f263-4b58-a537-8333878dd7c3"
},
{
    "attr": {
        "import_export": null
    },
    "href": "http://<host
ip>:8095/route-target/7668088d-e403-414f-8f5d-649ed80e0689",
    "to": [
        "target:64512:8000012"
    ],
    "uuid": "7668088d-e403-414f-8f5d-649ed80e0689"
},
{
    "attr": {
        "import_export": "import"
    },
    "href": "http://<host
ip>:8095/route-target/8f216064-8488-4486-8fce-b4afb87266bb",
    "to": [
        "target:1:2"
    ],
    "uuid": "8f216064-8488-4486-8fce-b4afb87266bb"
}
],
"routing_instance_is_default": true,
}
}

```

Hub VRF configuration:

curl -u admin:<password> http://<host ip>:8095/routing-instance/<hub-vrf-uuid> | python -m json.tool

```

{
  "routing-instance": {
    "display_name": "hub-vn",
    "fq_name": [
      "default-domain",
      "admin",
      "hub-vn",
      "hub-vn"
    ]
  }
}

```

```

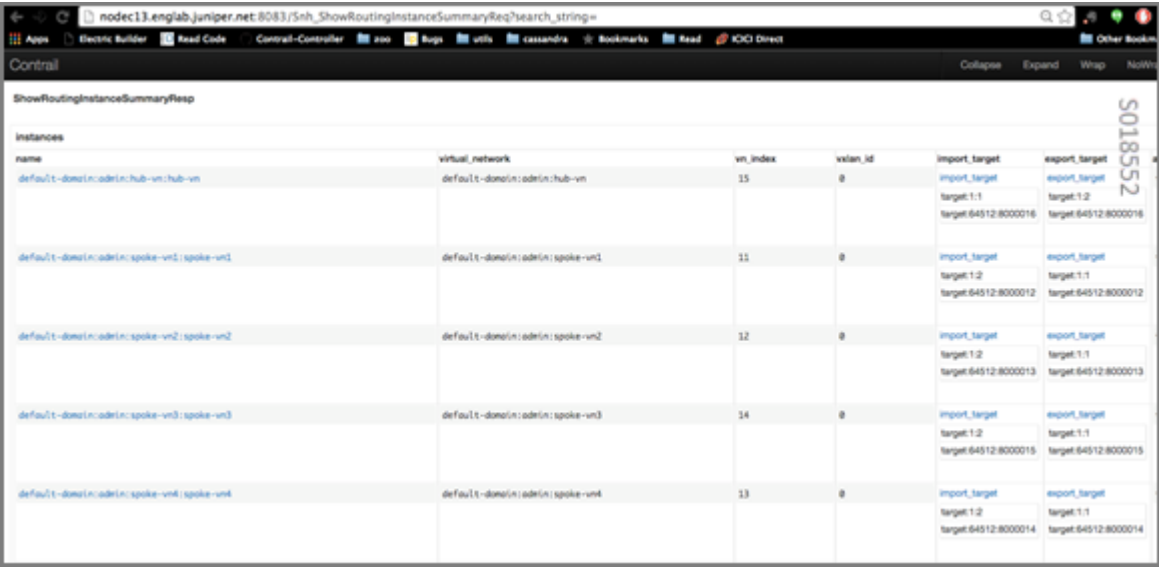
    ],
    "route_target_refs": [
      {
        "attr": {
          "import_export": "import"
        },
        "href": "http://<host
ip>:8095/route-target/446a3bbe-f263-4b58-a537-8333878dd7c3",
        "to": [
          "target:1:1"
        ],
        "uuid": "446a3bbe-f263-4b58-a537-8333878dd7c3"
      },
      {
        "attr": {
          "import_export": "export"
        },
        "href": "http://<host
ip>:8095/route-target/8f216064-8488-4486-8fce-b4afb87266bb",
        "to": [
          "target:1:2"
        ],
        "uuid": "8f216064-8488-4486-8fce-b4afb87266bb"
      },
      {
        "attr": {
          "import_export": null
        },
        "href": "http://<host
ip>:8095/route-target/a85fec19-eed2-430c-af23-9919aca1dd12",
        "to": [
          "target:64512:8000016"
        ],
        "uuid": "a85fec19-eed2-430c-af23-9919aca1dd12"
      }
    ],
    "routing_instance_is_default": true,
  }
}

```

Example: Using Contrail Control Introspect

Figure 4 on page 31 shows the import and export targets for **hub-vn** and **spoke-vns**, by invoking **contrail-control-introspect**.

Figure 4: Contrail Introspect



The screenshot shows a web browser window with the URL `nodec13.englab.juniper.net:8083/5nh_ShowRoutingInstanceSummaryReq?search_string=`. The page title is "Contrail" and the main content is a table titled "ShowRoutingInstanceSummaryResp". The table has columns: name, virtual_network, vn_index, vxlan_id, import_target, and export_target. The table lists several routing instances, including "default-domain:admin:hub-vn" and "default-domain:admin:spoke-vn1" through "default-domain:admin:spoke-vn4". The table is displayed in a grid format with alternating light and dark rows. A vertical label "5018552" is visible on the right side of the table.

name	virtual_network	vn_index	vxlan_id	import_target	export_target
default-domain:admin:hub-vn	default-domain:admin:hub-vn	15	0	import_target target:1:1 target:64512:8000016	export_target target:1:2 target:64512:8000016
default-domain:admin:spoke-vn1	default-domain:admin:spoke-vn1	11	0	import_target target:1:2 target:64512:8000012	export_target target:1:1 target:64512:8000012
default-domain:admin:spoke-vn2	default-domain:admin:spoke-vn2	12	0	import_target target:1:2 target:64512:8000013	export_target target:1:1 target:64512:8000013
default-domain:admin:spoke-vn3	default-domain:admin:spoke-vn3	14	0	import_target target:1:2 target:64512:8000015	export_target target:1:1 target:64512:8000015
default-domain:admin:spoke-vn4	default-domain:admin:spoke-vn4	13	0	import_target target:1:2 target:64512:8000014	export_target target:1:1 target:64512:8000014

Remote Compute

IN THIS SECTION

- Remote Compute Overview | 31
- Remote Compute Features | 32
- Remote Compute Operations | 32
- Provisioning a Remote Compute Cluster | 33

Contrail Release 5.0.1 introduces remote compute, a method of managing a Contrail deployment across many small distributed data centers efficiently and cost effectively.

Remote Compute Overview

The purpose of remote compute is to enable the deployment of Contrail in many small distributed data centers, up to hundreds or even thousands, for telecommunications point-of-presence (PoPs) or central offices (COs). Each small data center has only a small number of computes, typically 5-20 in a rack, running

a few applications such as video caching, traffic optimization, and virtual Broadband Network Gateway (vBNG). It is not cost effective to deploy a full Contrail controller cluster of nodes of control, configuration, analytics, database, and the like, in each distributed PoP on dedicated servers. Additionally, manually managing hundreds or thousands of clusters is not feasible operationally.

Remote Compute Features

Remote compute is implemented by means of a subcluster that manages compute nodes at remote sites to receive configurations and exchange routes.

The key concepts of Contrail remote compute include:

- Remote compute employs a subcluster to manage remote compute nodes away from the primary data center.
- The Contrail control cluster is deployed in large centralized data centers, where it can remotely manage compute nodes in small distributed small data centers.
- A lightweight version of the controller is created, limited to the control node, and the config node, analytics, and analytics database are shared across several control nodes.
- Many lightweight controllers are co-located on a small number of servers to optimize efficiency and cost.
- The control nodes peer with the remote compute nodes by means of XMPP and peer with local gateways by means of MP-eBGP.

Remote Compute Operations

A subcluster object is created for each remote site, with a list of links to local compute nodes that are represented as vrouter objects, and a list of links to local control nodes that are represented as BGP router objects, with an ASN as property.

The subclusters are identified in the provision script. The vrouter and bgp-router provision scripts take each subcluster as an optional argument to link or delink with the subcluster object.

It is recommended to spawn the control nodes of the remote cluster in the primary cluster, and they are IGBP-meshed among themselves within that subcluster. The control nodes BGP-peer with their respective SDN gateway, over which route exchange occurs with the primary control nodes.

Compute nodes in the remote site are provisioned to connect to their respective control nodes to receive configuration and exchange routes. Data communication among workloads between these clusters occurs

through the provider backbone and their respective SDN gateways. The compute nodes and the control nodes push analytics data to analytics nodes hosted on the primary cluster.

Subcluster Properties

The Contrail UI shows a list of subcluster objects, each with a list of associated vrouters and BGP routers that are local in that remote site and the ASN property.

General properties of subclusters include:

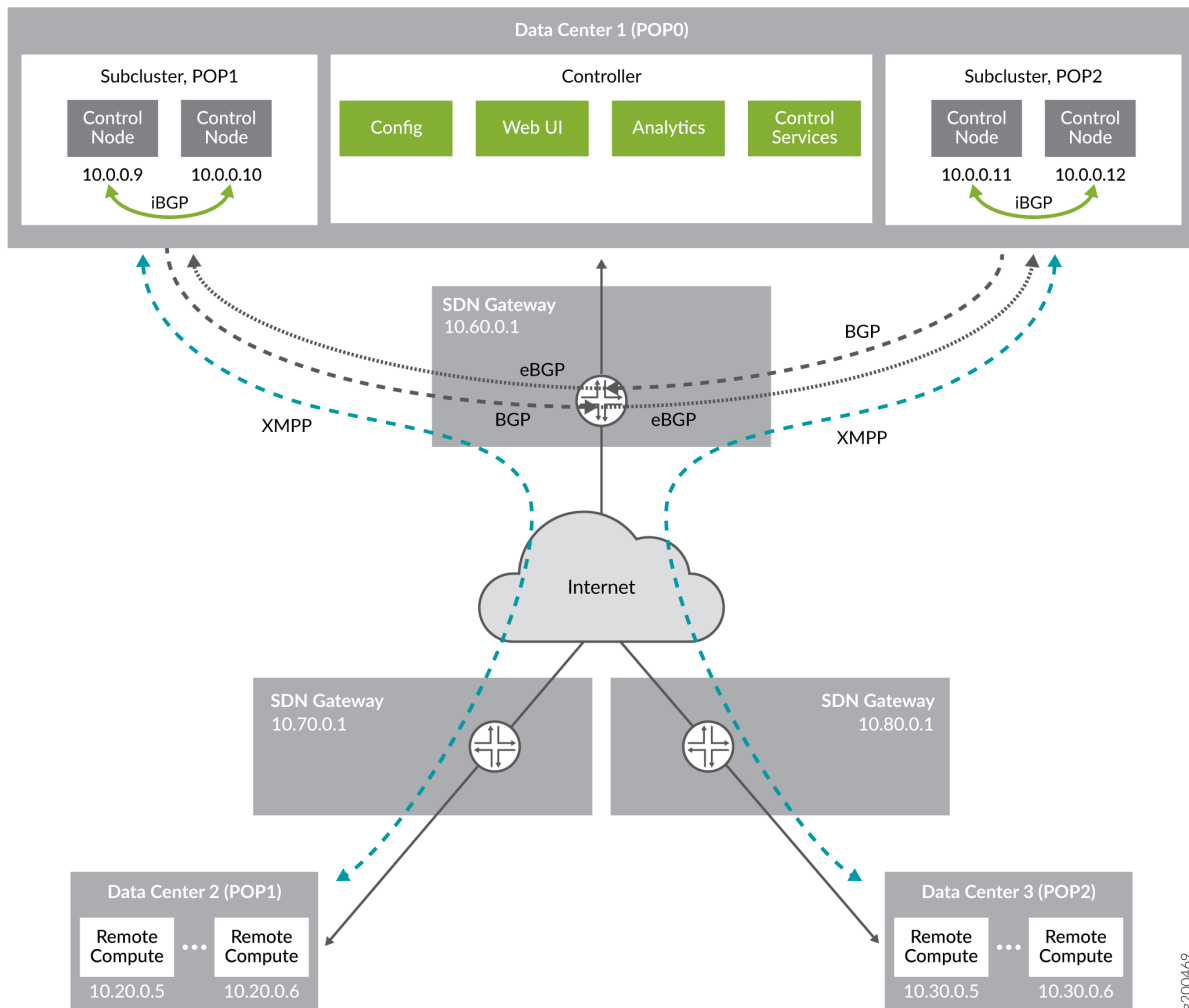
- A subcluster control node never directly peers with another subcluster control node or with primary control nodes.
- A subcluster control node has to be created, and is referred to, in virtual-router and bgp-router objects.
- A subcluster object and the control nodes under it should have the same ASN.
- The ASN cannot be modified in a subcluster object.

NOTE: Multinode service chaining across subclusters is not supported.

Provisioning a Remote Compute Cluster

With Contrail Release 5.0.1, you provision for remote compute using an `instances.yaml` file. *Installing Contrail Cluster using Contrail Command and instances.yml* shows a bare minimum configuration. The YAML file described in this section builds upon that minimum configuration and uses [Figure 5 on page 34](#) as an example data center network topology.

Figure 5: Example Multi-Cluster Topology



In this topology, there is one main data center (**pop0**) and two remote data centers (**pop1** and **pop2**.) **pop0** contains two subclusters: one for **pop1**, and the other for **pop2**. Each subcluster has two control nodes. The control nodes within a subcluster, for example 10.0.0.9 and 10.0.0.10, communicate with each other through iBGP.

Communication between the control nodes within a subcluster and the remote data center is through the SDN Gateway; there is no direct connection. For example, the remote compute in pop1 (IP address 10.20.0.5) communicates with the control nodes (IP addresses 10.0.0.9 and 10.0.0.10) in subcluster 1 through the SDN Gateway.

To configure remote compute in the YAML file:

1. First, create the remote locations or subclusters. In this example, we create data centers 2 and 3 (with the names **pop1** and **pop2**, respectively), and define unique ASN numbers for each. Subcluster names must also be unique.

```
remote_locations:
  pop1:
    BGP_ASN: 12345
    SUBCLUSTER: pop1
  pop2:
    BGP_ASN: 12346
    SUBCLUSTER: pop2
```

2. Create the control nodes for pop1 and pop2 and assign an IP address and role. These IP addresses are the local IP address. In this example, there are two control nodes for each sub-cluster.

```
control_1_only_pop1:      # Mandatory. Instance name
  provider: bms           # Mandatory. Instance runs on BMS
  ip: 10.0.0.9
  roles:
    control:
      location: pop1
control_2_only_pop1:      # Mandatory. Instance name
  provider: bms           # Mandatory. Instance runs on BMS
  ip: 10.0.0.10
  roles:
    control:
      location: pop1
control_1_only_pop2:      # Mandatory. Instance name
  provider: bms           # Mandatory. Instance runs on BMS
  ip: 10.0.0.11
  roles:                  # Optional.
    control:
      location: pop2
control_2_only_pop2:      # Mandatory. Instance name
  provider: bms           # Mandatory. Instance runs on BMS
  ip: 10.0.0.12
  roles:                  # Optional.
    control:
      location: pop2
```

- Now, create the remote compute nodes for **pop1** and **pop2** and assign an IP address and role. In this example, there are two remote compute nodes for each data center. The 10.60.0.x addresses are the management IP addresses for the control service.

```

compute_1_pop1:                # Mandatory. Instance name
  provider: bms                 # Mandatory. Instance runs on BMS
  ip: 10.20.0.5
  roles:
    openstack_compute:         # Optional.
    vrouter:
      CONTROL_NODES: 10.60.0.9,10.60.0.10
      VROUTER_GATEWAY: 10.70.0.1
      location: pop1
compute_2_pop1:                # Mandatory. Instance name
  provider: bms                 # Mandatory. Instance runs on BMS
  ip: 10.20.0.6
  roles:
    openstack_compute:         # Optional.
    vrouter:
      CONTROL_NODES: 10.60.0.9,10.60.0.10
      VROUTER_GATEWAY: 10.70.0.1
      location: pop1
compute_1_pop2:                # Mandatory. Instance name
  provider: bms                 # Mandatory. Instance runs on BMS
  ip: 10.30.0.5
  roles:
    openstack_compute:         # Optional.
    vrouter:
      CONTROL_NODES: 10.60.0.11,10.60.0.12
      VROUTER_GATEWAY: 10.80.0.1
      location: pop2
compute_2_pop2:                # Mandatory. Instance name
  provider: bms                 # Mandatory. Instance runs on BMS
  ip: 10.30.0.6
  roles:
    openstack_compute:         # Optional.
    vrouter:
      CONTROL_NODES: 10.60.0.11,10.60.0.12
      VROUTER_GATEWAY: 10.80.0.1
      location: pop2

```

The entire YAML file is contained below.

Example instance.yaml with sub-cluster configuration

```

provider_config:
  bms:
    ssh_pwd: <password>
    ssh_user: <root_user>
    ntpserver: 10.84.5.100
    domainsuffix: local
instances:
  openstack_node:
    # Mandatory. Instance name
    provider: bms
    # Mandatory. Instance runs on BMS
    ip: 10.0.0.4
    roles:
      # Optional.
      openstack:
  all_contrail_roles_default_pop:
    # Mandatory. Instance name
    provider: bms
    # Mandatory. Instance runs on BMS
    ip: 10.0.0.5
    roles:
      # Optional.
      config_database:
        # Optional.
      config:
        # Optional.
      control:
        # Optional.
      analytics_database:
        # Optional.
      analytics:
        # Optional.
      webui:
        # Optional.
  compute_3_default_pop:
    # Mandatory. Instance name
    provider: bms
    # Mandatory. Instance runs on BMS
    ip: 10.0.0.6
    roles:
      openstack_compute:
      vrouter:
        VROUTER_GATEWAY: 10.60.0.1
  compute_1_default_pop:
    # Mandatory. Instance name
    provider: bms
    # Mandatory. Instance runs on BMS
    ip: 10.0.0.7
    roles:
      openstack_compute:
      vrouter:
        VROUTER_GATEWAY: 10.60.0.1
  compute_2_default_pop:
    # Mandatory. Instance name
    provider: bms
    # Mandatory. Instance runs on BMS
    ip: 10.0.0.8
    roles:
      openstack_compute:
      vrouter:
        VROUTER_GATEWAY: 10.60.0.1

```

```

control_1_only_pop1:           # Mandatory. Instance name
    provider: bms               # Mandatory. Instance runs on BMS
    ip: 10.0.0.9
    roles:
        control:
            location: pop1
control_2_only_pop1:           # Mandatory. Instance name
    provider: bms               # Mandatory. Instance runs on BMS
    ip: 10.0.0.10
    roles:
        control:
            location: pop1
control_1_only_pop2:           # Mandatory. Instance name
    provider: bms               # Mandatory. Instance runs on BMS
    ip: 10.0.0.11
    roles:                      # Optional.
        control:
            location: pop2
control_2_only_pop2:           # Mandatory. Instance name
    provider: bms               # Mandatory. Instance runs on BMS
    ip: 10.0.0.12
    roles:                      # Optional.
        control:
            location: pop2
compute_1_pop1:                # Mandatory. Instance name
    provider: bms               # Mandatory. Instance runs on BMS
    ip: 10.20.0.5
    roles:
        openstack_compute:     # Optional.
        vrouters:
            CONTROL_NODES: 10.60.0.9,10.60.0.10
            VROUTER_GATEWAY: 10.70.0.1
            location: pop1
compute_2_pop1:                # Mandatory. Instance name
    provider: bms               # Mandatory. Instance runs on BMS
    ip: 10.20.0.6
    roles:
        openstack_compute:     # Optional.
        vrouters:
            CONTROL_NODES: 10.60.0.9,10.60.0.10
            VROUTER_GATEWAY: 10.70.0.1
            location: pop1
compute_1_pop2:                # Mandatory. Instance name
    provider: bms               # Mandatory. Instance runs on BMS

```

```

ip: 10.30.0.5
roles:
  openstack_compute:          # Optional.
  vrouter:
    CONTROL_NODES: 10.60.0.11,10.60.0.12
    VROUTER_GATEWAY: 10.80.0.1
    location: pop2
compute_2_pop2:                # Mandatory. Instance name
  provider: bms                 # Mandatory. Instance runs on BMS
  ip: 10.30.0.6
  roles:
    openstack_compute:        # Optional.
    vrouter:
      CONTROL_NODES: 10.60.0.11,10.60.0.12
      VROUTER_GATEWAY: 10.80.0.1
      location: pop2
global_configuration:
  CONTAINER_REGISTRY: 10.xx.x.81:5000
  REGISTRY_PRIVATE_INSECURE: True

contrail_configuration:          # Contrail service configuration section
  CONTRAIL_VERSION: <contrail_version>
  CONTROLLER_NODES: 10.60.0.5
  CLOUD_ORCHESTRATOR: openstack
  KEYSTONE_AUTH_HOST: 10.60.0.100
  KEYSTONE_AUTH_URL_VERSION: /v3
  RABBITMQ_NODE_PORT: 5673
  PHYSICAL_INTERFACE: eth1
  CONTROL_DATA_NET_LIST: 10.60.0.0/24,10.70.0.0/24,10.80.0.0/24

kolla_config:
  kolla_globals:
    network_interface: "eth1"
    enable_haproxy: "yes"
    contrail_api_interface_address: 10.60.0.5
    kolla_internal_vip_address: 10.60.0.100
    kolla_external_vip_address: 10.0.0.100
    kolla_external_vip_interface: "eth0"
  kolla_passwords:
    keystone_admin_password: <password>

remote_locations:
  pop1:
    BGP_ASN: 12345

```

```
SUBCLUSTER: pop1
pop2:
  BGP_ASN: 12346
  SUBCLUSTER: pop2
```

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

3

CHAPTER

Advanced Service Chain Configuration

Customized Hash Field Selection for ECMP Load Balancing | **42**

Routing Policy | **45**

Creating a Routing Policy With Extended Communities in Contrail Command | **60**

Service Instance Health Checks | **64**

ECMP Support in Service Chain | **71**

Route Reflector Support in Contrail Control Node | **72**

Configuring Route Reflectors from Contrail Command | **73**

BGP as a Service | **76**

Fat Flows | **85**

Use Case: Configuring Fat Flows from Contrail Command | **98**

Understanding Flow Sampling | **116**

Customized Hash Field Selection for ECMP Load Balancing

Overview: Custom Hash Feature

Starting with Contrail Release 3.0, it is possible to configure the set of fields used to hash upon during equal-cost multipath (ECMP) load balancing.

Earlier versions of Contrail had this set of fields fixed to the standard 5-tuple set of: source L3 address, destination L3 address, L4 protocol, L4 SourcePort, and L4 DestinationPort.

With the custom hash feature, users can configure an exact subset of fields to hash upon when choosing the forwarding path among a set of eligible ECMP candidates.

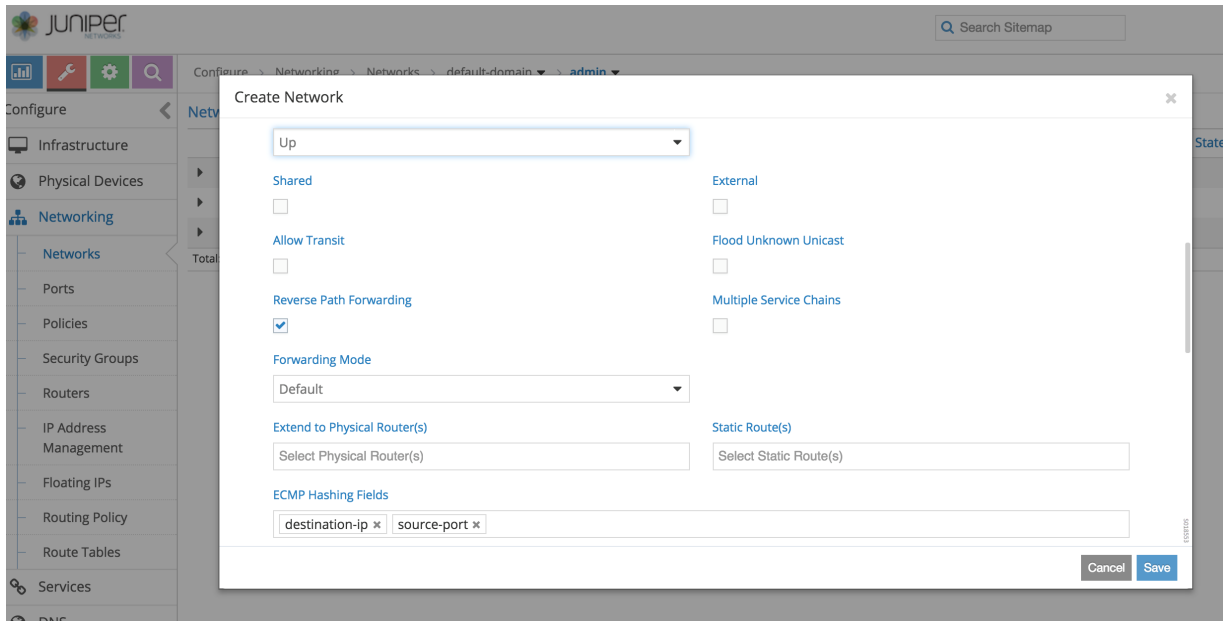
The custom hash configuration can be applied in the following ways:

- globally
- per virtual network (VN)
- per virtual machine interface (VMI)

VMI configurations take precedence over VN configurations, and VN configurations take precedence over global level configuration (if present).

Custom hash is useful whenever packets originating from a particular source and addressed to a particular destination must go through the same set of service instances during transit. This might be required if source, destination, or transit nodes maintain a certain state based on the flow, and the state behavior could also be used for subsequent new flows, between the same pair of source and destination addresses. In such cases, subsequent flows must follow the same set of service nodes followed by the initial flow.

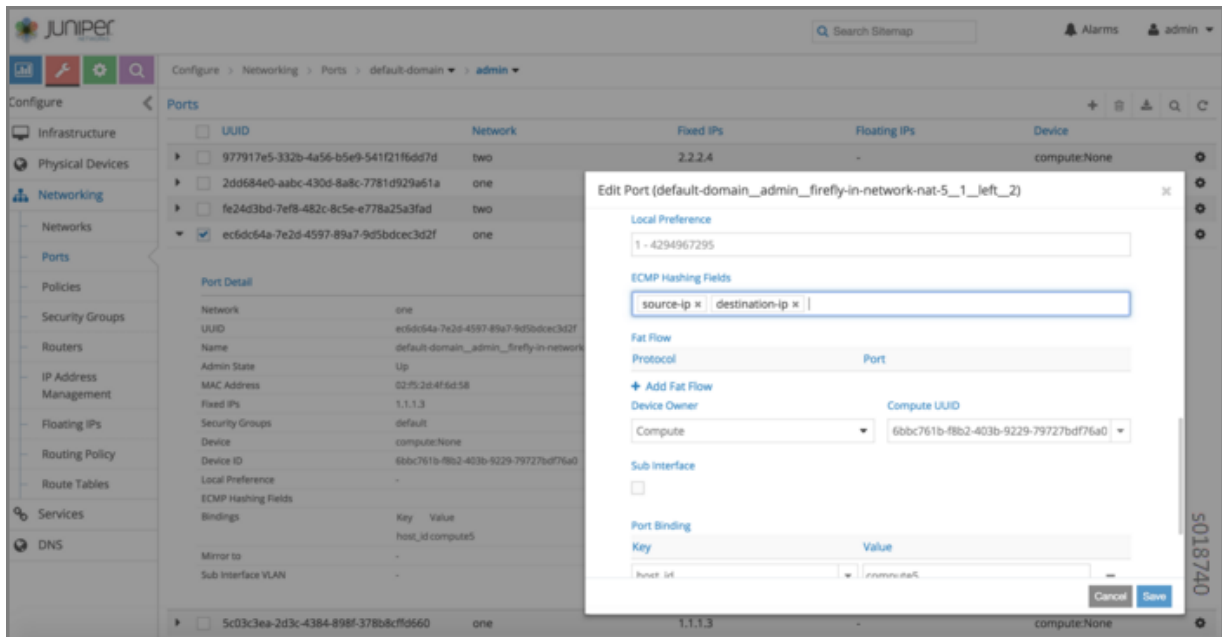
You can use the Contrail UI to identify specific fields in the network upon which to hash at the **Configure > Networking > Network, Create Network** window, in the **ECMP Hashing Fields** section as shown in the following figure.



If the hashing fields are configured for a virtual network, all traffic destined to that VN will be subject to the customized hash field selection during forwarding over ECMP paths by vRouters. This may not be desirable in all cases, as it could potentially skew all traffic to the destination network over a smaller set of paths across the IP fabric.

A more practical scenario is one in which flows between a source and destination must go through the same service instance in between, where one could configure customized ECMP fields for the virtual machine interface (VMI) of the service instance. Then, each service chain route originating from that VMI

would get the desired ECMP field selection applied as its path attribute, and eventually get propagated to the ingress vRouter node. See the following example.



Using ECMP Hash Fields Selection

Custom hash fields selection is most useful in scenarios where multiple ECMP paths exist for a destination. Typically, the multiple ECMP paths point to ingress service instance nodes, which could be running anywhere in the Contrail cloud.

Configuring ECMP Hash Fields Over Service Chains

Use the following steps to create customized hash fields with ECMP over service chains.

1. Create the virtual networks needed to interconnect using service chaining, with ECMP load-balancing.
2. Create a service template and enable scaling.
3. Create a service instance, and using the service template, configure by selecting:
 - the desired number of instances for scale-out
 - the left and right virtual network to connect
 - the shared address space, to make sure that instantiated services come up with the same IP address for left and right, respectively

This configuration enables ECMP among all those service instances during forwarding.

4. Create a policy, then select the service instance previously created and apply the policy to the desired VMIs or VNs.
5. After the service VMs are instantiated, the ports of the left and right interfaces are available for further configuration. At the Contrail UI Ports section under Networking, select the left port (VMI) of the service instance and apply the desired ECMP hash field configuration.

NOTE: Currently the ECMP field selection configuration for the service instance left or right interface must be applied by using the Ports (VMIs) section under Networking and explicitly configuring the ECMP fields selection for each of the instantiated service instances' VMIs. This must be done for all service interfaces of the group, to ensure the end result is as expected, because the load balance attribute of only the best path is carried over to the ingress vRouter. If the load balance attribute is not configured, it is not propagated to the ingress vRouter, even if other paths have that configuration.

When the configuration is finished, the vRouters get programmed with routing tables with the ECMP paths to the various service instances. The vRouters are also programmed with the desired ECMP hash fields to be used during load balancing of the traffic.

Routing Policy

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

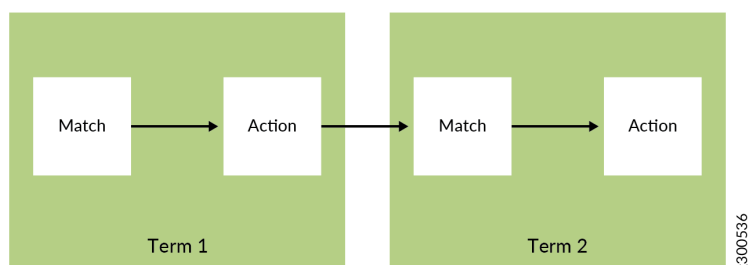
The routing policy contains list terms. A term can be a terminal rule, meaning that upon a match on the specified term, no further terms are evaluated and the route is dropped or accepted, based on the action in that term.

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

The list terms are structured as in the following example.

```
Policy {
  Term-1
  Term-2
}
```

The matches and actions of the policy term lists operate similarly to the Junos language match and actions operations. A visual representation is the following.



Each term is represented as in the following:

```

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

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

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

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

Applying Routing Policy

The routing policy evaluation has the following key points:

- If the term of a routing policy consists of multiple match conditions, a route must satisfy all match conditions to apply the action specified in the term.
- If a term in the policy does not specify a match condition, all routes are evaluated against the match.

- If a match occurs but the policy does not specify an accept, reject, or next term action, one of the following occurs:
 - The next term, if present, is evaluated.
 - If no other terms are present, the next policy is evaluated.
 - If no other policies are present, the route is accepted. The default routing policy action is “accept”.
- If a match does not occur with a term in a policy, and subsequent terms in the same policy exist, the next term is evaluated.
- If a match does not occur with any terms in a policy, and subsequent policies exist, the next policy is evaluated.
- If a match does not occur by the end of a policy or all policies, the route is accepted.

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

Each route is evaluated against the policy as follows:

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

Match Condition: From

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

The following table describes the match conditions supported by Contrail.

Match Condition	User Input	Description
Prefix	List of prefixes to match	<p>Each prefix in the list is represented as prefix and match type, where the prefix match type can be:</p> <ul style="list-style-type: none"> • exact • orlonger • longer <p>Example: 1.1.0.0/16 orlonger</p> <p>A route matches this condition if its prefix matches any of the prefixes in the list.</p>
Community	Community string to match	<p>Represented as either a well-known community string with no export or no reoriginate, or a string representation of a community (64512:11).</p>
Protocol	Array of path source or path protocol to match	<p>BGP XMPP StaticRoute ServiceChain Aggregate.</p> <p>A path is considered as matching this condition if the path protocol is one of protocols in the list.</p>

Routing Policy Action and Update Action

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

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

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

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

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

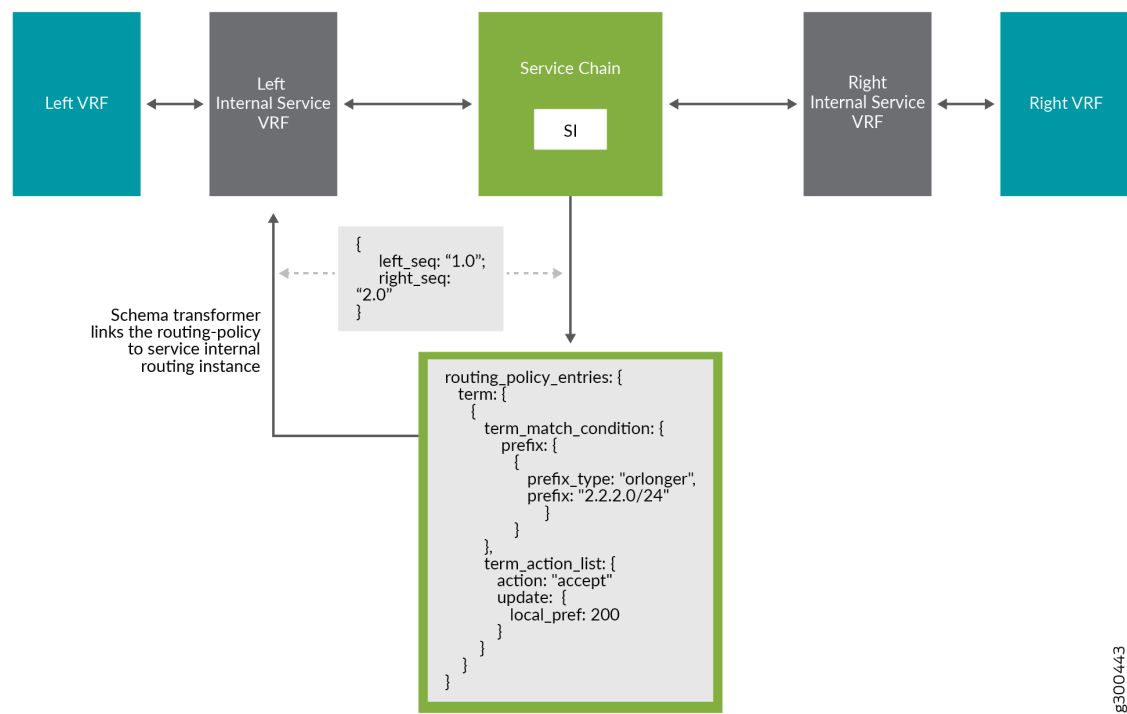
Update Action	User Input	Description
Community	List of community	<p>As part of the policy update, the following actions can be taken for community:</p> <ul style="list-style-type: none"> • Add a list of community to the existing community. • Set a list of community. • Remove a list of community (if present) from the existing community.
MED	Update the MED of the BgpPath	Unsigned integer representing the MED
local-pref	Update the local-pref of the BgpPath	Unsigned integer representing local-pref

Routing Policy Configuration

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

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

The following figure illustrates a routing policy configuration.



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

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

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

Configuring and Troubleshooting Routing Policy

IN THIS SECTION

- Create Routing Policy | 51
- Configure Service Instance | 52
- Configure the Network Policy for the Service Chain | 52

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.

The screenshot shows a 'Create Routing Policy' dialog box. The 'Name' field is filled with 'failover'. The 'Term(s)' section contains a single term: 'from: { prefix 2.2.2.0/24 orlonger } then: { local-preference 200 }'. This term is expanded to show its components. The 'From' section has a dropdown set to 'prefix', a text field with '2.2.2.0/24', and another dropdown set to 'orlonger'. The 'Then' section has a dropdown set to 'local-preference' and a text field with '200'. There are 'x' and '+' icons for removing and adding terms. At the bottom right, there is a 'Cancel' button and a 'Save' button. A vertical text 's018729' is visible on the right side of the dialog.

NOTE: The Contrail UI and REST APIs enable you to configure a BGP routing policy and then assign it to a virtual network, but the routing policy will not be applied if the virtual network is attached to an L3VPN.

Configure Service Instance

Create a service instance and attach the routing policy to both the left and right interfaces. The order of the policy is calculated by the UI, based on the order of the policy specified in the list.

Create Service Instance

ha-chain st-with-policy - [transparent (left, right)] - v1

▼ Interface Details

Interface Type Virtual Network

left Auto Configured

right Auto Configured

▼ Advanced Options

▼ Routing Policy

Interface Type	Routing Policy
left	failover
right	failover

Cancel Save

Configure the Network Policy for the Service Chain

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

Edit Policy (service-chain-policy)

Policy Name

service-chain-policy

Policy Rules

Action	Protocol	Source	Ports	Direction	Destination	Ports	Log	Services	Mirror
PASS	ANY	left	ANY	left to right	right	ANY	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Service Instance

si-aggregate ha-chain

+ Add Rule

Cancel Save

Using a VNC Script to Create Routing Policy

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

```

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

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

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

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

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

```

Verify Routing Policy in API Server

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

```
- routing_policy_entries: {
  - term: {
    - {
      - term_match_condition: {
        - prefix: {
          - {
            prefix_type: "orlonger",
            prefix: "2.2.2.0/24"
          }
        }
      },
      - term_action_list: {
        action: "accept",
        - update: {
          local_pref: 200
        }
      }
    }
  },
  + id_perms: {--},
  - routing_instance_refs: [
    - {
      - to: [
        "default-domain",
        "admin",
        "right",
        "service-ace7ae00-56e3-42d1-96ec-7fe7708d97f-default-domain_admin_ha-chain"
      ],
      href: "http://nodes27.englab.juniper.net:8082/routing-instance/32b7eed4-57ce-4c44-bbb0-513f78db6068",
      - attr: {
        sequence: "1"
      },
      uuid: "32b7eed4-57ce-4c44-bbb0-513f78db6068"
    },
    - {
      - to: [
        "default-domain",
        "admin",
        "left",
        "service-ace7ae00-56e3-42d1-96ec-7fe7708d97f-default-domain_admin_ha-chain"
      ],
      href: "http://nodes27.englab.juniper.net:8082/routing-instance/6ad868d1-a412-4765-b8c4-f93ec5d9f4b2",
      - attr: {
        sequence: "1"
      },
      uuid: "6ad868d1-a412-4765-b8c4-f93ec5d9f4b2"
    }
  ],
  - service_instance_refs: [
    - {
      - to: [
        "default-domain",
        "admin",
        "ha-chain"
      ],
      href: "http://nodes27.englab.juniper.net:8082/service-instance/983bb90b-b3f4-4d6c-be54-33a474eee7de",
      - attr: {
        left_sequence: "1",
        right_sequence: "1"
      },
      uuid: "983bb90b-b3f4-4d6c-be54-33a474eee7de"
    }
  ],
  name: "failover"
}
```

s018732

See the following example.

routing_policies				
name	generation	ref_count	terms	deleted
default-domain:admin:failover	0	2	<div> <div>terms</div> <div> <div>terminal</div> <div> <div>matches</div> <div>actions</div> </div> </div> <div> <div>true</div> <div> <div>matches</div> <div>actions</div> </div> <div> <div>prefix [2.2.0/24 orlonger]</div> <div> <div>accept</div> <div>local-pref 200</div> </div> </div> </div> </div>	false
default-domain:default-project:default-routing-policy	0	0	<div> <div>terms</div> </div>	false

Verify Routing Policy Configuration in the Control Node

See the following example.

ShowBgpRoutingPolicyConfigResp		
routing_policies		
name	terms	
default-domain:admin:failover	terms	
	match	action
	from { prefix 2.2.2.0/24 orlonger }	then { local-preference 200 accept }

Verify Routing Policy Configuration on the Routing Instance

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

Point your browser to:

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

See the following example.

service_chain_info					static_routes		aggregate_routes		routing_policies	
family	routing_instance	chain_address	prefixes	service_instance	static_routes	aggregate_routes	routing_policies		policy_name	sequence
inet	default-domain:admin:right:right	1.1.1.6	prefixes 2.2.2.0/24	default-domain:admin:ha-chain					default-domain:admin:failover	1

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

Point your browser to:

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

See the following example.

routing_policies	
routing_policies	
policy_name	generation
default-domain:admin:failover	0

Control for Route Reorigination

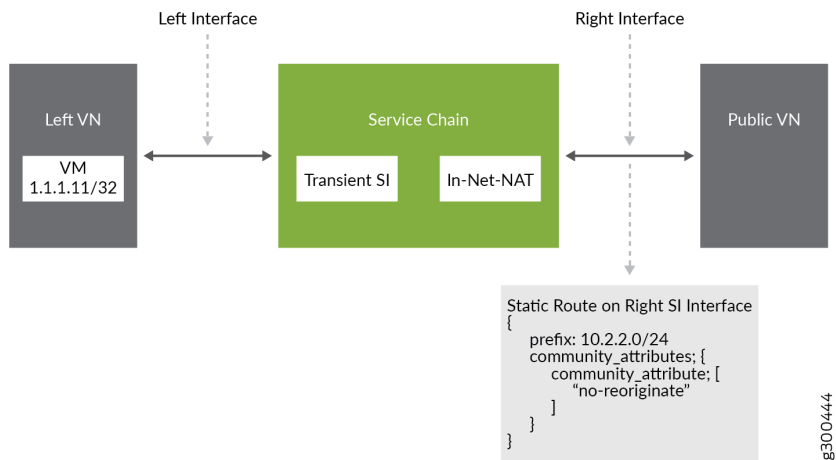
The ability to prevent reorigination of interface static routes is typically required when routes are configured on an interface that belongs to a service VM.

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

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

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

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



Configuring and Troubleshooting Reorigination Control

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

Create Service Instance

Name

si-with-static

Service Template

st-with-static - [in-network-nat (left, right)] - v1

Interface Type

left

Virtual Network

Select Virtual Network

Interface Type

right

Virtual Network

Select Virtual Network

+ Add Static Routes

Prefix

10.2.2.0/24

Next Hop

Interface 2

Community

no-reoriginate

Routing Policy

Interface Type

Routing Policy

Cancel

Save

5018737

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

```
{
  - service-instance: {
    + virtual_machine_back_refs: [...],
    + fq_name: [...],
    uuid: "a6e1e71f-f828-43de-a493-b193bdb73ded",
    parent_type: "project",
    parent_uuid: "634f90d9-da62-4c2f-a238-7cc1c1a055a5",
    parent_href: "http://nodeg2:8082/project/634f90d9-da62-4c2f-a2",
    - service_instance_properties: {
      right_virtual_network: "default-domain:admin:twig",
      - interface_list: [
        - {
          virtual_network: "default-domain:admin:fifo"
        },
        - {
          virtual_network: "default-domain:admin:twig",
          - static_routes: {
            - route: [
              - {
                prefix: "10.2.2.0/24",
                next_hop: null,
                - community_attributes: {
                  - community_attribute: [
                    "no-reoriginate"
                  ]
                },
                next_hop_type: null
              }
            ]
          }
        }
      ]
    },
    left_virtual_network: "default-domain:admin:fifo",
    - scale_out: {
      max_instances: 1
    }
  },
}
```

s018738

RELATED DOCUMENTATION

Creating a Routing Policy With Extended Communities in Contrail Command | 60

Creating a Routing Policy With Extended Communities in Contrail Command

Contrail Release 5.1 supports extended communities on the import routing policy function. Release 5.1 allows import routing policy terms to match on extended communities and import routing policy actions to add, set, and remove extended communities. Filtering routes based on extended communities prevent advertising unnecessary service interface and static routes from the control node.

The following extended communities are supported:

- Route Target
- Encapsulation
- Security Group
- Origin VN
- MAC Mobility
- Load Balance
- Tag

For information on these extended communities, see [BGP Extended Communities](#).

Creating a Routing Policy

This section shows how to create a routing policy for a virtual network with extended communities.

1. Click the **Add** button in **Overlay > Routing > Routing Policies**.
2. Enter routing policy information according to the guidelines provided in [Table 3 on page 61](#)
3. Click **Create** to create the routing policy.

The **Routing Policies** tab is displayed listing the newly created policy.

4. Navigate to the **Overlay > Virtual Networks** page.
5. Select the check box for the virtual network that you want to attach the routing policy to, and click the **Edit** icon.

The **Edit Virtual Network** page appears.

6. Scroll down to the **Routing, Bridging, and Policies** section in the **Network** tab, and select the newly created routing policy in the **Routing Policies** field.
7. Click **Save** to add the routing policy to the virtual network.

Table 3: Create Routing Policy

Field	Guidelines
Name	Enter a name for the routing policy.
<i>Term(s)</i>	
Community	Select the community string to match for the routing policy. The community string is represented with accept-own , no-advertise , no-export , no-export-subconfed , no-reoriginate ..
Match All	Select the check box to match all the community strings.
Extended Community	Select the extended community string to match for the routing policy.
Match All	Select the check box to match the extended community strings.
Protocol	Select the protocol for the routing policy which is an array of path source or path protocol to match. The protocols are interface , aggregate , bgp , BGPaaS , interface-static , service-chain , service-interface , static , and xmpp . A path is considered as matching this condition if the path protocol is one of protocols in the list.
Prefixes	<p>Select a list of prefixes to match.</p> <p>Each prefix in the list is represented as prefix and match type, where the prefix match type can be:</p> <ul style="list-style-type: none"> • exact • orlonger • longer <p>Example: 1.1.0.0/16 orlonger</p> <p>A route matches this condition if its prefix matches any of the prefixes in the list.</p>
<i>Then</i>	

Table 3: Create Routing Policy (*continued*)

Field	Guidelines
Actions	

Table 3: Create Routing Policy (*continued*)

Field	Guidelines																
	<p>Select the actions to be performed on the matching routes. The supported actions and the values are:</p> <table> <tr> <th>Action</th><th>Value</th></tr> <tr> <td>action</td><td> <p>Reject-Reject the route that matches this term. No more terms are evaluated after hitting this term.</p> <p>Accept-Accept the route that matches this term. No more terms are evaluated after hitting this term.</p> <p>Next-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.</p> </td></tr> <tr> <td> add community Add a list of community to the existing community. </td><td> <p>The community is of type unsigned 32 bit integer:unsigned 32 bit integer.</p> <p>For example, 64512:55555.</p> </td></tr> <tr> <td> add extended community Add a list of extended community to the existing community. </td><td> <p>An eight octets string of value type:administrator:assigned-number where type is two octets, administrator is four octets, and assigned number is two octets.</p> </td></tr> <tr> <td> as-path Select different AS paths to control routing decisions </td><td> <p>Unsigned 32-bit integer representing the as-path.</p> <p>For example, 444.</p> </td></tr> <tr> <td> local-preference Select the local preference to distinguish routes and take further action. </td><td> <p>Unsigned 32-bit integer representing local-preference.</p> <p>For example, 444.</p> </td></tr> <tr> <td> med Select the MED of the BgpPath. </td><td> <p>Unsigned 32-bit integer representing the MED.</p> <p>For example, 444.</p> </td></tr> <tr> <td> remove community Remove a list of community (if present) from the existing community. </td><td> <p>The community is of type unsigned 32 bit integer:unsigned 32 bit integer.</p> </td></tr> </table>	Action	Value	action	<p>Reject-Reject the route that matches this term. No more terms are evaluated after hitting this term.</p> <p>Accept-Accept the route that matches this term. No more terms are evaluated after hitting this term.</p> <p>Next-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.</p>	add community Add a list of community to the existing community.	<p>The community is of type unsigned 32 bit integer:unsigned 32 bit integer.</p> <p>For example, 64512:55555.</p>	add extended community Add a list of extended community to the existing community.	<p>An eight octets string of value type:administrator:assigned-number where type is two octets, administrator is four octets, and assigned number is two octets.</p>	as-path Select different AS paths to control routing decisions	<p>Unsigned 32-bit integer representing the as-path.</p> <p>For example, 444.</p>	local-preference Select the local preference to distinguish routes and take further action.	<p>Unsigned 32-bit integer representing local-preference.</p> <p>For example, 444.</p>	med Select the MED of the BgpPath.	<p>Unsigned 32-bit integer representing the MED.</p> <p>For example, 444.</p>	remove community Remove a list of community (if present) from the existing community.	<p>The community is of type unsigned 32 bit integer:unsigned 32 bit integer.</p>
Action	Value																
action	<p>Reject-Reject the route that matches this term. No more terms are evaluated after hitting this term.</p> <p>Accept-Accept the route that matches this term. No more terms are evaluated after hitting this term.</p> <p>Next-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.</p>																
add community Add a list of community to the existing community.	<p>The community is of type unsigned 32 bit integer:unsigned 32 bit integer.</p> <p>For example, 64512:55555.</p>																
add extended community Add a list of extended community to the existing community.	<p>An eight octets string of value type:administrator:assigned-number where type is two octets, administrator is four octets, and assigned number is two octets.</p>																
as-path Select different AS paths to control routing decisions	<p>Unsigned 32-bit integer representing the as-path.</p> <p>For example, 444.</p>																
local-preference Select the local preference to distinguish routes and take further action.	<p>Unsigned 32-bit integer representing local-preference.</p> <p>For example, 444.</p>																
med Select the MED of the BgpPath.	<p>Unsigned 32-bit integer representing the MED.</p> <p>For example, 444.</p>																
remove community Remove a list of community (if present) from the existing community.	<p>The community is of type unsigned 32 bit integer:unsigned 32 bit integer.</p>																

Table 3: Create Routing Policy (continued)

Field	Guidelines	
	remove extended community Remove a list of extended community (if present) from the existing community.	An eight octets string of value type:administrator:assigned-number where type is two octets, administrator is four octets, and assigned number is two octets.
	set community Set a list of community.	The community is of type unsigned 32 bit integer:unsigned 32 bit integer .
	set extended community Set a list of extended community.	An eight octets string of value type:administrator:assigned-number where type is two octets, administrator is four octets, and assigned number is two octets.

RELATED DOCUMENTATION

| [Routing Policy](#) | 45

Service Instance Health Checks

IN THIS SECTION

- [Health Check Object](#) | 65
- [Bidirectional Forwarding and Detection Health Check over Virtual Machine Interfaces](#) | 69
- [Bidirectional Forwarding and Detection Health Check for BGPaaS](#) | 69
- [Health Check of Transparent Service Chain](#) | 70
- [Service Instance Fate Sharing](#) | 70

In Contrail Release 3.0 and greater, a service instance health check can be used to determine the liveliness of a service provided by a virtual machine (VM).

Health Check Object

IN THIS SECTION

- [Health Check Overview | 65](#)
- [Health Check Object Configuration | 65](#)
- [Creating a Health Check with the Contrail User Interface | 67](#)
- [Using the Health Check | 68](#)
- [Health Check Process | 68](#)

Health Check Overview

The service instance health check is used to determine the liveliness of a service provided by a VM, checking whether the service is operationally up or down. The vRouter agent uses ping and an HTTP URL to the link-local address to check the liveliness of the interface.

If the health check determines that a service is no longer operational, it removes the routes for the VM, thereby disabling packet forwarding to the VM.

The service instance health check is used with service template version 2.

Health Check Object Configuration

[Table 4 on page 65](#) shows the configurable properties of the health check object.

Table 4: Health Check Configurable Parameters

Field	Description
- enabled	Indicates that health check is enabled. The default is False .
- health-check-type	Indicates the health check type: link-local , end-to-end , bgp-as-a-service , and so on.. The default is link-local .
- monitor-type	The protocol type to be used: PING or HTTP .
- delay	The delay, in seconds, to repeat the health check.
- timeout	The number of seconds to wait for a response.

Table 4: Health Check Configurable Parameters (*continued*)

Field	Description
- max-retries	The number of retries to attempt before declaring an instance health down.
- http-method	When the monitor protocol is HTTP, the type of HTTP method used, such as GET, PUT, POST, and so on.
- url-path	When the monitor protocol is HTTP, the URL to be used. For all other cases, such as ICMP, the destination IP address.
- expected-codes	When the monitor protocol is HTTP, the expected return code for HTTP operations.

Health Check Modes

The following modes are supported for the service instance health check:

- **link-local**—A local check for the service VM on the vRouter where the VM is running. In this case, the source IP of the packet is the service chain IP.
- **end-to-end**—A remote address or URL is provided for a service health check through a chain of services. The destination of the health check probe is allowed to be outside the service instance. However, the health check probe must be reachable through the interface of the service instance where the health check is attached. The end-to-end health check probe is transmitted all the way to the actual destination outside the service instance. The response to the health check probe is received and processed by the service health check to evaluate the status.

Restrictions include:

- This check is applicable for a chain where the services are not scaled out.
- When this mode is configured, a new health check IP is allocated and used as the source IP of the packet.
- The health check IP is allocated per **virtual-machine-interface** of the service VM where the health check is attached.
- The agent relies on the **service-health-check-ip** flag to use as the source IP.

NOTE: In versions prior to Contrail 4.1, end-to-end health check is not supported on a transparent service chain. However, a link-local health check is possible on a transparent service instance if the corresponding service instance interface is configured with its IP address. Contrail 4.1 supports a segment-based health check for transparent service chain.

Creating a Health Check with the Contrail User Interface

To create a health check with the Contrail Web UI:

- 1. Navigate to **Configure > Services > Health Check Service**, and click to open the **Create** screen. See [Figure 6 on page 67](#).

Figure 6: Create Health Check Screen

Create

Health Check ServicePermissions

Name

ext_hc_service

Protocol

PING

Monitor Target

8.8.8.8

Delay (secs)

3

Timeout (secs)

5

Retries

2

Health Check Type

End-To-End

Cancel

Save

- 2. Complete the fields to define the permissions for the health check, see [Table 5 on page 67](#).

Table 5: Create Health Check Fields

Field	Description
Name	Enter a name for the health check service you are creating.
Protocol	Select from the list the protocol to use for the health check, PING, HTTP, BFD, and so on.
Monitor Target	Select from the list the address of the target to be monitored by the health check.
Delay (secs)	The delay, in seconds, to repeat the health check.

Table 5: Create Health Check Fields (*continued*)

Field	Description
Timeout (secs)	The number of seconds to wait for a response.
Retries	The number of retries to attempt before declaring an instance health down.
Health Check Type	Select from the list the type of health check—link-local, end-to-end, segment-based, bgp-as-a-service, and so on.

Using the Health Check

A REST API can be used to create a health check object and define its associated properties, then a link is added to the VM interface.

The health check object can be linked to multiple VM interfaces. Additionally, a VM interface can be associated with multiple health check objects. The following is an example:

```
HealthCheckObject 1 ----- VirtualMachineInterface 1 -----
HealthCheckObject 2
|
|
VirtualMachineInterface 2
```

Health Check Process

The Contrail vRouter agent is responsible for providing the health check service. The agent spawns a Python script to monitor the status of a service hosted on a VM on the same compute node, and the script updates the status to the vRouter agent.

The vRouter agent acts on the status provided by the script to withdraw or restore the exported interface routes. It is also responsible for providing a link-local metadata IP for allowing the script to communicate with the destination IP from the underlay network, using appropriate NAT translations. In a running system, this information is displayed in the vRouter agent introspect at:

http://<compute-node-ip>:8085/Snh_HealthCheckSandeshReq?uuid=

NOTE: Running health check creates flow entries to perform translation from underlay to overlay. Consequently, in a heavily loaded environment with a full flow table, it is possible to observe false failures.

Bidirectional Forwarding and Detection Health Check over Virtual Machine Interfaces

Contrail Release 4.1 and later support for BFD-based health checks for VMIs.

Health check for VMIs is already supported as poll-based checks with ping and curl commands. When enabled, these health checks run periodically, once every few seconds. Consequently, failure detection times can be quite large, always in seconds.

Health checks based on the BFD protocol provide failure detection and recovery in sub-second intervals, because applications are notified immediately upon BFD session state changes.

If BFD-based health check is configured, whenever a BFD session status is detected as **Up** or **Down** by the health-checker, corresponding logs are generated.

Logging is enabled in the `contrail-vrouter-agent.conf` file with the log severity level **SYS_NOTICE**.

You can view the log file in the location `/var/log/contrail/contrail-vrouter-agent.log`

Snippet of sample log message related to BFD session events

```
2019-02-26 Tue 14:38:49:417.479 SYS_NOTICE BFD session Down interface:
test-bfd-hc-vmi.st2 vrf: default-domain:admin:VN.hc.st2:VN.hc.st2
2019-02-26 Tue 14:38:49:479.733 PST SYS_NOTICE BFD session Up interface:
test-bfd-hc-vmi.st2 vrf: default-domain:admin:VN.hc.st2:VN.hc.st2
```

Bidirectional Forwarding and Detection Health Check for BGPaaS

Contrail Release 4.1 adds support for BFD-based health check for BGP as a Service (BGPaaS) sessions.

This health check should not be confused with the BFD-based health check over VMIs feature, also introduced in Release 4.1. The BFD-based health check for VMIs cannot be used for a BGPaaS session, because the session shares a tenant destination address over a set of VMIs, with only one VMI active at any given time.

When the BFD-based health check for BGP as a Service (BGPaaS) is configured, any time a BFD-for-BGP session is detected as down by the health-checker, corresponding logs and alarms are generated.

To enable this health check, configure the **ServiceHealthCheckType** property and associate it with a `bgp-as-a-service` configuration object. This can also be accomplished in the Contrail WebUI.

Health Check of Transparent Service Chain

Contrail 4.1 enhances service chain redundancy by implementing an end-to-end health check for the transparent service chain. The service health check monitors the status of the service chain and if there is a failure, the control node no longer considers the service chain as a valid next hop, triggering traffic failover.

A segment-based health check is used to verify the health of a single instance in a transparent service chain. The user creates a service-health-check object, with type segment-based, and attaches it to either the left or right interface of the service instance. The service health check packet is injected to the interface to which it is attached. When the packet comes out of the other interface, a reply packet is injected on that interface. If health check requests fail after 30-second retries, the service instance is considered unhealthy and the service VLAN routes of the left and right interfaces are removed. When the agent receives health check replies successfully, it adds the retracted routes back onto both interfaces, which triggers the control node to start reoriginating routes to other service instances on that service chain.

For more information, see https://github.com/Juniper/contrail-specs/blob/master/transparent_sc_health_check.md

Service Instance Fate Sharing

A service chain contains multiple service instances (SI) and the failure of a single SI can cause a null route. In releases prior to Contrail Release 5.0, when an SI fails, the service chain continues to forward packets and routes reoriginate on both sides of the service chain. The packets are dropped in the SI or by the vRouter causing a null route.

Starting in Contrail Release 5.0, when one or more than one SI in a service chain fails, reorigination of routes on both sides of the service chain is stopped and routes automatically converge to a backup service chain that is part of another Contrail cluster. SI fate sharing brings down the service chain and the gateway nodes automatically reroutes traffic to an alternate cluster.

Starting in Contrail Release 4.1, **segment-based** health check type is used to verify the health of a SI in a service chain. To identify a failure of an SI, segment-based health check is configured either on the egress or ingress interface of the SI. When SI health check fails, the vRouter agent drops an SI route or a connected route. A connected route is also dropped if the vRouter agent restarts due to a software failure, when a compute node reboots, or when long-lived graceful restart (LLGR) is not enabled. You can detect an SI failure by keeping track of corresponding connected routes of the service chain address.

NOTE: When an SI is scaled out, the connected route for an SI interface goes down only when all associated VMs have failed.

The control node uses the **service-chain-id** in **ServiceChainInfo** to link all SIs in a service chain. When the control node detects that any SI of the same service-chain-id is down, it stops reoriginating routes in egress and ingress directions for all SIs. The control node reoriginates routes only when the connected routes of all the SIs are up.

ECMP Support in Service Chain

IN THIS SECTION

- [Service Chain with Equal-Cost Multipath in Active-Active Mode | 71](#)
- [Service Chain with Health Check | 71](#)

Equal-cost multipath (ECMP) can be used to distribute traffic across VMs.

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

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 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 [“Health Check Object” on page 65](#).

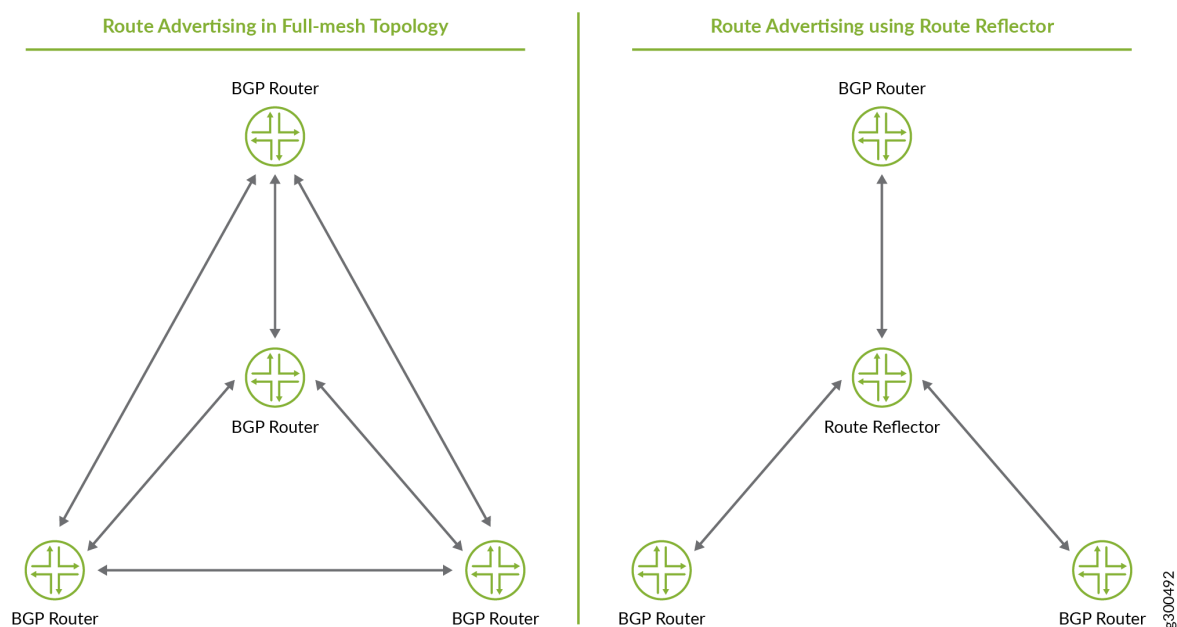
RELATED DOCUMENTATION

Route Reflector Support in Contrail Control Node

Contrail Release 5.1 supports Route Reflector (RR) functionality in the Control node for Internal Border Gateway Protocol (iBGP) peers. Route reflection is a BGP feature that enables BGP routers to acquire route information from one iBGP router and reflect or advertise the information to other iBGP peers in the same autonomous system (AS).

In previous releases, all BGP routers peering with the Control Node were deployed in a full-mesh topology. In a large scale AS, full-mesh topology affects scalability and leads to maintenance and configurational issues. The issues are caused while exchanging large volumes of routing information and maintaining connectivity among a large number of devices in the AS. A Route Reflector provides a scalable alternative to full-mesh internal BGP peering. See [Figure 7 on page 72](#).

Figure 7: Advantage of BGP Route Reflector over Full-mesh Topology



You can create any number of RRs in the network. Each RR must have a unique cluster ID to prevent looping among the RRs.

Contrail nodes support Edge Reflection Multicasting Virtual Private Networking (ERMVPN) of the BGP Address Family. As non-Contrail nodes do not support ERMVPN, RR is configured separately for Contrail nodes, and external BGP speakers. If a single RR is deployed between Contrail nodes and non-Contrail nodes, the RR cannot advertise ERMVPN routes to the Contrail nodes. Contrail deploys a separate RR peering session among Contrail nodes that supports ERMVPN and helps in propagating routes among all Contrail nodes.

Benefits of RRs in Contrail

- RRs can be deployed at multiple locations in the network, which helps to scale the BGP network at lower cost.
- The RR feature conserves data center rack space by replacing physical route reflectors.

RELATED DOCUMENTATION

[Configuring Route Reflectors from Contrail Command | 73](#)

Configuring the Control Node with BGP

Configuring Route Reflectors from Contrail Command

Starting with Release 5.1, you can configure a control node as a route reflector from the Contrail Command user interface (UI).

Follow these steps to configure a route reflector from Contrail Command UI:

1. Click the **Infrastructure > Cluster** page.

The **Overview** tab is displayed.

2. Click the **Advanced Options** tab.

The **Global Config** tab is displayed, which lists all system configuration information.

3. Click the **BGP Routers** tab.

A list of control nodes are displayed. See [Figure 8 on page 74](#).

Figure 8: BGP Routers List

IP ADDRESS	ROUTER TYPE	VENDOR	HOST NAME	CONTROL NODE ZONE
<input checked="" type="checkbox"/> 192.0.2.241	control-node	contrail	nodec4	Edit
<input type="checkbox"/> 192.0.2.230	control-node	contrail	nodec5	...
<input type="checkbox"/> 192.0.2.233	control-node	contrail	nodec6	...

1 item selected | [Select all](#) | [Deselect all](#)

4. Select the desired control node from the check box and click the **Edit** icon.

The **BGP** tab in the **Edit BGP Router** page is displayed.

5. To configure the control node as route reflector, you must assign a **Cluster ID** value in IPv4 format.
See [Figure 9 on page 75](#).

Figure 9: Add Cluster ID value

INFRASTRUCTURE > Cluster > Advanced > Edit BGP Router

default-domain > default-project > Admin

BGP Tags Permissions

Router Type: Control Node

Host Name*: nodec4

Vendor ID*: contrail

IP Address*: 192.0.2.241

Router ID*: 192.0.2.241

Autonomous System*: 64512

BGP Router ASN: ASN Range: 1-65535

Address Families: route-target, inet-vpn, erm-vpn, inet6-vpn

Cluster Id: 192.0.2.241

Associate Peers

Advanced Options

Save Cancel

6. Click **Save**.

The **Cluster ID** information is saved and the **BGP Routers** tab is displayed.

The selected control node starts functioning as a route reflector.

RELATED DOCUMENTATION

[Route Reflector Support in Contrail Control Node](#) | 72

BGP as a Service

IN THIS SECTION

- [Understanding BGP as a Service | 76](#)
- [Configuring BGPaaS using VNC API | 80](#)
- [Configuring BGPaaS from Contrail Web UI | 81](#)
- [Configuring BGPaaS from Contrail Command | 82](#)

Understanding BGP as a Service

IN THIS SECTION

- [Contrail BGPaaS Features | 76](#)
- [BGPaaS Use Cases | 78](#)

The BGP as a Service (BGPaaS) feature allows a guest virtual machine (VM) to place routes in its own virtual routing and forwarding (VRF) instance using BGP.

Contrail BGPaaS Features

Using BGPaaS with Contrail requires the guest VM to have connectivity to the control node and to be able to advertise routes into the VRF instance.

With the BGPaaS feature:

- The vRouter agent is able to accept BGP connections from the VMs and proxy them to the control node.
- The vRouter agent always selects one of the control nodes that it is using as an XMPP server.

Starting with Contrail Release 3.0, the following features have been added to BGPaaS:

- All BGPaaS sessions are configured to have bidirectional exchange of routes.
- If inet6 routes are being advertised to the tenant VM, they are advertised with the IPv6 subnet's default gateway address as the BGP next hop.

- If multiple tenant VMs in the same virtual network have BGPaaS sessions and they use eBGP, standard loop prevention rules prevent routes advertised by one tenant VM from being advertised to other tenant VMs

A second BGP session for high availability can also be configured appropriately using one more BGP router object in the Contrail configuration and the peering session (from the VNF's point of view) to the DNS IP address (reserved by Contrail).

The following are caveats:

- BGP sessions must use IPv4 transport.
- The VNF must support RFC 2545, *Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing*, to carry IPv6 routes over the IPv4 peer.
- Only IPv4 (inet) and IPv6 (inet6) address families are supported.

The initial implementation of BGPaaS Version 1, supported in Contrail Release 3.0, allowed a tenant VM to establish BGP sessions to the default gateway and DNS server in the VM's subnet. A limitation of this implementation was that the tenant VM could advertise routes into the virtual network to which the VM belonged, however, the VM could not receive any routes. The tenant VM was required to use a static default route, with the subnet's default gateway as the next hop.

Contrail Release 3.1 eliminates the previous limitation and provides route export functionality for BGPaaS sessions. The next hop for all routes advertised to the tenant VM is set to the default gateway address of the subnet of the tenant VM. This allows the tenant BGP implementation to be relatively simple, by not requiring support for recursive resolution of BGP next hops.

The BGPaaS object is associated with a virtual machine interface (VMI), not just a virtual machine (VM), which enables a tenant VM to have BGP sessions in multiple virtual networks, if required.

Starting with Contrail Release 3.1, the following features and properties have been added to BGPaaS:

- By default, all BGPaaS sessions are configured to have bidirectional exchange of routes. The Boolean property **bgpaas-suppress-route-advertisement** ensures no advertisement of routes to the tenant VM.
- If inet6 routes are being advertised to the tenant VM, they are advertised with the IPv6 subnet's default gateway address as the BGP next hop. A Boolean property, **bgpaas-ipv4-mapped-ipv6-nexthop**, causes the IPv4 subnet's default gateway, in IPv4-mapped IPv6 format, to be used instead as the next hop.
- If multiple tenant VMs in the same virtual network have BGPaaS sessions and they use eBGP, the standard BGP AS path loop prevention rules prevent routes advertised by one tenant VM from being advertised to the other tenant VMs. The **as-override** field, added to the existing **BgpSessionAttributes** in the BGPaaS object, causes the control node to replace the AS number of the tenant VM with its own AS number, when advertising routes learned from a tenant VM to another tenant VM in the same virtual network. The tenant VM does not need to implement any new functionality.

Starting with Contrail Networking Release 5.1, to better support high availability (HA) architectures, BGPaaS supports control node zone selection, with options available to configure BGPaaS control node zone peers.

This capability enables you to set up primary and secondary control node zones, which can have one or more control nodes. The reason for this is because BGPaaS is often being relied upon to provide routing to and from VNFs, which are comprised of several nodes across different computes, and the VNFs usually rely upon two BGP peers for HA. These control node zone features increase the robustness and failover capabilities for BGPaaS in Contrail.

For BGPaaS configuration in Contrail Networking Release 5.1, the following features and properties are supported:

- Global-System-Config has an option to add, modify, or delete control node zones
- Control-Node-Zone has an option to add, modify, or delete control nodes
- Control node has an option to add, modify or delete a control node zone and it can have only one control node zone
- BGPaaS has an option to add, modify, or delete a primary or secondary control node zone
- If control node zone has more than one control-node, selection of control-node for BGP Peering is random in a control node zone
- Using just one control node in each zone, VNF can predictably establish bgp-peering to that particular control node. In prior releases, selection of control-node was random, which could cause issues during planning and operation.

BGPaaS Use Cases

IN THIS SECTION

- [Dynamic Tunnel Insertion Within a Tenant Overlay | 78](#)
- [Dynamic Network Reachability of Applications | 79](#)
- [Liveness Detection for High Availability | 79](#)

This section provides example scenarios for implementing BGPaaS with Contrail.

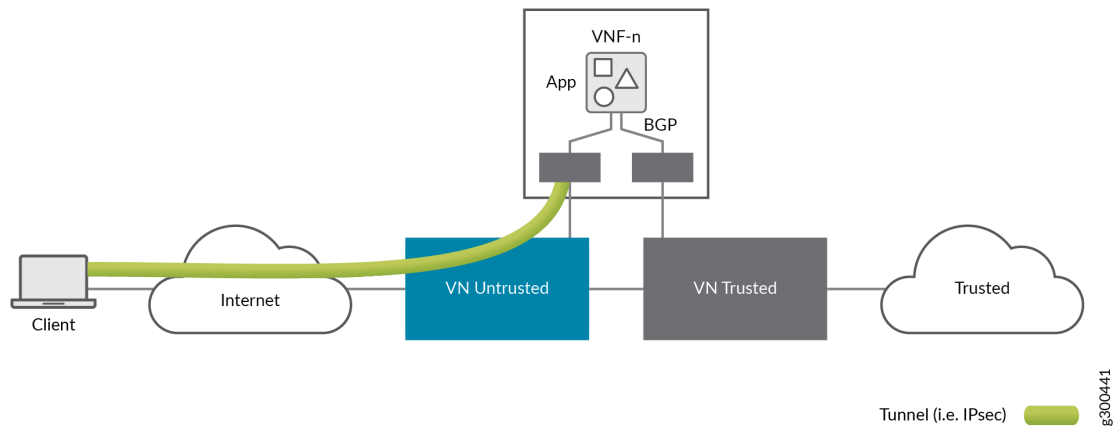
Dynamic Tunnel Insertion Within a Tenant Overlay

Various applications need to insert dynamic tunnels into virtual networks. Virtual network functions (VNFs) provide the function of tunnel termination. Tunnel termination types vary across application types, such as business VPN, mobility small site backhaul, VPC, and the like. The key requirement is that tunnels need to insert dynamically new network reachability information into the virtual network. The predominant methods of tunnel network reachability insertion use BGP.

BGPaaS allows the migration of brownfield VNFs into Contrail, preserving the application behavior and requirement for BGP, without rewriting the application.

Figure 10 on page 79 shows the need to insert a dynamic tunnel into a virtual network.

Figure 10: Dynamic Tunnel Insertion



Dynamic Network Reachability of Applications

The Domain Name System (DNS) is a widespread application that uses BGP as a mechanism to tune reachability of its services, based on metrics such as load, maintenance, availability, and the like. As DNS services are migrated to environments using overlays, a mechanism to preserve the existing application behavior and requirements is needed, including the ability to announce and withdraw reachability to the available application.

This requirement is not limited to DNS. Other applications, such as virtualized evolved packet core (vEPC) and others, use BGP as a mechanism for network reachability based on availability and load.

Liveness Detection for High Availability

Various keepalive mechanisms for tenant reachability have been provided by network components such as BGP, OSPF, PING, VRRP, BFD, or application-specific mechanisms. With BGP on the vRouter agent, BGP can be used to provide a liveness detection mechanism between the tenant on the local compute node and the services that the specific tenant VM is providing.

Configuring BGPaaS using VNC API

To configure BGPaaS using VNC APIs:

1. Access the default project.

```
default_project = self._vnc_lib.project_read(fq_name=[u'default-domain', 'bgpaas-tenant'])
```

2. Create a BGPaaS object.

```
bgpaas_obj = BgpAsAService(name='bgpaas_1', parent_obj=default_project)
```

3. Attach the BGP object to a precreated VMI.

```
bgpaas_obj.add_virtual_machine_interface(vmi)
```

4. Set the ASN. It must be an eBGP session.

```
bgpaas_obj.set_autonomous_system('65000')
```

If the ASN is not set, the primary instance IP will be chosen.

```
bgpaas_obj.set_bgpaas_ip_address(u'10.1.1.5')
```

5. Set session attributes.

```
bgp_addr_fams = AddressFamilies(['inet', 'inet6'])
```

```
bgp_sess_attrs = BgpSessionAttributes(address_families=bgp_addr_fams,hold_time=60)
```

```
bgpaas_obj.set_bgpaas_session_attributes(bgp_sess_attrs)
```

```
self._vnc_lib.bgp_as_a_service_create(bgpaas_obj)
```


To delete a BGPaaS object, follow the given code:

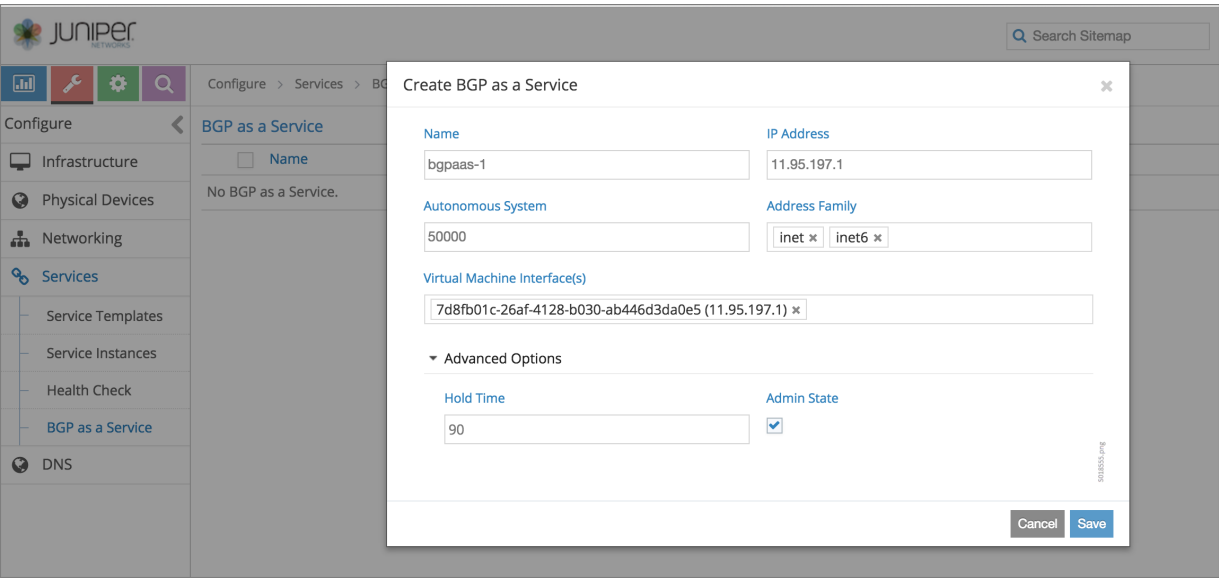
```
fq_name=[u'default-domain', 'bgpaas-tenant', 'bgpaas_1']
bgpaas_obj = self._vnc_lib.bgp_as_a_service_read(fq_name=fq_name)
bgpaas_obj.del_virtual_machine_interface(vmi)
self._vnc_lib.bgp_as_a_service_update(bgpaas_obj)
self._vnc_lib.bgp_as_a_service_delete(id=bgpaas_obj.get_uuid())
```

Configuring BGPaaS from Contrail Web UI

To configure BGPaaS within a tenant:

- 1. Select **Configure > Services > BGP as a Service** from the Contrail Web User Interface (UI). The BGP as a Service page is displayed.
- 2. Click the + button on the **BGPaaS** page. The **Create BGP as a Service** page is displayed. See [Figure 11 on page 81](#).

Figure 11: Create BGP as a Service



- 3. In the **Create BGPaaS** page, populate the fields with the following values to create your service.

Fields	Description
Name	Enter a name for the BGP service The name can be a unique string of not more than 15 characters that contains alphanumeric characters and hyphen (-).

Fields	Description
IP Address	Enter the IPv4 or IPv6 source-address on the BGPaaS VM.
Virtual Machine Interface	Enter IP address of a virtual machine interface.
Address Family	Choose inet or inet6 from the Address Family list according to your requirement.
Autonomous System	Enter AS number in the range 1- 65,534.
Advanced Options	
Hold Time	Enter the maximum time a BGP session remains active if no Keepalives are received.
Admin State	Select the Admin state box to enable the state as UP and deselect the box to disable the state to DOWN.

4. Click **Save** to create the BGP object.

Configuring BGPaaS from Contrail Command

To configure BGPaaS within a tenant:

1. Select **Services > BGPaaS**. from the Contrail Command User Interface (UI) . The BGPaaS page is displayed.
2. Click the **Create** button on the **BGPaaS** page. The **Create BGPaaS** page is displayed. See [Figure 12 on page 83](#).

Figure 12: Create BGPaaS

The screenshot shows the 'Create BGPaaS' form in the Contrail Command interface. The form is titled 'BGP as a Service' and includes the following fields and options:

- Name***: A text input field.
- Virtual Machine Interface(s)**: A dropdown menu.
- Address Family***: A dropdown menu.
- Autonomous System***: A text input field with a hint 'ASN Range: 1-65535'.
- Advanced Options**:
 - IP Address**: A text input field with a hint 'Enter valid IPv4'.
 - Shared**: A checkbox.
 - Route Origin**: A dropdown menu with 'IGP' selected.
 - Route Origin Override**: A checkbox.
 - Service Health Check**: A dropdown menu.
 - Hold Time**: A text input field.
 - Loop Count**: A text input field.
 - Local ASN**: A text input field with a hint 'ASN Range: 1-65535'.
 - Admin State**: A checked checkbox.
 - AS Override**: An unchecked checkbox.
 - Use IPv4-mapped IPv6 Nexthop**: An unchecked checkbox.
 - Suppress Route Advertisement**: An unchecked checkbox.
 - Primary Control Node Zone**: A dropdown menu.
 - Secondary Control Node Zone**: A dropdown menu.

At the bottom of the form are 'Create' and 'Cancel' buttons.

3. In the **Create BGPaaS** page, populate the fields with the following values to create your BGP object.

Fields	Description
Name	Enter a name for the BGP service The name can be a unique string of not more than 15 characters that contains alphanumeric characters and hyphen (-).
Virtual Machine Interface	Enter IP address of a virtual machine interface.
Address Family	Choose inet or inet6 from the Address Family list according to your requirement.
Autonomous System	Enter AS number in the range 1-65535.
Advanced Options	
IP Address	Enter the IPv4 or IPv6 source-address on the BGPaaS VM.
Shared	Select this check box to link all VMIs with the common bgp-router object. If this box remains unchecked, each VMI individually links to it's own bgp-router object.

Fields	Description
Route Origin	<ul style="list-style-type: none"> • Choose BGP from the list if the route originated on a BGP router. • Choose EGP from the list if the route originated from an External Gateway Protocols (EGP) session. • Choose Incomplete from the list if the Network Layer Reachability Information (NLRI) is learned through some other means other than BGP, such as, redistribution of the routes into BGP.
Route Origin Override	Select this check box to override the origin attribute of the advertised route origin into Incomplete.
Service Health Check	Select any Service Health Check object from the list according to your requirement.
Hold Time	Enter the maximum time a BGP session remains active if no Keepalives are received.
Loop Count	Enter the number of times the same ASN can be seen in a route-update. The route is discarded when the loop count is exceeded.
Local ASN	Enter Local AS number in the range 1-65535.
AS Override	Select this check box to replace the AS number of the control node with the AS number of the tenant VM.
Use IPv4-mapped IPv6 Nexthop	Select this check box to use IPv4-mapped IPv6 format as the next hop instead of the IPv4 subnet's default gateway.
Suppress Route Advertisement	Select this check box to prevent advertisement of routes to tenant VM.
Primary Control Node Zone	You can choose the control-node with which the BGPaaS VM can perform a BGP session.
Secondary Control Node Zone	You can choose the control-node with which the BGPaaS VM can perform a BGP session. with

4. Click the **Create** button to create the BGP object.

Fat Flows

IN THIS SECTION

- [Understanding Fat Flow | 85](#)
- [Configuring Fat Flow from Contrail Command | 86](#)
- [Limitations of Fat Flow | 98](#)

Service Providers provide services to several subscribers and as a result, large volume of flows are processed at the Contrail vRouter-level and Contrail Agent-level. Processing large volume of flows affects the flow setup rate and increases latency. Fat flow helps reduce the number of flows that are handled by Contrail.

Understanding Fat Flow

With Release 2.22, Contrail optimizes the number of flows that are sent or received by a virtual machine by reusing a flow. A single flow pair or a fat flow comprises of a single forward and single reverse flow entry. A fat flow is used for a number of sessions between two end points that use the same application protocol.

For example, multiple DNS sessions from a client to a server can be set up by using a single flow pair. The flow key is reduced from five tuples to four tuples consisting of source IP address, destination IP address, server port, and internet protocol. This can be configured by specifying the fat flow protocol on the virtual machine interface (VMI). The client port, however, is not used in the flow key. With Contrail Release 5.0, the fat flow key can be further reduced to two tuples.

You can configure fat flows by specifying the list of fat-flow protocols on a virtual machine interface (VMI). For each such application protocol, the list contains the protocol and port pairs. If you want to enable the fat flow feature on the client side, the configuration must be applied on the client VMI as well. Contrail Release 5.0 supports configuring fat flow at the virtual network (VN) level. When configured at the VN level, the fat flow configuration is applied to all VMIs under the configured VN.

With Release 5.0, Contrail supports aggregation of multiple flows into a single flow by ignoring source and destination ports or IP addresses, with the following possible options:

- ignore source and/or destination ports
- ignore source and/or destination IP addresses

- ignore a combination of source and/or destination ports and IP addresses

Prefix-Based Fat Flow

In Contrail Release 5.0, you can configure the Ignore Address field to help reduce the number of flows. Starting in Contrail Release 5.1, you can create fat flows by configuring prefix length. Service provider subscribers in a common IP address pool can access any IP address in the pool. With the introduction of prefix-based fat flow in Release 5.1, Contrail supports mask processing where you can create flows based on a group of subscribers. This ensures that continuous flows in the same subnet are grouped into a common fat flow that is configured with the same protocol and port numbers. You can apply prefix length-based fat flow on source IP address while the Ignore Address option is configured on the destination IP address, resulting in a reduction of flow processing.

For example, you use prefix-based fat flow to create one flow for 255 IP end points in a /24 subnet (aggregate) mask or one flow for 65,535 IP end points in a /16 subnet (aggregate) mask. This results in a huge reduction on the number of flows created, and a corresponding increase in the number of traffic flows going through vRouter without being limited by vRouter flow setup rate.

Configuring Fat Flow from Contrail Command

You use the Contrail Command user interface (UI) to configure fat flow.

You can configure fat flow from:

- **Overlay>Ports** or
- **Overlay>Virtual Networks**

Configuring Fat Flow from Overlay>Ports

To configure fat flow from **Overlay>Ports**:

- 1. Click **Overlay>Ports**.

The Ports page is displayed. See [Figure 13 on page 87](#).

Figure 13: Ports Page

OVERLAY ▶ Ports

🔔

Default ▶ admin ▼

admin

Ports

🔍 ↺ 🗑

Create

<input type="checkbox"/>	NAME	UUID	TA...	N...	FL...	FL...	DE...	VL...	
▶ <input type="checkbox"/>	fa085e76-e0b7-43cb-8ca7-e3d56f03c6aa	fa085e76-e0b7-43cb-8ca7-e3d56f03c6aa	-	ri...	2...	-	b...	v...	...
▶ <input type="checkbox"/>	left_vmi_1	1045e12f-6164-46ad-8bc6-7574b629ed4c	-	le...	1...	-	n...		...
▶ <input type="checkbox"/>	left_vmi_10	a2d17fec-53b0-4636-8e71-1e99f5b49971	-	le...	1...	-	n...		...
▶ <input type="checkbox"/>	left_vmi_11	8124fe0a-3c01-4711-a343-1e4b19c3b5b2	-	le...	1...	-	n...		...
▶ <input type="checkbox"/>	left_vmi_12	9277bd19-844b-4e22-9097-f584e42994aa	-	le...	1...	-	n...		...

- 2. Select the port you want to configure by selecting the check box next to the name of the port, and then click the **Edit** icon.

The Edit Port page is displayed. See [Figure 14 on page 88](#).

Figure 14: Edit Port Page

OVERLAY ▶ Ports ▶ Edit Port 🔔 🌐 Default > 📄 admin

Port

Tags

Permissions

Port Name*

left_vmi_12

Network*

left_vn_12 ▾

Security Group

▾

Floating IPs

▾

▼ Expand All

▲ Collapse All

▶ Advanced Options

▶ DHCP Option(s)

▶ Fat Flow(s)

Save

Cancel

3. Click **Fat Flow(s)** to display the fields that you can edit.

You can edit the fields listed in [Table 6 on page 88](#).

Table 6: Edit Fat Flow(s)

Field	Action
Protocol	<div>Change the protocol that is currently being used to any one of the following protocols given in the Protocol list:</div> <ul style="list-style-type: none">• ICMP• SCTP• TCP (default)• UDP <div>You can select ICMP for both IPv4 and IPv6 traffic.</div>

Table 6: Edit Fat Flow(s) (continued)

Field	Action
Port	<p>Edit the Port field to any value between 0 through 65,535.</p> <p>Enter 0 to ignore both source and destination port numbers.</p> <p>NOTE: If you select ICMP as the protocol, the PORT field is not enabled.</p>
Ignore Address	<p>Change the Ignore Address field to any one of the following options:</p> <ul style="list-style-type: none"> • Destination—If you choose Destination as the option, Prefix Aggregation Source fields are only enabled. See Figure 15 on page 92. • None (default)—If you choose None as the option, both Prefix Aggregation Source and Prefix Aggregation Destination fields are enabled. See Figure 16 on page 92. • Source—If you choose Source as the option, Prefix Aggregation Destination fields are only enabled. See Figure 17 on page 92. <p>NOTE: With Contrail Release 5.0, fat flow is enhanced with the option to support aggregation of multiple flows into a single flow by ignoring source and destination ports or IP addresses.</p>

Table 6: Edit Fat Flow(s) (continued)

Field		Action
Prefix Aggregation Source	Source Subnet	<p>Edit source IP subnet.</p> <p>Ensure that the source subnet of the flows match. For example, to create fat flows with 192.0.2.0/24 as the subnet, enter 192.0.2.0/24 in the Source Subnet field.</p> <p>Valid range of the subnet mask: /8 through /32.</p> <p>For more information, refer to the Understanding Source and Destination section.</p>
	Prefix	<p>Edit source subnet prefix length.</p> <p>The prefix length you enter is used to aggregate flows matching the source subnet. For example, when the source subnet is 10.1.0.0/16 and prefix length is 24, the flows matching the source subnet is aggregated to 10.1.x.0/24 flows.</p> <p>Valid range of the prefix length: /(subnet mask of the source subnet) through /32.</p> <p>For more information, refer to the Understanding Source and Destination section.</p> <p>NOTE: Starting with Contrail Release 5.1, you can configure subnet and prefix length.</p>

Table 6: Edit Fat Flow(s) (continued)

Field		Action
Prefix Aggregation Destination	Destination Subnet	<p>Edit destination IP address.</p> <p>Ensure that the destination subnet of the flows match. For example, to create fat flows with 192.0.2.0/24 as the subnet, enter 192.0.2.0/24 in the Destination Subnet field.</p> <p>Valid range of the subnet mask: /8 through /32.</p> <p>For more information, refer to the Understanding Source and Destination section.</p>
	Prefix	<p>Edit destination subnet prefix length.</p> <p>The prefix length you enter is used to aggregate flows matching the destination subnet. For example, when the destination subnet is 10.1.0.0/16 and prefix length is 24, the flows matching the destination subnet is aggregated to 10.1.x.0/24 flows.</p> <p>Valid range of the prefix length: /(subnet mask of the destination subnet) through /32.</p> <p>For more information, refer to the Understanding Source and Destination section.</p> <p>NOTE: Starting with Contrail Release 5.1, you can configure subnet and prefix length.</p>

Figure 15: Ignore Address—Destination

▼ Fat Flow(s)

Protocol

TCP

Port*

3002

Ignore Address

Destination

Prefix Aggregation Source

Source Subnet

192.0.2.0/24

Prefix

24

+ Add

Figure 16: Ignore Address—None

▼ Fat Flow(s)

Protocol

TCP

Port*

3002

Ignore Address

None

Prefix Aggregation Source

Source Subnet

192.0.2.0/24

Prefix

24

Prefix Aggregation Destination

Destination Subnet

198.51.100.0/24

Prefix

24

+ Add

Figure 17: Ignore Address—Source

▼ Fat Flow(s)

Protocol

TCP

Port*

3002

Ignore Address

Source

Prefix Aggregation Destination

Destination Subnet

192.0.2.0/24

Prefix

24

+ Add

4. Click **Save** to update new configuration information.

NOTE:

Understanding Source and Destination

- **Source**—For packets from the local virtual machine, source refers to the source IP of the packet. For packets from the physical interface, source refers to the destination IP of the packet.
- **Destination**—For packets from the local virtual machine, destination refers to the destination IP of the packet. For packets from the physical interface, destination refers to the source IP of the packet.

Configuring Fat Flow from Overlay>Virtual Networks

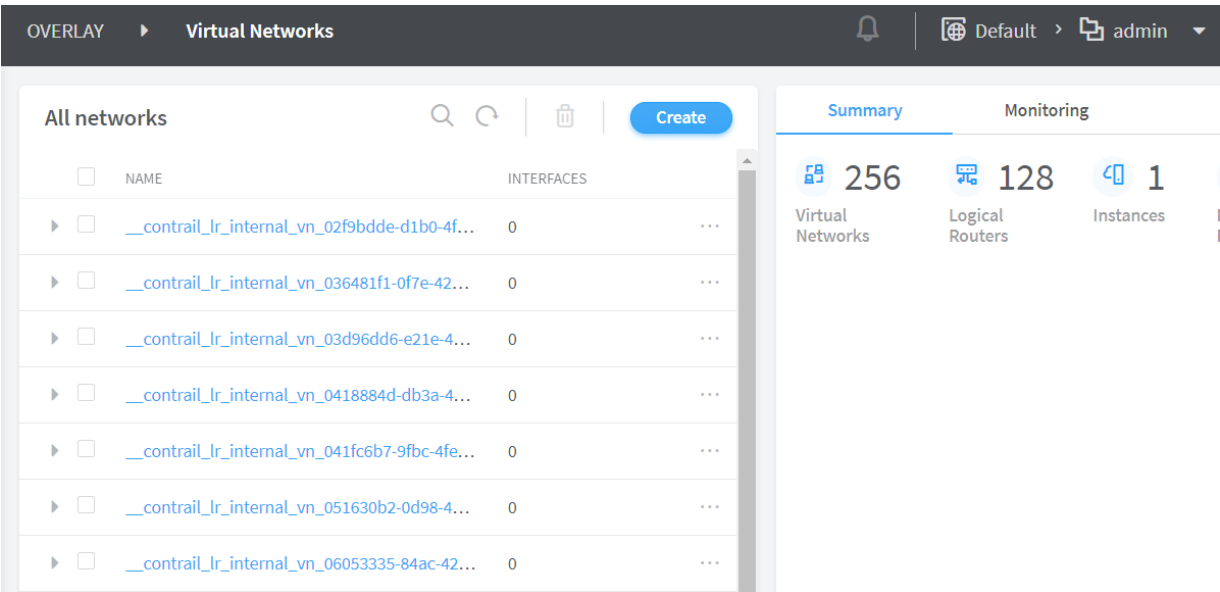
Starting with Contrail Release 5.0, you can also configure fat flow at the virtual network (VN) level. When you configure fat flow from the VN level, the fat flow configuration is applied to all VMIs under the configured VN.

To configure fat flow from **Overlay>Virtual Networks**:

1. Click **Overlay>Virtual Networks**.

The Virtual Networks page is displayed. See [Figure 18 on page 93](#).

Figure 18: Virtual Networks Page



2. Select the virtual network you want to edit by selecting the check box next to the name of the virtual network, and then click the **Edit** icon.

The Edit Virtual Network page is displayed. See [Figure 19 on page 94](#).

Figure 19: Edit Virtual Network Page

OVERLAY ▶ Virtual Networks ▶ Edit Virtual Network

Network Tags Permissions

Allocation Mode

User defined subnet only ▾

VxLAN Network Identifier

115

Subnets

+ Add

▼ Expand All ▲ Collapse All

▶ Floating IP pools

▶ Fat Flows

▶ Routing, Bridging and Policies

▶ Advanced

Save

Cancel

3. Click **Fat Flows** to display the fields that you can edit.

You can edit the fields listed in [Table 7 on page 94](#).

Table 7: Edit Fat Flows

Field	Action
Protocol	<div>Change the protocol that is currently being used to any one of the following protocols given in the Protocol list:</div> <ul style="list-style-type: none">• ICMP• SCTP• TCP (default)• UDP <div>You can select ICMP for both IPv4 and IPv6 traffic.</div>

Table 7: Edit Fat Flows (*continued*)

Field	Action
Port	<p>Edit the Port field to any value between 0 through 65,535.</p> <p>Enter 0 to ignore both source and destination port numbers.</p> <p>NOTE: If you select ICMP as the protocol, the PORT field is not enabled.</p>
Ignore Address	<p>Change the Ignore Address field to any one of the following options:</p> <ul style="list-style-type: none"> • Destination—If you choose Destination as the option, Prefix Aggregation Source fields are only enabled. See Figure 15 on page 92. • None (default)—If you choose None as the option, both Prefix Aggregation Source and Prefix Aggregation Destination fields are enabled. See Figure 16 on page 92. • Source—If you choose Source as the option, Prefix Aggregation Destination fields are only enabled. See Figure 17 on page 92. <p>NOTE: With Contrail Release 5.0, fat flow is enhanced with the option to support aggregation of multiple flows into a single flow by ignoring source and destination ports or IP addresses.</p>

Table 7: Edit Fat Flows (continued)

Field		Action
Prefix Aggregation Source	Source Subnet	<p>Edit source IP address.</p> <p>Ensure that the source subnet of the flows match. For example, to create fat flows with 192.0.2.0/24 as the subnet, enter 192.0.2.0/24 in the Source Subnet field.</p> <p>Valid range of the subnet mask: /8 through /32.</p> <p>For more information, refer to the Understanding Source and Destination section.</p>
	Prefix	<p>Edit source subnet prefix length.</p> <p>The prefix length you enter is used to aggregate flows matching the source subnet. For example, when the source subnet is 10.1.0.0/16 and prefix length is 24, the flows matching the source subnet is aggregated to 10.1.x.0/24 flows.</p> <p>Valid range of the prefix length: /(subnet mask of the source subnet) through /32.</p> <p>For more information, refer to the Understanding Source and Destination section.</p> <p>NOTE: Starting with Contrail Release 5.1, you can configure subnet and prefix length.</p>

Table 7: Edit Fat Flows (continued)

Field		Action
Prefix Aggregation Destination	Destination Subnet	<p>Edit destination IP address.</p> <p>Ensure that the destination subnet of the flows match. For example, to create fat flows with 192.0.2.0/24 as the subnet, enter 192.0.2.0/24 in the Destination Subnet field.</p> <p>Valid range of the subnet mask: /8 through /32.</p> <p>For more information, refer to the Understanding Source and Destination section.</p>
	Prefix	<p>Edit destination subnet prefix length.</p> <p>The prefix length you enter is used to aggregate flows matching the destination subnet. For example, when the destination subnet is 10.1.0.0/16 and prefix length is 24, the flows matching the destination subnet is aggregated to 10.1.x.0/24 flows.</p> <p>Valid range of the prefix length: /(subnet mask of the destination subnet) through /32.</p> <p>For more information, refer to the Understanding Source and Destination section.</p> <p>NOTE: Starting with Contrail Release 5.1, you can configure subnet and prefix length.</p>

4. (Optional) If you have not already added fat flow information, you can add information by clicking **+Add**. You can enter information as given in [Table 7 on page 94](#).
5. Click **Save** to add new configuration information.

NOTE:

- A service virtual machine (SVM) is a virtualized network function (VNF) that is a part of a service chain. Fat flow configuration on SVM is supported when:
 - Left VMI: Ignore source address and/or Prefix aggregation destination
 - Right VMI: Ignore destination address and/or Prefix aggregation source
- Fat flow on service virtual machine interfaces (SVMIs) in scale-out mode is supported when all SVMIs are on the same compute, and not on the source or destination compute.
- Fat flow configuration across all SVMs must be consistent.

Limitations of Fat Flow

The following are the limitations of fat flow.

- Drop in packet per second (pps) performance depends on the number of rules or configuration.
- Network policy configuration must be consistent with fat flow configuration.

RELATED DOCUMENTATION

[Understanding Flow Sampling | 116](#)

[Understanding Contrail Analytics](#)

[Query > Flows](#)

Use Case: Configuring Fat Flows from Contrail Command

IN THIS SECTION

- [Overview | 99](#)
- [Prerequisites | 101](#)

- [Getting Started | 102](#)
- [Configuration | 103](#)

This topic provides step-by-step instructions to create an in-network service chain and configure fat flows.

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. 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*. You can also configure fat flows while you create an in-network-NAT or transparent service chain.

Overview

IN THIS SECTION

- [Ignore Address - Source, Destination | 100](#)
- [Ignore Address - None | 101](#)

Service Providers provide services to several subscribers and as a result, large volume of flows are processed at the Contrail vRouter-level and Contrail Agent-level. Processing large volume of flows affects the flow setup rate and increases latency. Fat flow helps reduce the number of flows that are handled by Contrail.

Starting in Contrail Release 5.0, you can configure the Ignore Address field to help reduce the number of flows. With Contrail Release 5.1, you can create fat flows by configuring prefix length. Service provider subscribers in a common IP address pool can access any IP address in the pool. With the introduction of prefix-based fat flow in Release 5.1, Contrail supports mask processing where you can create flows based on a group of subscribers. This ensures that continuous flows in the same subnet are grouped into a common fat flow that is configured with the same protocol and port numbers. You can apply prefix length-based fat flow on source IP address while the Ignore Address option is configured on the destination IP address, resulting in a reduction of flow processing.

Topology Information

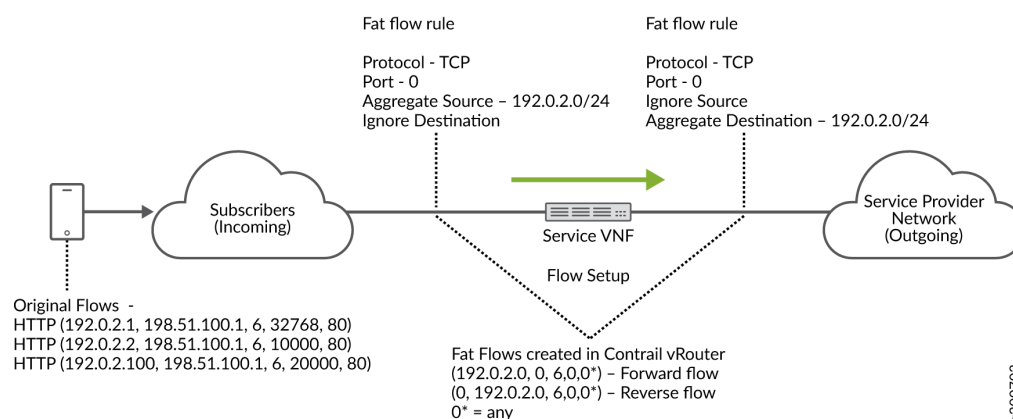
These topologies provide information on how you can configure the Ignore Address field to reduce the number of flows.

Ignore Address - Source, Destination

Figure 20 on page 100 depicts a scenario where you have selected the following options from the **Ignore Address** list.

- **Destination**—for the test-left-VN (subscribers network).
- **Source**—for the test-right-VN (service provider network).

Figure 20: Ignore Source, Destination



Understanding Source and Destination

- **Source**—For packets from the local virtual machine, source refers to the source IP of the packet.
- **Destination**—For packets from the local virtual machine, destination refers to the destination IP of the packet.

By choosing **Destination** in the subscribers network, the Prefix Aggregation Source fields are enabled in the network. And by choosing **Source** in the service providers network, the Prefix Aggregation Destination fields are enabled in the network. When you configure Ignore Address, Contrail helps you to aggregate multiple flows into a single flow by ignoring source and/or destination ports.

To create fat flows in subscribers network with 192.0.2.0/24 as the subnet, enter 192.0.2.0/24 in the **Source Subnet** field and 24 in the **Prefix** field. The prefix length, 24, is used to aggregate flows matching the source subnet. The flows matching the source subnet is aggregated to 192.0.2.X/24 flows.

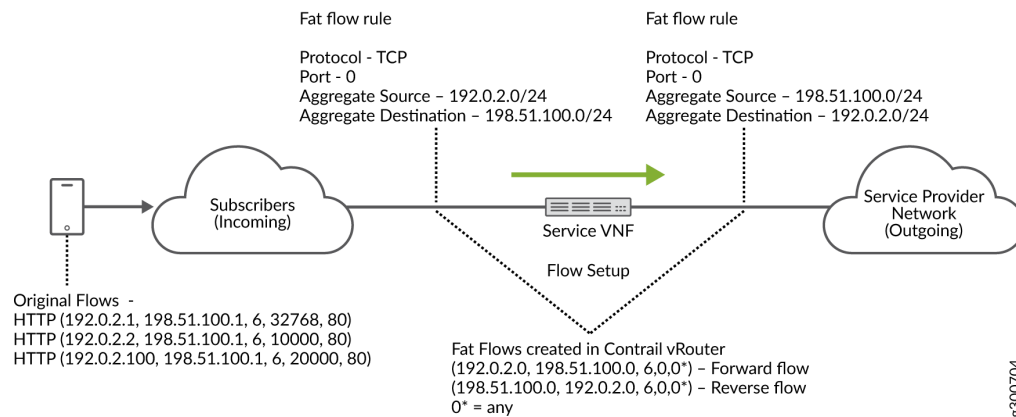
Similarly to create fat flows in service provider network with 192.0.2.0/24 as the subnet, enter 192.0.2.0/24 in the **Destination Subnet** field and 24 in the **Prefix** field. The prefix length, 24, is used to aggregate flows

matching the destination subnet. The flows matching the destination subnet is aggregated to 192.0.2.X/24 flows.

Ignore Address - None

Figure 21 on page 101 depicts a scenario where you have selected **None** from the **Ignore Address** list.

Figure 21: Ignore None



By choosing **None** in the subscribers network and service providers network, the Prefix Aggregation Destination fields and Prefix Aggregation Source fields are enabled in both networks.

In this scenario, the subnet that you enter in the Source Subnet field of the subscribers network matches the subnet that you enter in Destination Subnet field of the service providers network. Similarly, the subnet that you enter in the Destination Subnet field of the subscribers network matches the subnet that you enter in the Source Subnet field of the service providers network.

Prerequisites

Before you begin, ensure that the following prerequisites are met:

- **Hardware Requirements**

- Processor: 4 core x86
- Memory: 32GB RAM
- Storage: at least 128GB hard disk

- **Software Requirements**

- Contrail Release 5.0 or later

- Create three network IPAMs (IP Address Management).

You can create a new Network IPAM by following these steps:

1. Click **Overlay>IPAM**.

The IP Address Management page is displayed.

2. Click **Create** to create a new network IPAM.

3. In the **Name** field, enter a name for the IPAM.

For left network, enter **test-left-IPAM**. For right network, enter **test-right-IPAM**. For management network, enter **mgmt-right-IPAM**.

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.

Getting Started

The instructions provided in the topics given below will help you to

1. Create the following virtual networks:

- Left Virtual Network
- Right Virtual Network
- Management Virtual Network

For steps to create virtual networks, see [“Create Virtual Network” on page 104](#).

2. Create three virtual machines.

Each virtual machine must be created with left, right, and management interfaces.

- Left Virtual Machine
- Right Virtual Machine
- Management Virtual Machine

For steps to create virtual machines by using OpenStack, see [“Create Virtual Machines by using OpenStack” on page 105](#).

For steps to create virtual machines by using Contrail Command, see [“Create Virtual Machines by using Contrail Command” on page 106](#).

3. Create a service template.

For steps to create a service template, see [“Create Service Template” on page 108](#).

4. Add a service instance.

For steps to add a service instance, see [“Add Service Instance” on page 109](#).

5. Configure fat flows for these virtual networks.

- Left Virtual Network
- Right Virtual Network

For steps to configure fat flows, see [“Configure Fat Flow” on page 111](#).

6. Create a service policy for the left virtual network and right virtual network.

For steps to create a service policy, see [“Create Service Policy” on page 113](#).

7. Attach the service policy to the left virtual network and right virtual network.

For steps to attach a service policy to a virtual network, see [“Attach Service Policy” on page 114](#).

8. Ping right virtual machine from left virtual machine.

For steps to ping the right virtual machine by using OpenStack, see [“Launch a Virtual Machine from OpenStack” on page 115](#).

For steps to ping the right virtual machine by using Contrail Command, see [“Launch a Virtual Machine from Contrail Command” on page 115](#).

Configuration

IN THIS SECTION

- [Create Virtual Network | 104](#)
- [Create Virtual Machine | 105](#)

- [Create Service Template | 108](#)
- [Add Service Instance | 109](#)
- [Configure Fat Flow | 111](#)
- [Create Service Policy | 113](#)
- [Attach Service Policy | 114](#)
- [Launch Virtual Machine | 114](#)

These topics provide instructions to configure fat flows by creating an in-network service chain.

Create Virtual Network

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.

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 through 7 to create the right virtual network (**test-right-VN**) and management virtual network (**test-mgmt-VN**).

Create Virtual Machine

IN THIS SECTION

- [Create Virtual Machines by using OpenStack | 105](#)
- [Create Virtual Machines by using Contrail Command | 106](#)

You use OpenStack or Contrail Command to create virtual machines for left, right, and management networks. You create the virtual networks with left, right, and management interfaces.

Create Virtual Machines by using OpenStack

Follow these steps to create left virtual machine by using OpenStack.

1. Click **Project>Compute>Instances**.

The Instances page is displayed.

2. Click **Launch Instance** to create an instance.

The Details tab of the Launch Instance page is displayed.

3. Enter **test-left-VM** for the left virtual machine in the **Instance Name** field and click the **Source** tab.

The Source tab of the Launch Instance page is displayed.

4. Select an **vSRX** image from the Available list by clicking the add (+) icon next to the image file.

5. Click the **Flavor** tab.

The Flavor tab of the Launch Instance page is displayed.

NOTE: vSRX image with M1.large flavor is recommended for in-network virtual machine.

6. Select **M1.large** as the flavor from the Available list by clicking the add (+) icon next to the flavor name.

7. Click the **Networks** tab.

The Network tab of the Launch Instance page is displayed.

8. Select a network you want to associate with the virtual machine instance by clicking the add (+) icon next to the network name.

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**.

9. Click **Launch Instance** to launch the virtual machine instance.

The Instances page is displayed.

All virtual machine instances that you created are displayed on the Instances page.

Repeat steps 2 through 9 to create the right virtual machine (**test-right-VM**) and management virtual machine (**test-mgmt-VM**).

Create Virtual Machines by using Contrail Command

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 through 11 to create right virtual machine instance (**test-right-VM**) and management virtual machine instance (**test-mgmt-VM**).

Create Service Template

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.

Figure 22: Adding Interfaces

The screenshot shows the 'Create VNF Service Template' page. The breadcrumb navigation is 'SERVICES > Catalog > Create VNF Service Template'. There are three tabs: 'Service Template' (active), 'Tags', and 'Permissions'. The form contains the following fields:

- Name***: A text input field containing 'test-service-template'.
- Version***: A dropdown menu with 'v2' selected.
- Virtualization Type***: A dropdown menu with 'Virtual Machine' selected.
- Service Mode***: A dropdown menu with 'In-Network' selected.
- Service Type***: A dropdown menu with 'Firewall' selected.

Below these fields is a section titled 'Interface' containing two 'Interface Type' dropdown menus. The first dropdown is set to 'left' and the second is set to 'right'. At the bottom of the form are two buttons: 'Create' (in blue) and 'Cancel' (in light blue).

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

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.

Figure 23: Adding Service Instance

SERVICES > Deployments > Create VNF Service Instance

Service Instances Tags Permissions

Name*
test-service-instance

Service Template*
test-service-template - ...

Interface Type Virtual Network*
left test-left-VN

Interface Type Virtual Network*
right test-right-VN

Interface Type Virtual Network*
management test-mgmt-VN

▼ Expand All ▲ Collapse All

► Port Tuples

Create Cancel

6. 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.

Configure Fat Flow

Starting with Contrail Release 5.0, you can also configure fat flow at the virtual network (VN) level. When you configure fat flow from the VN level, the fat flow configuration is applied to all VMIs under the configured VN.

For more information, see [“Fat Flows” on page 85](#).

Follow these steps to configure fat flows by using the Contrail Command UI.

1. Click **Overlay>Virtual Networks**.

The Virtual Networks page is displayed.

2. Select **test-left-VN** by selecting the check box next to the name of the virtual network, and then click the **Edit** icon.

The Edit Virtual Network page is displayed.

NOTE: You must configure fat flows on all the virtual networks that you created.

3. Click **Fat Flow(s)** to display the fields that you can edit.

You can edit the fields listed in [Table 6 on page 88](#).

Table 8: Edit Fat Flow(s)

Field	Action
Protocol	<p>Select ICMP from the Protocol list.</p> <p>You can select ICMP for both IPv4 and IPv6 traffic.</p>
Port	<p>Edit the Port field to any value between 0 through 65,535.</p> <p>Enter 0 to ignore both source and destination port numbers.</p> <p>NOTE: If you select ICMP as the protocol, the PORT field is not enabled.</p>

Table 8: Edit Fat Flow(s) (continued)

Field		Action
Ignore Address		<p>Select None from the Ignore Address list.</p> <p>For more information on Destination and Source options, see “Fat Flows” on page 85.</p> <p>NOTE: With Contrail Release 5.0, fat flow is enhanced with the option to support aggregation of multiple flows into a single flow by ignoring source and destination ports or IP addresses.</p>
Prefix Aggregation Source	Source Subnet	<p>For test-left-VN, enter 192.0.2.0/24 in the Source Subnet field. See Figure 24 on page 112.</p> <p>For test-right-VN, enter 198.51.100.0/24 in the Source Subnet field. See Figure 25 on page 113.</p>
	Prefix	Enter 24 in the Prefix field.
Prefix Aggregation Destination	Destination Subnet	<p>For test-left-VN, enter 198.51.100.0/24 in the Source Subnet field. See Figure 24 on page 112.</p> <p>For test-right-VN, enter 192.0.2.0/24 in the Source Subnet field. See Figure 25 on page 113.</p>
	Prefix	Enter 24 in the Prefix field.

Figure 24: Configure Fat Flows for test-left-VN

▼ Fat Flows

Protocol

Port*

Ignore Address

ICMP

0

None

Prefix Aggregation Source

Prefix Aggregation Destination

Source Subnet

Prefix

Destination Subnet

Prefix

192.0.2.0/24

24


198.51.100.0/24

24

+ Add

Figure 25: Configure Fat Flows for test-right-VN

▼ Fat Flows

Protocol ICMP	Port* 0	Ignore Address None	
Prefix Aggregation Source		Prefix Aggregation Destination	
Source Subnet 198.51.100.0/24	Prefix 24	Destination Subnet 192.0.2.0/24	Prefix 24
+ Add			

- Click **Save** to update new configuration information.

The All Networks page is displayed.

Repeat steps 2 through 4 to configure fat flows for the **test-right-VN**.

Create Service Policy

Follow these steps to create a service policy by using the Contrail Command UI.

- Click **Overlay > Network Policies**.

The Network Policies page is displayed.

- Click **Create**.

The Network Policy tab of the Create Network Policy page is displayed.

- Enter **test-network-policy** in the **Policy Name** field.

- 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

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.

To attach service policy,

- Select the check box next to the name of the virtual network.
- Hover over to the end of the selected row and click the **Edit** icon.

The Edit Virtual Network page is displayed.

- 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

IN THIS SECTION

- [Launch a Virtual Machine from OpenStack | 115](#)
- [Launch a Virtual Machine from Contrail Command | 115](#)

You can launch a virtual machine from OpenStack or from Contrail Command UI.

Launch a Virtual Machine from OpenStack

You can launch virtual machines from OpenStack 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 Machines by using OpenStack” on page 105](#).
2. Launch the right virtual machine in right virtual network. See [“Create Virtual Machines by using OpenStack” on page 105](#).
3. Ping the left virtual machine IP address from the right virtual machine.

Follow these steps to ping a virtual machine:

1. Click **Project > Compute > Instances**.

All virtual machine instances that you created are displayed on the Instances page.

2. From the list of virtual machines, click **test-right-VM**.

The Overview tab of the test-right-VM is displayed.

3. Click the **Console** tab.

The Instance Console is displayed.

4. Log in using the **root** user credentials.

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

See [Figure 26 on page 116](#) for a sample output.

Launch a Virtual Machine from Contrail Command

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 Machines by using Contrail Command” on page 106](#).
2. Launch the right virtual machine in right virtual network. See [“Create Virtual Machines by using Contrail Command” on page 106](#).
3. Ping the left virtual machine IP address from the right virtual machine.

Follow these steps to ping a virtual machine:

- 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 the **root** user credentials.

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

See [Figure 26 on page 116](#) for a sample output.

Figure 26: Ping test-left-VM

```
root@test-right-vm:~# ping -c 5 192.0.2.3
PING 192.0.2.3 (192.0.2.3) 56(84) bytes of data.
64 bytes from 192.0.2.3: icmp_seq=1 ttl=63 time=0.238 ms
64 bytes from 192.0.2.3: icmp_seq=2 ttl=63 time=0.208 ms
64 bytes from 192.0.2.3: icmp_seq=3 ttl=63 time=0.231 ms
64 bytes from 192.0.2.3: icmp_seq=4 ttl=63 time=0.210 ms
64 bytes from 192.0.2.3: icmp_seq=5 ttl=63 time=0.210 ms

--- 192.0.2.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 0.208/0.219/0.238/0.018 ms
root@test-right-vm:~#
```

RELATED DOCUMENTATION

| [Fat Flows](#) | 85

Understanding Flow Sampling

IN THIS SECTION

- [Flow Sampling](#) | 117
- [Flow Handling](#) | 118

- [Flow Aging | 118](#)
- [TCP State-Based Flow Handling and Aging | 119](#)

This topic describes how flow records are sampled and exported to the Contrail collector, flow handling, and flow aging.

Flow Sampling

The Contrail vRouter agent exports flow records to the Contrail collector when a flow is created or deleted. It also updates flow statistics at regular intervals.

If all flow records are exported from the agent, depending on the scale of flows, some of the exported flows might be dropped due to queue overflow.

In Contrail Release 2.22 and later, to reduce queue overflow, flow records are sampled and exported to the Contrail Collector based on sampling.

The flows that are exported are selected based on the following parameters used in the algorithm:

- The configured flow export rate. This is configured as part of the **global-vrouter-config** object.
- The actual flow export rate.
- The sampling threshold. This is a dynamic value calculated internally. If the flow statistics in a flow sample are above this threshold, the flow record is exported.

Each flow is subjected to the following algorithm at regular intervals. The algorithm determines whether to export the sample or not.

- Flows with traffic that is greater than or equal to the sampling threshold are always exported. The byte and packet counts are reported without modification.
- Flows with traffic that is less than the sampling threshold are exported according to the probability. The byte and packet counts are adjusted upwards according to the probability.

The probability is calculated as (bytes during the interval) / (sampling threshold).

- The system generates a random number less than the sampling threshold. If the byte count during the interval is less than the random number, then the flow sample is not exported.
- If none of these conditions are met, the flow sample is exported after normalizing the byte count and packet count during the interval. Normalization is done by dividing the byte count and packet count

during the interval by the probability. This normalization is used as a heuristic to account for statistics of flow samples that are dropped.

The actual flow export rate is close to the configured export rate. If there is a large deviation, the sampling threshold is adjusted to bring the actual flow export rate close to the configured flow export rate.

Flow Handling

When a virtual machine sends or receives IP traffic, forward and reverse flow entries are set up. When the first packet arrives, a flow key is used to hash into a flow table (identify a hash bucket). The flow key is based on five-tuples consisting of source and destination IP addresses, ports, and the IP protocol.

A flow entry is created and the packet is sent to the Contrail vRouter agent. Configured policies are applied and the flow action is updated. The agent also creates a flow entry for the reverse direction where relevant. Subsequent packets match the established flow entries and are forwarded, dropped, NAT translated, and so on, based on the flow action.

When the hash bucket is full, entries are created in an overflow table. In releases earlier than Contrail Release 2.22, the overflow table was a global table, which is searched sequentially. In Contrail Release 2.22 and later, the overflow entries are maintained as a list against the hash bucket.

By default, the maximum number of flow table and overflow table entries are 512,000 and 8000 respectively. These can be modified by configuring them as vRouter module parameters, for example: **vr_flow_entries** and **vr_oflow_entries**.

For more information about the vRouter module parameters, see <https://github.com/Juniper/contrail-controller/wiki/Vrouter-Module-Parameters>.

Flow Aging

Flows are aged out based on inactivity for a specified period of time. By default, the timeout value is 180 seconds. This can be modified by configuring the **flow_cache_timeout** parameter under the **DEFAULT** section in the **/etc/contrail/contrail-vrouter-agent.conf** file.

TCP State-Based Flow Handling and Aging

IN THIS SECTION

- [TCP State-Based Flow Handling | 119](#)
- [Protocol-Based Flow Aging | 119](#)

TCP State-Based Flow Handling

In Contrail Release 2.22 and later, the Contrail vRouter monitors TCP flows to identify entries that can be reused without going through the standard aging cycle.

All flow entries that match TCP flows that have experienced a connection teardown, either through the standard TCP closure cycle (FIN/ACK-FIN/ACK) or the RST indicator, are torn down by the vRouter and are immediately available for use by new qualified flows.

The vRouter also keeps track of connection establishment cycles and exports the necessary information to the vRouter agent, such as SYN/ACK and a digested established flag. This allows the vRouter agent to tear down flows that do not experience a full connection cycle.

Flows that the vRouter identifies as reuse candidates, or eviction candidates, are marked as such in the flow entry. Flows are in the evicted state when they become available for other new flows to be reused.

This two-step transition is used so that the flow entry remains valid until the packet reaches the destination, preventing the flow from getting remapped to another flow entry in the interim.

Protocol-Based Flow Aging

Although TCP flows are deleted based on TCP state, you are sometimes required to age out specific protocol flows more aggressively. One example is when a DNS server is run in one VM. In this case, multiple flows are set up for DNS. A pair of flows are set up to serve each query. Because the flows are no longer required after the query is served, the timeout can be lower for these flows. To handle these cases, protocol-based flow aging is used.

With protocol-based flow aging, the aging timeout can be configured per protocol. All other protocols continue to use the default aging timeout.

Protocol-based flow aging is supported in Contrail Release 2.22 and later.

The configuration for protocol-based flow aging can be done in the **global-vrouter-config** object. For example, to have all DNS flows aged out in five seconds, use the following entry: **protocol = udp, port = 53 will be set an aging timeout of 5 seconds.**

RELATED DOCUMENTATION

Query > Flows

[Fat Flows](#) | 85