JUNIPER NETWORKS | Engineering Simplicity

Contrail® Networking

Contrail Networking Installation and Upgrade Guide

Published
2025-03-24

RELEASE
21.4

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

Contrail® Networking Contrail Networking Installation and Upgrade Guide
21.4

YEAR 2000 NOTICE

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

END USER LICENSE AGREEMENT

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

Table of Contents

About This Guide

Use this guide to install and upgrade the Contrail Networking solution for your environment. This guide covers various installation scenarios including:

- Contrail Command
- Contrail with Contrail Insights
- Contrail with Openstack, including Openstack, Red Hat Openstack, and Canonical Openstack

For information on Contrail Networking installations and upgrades in containerized environments using Kubernetes orchestration, see the Contrail Networking for Container Networking Environments User Guide.

Contrail Networking product documentation is organized into multiple guides as shown in Table 1 on page xiv, according to the task you want to perform or the deployment scenario.

Table 1: Contrail Networking Guides

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

RELATED DOCUMENTATION

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

PART 1

Installing and Upgrading Contrail

CHAPTER 1

Understanding Contrail

IN THIS CHAPTER

Understanding Contrail Networking

Contrail Networking provides dynamic end-to-end networking policy and control for any cloud, any workload, and any deployment, from a single user interface. It translates abstract workflows into specific policies, simplifying the orchestration of virtual overlay connectivity across all environments.

It unifies policy for network automation with seamless integrations for systems such as: Kubernetes, OpenShift, Mesos, OpenStack, VMware, a variety of popular DevOps tools like Ansible, and a variety of Linux operating systems with or without virtualization like KVM and Docker containers.

Contrail Networking is a fundamental building block of Contrail Enterprise Multicloud for enterprises. It manages your data center networking devices, such as QFX Series Switches, Data Center Interconnect (DCI) infrastructures, as well as public cloud gateways, extending the continuous connectivity from your on-premises to private and public clouds.

Contrail Networking reduces the friction of migrating to cloud by providing a virtual networking overlay layer that delivers virtual routing, bridging, and networking services (IPAM, NAT, security, load balancing, VPNs, etc.) over any existing physical or cloud IP network. It also provides multitenant structure and API compatibility with multitenant public clouds like Amazon Web Services (AWS) virtual private clouds (VPCs) for truly unifying policy semantics for hybrid cloud environments.

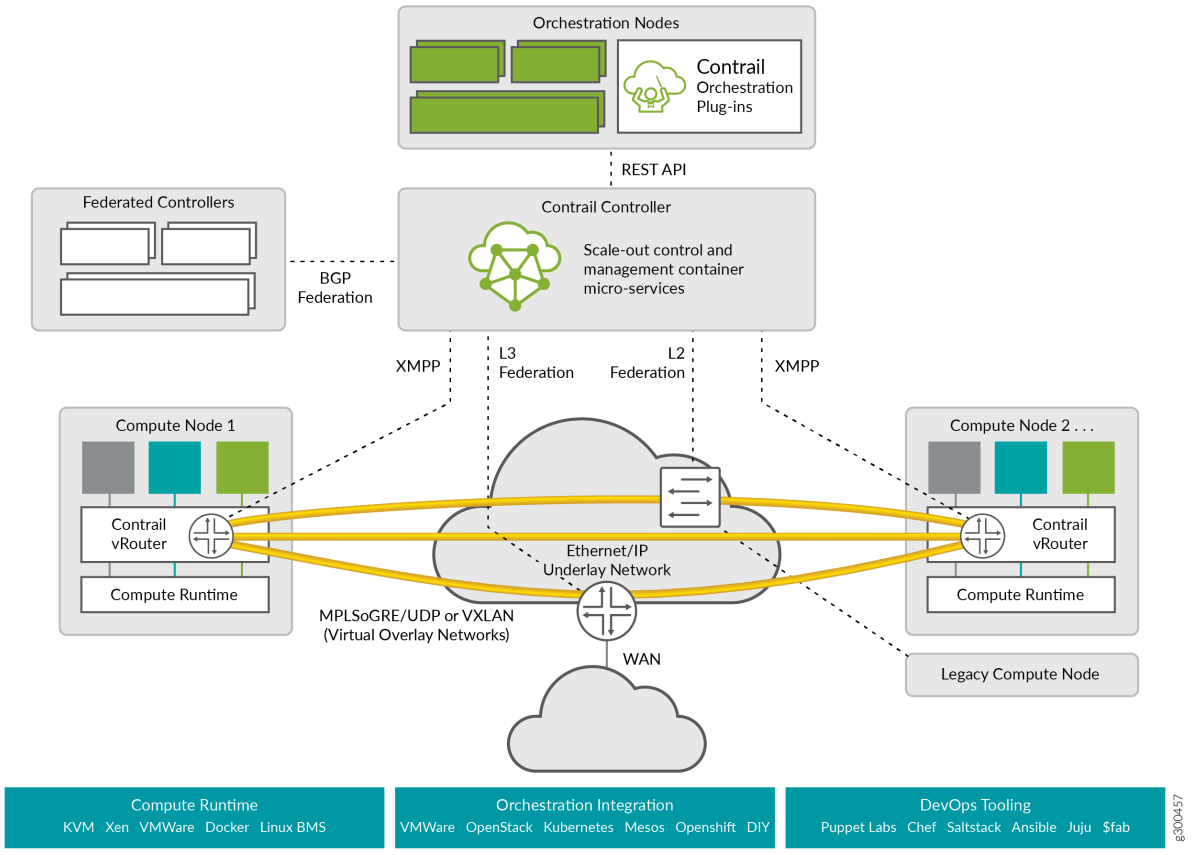
For service providers, Contrail Networking automates network resource provisioning and orchestration to dynamically create highly scalable virtual networks and to chain a rich set of Juniper Networks or third-party virtualized network functions (VNFs) and physical network functions (PNFs) to form differentiated service chains on demand.

Contrail Networking is also integrated with Contrail Cloud for service providers. It enables you to run high-performance Network Functions Virtualization (NFV) with always-on reliability so that you can deliver innovative services with greater agility.

Contrail Networking is equipped with always-on advanced analytics capabilities to provide deep insights into application and infrastructure performance for better visualization, easier diagnostics, rich reporting, custom application development, and machine automation. It also supports integration with other analytics platforms like Juniper Networks Contrail Insights and streaming analytics through technologies like Apache Kafka and its API.

Contrail Networking also provides a Graphical User Interface (GUI).This GUI is built entirely using the REST APIs.

Figure 1: Contrail Networking Architecture



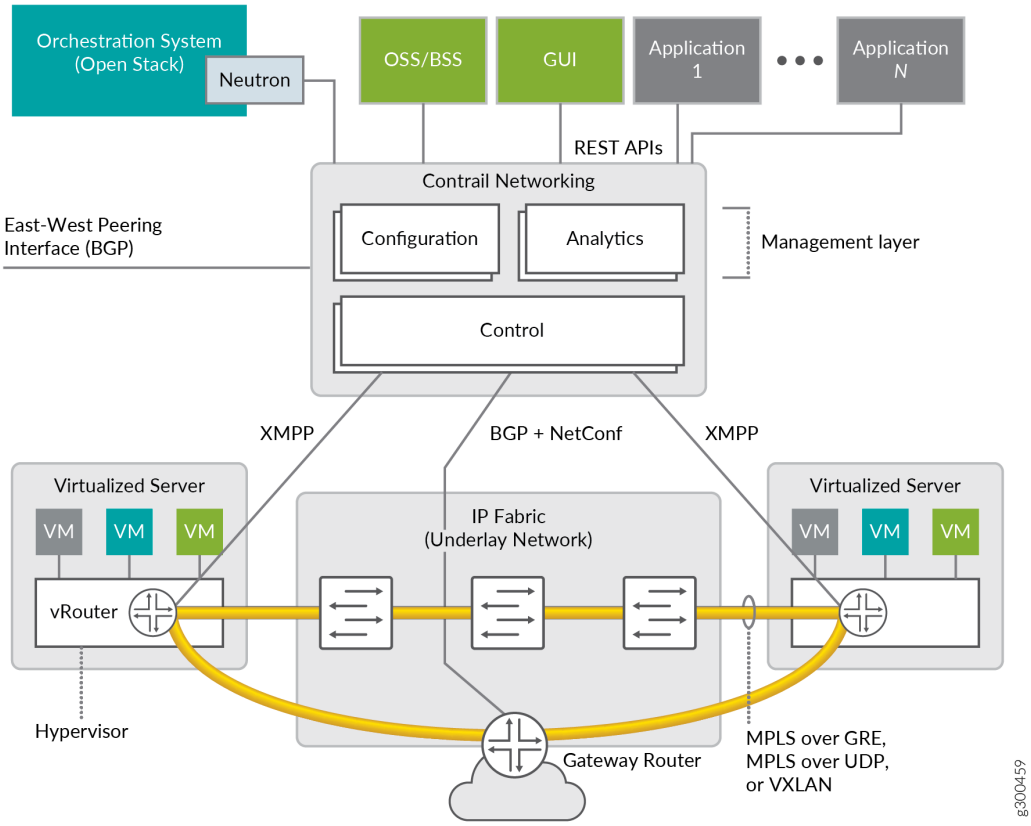
RELATED DOCUMENTATION

Understanding Contrail Networking Components

Contrail Networking is comprised of the following key components:

- *Contrail Networking management Web GUI and plug-ins* integrate with orchestration platforms such as Kubernetes, OpenShift, Mesos, OpenStack, VMware vSphere, and with service provider operations support systems/business support systems (OSS/BSS). Many of these integrations are built, certified, and tested with technology alliances like Red Hat, Mirantis, Canonical, NEC, and more. Contrail Networking sits under such orchestration systems and integrates northbound via published REST APIs. It can be automatically driven through the APIs and integrations, or managed directly using the Web GUI, called Contrail Command GUI.
- *Contrail Networking control and management system*, commonly called the controller, have several functions. Few of the major functions are:
  - *Configuration Nodes*—This function accepts requests from the API to provision workflows like adding new virtual networks, new endpoints, and much more. It converts these abstract high-level requests, with optional detail, into low-level directions that map to the internal data model.
  - *Control Nodes*—This function maintains a scalable, highly available network model and state by federating with other peer instances of itself. It directs network provisioning for the Contrail Networking vRouters using Extensible Messaging and Presence Protocol (XMPP). It can also exchange network connectivity and state with peer physical routers using open industry-standard MP-BGP which is useful for routing the overlay networks and north-south traffic through a high-performance cloud gateway router.
  - *Analytics Nodes*—This function collects, stores, correlates, and analyzes data across network elements. This information, which includes statistics, logs, events, and errors, can be consumed by end-user or network applications through the northbound REST API or Apache Kafka. Through the Web GUI, the data can be analyzed with SQL style queries.
- *Contrail Networking vRouter* runs on the compute nodes of the cloud or NFV infrastructure. It gets network tenancy, VPN, and reachability information from the control function nodes and ensures native Layer 3 services for the Linux host on which it runs or for the containers or virtual machines of that host. Each vRouter is connected to at least two control nodes to optimize system resiliency. The vRouters run in one of two high performance implementations: as a Linux kernel module or as an Intel Data Plane Development Kit (DPDK)-based process.

Figure 2: Contrail Networking Overview



RELATED DOCUMENTATION

Understanding Contrail Networking | 2

Understanding Contrail Containers

IN THIS SECTION

Some subsystems of Contrail Networking solution are delivered as Docker containers.

Contrail Containers

The following are key features of the architecture of Contrail containers:

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

> **NOTE**: Starting in Contrail Release 21.4, the Red Hat Universal Base Image 8 (ubi8) is used as the base for Contrail container images.

Understanding Contrail Microservices Architecture

IN THIS SECTION

What is Contrail Microservices Architecture?

Employing microservices provides a number of benefits which includes:

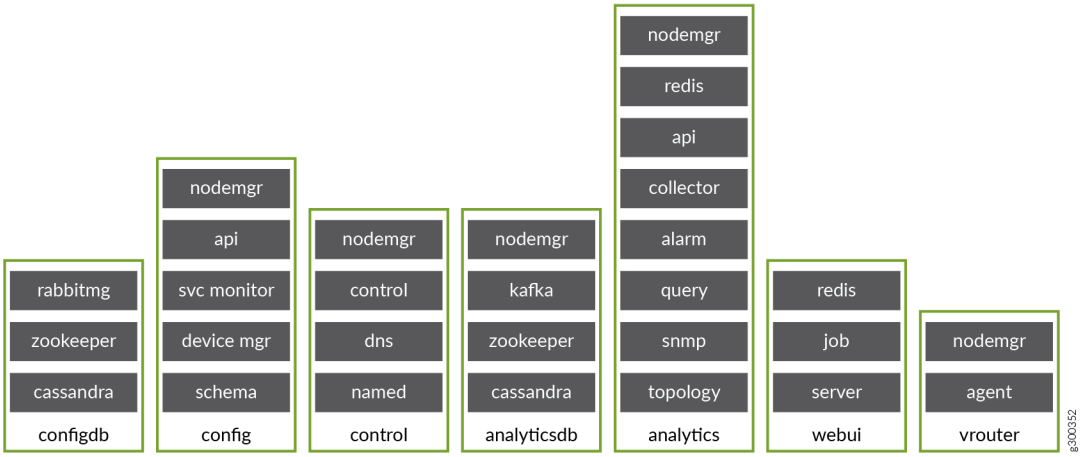
- Deploying patches without updating the entire Contrail deployment.
- Better ways to manage the lifecycles of containers.
- Improved user experiences with Contrail provisioning and upgrading.
- Provisioning with minimum information provided.

- Configuring every feature.
- Simplify application complexity by implementing small, independent processes.

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

Figure 3 on page 7 shows how the Contrail containers and microservices are grouped into a pod structure upon installation.

Figure 3: Contrail Containers, Pods, and Microservices



Installing Contrail with Microservices Architecture

These procedures help you to install and manage Contrail with microservices architecture. Refer to the following topics for installation for the operating system appropriate for your system:

- "Understanding contrail-ansible-deployer used in Contrail Command" on page 7

Understanding contrail-ansible-deployer used in Contrail Command

IN THIS SECTION

This topic provides an overview of `contrail-ansible-deployer` used by *Contrail Command* tool. It is used for installing Contrail Networking with microservices architecture.

To understand Contrail microservices, refer to "Understanding Contrail Microservices Architecture" on page 6. For step by step procedure on how to install Contrail using Contrail Command deployer, refer to "Installing a Contrail Cluster using Contrail Command and instances.yml" on page 78.

What is the contrail-ansible-deployer?

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

The `contrail-ansible-deployer` contains three playbooks:

provision_instances.yml

This playbook provisions the operating system instances for hosting the containers. It supports the following infrastructure providers:

- kvm.
- gce.
- aws.

configure_instances.yml

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

install_contrail.yml

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

Preparing to Install with Contrail Command

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

Prerequisites

Make sure your system meets the following requirements before running `contrail-command-deployer`.

- Confirm that you are running compatible versions of CentOS, Ansible, Docker, and any other software component for your system in your environment. See Contrail Networking Supported Platforms List.
- Name resolution is operational for long and short host names of the cluster nodes, through either DNS or the host file.
- For high availability (HA), confirm that the time is in sync between the cluster nodes.
- The time must be synchronized between the cluster nodes using Network Time Protocol (ntp).

Supported Providers

The playbooks support installing Contrail Networking on the following providers:

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

Configure the instances.yaml File for Your Environment

The configuration for all three playbooks is contained in a single file, `config/instances.yaml`.

The configuration has multiple main sections, including:

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

Provider Configuration

The section `provider_config` configures provider-specific settings.

KVM Provider Example

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

NOTE: Passwords are provided in this output for illustrative purposes only. We suggest using unique passwords in accordance with your organization's security guidelines in your environment.

```
provider_config:                                  # the provider section contains all provider
relevant configuration
  kvm:                                                  # Mandatory.
    image: CentOS-7-x86_64-GenericCloud-1710.qcow2.xz     # Mandatory for provision play. Image
to be deployed.
    image_url: https://cloud.centos.org/centos/7/images/  # Mandatory for provision play. Path/
url to image.
    ssh_pwd: contrail123                                  # Mandatory for provision/
configuration/install play. Ssh password set/used.
    ssh_user: centos                                      # Mandatory for provision/
configuration/install play. Ssh user set/used.
    ssh_public_key: /home/centos/.ssh/id_rsa.pub          # Optional for provision/configuration/
install play.
    ssh_private_key: /home/centos/.ssh/id_rsa             # Optional for provision/configuration/
install play.
    vcpu: 12                                              # Mandatory for provision play.
    vram: 64000                                           # Mandatory for provision play.
    vdisk: 100G                                           # Mandatory for provision play.
    subnet_prefix: ip-address                           # Mandatory for provision play.
    subnet_netmask: subnet-mask                         # Mandatory for provision play.
    gateway: gateway-ip-address                                 # Mandatory for provision play.
    nameserver: dns-ip-address                             # Mandatory for provision play.
    ntpserver: ntp-server-ip-address                              # Mandatory for provision/
configuration play.
    domainsuffix: local                                   # Mandatory for provision play.
```

BMS Provider Example

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

NOTE: Passwords are provided in this output for illustrative purposes only. We suggest using unique passwords in accordance with your organization's security guidelines in your environment.

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

CAUTION: SSH *Host Identity Keys* must be accepted or installed on the Deployer node before proceeding with Contrail installation.

To do so:

- Make SSH connection to each target machine from the Deployer VM using Deployer user credentials and click **Yes** to accept the SSH *Host Key*.

  or

- Set the environmental variable *ANSIBLE_HOST_KEY_CHECKING* value to **False**.

  ```
  ANSIBLE_HOST_KEY_CHECKING=false
  ```

  or

- Set *[defaults] host_key_checking* value to **False** in **ansible.cfg** file.

  ```
  [defaults] host_key_checking=false
  ```

AWS Provider Example

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

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

GCE Provider Example

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

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

```
    zone: us-west1-aA           # Optional.  Defaults to  ?
    disk_size: 50               # Mandatory. Default is too small
```

Global Services Configuration

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

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

Contrail Services Configuration

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

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

For a complete list of parameters available for contrail_configuration.md, see Contrail Configuration Parameters for Ansible Deployer.

Kolla Services Configuration

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

```
kolla_config:
```

Instances Configuration

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

KVM Contrail Plane Instance

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

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

GCE Default All-in-One Instance

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

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

AWS Default Three Node HA Instance

The following example uses three AWS EC2 instances to deploy a three node high availability setup with all roles and default parameters.

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

More Examples

Refer to the following for more configuration examples for instances.

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

Installing a Contrail System

To perform a full installation of a Contrail system, refer to the installation instructions in: "Installing a Contrail Cluster using Contrail Command and instances.yaml" on page 78.

CHAPTER 2

Supported Platforms and Server Requirements

IN THIS CHAPTER

Server Requirements and Supported Platforms

This topic discusses server requirements in a Contrail Networking cluster.

Each server must have a minimum of:

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

A server can either be a physical device or a virtual machine. For scalability and availability reasons, it is highly recommended to use physical servers in most use cases whenever possible.

Server role assignments vary by environment. All non-compute roles can be configured in each controller node if desired in your topology.

All installation images are available in repositories and can also be downloaded from Contrail Downloads page.

The Contrail image includes the following software:

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

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

For a list of supported platforms for all Contrail Networking releases, see Contrail Networking Supported Platforms List.

Access *Container Tags* are located at README Access to Contrail Registry 21XX.

If you need access to Contrail docker private secure registry, e-mail **contrail-registry@juniper.net** for Contrail container registry credentials.

RELATED DOCUMENTATION

Hardware and Server Requirements | 17

Hardware and Server Requirements

IN THIS SECTION

The following tables list the minimum and total memory and disk requirements per x86 server or per virtual machine for installing Contrail Networking.

Hardware Requirements for Contrail Networking Release 2011

IN THIS SECTION

The following tables list the minimum and total memory and disk requirements for installing Contrail Networking Release 2011.

Baseline Server Requirement for an All-In-One Setup

Baseline for 32 switches.

Role	vCPUs (hyper threaded)	Memory (GB)	Disk (GB)	Networking
Contrail Command	4	32	100	1 Ethernet port
Control Host	20	48	350	2 Ethernet ports
Contrail Insights + Flow Node	10	50	300	2 Ethernet ports

Contrail Controller for Object Scale (Number of VLANs)

Baseline for 256 switches.

Number of Devices	Number of VMIs (2500 VPGs each with 102 VLANs)	vCPUs (hyper threaded)	RAM (GB)	Disk (GB)
256	256K	38	135	350

Baseline Server Requirement for a HA Setup

Role	vCPUs (hyper threaded)/ Node	Memory (GB)/ Node	Disk (GB)/ Node	Networking
Contrail Command	4	32	100	1 Ethernet port
Control Host (x3)	20	48	350	2 Ethernet ports

Contrail Insights + Flow Node (x3) Note: Number of sampled flows: 500K	10	50	300	2 Ethernet ports

Contrail Insights Scale

Number of Contrail Insight Nodes	Number of VMs	Number of Compute Nodes	vCPUs	RAM (GB)	Disk (TB)
3	10K	300	64	64	2
5	100K	1000	64	128	4

Hardware Requirements for Contrail Networking Release 2008

IN THIS SECTION

The following tables list the minimum and total memory and disk requirements for installing Contrail Networking Release 2008.

Baseline Server Requirement for an All-In-One Setup

Baseline for 32 switches.

Role	vCPUs (hyper threaded)	Memory (GB)	Disk (GB)	Networking
Contrail Command	4	32	100	1 Ethernet port

Control Host	20	48	350	2 Ethernet ports
Contrail Insights + Flow Node	10	50	300	2 Ethernet ports

Contrail Controller for Object Scale (Number of VLANs)

Baseline for 128 switches.

Number of Devices	Number of VMIs (2500 VPGs each with 102 VLANs)	vCPUs (hyper threaded)	RAM (GB)	Disk (GB)
128	256K	35	133	350

Baseline Server Requirement for a HA Setup

Role	vCPUs (hyper threaded)/ Node	Memory (GB)/ Node	Disk (GB)/ Node	Networking
Contrail Command	4	32	100	1 Ethernet port
Control Host (x3)	20	48	350	2 Ethernet ports
Contrail Insights + Flow Node (x3) Note: Number of sampled flows: 500K	10	50	300	2 Ethernet ports

Contrail Insights Scale

Number of Contrail Insight Nodes	Number of VMs	Number of Compute Nodes	vCPUs	RAM (GB)	Disk (TB)
3	10K	300	64	64	2

5	100K	1000	64	128	4

RELATED DOCUMENTATION

Server Requirements and Supported Platforms | **16**

CHAPTER 3

Contrail Command

IN THIS CHAPTER

How to Install Contrail Command and Provision Your Contrail Cluster

IN THIS SECTION

Use this document to install Contrail Command—the graphical user interface for Contrail Networking—and provision your servers or VMs as nodes in a Contrail cluster. Servers or VMs are provisioned into compute nodes, control nodes, orchestrator nodes, Contrail Insights nodes, Contrail Insights Flows nodes, or service nodes to create your Contrail cluster using this procedure.

NOTE: Contrail Insights and Contrail Insights Flows were previously named Appformix and Appformix Flows.

When to Use This Document

We strongly recommend Contrail Command as the primary interface for configuring and maintaining Contrail Networking.

You should, therefore, complete the procedures in this document as an initial configuration task in your Contrail Networking environment.

Server Requirements

A Contrail Networking environment can include physical servers or VMs providing server functions, although we highly recommended using physical servers for scalability and availability reasons whenever possible.

Each server in a Contrail environment must have a minimum of:

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

For additional information on server requirements for Contrail Networking, see "Server Requirements and Supported Platforms" on page 16.

Software Requirements

- *Contrail Command and Contrail Networking*

  Contrail Command and Contrail Networking are updated simultaneously and always run the same version of Contrail Networking software.

  Each Contrail Networking release has software compatibility requirements based on the orchestration platform version, the deployer used to deploy the orchestration platform, the supported server operating system version, and other software requirements.

For a list of supported platforms for all Contrail Networking releases and additional environment-specific software requirements, see Contrail Networking Supported Platforms List.

- *Contrail Insights and Contrail Insights Flows*

  Starting in Contrail Release 2005, the Contrail Insights and Contrail Insights Flows images that support a Contrail Networking release are automatically provisioned within Contrail Command. When you download your version of Contrail Command, Contrail Command pulls the Contrail Insights and Contrail Insights Flows images for your Contrail Networking version automatically from within the Juniper Contrail registry. You do not, therefore, need to separately download any individual Contrail Insights software or have awareness of Contrail Insights or Contrail Insights version numbers for your installation.

How to Obtain Contrail Images

The procedures used in this document download the Contrail Command, Contrail Insights, and Contrail Insights Flows software from the Juniper Networks Contrail docker private secure registry at *hub.juniper.net*. Email mailto:contrail-registry@juniper.net to obtain access credentials to this registry.

You will need to know the *Container Tags* for your Contrail image to retrieve Contrail images from the Contrail registry. See README Access to Contrail Registry 21XX.

Contrail Networking images are also available at the Contrail Downloads page. Enter *Contrail Networking* as the product name.

Contrail Insights and Contrail Insights Flows images are also available at the Contrail Insights Download page. Enter *Contrail Insights* as the product name.

How to Install Contrail Command

IN THIS SECTION

Contrail Command is a single pane-of-glass GUI interface for Contrail Networking. For an optimized Contrail Networking experience, we strongly recommend installing Contrail Command before creating your Contrail clusters. Contrail Command is installed using these instructions.

For additional information on Contrail Command, see "Understanding Contrail Networking Components" on page 4.

Before You Begin

Ensure your Contrail Command server—the server that will host Contrail Command—is a virtual machine (VM) or a physical x86 server that meets these minimum system requirements:

- 4 vCPUs
- 32 GB RAM
- 100 GB disk storage with all user storage in the "/" partition.

  If the "**/home**" partition exists, remove it and increase the "/" partition by the amount of freed storage.
- Meets the specifications listed in "Server Requirements" on page 23.
- Runs a version of CentOS that supports your version of Contrail Networking.

  For a list of CentOS versions that are supported with Contrail Networking and orchestration platform combinations, see Contrail Networking Supported Platforms List.

  You can install CentOS with updated packages using the `yum update` command.
- Has access to the Contrail Container registry at *hub.juniper.net*. This access is needed because the Contrail Command deployer, which includes the Contrail Command docker images, is retrieved from this registry during this installation procedure.

  If you do not have access to the Contrail Container registry, email mailto:contrail-registry@juniper.net to obtain access credentials. See README Access to Contrail Registry 21XX for additional information about accessing this registry.
- Has an active connection to the Internet.
- Includes at least one active IP interface attached to the management network. Contrail Command manages Contrail and orchestrator clusters over a management IP interface.

Obtain the container tag for the release that you are installing. A container tag is necessary to identify the Contrail Command container files in the *hub.juniper.net* repository that are installed during this procedure.

The container tag for any Contrail Release 21-based image can be found in README Access to Contrail Registry 21XX.

Preparing Your Contrail Command Server for the Installation

To prepare your servers or VMs for the installation:

1. Log onto the server that will host Contrail Command and all servers in your Contrail cluster. The servers in your Contrail cluster are the devices that will be provisioned into compute, control, orchestrator, Contrail Insights, Contrail Insights Flows, or service node roles.
2. Verify the hosts in the hosts file, and add the name and IP address of each host that you are adding to the file.

   In this example, the hosts file is edited using VI to include the name and IP address of the three other servers—contrail-cluster, insights, and insights-flows—that will be provisioned into the contrail cluster during this procedure.

```
[root@ix-cn-ccmd-01 ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

[root@ix-cn-ccmd-01 ~]# vi /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.1.1.2 contrail-cluster
10.1.1.3 insights
10.1.1.4 insights-flows
```

   NOTE: The hosts file is typically overwritten during the provisioning process. This step can be skipped in most Contrail cluster provisioning scenarios, but is recommended as a precaution.

3. Verify the hostname file listing on the Contrail Command server.

```
[root@ix-cn-ccmd-01 ~]# cat /etc/hostname
ix-cn-ccmd-01
```

   If needed, update the Contrail Command hostname accordingly to match the hostname that you will use in the Contrail Command cluster.

   NOTE: The hostname file is typically overwritten during the provisioning process. This step can be skipped in most Contrail cluster provisioning scenarios, but is recommended as a precaution.

4. If you haven't already generated a shared RSA key for the servers in the cluster, generate and share the RSA key.

```
[root@ix-cn-ccmd-01 ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:$ABC123
root@ix-cn-ccmd-01
The key's randomart image is:
+---[RSA 2048]----+
|         .o=o*.  |
|        . =o+ o  |
|       o. o= +   |
|      .o.+ +B    |
|      ..S.+++.   |
|       ...o..o.. |
|         .. oo=oo|
|         . ..=o*=|
|          E...=+=|
+----[SHA256]-----+

[root@ix-cn-ccmd-01 ~]# ssh-copy-id -i /root/.ssh/id_rsa.pub 10.1.1.2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that
are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is
to install the new keys
root@10.1.1.2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '10.1.1.2'"
and check to make sure that only the key(s) you wanted were added.

[root@ix-cn-ccmd-01 ~]# ssh-copy-id -i /root/.ssh/id_rsa.pub 10.1.1.3
...
```

```
[root@ix-cn-ccmd-01 ~]# ssh-copy-id -i /root/.ssh/id_rsa.pub 10.1.1.4
...
```

5. SSH into each server that will be provisioned into the Contrail cluster to confirm reachability and accessibility:

```
[root@ix-cn-ccmd-01 ~]# ssh 10.1.1.2
[root@contrail-cluster ~]# exit
logout
Connection to 10.1.1.2 closed.

[root@ix-cn-ccmd-01 ~]# ssh 10.1.1.3
[root@insights ~]# exit
logout
Connection to 10.1.1.3 closed.

[root@ix-cn-ccmd-01 ~]# ssh 10.1.1.4
[root@insights-flows ~]# exit
logout
Connection to 10.1.1.4 closed.
```

6. Verify that routes to each server are established on your server.

> **NOTE**: The routes connecting the servers are created outside the Contrail Networking environment and the process to create the routes varies by environment. This procedure, therefore, does not provide the instructions for creating these routes.

In this example, the routes are verified on the Contrail Command server.

```
[root@ix-cn-ccmd-01 ~]# ip route
default via 10.102.70.254 dev ens192 proto static metric 100
10.1.1.0/24 dev ens224 proto kernel scope link src 10.1.1.1 metric 101
10.102.70.0/24 dev ens192 proto kernel scope link src 10.102.70.216 metric 100
```

Perform this step on the Contrail Command server and all servers in your Contrail cluster.

7. Ping each server to verify connectivity.

In this example, each node in the Contrail cluster is pinged from the Contrail Command server.

```
[root@ix-cn-ccmd-01 ~]# ping 10.1.1.2
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.
64 bytes from 10.1.1.2: icmp_seq=1 ttl=64 time=0.602 ms
64 bytes from 10.1.1.2: icmp_seq=2 ttl=64 time=0.220 ms
^C
--- 10.1.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.220/0.411/0.602/0.191 ms

[root@ix-cn-ccmd-01 ~]# ping 10.1.1.3
PING 10.1.1.3 (10.1.1.3) 56(84) bytes of data.
64 bytes from 10.1.1.3: icmp_seq=1 ttl=64 time=0.435 ms
64 bytes from 10.1.1.3: icmp_seq=2 ttl=64 time=0.299 ms
64 bytes from 10.1.1.3: icmp_seq=3 ttl=64 time=0.260 ms
^C
--- 10.1.1.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.260/0.331/0.435/0.076 ms

[root@ix-cn-ccmd-01 ~]# ping 10.1.1.4
PING 10.1.1.4 (10.1.1.4) 56(84) bytes of data.
64 bytes from 10.1.1.4: icmp_seq=1 ttl=64 time=0.248 ms
64 bytes from 10.1.1.4: icmp_seq=2 ttl=64 time=0.266 ms
64 bytes from 10.1.1.4: icmp_seq=3 ttl=64 time=0.232 ms
^C
--- 10.1.1.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.232/0.248/0.266/0.022 ms
[root@ix-cn-ccmd-01 ~]#
```

Perform this step on the Contrail Command server and all servers in your Contrail cluster.

8. Check the Linux kernel version and, if needed, update the Linux kernel version. If a kernel version update is performed, reboot the server to complete the update.

NOTE: Obtaining the Linux kernel is not shown in this document.

In this example, the Linux kernel is verified on the Contrail Command server.

```
[root@ix-cn-ccmd-01 ~]# uname -a
Linux ix-cn-ccmd-01 3.10.0-1062.el7.x86_64 #1 SMP Wed Aug 7 18:08:02 UTC 2019 x86_64 x86_64
x86_64 GNU/Linux

[root@ix-cn-ccmd-01 ~]# ls
anaconda-ks.cfg  kernel-3.10.0-1062.12.1.el7.x86_64.rpm

[root@ix-cn-ccmd-01 ~]# rpm -ihv kernel-3.10.0-1062.12.1.el7.x86_64.rpm
warning: kernel-3.10.0-1062.12.1.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID
f4a80eb5: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:kernel-3.10.0-1062.12.1.el7      ################################# [100%]

[root@ix-cn-ccmd-01 ~]# shutdown -r now
```

After the server reboots, confirm that the kernel is updated.

```
[root@ix-cn-ccmd-01 ~]# uname -a
Linux ix-cn-ccmd-01 3.10.0-1062.12.1.el7.x86_64 #1 SMP Tue Feb 4 23:02:59 UTC 2020 x86_64
x86_64 x86_64 GNU/Linux
```

Perform this step on the Contrail Command server and all servers in your Contrail cluster.

Installing Contrail Command

To install Contrail Command onto a server:

1. Log into the server that will host the Contrail Command containers. This server will be called the Contrail Command server for the remainder of this procedure.

```
$ ssh root@10.12.70.192
root@10.12.70.192's password: password
```

2. Remove all installed Python Docker libraries—`docker` and `docker-py`—from the Contrail Command server:

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

The Python Docker libraries will not exist on the server if a new version of CentOS 7-based software was recently installed. Entering this command when no Python Docker libraries are installed does not harm any system functionality.

The Contrail Command Deployer, which is deployed later in this procedure, installs all necessary libraries, including the Python Docker libraries.

3. Install and start the Docker Engine.

   There are multiple ways to perform this step. In this example, Docker Community Edition version 18.03 is installed using `yum install` and `yum-config-manager` commands and started using the `systemctl start docker` command.

   > **NOTE**: The Docker version supported with Contrail Networking changes between Contrail releases and orchestration platforms. See Contrail Networking Supported Platforms List. The **yum install -y docker-ce-18.03.1.ce** command is used to illustrate the command for one version of Docker.

   ```
   yum install -y yum-utils device-mapper-persistent-data lvm2
   yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
   yum install -y docker-ce-18.03.1.ce
   systemctl start docker
   ```

4. Retrieve the `contrail-command-deployer` Docker image by logging into *hub.juniper.net* and entering the `docker pull command`.

   ```
   docker login hub.juniper.net --username <container_registry_username> --password
   <container_registry_password>
   docker pull hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
   ```

   Variables:

   - *<container_registry_username>* and *<container_registry_password>*—Registry access credentials. You can email mailto:contrail-registry@juniper.net to obtain your username and password credentials to access the Contrail Container registry.
   - *<container_tag>*—container tag for the Contrail Command (UI) container deployment for the release that you are installing. The *<container_tag>* for any Contrail Release 21xx image can be found in README Access to Contrail Registry 21XX.

5. Create and save the `command_servers.yml` configuration file on the Contrail Command server.

   The configuration of the `command_servers.yml` file is unique to your environment and complete documentation describing all `command_servers.yml` configuration options is beyond the scope of this

document. Two sample command_servers.yml files for a Contrail environment are provided with this document in "Sample command_servers.yml Files for Installing Contrail Command" on page 54 to provide configuration assistance.

Be aware of the following key configuration parameters when configuring the command_servers.yml file for Contrail Command:

- The contrail_config: hierarchy defines the Contrail Command login credentials:

```
contrail_config:
            database:
                type: postgres
                dialect: postgres
                password: contrail123
            keystone:
                assignment:
                    data:
                    users:
                        admin:
                          password: contrail123
```

> **CAUTION**: For security purposes, we strongly recommend creating unique username and password combinations in your environment.

- (Contrail Networking Release 2003 or earlier) The following configuration lines must be entered if you want to deploy Contrail Insights and Contrail Insights Flows:

> **NOTE**: Appformix and Appformix Flows were renamed Contrail Insights and Contrail Insights Flows. The Appformix naming conventions still appear during product usage, including within these directory names.

```
---
user_command_volumes:
- /opt/software/appformix:/opt/software/appformix
- /opt/software/xflow:/opt/software/xflow
```

The configuration lines must be entered outside of the “command_servers” hierarchy, either immediately after the "---" at the very top of the file or as the last two lines at the very bottom of

the file. See "Complete command_servers.yml File" on page 56 for an example of these lines added at the beginning of the `command_servers.yml` file.

This step is not required to install Contrail Insights and Contrail Insights Flows starting in Contrail Networking Release 2005.

6. Run the `contrail-command-deployer` container to deploy Contrail Command.

```
docker run -td --net host -v <ABSOLUTE_PATH_TO_command_servers.yml_FILE>:/command_servers.yml
--privileged --name contrail_command_deployer hub.juniper.net/contrail/contrail-command-
deployer:<container_tag>
```

where *<ABSOLUTE_PATH_TO_command_servers.yml_FILE>* is the absolute path to the **command_servers.yml** file that you created in step 5, and *<container_tag>* is the container tag for the Contrail Command (UI) container deployment for the release that you want to install. The *<container_tag>* for any Contrail Release 21xx image can be found in README Access to Contrail Registry 21XX.

7. (Optional) Track the progress of step 6.

```
docker logs -f contrail_command_deployer
```

8. Verify that the Contrail Command containers are running:

```
[root@centos254 ~]# docker ps -a
CONTAINER ID  IMAGE    <trimmed>   STATUS  <trimmed>  NAMES
2e62e778aa91  hub.juniper.net/...  Up      <trimmed>  contrail_command
c8442860e462  circleci/postgre...  Up      <trimmed>  contrail_psql
57a666e93d1a  hub.juniper.net/...  Exited  <trimmed>  contrail_command_deployer
```

The `contrail_command` container is the GUI and the `contrail_psql` container is the database. Both containers should have a STATUS of `Up`.

The `contrail-command-deployer` container should have a STATUS of Exited because it exits when the installation is complete.

9. Open a web browser and enter `https://<Contrail-Command-Server-IP-Address>:9091` as the URL. The Contrail Command home screen appears.

Enter the username and password combination specified in the `command_servers.yml` file in step 5. If you use the sample `command_servers.yml` files in "Sample command_servers.yml Files for Installing Contrail Command" on page 54, the username is **admin** and the password is **contrail123**.

> ⚠ **CAUTION**: For security purposes, we strongly recommend creating unique username and password combinations in your environment.

For additional information on logging into Contrail Command, see "How to Login to Contrail Command" on page 62.

How to Provision Servers into the Contrail Cluster

IN THIS SECTION

Use this procedure to provision servers into your Contrail cluster. A Contrail cluster is a collection of interconnected servers that have been provisioned as compute nodes, control nodes, orchestrator nodes, Contrail Insights nodes, Contrail Insights Flows nodes, or service nodes in a cloud networking environment.

Before You Begin

Before you begin:

- Plan your topology.
- Ensure an out-of-band management network is established.
- Ensure Contrail Command is installed. See "How to Install Contrail Command" on page 24.
- Ensure all servers hosting Contrail cluster functions meet the specifications listed in "Server Requirements" on page 23.

How to Provision the Contrail Cluster

To provision the Contrail cluster:

1. (Contrail Networking Release 2003 target release installations using Appformix only) Download the Appformix and—if your also using Appformix Flows—the Appformix Flows images from the Contrail Appformix Download page.

NOTE: Appformix and Appformix Flows were renamed Contrail Insights and Contrail Insights Flows. The Appformix filename conventions are used to name these files for use with Contrail Networking Release 2003.

For Contrail Release 2003, the supported AppFormix version is 3.1.15 and the supported AppFormix Flows version is 1.0.7.

```
appformix-<version>.tar.gz
appformix-platform-images-<version>.tar.gz
appformix-dependencies-images-<version>.tar.gz
appformix-network_device-images-<version>.tar.gz
appformix-openstack-images-<version>.tar.gz
appformix-flows-<version>.tar.gz
appformix-flows-ansible-<version>.tar.gz
```

- Copy the `tar.gz` files to the **/opt/software/appformix/** directory on the Contrail Command server.
- Copy your AppFormix license to the **/opt/software/appformix/** directory.
- (Appformix Flows environments only) Copy the two `appformix-flows` files to the **/opt/software/xflow** directory.

  You can ignore this step if you are not using Appformix Flows.

You can skip this step if you are using Contrail Networking Release 2005 or later or are not using Appformix or Appformix Flows in your environment.

2. Login to Contrail Command at `https://<Contrail-Command-Server-IP-Address>:9091` in most scenarios. See "How to Login to Contrail Command" on page 62 if you are not seeing the Contrail Command login screen at this URL.

   Leave the **Select Cluster** field blank to enter Contrail Command in a wizard that guides you through the cluster provisioning process. If Contrail Command is not currently managing a cluster, this is your only Contrail Command login option.

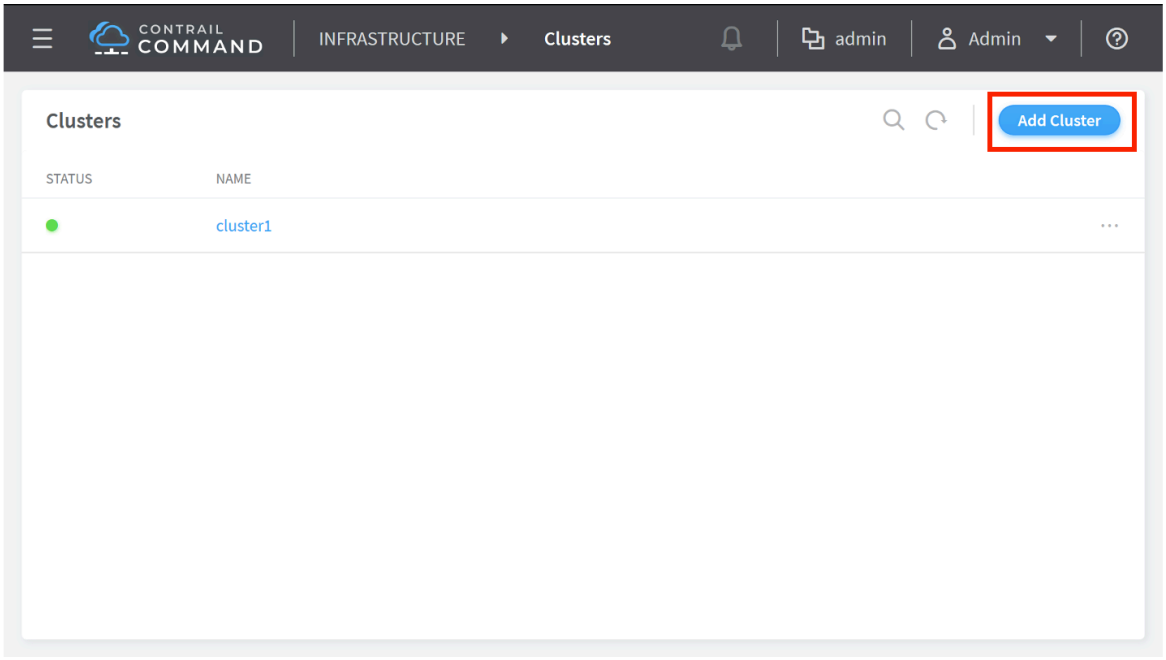
   Your Contrail Command access credentials were specified in the *command_servers.yml* files in step 5 when you installed Contrail Command. If you used the sample *command_servers.yml* file to enable Contrail Command, your username is **admin** and your password is **contrail123**.

NOTE: Username and password combinations are provided in this document for illustrative purposes only. We suggest using unique username and password

combinations to maximize security in accordance with your organization's security guidelines.

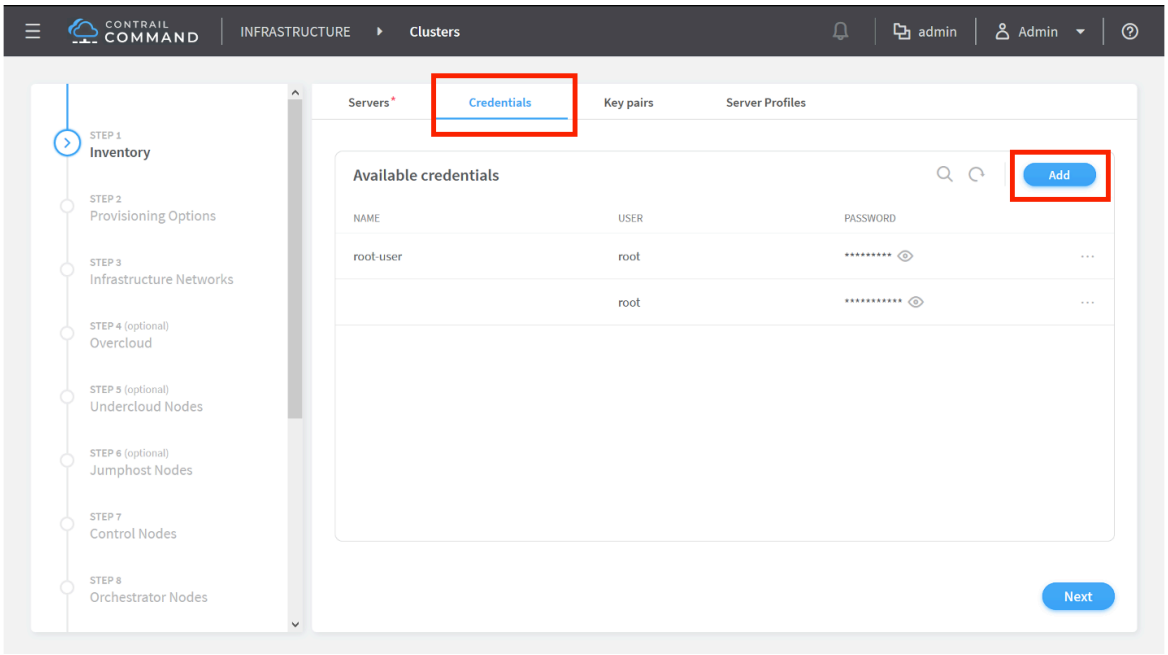
3. You are placed into the **Infrastructure** > **Clusters** menu upon login. Click the **Add Cluster** button to start the cluster provisioning process.

Figure 4: Clusters



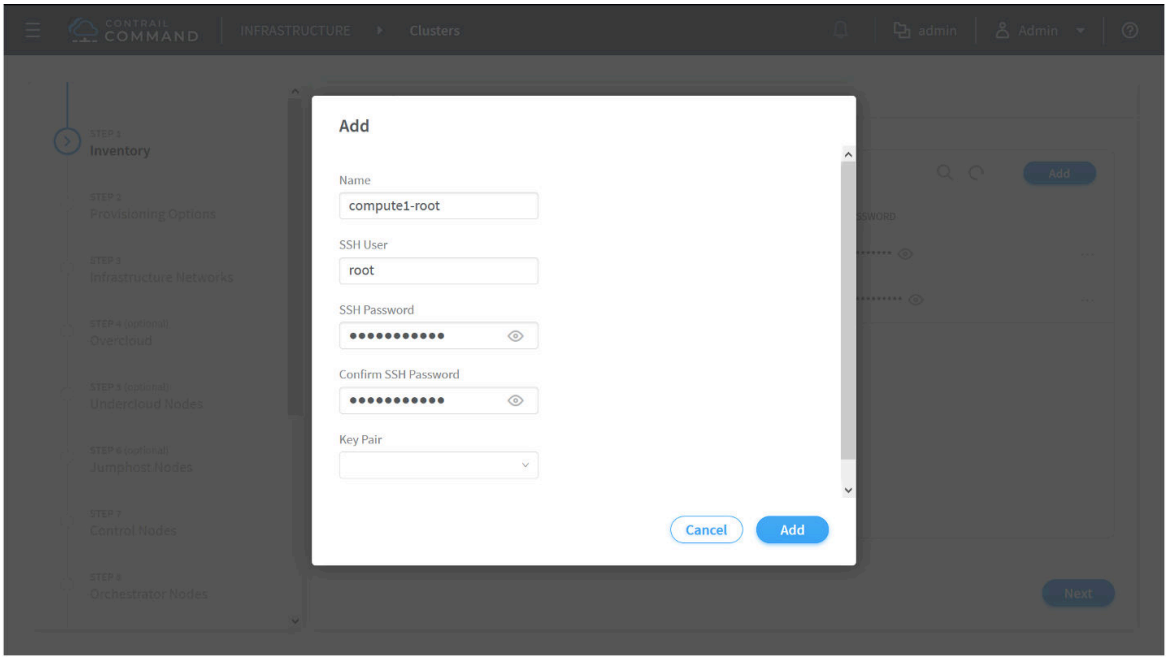
4. Click the **Credentials** tab to move to the **Credentials** box, then the **Add** button to add access credentials for a device that will be added to the cluster.

Figure 5: Available Credentials



5. In the **Add** box, add the access credentials for a device in your cluster. Click the **Add** button to complete the process and add the access credentials.

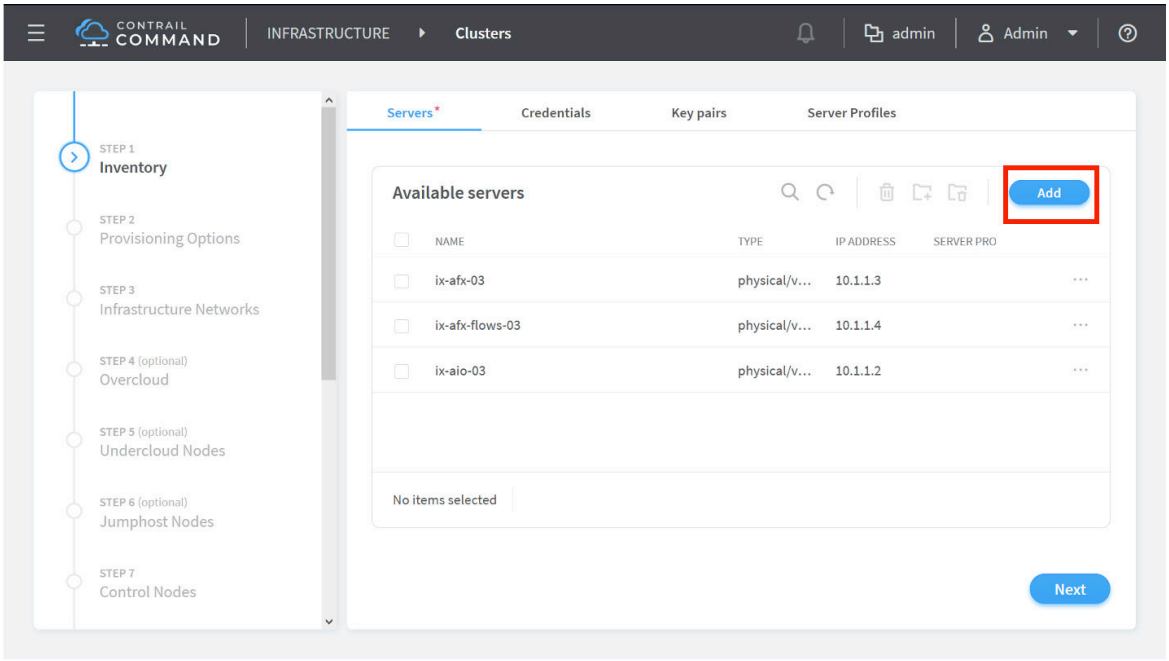
Figure 6: Add Dialog



Repeat steps 4 and 5 to add the access credentials for each server or VM in your cluster.

6. After clicking the **Add** button to add the credentials of your last server or VM, click the **Server** tab to return to the **Available servers** box.
7. Click the **Add** button in the **Available Servers** box.

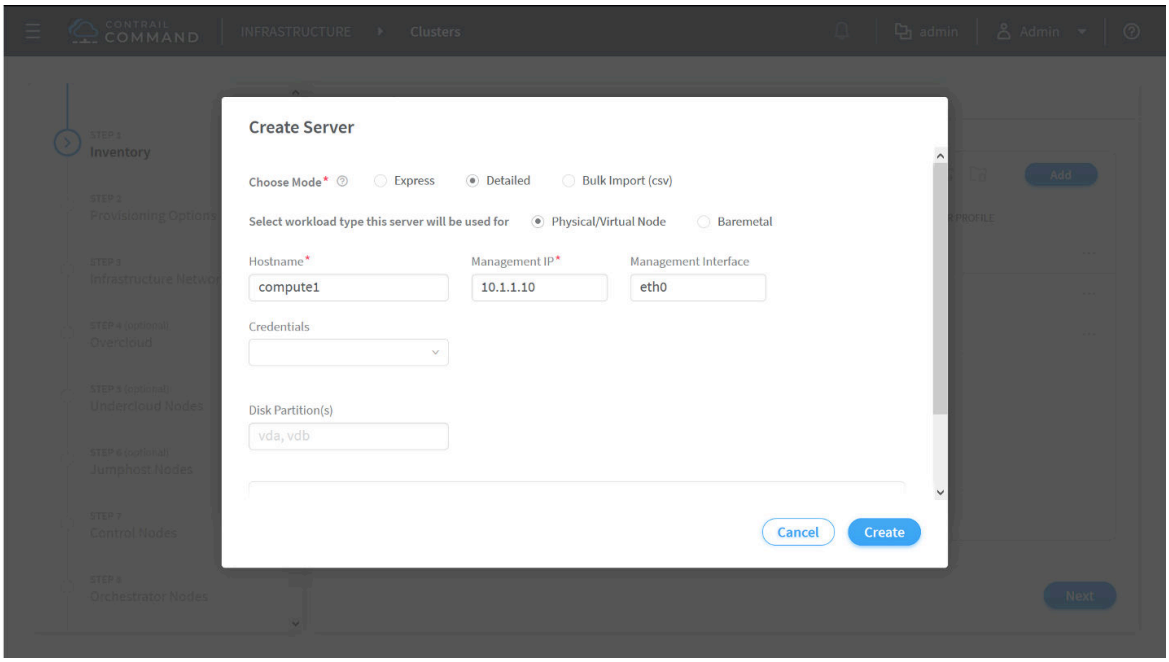
Figure 7: Available Servers



The **Create Server** dialog box appears.

8. Complete the fields in the **Create Server** dialog box for each physical server or VM in your Contrail cluster. Each physical server or VM that will function as a compute node, control node, orchestrator node, Contrail Insights node, Contrail Insights Flows node, or service node in your cluster must be added as a server at this stage of the provisioning process.

Figure 8: Create Server



Field descriptions:

- *Choose Mode*—Options include: *Express*, *Detailed*, or *Bulk Import (CSV)*. We recommend using the *Detailed* or *Bulk Import (CSV)* modes in most environments to ensure all server field data is entered and to avoid performing manual configuration tasks later in the procedure.
  - *Express*—includes a limited number of required fields to enter for each server or VM.
  - *Detailed*—provides all fields to enter for each server or VM.
  - *Bulk Import (CSV)*—Import the physical server or VM fields from a CSV file.
- *Select workload type this server will be used for*
  - *Physical/Virtual Node*—A virtualized physical server or a VM. This is the option used for most servers or VMs in Contrail Networking environments.
  - *Baremetal*—A non-virtualized server.
- *Hostname*—the name of the physical server or VM.
- *Management IP*—the management IP address of the physical server or VM.
- *Management Interface*—the name of the management-network facing interface on the physical server or VM.
- *Credentials*—Select any credentials that appear in the drop-down menu.

- *Disk Partition(s)*—(Optional) Specify the disk partitions that you want to use.

  This field is often left blank.

- *Name* (Network interfaces)—the name of a network-facing interface on the physical server or VM.

- *IP Address* (Network interfaces)—the IP address of the network-facing interface on the physical server or VM.

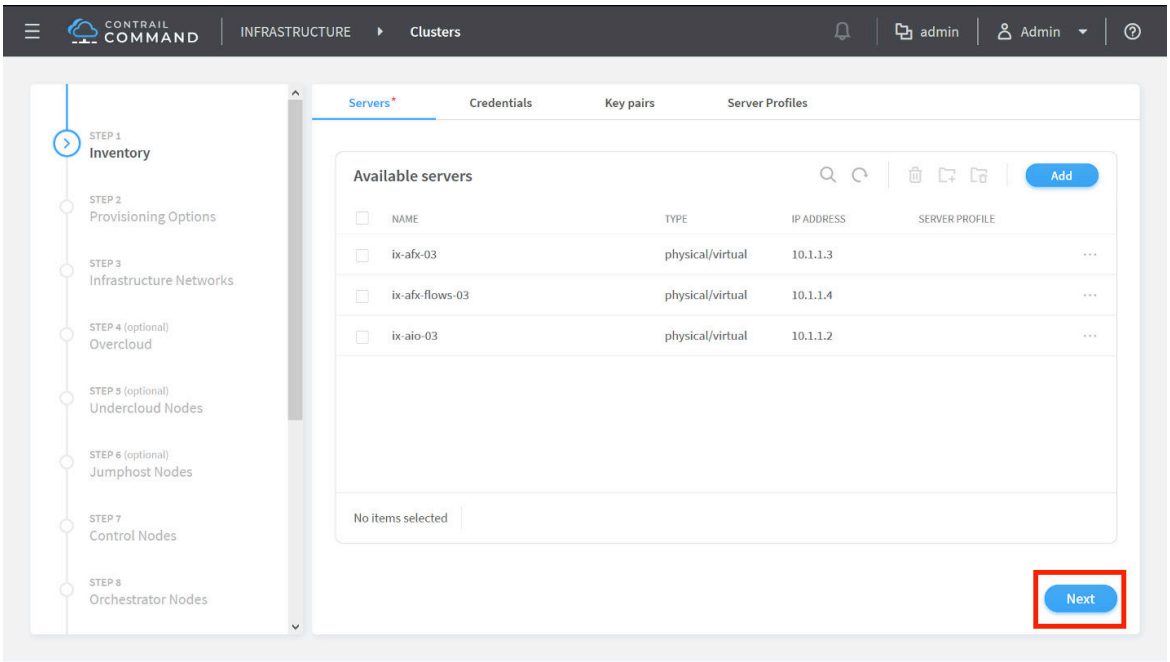
Click **Add** in the *Network Interfaces* box to add additional network interfaces for the server or VM.

Click the **Create** button after completing all fields to add the server or VM.

Repeat this step for each physical server or VM that will function as a compute node, control node, orchestrator node, Contrail Insights node, Contrail Insights Flows node, or service node in the Contrail cluster.

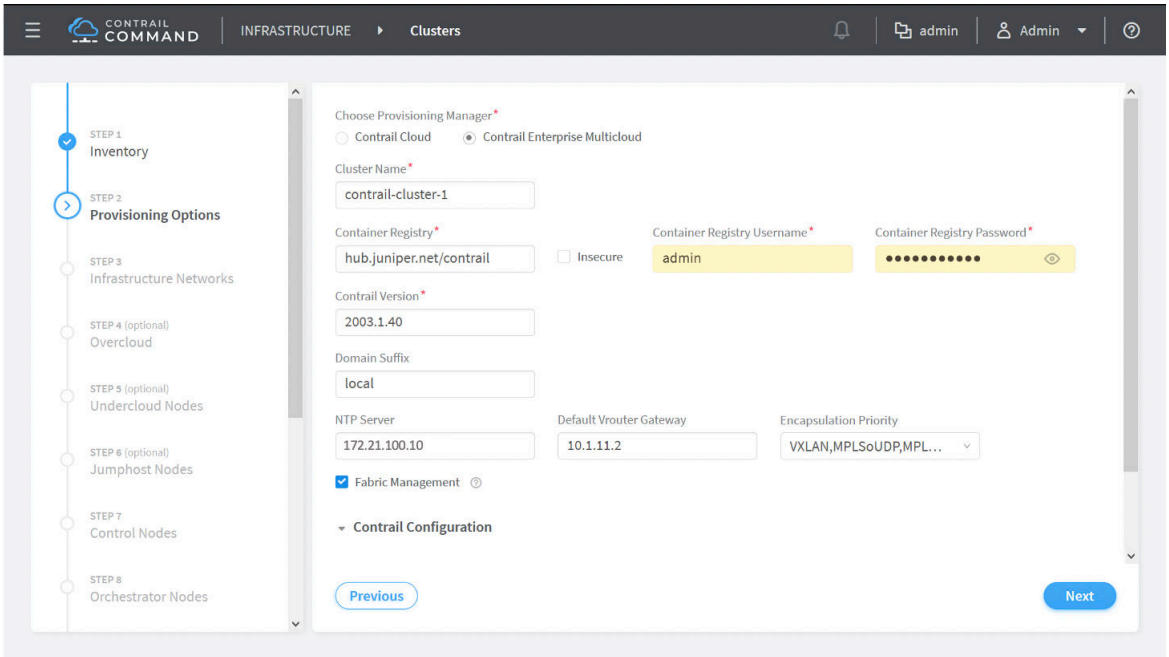
9. You are returned to the **Infrastructure** > **Clusters** > **Servers** menu after adding the final server. Click the **Next** button to proceed to the **Provisioning Options** page.

Figure 9: Available Servers



10. Complete the fields on the **Provisioning Options** page.

Figure 10: Provisioning Options



Field Descriptions:

- *Choose Provisioning Manager*
  - *Contrail Cloud*—Contrail Cloud Provisioning Manager. Do not use this provisioning manager option.
  - *Contrail Enterprise Multicloud*—(Default) Contrail Enterprise Multicloud Provisioning Manager. Select Contrail Enterprise Multicloud as your provisioning manager.

    The remaining steps of this procedure assume Contrail Enterprise Multicloud is selected as the provisioning manager.
- *Cluster Name*—Name the Contrail cluster.
- *Container Registry*—Path to the container registry to obtain the Contrail Networking image. The path to the Juniper container registry is *hub.juniper.net/contrail* and is set as the default container registry path. Enter this path or the path to the repository used by your organization.
- *Insecure* checkbox—This option should only be selected if you want to connect to an insecure registry using a non-secure protocol like HTTP.

  This box is unchecked by default. Leave this box unchecked to connect to the Juniper container registry at *hub.juniper.net/contrail* or to access any other securely-accessible registry.
- *Container Registry Username*—Username to access the container registry.

The Juniper container registry is often used in this field to obtain the Contrail Networking image. Email mailto:contrail-registry@juniper.net to receive a registry username and password combination to access the Juniper container registry.

- *Container Registry Password*—Password to access the container registry.

  The Juniper container registry is often used in this field to obtain the Contrail Networking image. Email mailto:contrail-registry@juniper.net to receive a registry username and password combination to access the Juniper container registry.

- *Contrail Version*—Specify the version of the Contrail Networking image to use for the upgrade that is in the repository.

  You can use the *latest* tag to retrieve the most recent image in the repository, which is the default setting. You can also specify a specific release in this field using the version's release tag.

  See README Access to Contrail Registry 21XX to obtain the release tag for any Contrail Networking Release 21XX release tag.

- *Domain Suffix*—(Optional) Domain name for the cluster.

- *NTP Server*—The IP address of the NTP server.

- *Default vRouter Gateway*—The IP address of the default vRouter gateway.

  This address is typically the IP address of the interface on the leaf device in the fabric that connects to the server's network-facing interface.

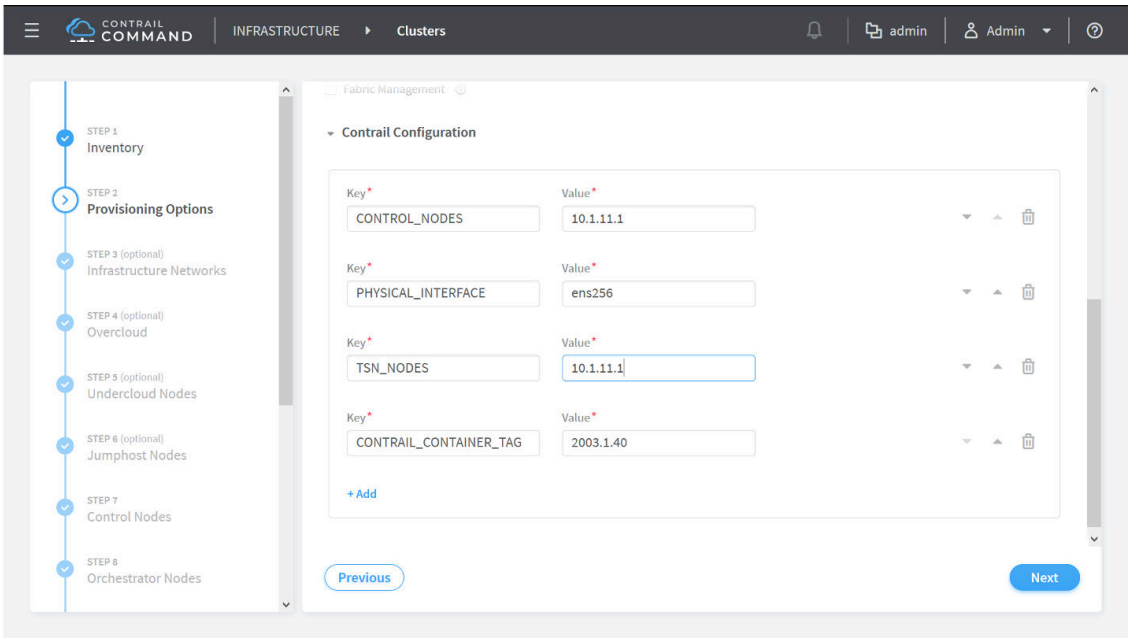
- *Encapsulation Priority*—Select the Encapsulation priority order from the drop down menu.

  Select *VXLAN, MPLSoUDP, MPLSoGRE* in most Contrail Networking environments.

- *Fabric Management* checkbox—Select this option if your deploying in an environment using Openstack for orchestration.

- To fill the Contrail Configuration details, click **Add**. This adds the Key-Value pair to the Contrail Configuration section as shown below:

Figure 11: Contrail configuration



Enter the **Key**/**Value** pair details following the table given below:

Table 2: Contrail Configuration Section

Key	Value
CONTROL_NODES	List of comma-separated fabric underlay interface IP addresses of the Contrail Control node. For example, using Value as '10.1.11.1' implies you'll be installing the control node on the Contrail Cluster server. This IP address is therefore the IP address that connects the Contrail Cluster server to the fabric underlay.
PHYSICAL_INTERFACE	Name of the interface that connects to the fabric underlay.
TSN_NODES	This field is 'OPTIONAL'. It specifies the list of comma-separated fabric underlay interface IP addresses of the Contrail Service node. For example, you will be installing the service node, '10.1.11.1' on the Contrail Cluster server. This IP address is therefore the IP address that connects the Contrail Cluster server to the fabric underlay.

Table 2: Contrail Configuration Section *(Continued)*

Key	Value
CONTRAIL_CONTAINER_TAG	The container tag for the desired Contrail and OpenStack release combination as specified in README Access to Contrail Registry 21XX.
API__DEFAULTS__enable_latency_stats_log	(Optional. Available starting in Contrail Networking Release 2008) Enable logging and storing of latency statistics in Contrail Networking and Contrail Insights for calls to Cassandra, Zookeeper, and Keystone from the API server.
API__DEFAULTS__enable_api_stats_log	(Optional Available starting in Contrail Networking Release 2008) Enable logging and storing of latency statistics and call time statistics in Contrail Networking and Contrail Insights for Rest API calls.

For more information about the key-value descriptions, see the following references:

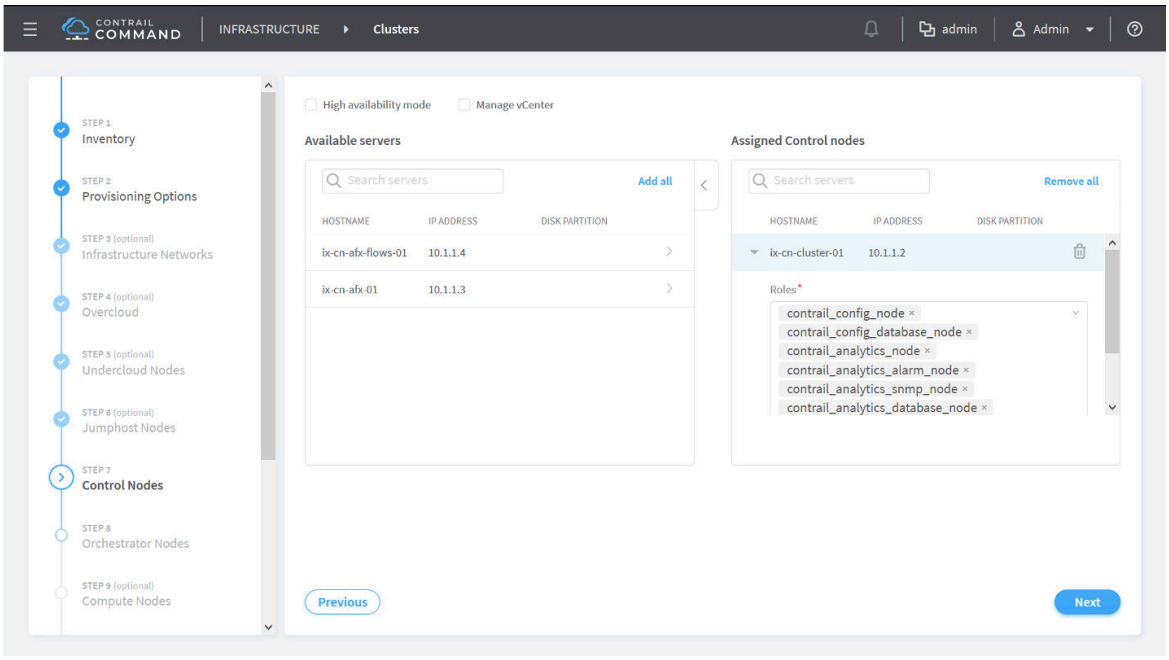
- Contrail Networking Quick Start Guide
- Contrail Cloud Getting Started Guide
- Contrail Enterprise Multicloud: Contrail Service Node (CSN) Overview

Click the **Next** button to proceed to the **Control Nodes** provisioning page.

11. From the **Control Nodes** provisioning page, assign any server that you created in step 8 as a control node by clicking the **>** icon next to the server to move it into the **Assigned Control Nodes** box.

    You have the option to remove roles from a control node within the **Assigned Control Nodes**. There is no need to remove control node roles in most deployments and you should only remove roles if you are an expert user familiar with the consequences.

Figure 12: Assigned Control Nodes



(Installations using VMWare vCenter only) Complete the following steps to install a control node that is integrated with VMware vCenter. For additional information on vCenter integration with Contrail Networking, see Understanding VMware-Contrail Networking Fabric Integration.

NOTE: Starting from Contrail Networking Release 21.4.L3, Juniper has stopped VMWare support with Contrail Networking.

Prerequisites:

- Installed vCenter version 6.5 or later.
- Installed ESX version 6.5 or later.
- A vCenter license with Distributed Virtual Switch (DVS) support.
- Login credentials for vCenter.

To perform the integration:

a. Select the **Manage vCenter** check box.

   The vCenter Credentials section is displayed.

b. Enter the following information:

   - Enter the vCenter IP address in the **vCenter IP Address** field.

- In the **Data Center Name** field, enter the name of the data center under vCenter that CVFM will work on.
- Enter the vCenter username in the **Username** field.
- Enter the vCenter password in the **Password** field.

c. Click **>**, next to the name of the server, to assign a server from the Available Servers table as a control node. The server is then added to the Assigned Control Nodes table.

Note that the `contrail_vcenter_fabric_manager_node` is added to the list of roles.

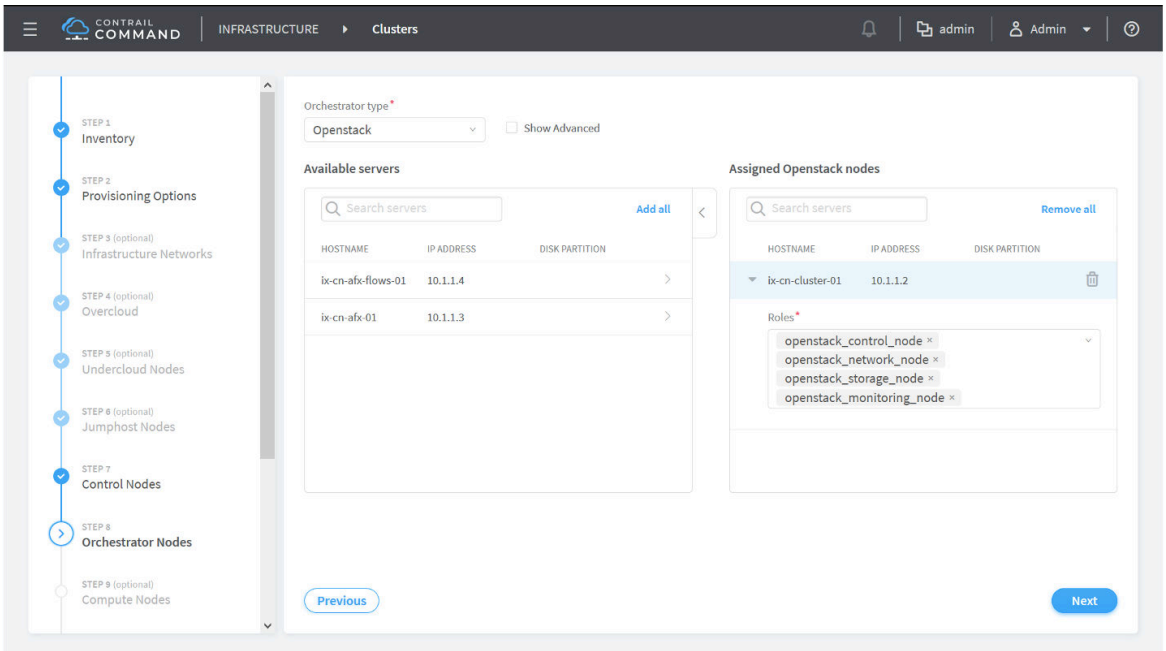
d. Click **Next**.

After assigning all control nodes, click the **Next** button to move to the **Orchestrator Nodes** provisioning page.

12. Select your orchestration platform from the **Orchestrator Type** drop-down menu.

Assign any one of the servers that you created in step 8 as an orchestrator node by clicking the **>** icon next to the server to move it into the **Assigned nodes** box.

Figure 13: Assigned nodes



The remaining processes for this step depend on your orchestration platform:

- *Openstack*

  Click the **Show Advanced** box then scroll to **Kolla Globals** and click **+Add**.

Add the following Kolla global **Key** and **Value** pairs in most environments:

Table 3: Kolla Globals Key Value Pairs

Key	Value
enable_haproxy	no
enable_ironic	no
enable_swift	yes
swift_disk_partition_size	20GB

After assigning all orchestrator nodes and Kolla global keys and values, click the **Next** button to progress to the **Compute Nodes** provisioning page.

- *Kubernetes*

  Select the Kubernetes nodes from the list of available servers and assign corresponding roles to the servers.

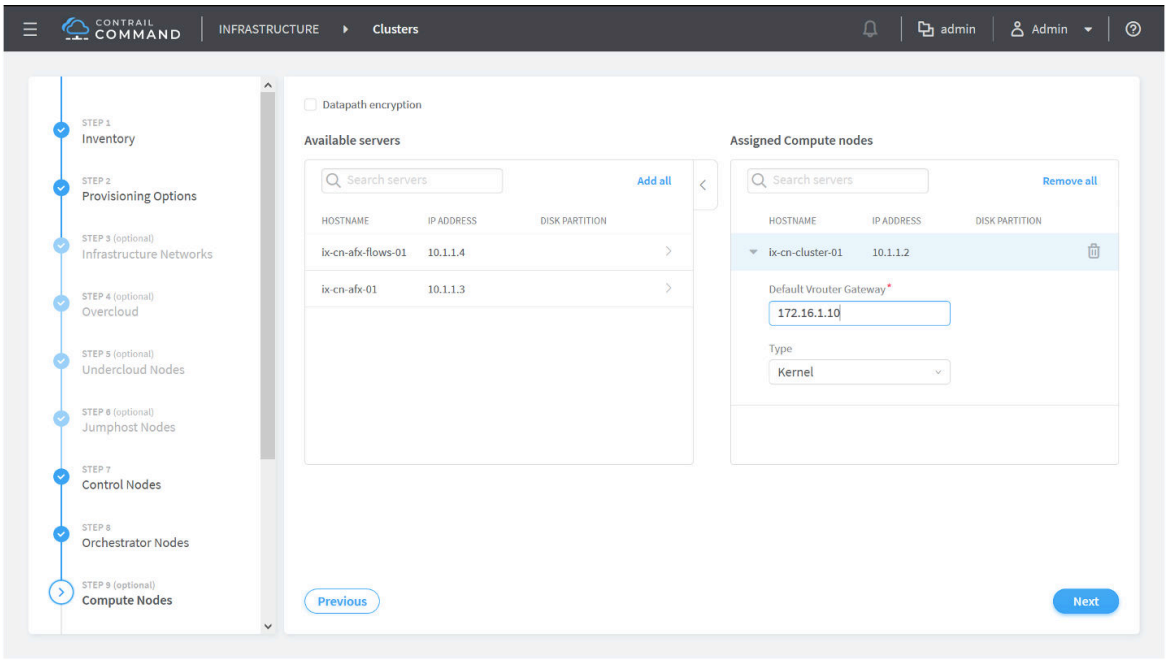
  By default, the Kubernetes nodes are assigned the kubernetes_master_node, kubernetes_kubemanager_node, and kubernetes_node roles.

  After assigning roles to all nodes, click the **Next** button to progress to the **Compute Nodes** provisioning page.

13. Assign any server that you created in step 8 as a compute node by clicking the **>** icon next to the server to move it into the **Assigned Compute nodes** box.

    Enter the default vRouter gateway IP Address in the **Default Vrouter Gateway** box after moving the server into the **Assigned Compute nodes** box.

Figure 14: Assigned Compute nodes



After assigning all compute nodes, click the **Next** button to progress to the **Contrail Service Nodes** provisioning page.

14. Assign any server that you created in step 8 as a Contrail Services node by clicking the **>** icon next to the server to move it into the **Assigned Service Nodes** box.

    Contrail service nodes are only used in environments with bare metal servers. If you are not using Contrail Service nodes in your environment, click the **Next** button without assigning any servers into the **Assigned Service Nodes** box.

    The default vRouter gateway IP Address might be autocompleted in the **Default Vrouter Gateway** box. This default vRouter gateway is typically the IP address of a leaf device in the fabric that is directly connected to the server fulfilling the service node role.

    After assigning all Contrail Service nodes, click the **Next** button to progress to the **Insights Nodes** provisioning page.

> **NOTE**: The **Insights Nodes** provisioning workflow is called the **Appformix Nodes** workflow in Contrail Networking Release 2005 and earlier releases.

15. Contrail Insights is an optional product that isn't used in all environments. If your are not using Contrail Insights in your environment, simply click the **Next** button without assigning a server as an Appformix node in this step.

NOTE: Appformix was renamed Contrail Insights. The Appformix naming is still used in some Contrail Command screens.

- *Contrail Insights*

  If you are using Contrail Insights in your environment, click the **>** icon next to the server or VM in the **Available servers** box to move it into the **Assigned Insights Nodes** box.

  NOTE: The **Assigned Insights Nodes** box is called **Assigned Appformix Nodes**in Contrail Networking Release 2005 and earlier releases.

  By default, the server is assigned the *appformix_platform_node* role. You can maintain this default setting in most environments. If the role needs to be changed, click within the **Roles** drop-down menu and select from the available roles.

- *Contrail Insights Flows*

  If you are also using Contrail Insights Flows in your environment, click the **>** icon next to the server or VM in the **Available servers** box to move it into the **Assigned Insights Nodes** box.

  NOTE: The **Assigned Insights Nodes** box is called **Assigned Appformix Nodes**in Contrail Networking Release 2005 and earlier releases.

  Click within the **Roles** drop-down menu and uncheck the default *appformix_platform_node* role selection. Select *appformix_bare_host_node* from within the **Roles** drop-down menu to set it as the role.

Click the **Next** button to progress to the **Appformix Flows** provisioning page.

16. Contrail Insights Flows is an optional product that isn't used in all environments. If your are not using Contrail Insights Flows in your environment, simply click the **Next** button without assigning a server as an Appformix Flows node in this step.

NOTE: Appformix Flows was renamed Contrail Insights Flows. The Appformix Flows naming is still used on this Contrail Command page.

If you are using Contrail Insights Flows in your environment, make the following configuration selections:

- *Appformix Flows Node Provisioning Type*:

  - *Out-of-Band*—(Default) The Appformix Flows node is managed from an out-of-band management network.
  - *In-Band*—The Appformix Flows node is managed from an in-band connection.
- *Virtual IP Address*—The virtual IP address management address on the Appformix Flows node that connects the node to the management network.

(Contrail Insights and Contrail Insights Flows on same server only) Starting in Contrail Networking Release 2008, you can enable Contrail Insights and Contrail Insights Flows on the same server node.

Perform these steps if you are enabling Contrail Insights and Contrail Insights Flows on the same node:

a. Click the **Show Advanced** box. The advanced configuration options appear.

b. From the **AppFormix Flows Configuration Parameters** box, click the **+Add** option to open the **Key** and **Value** configuration options.

   Add the following key value pairs:

   - **Key**: health_port

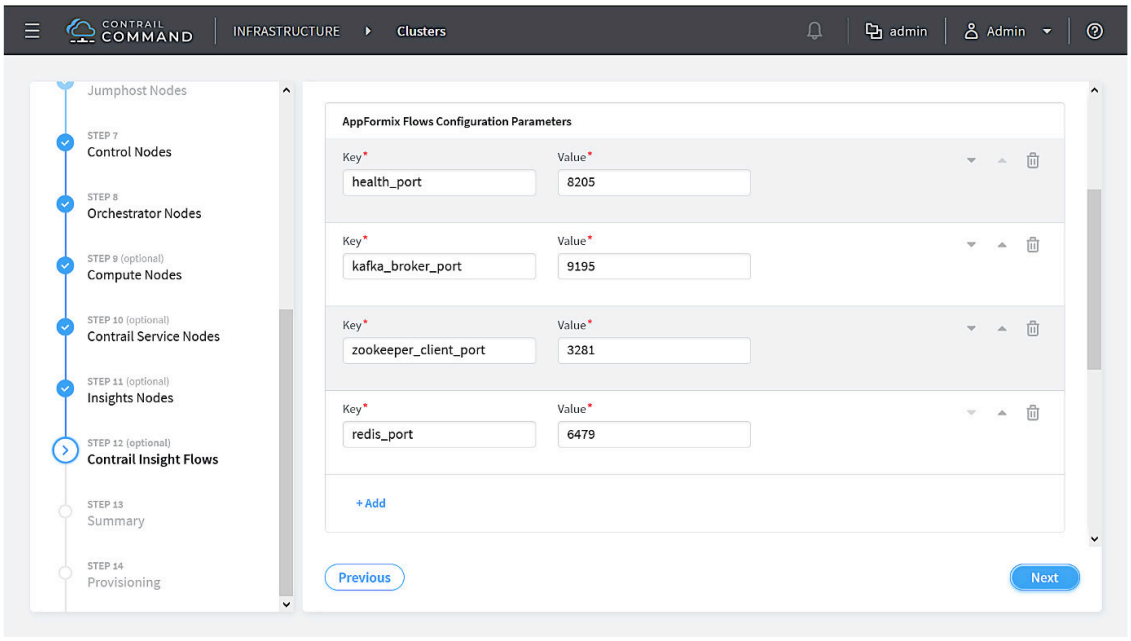
     Value: 8205
   - **Key**: kafka_broker_port

     Value: 9195
   - **Key**: zookeeper_client_port

     Value: 3281
   - **Key**: redis_port

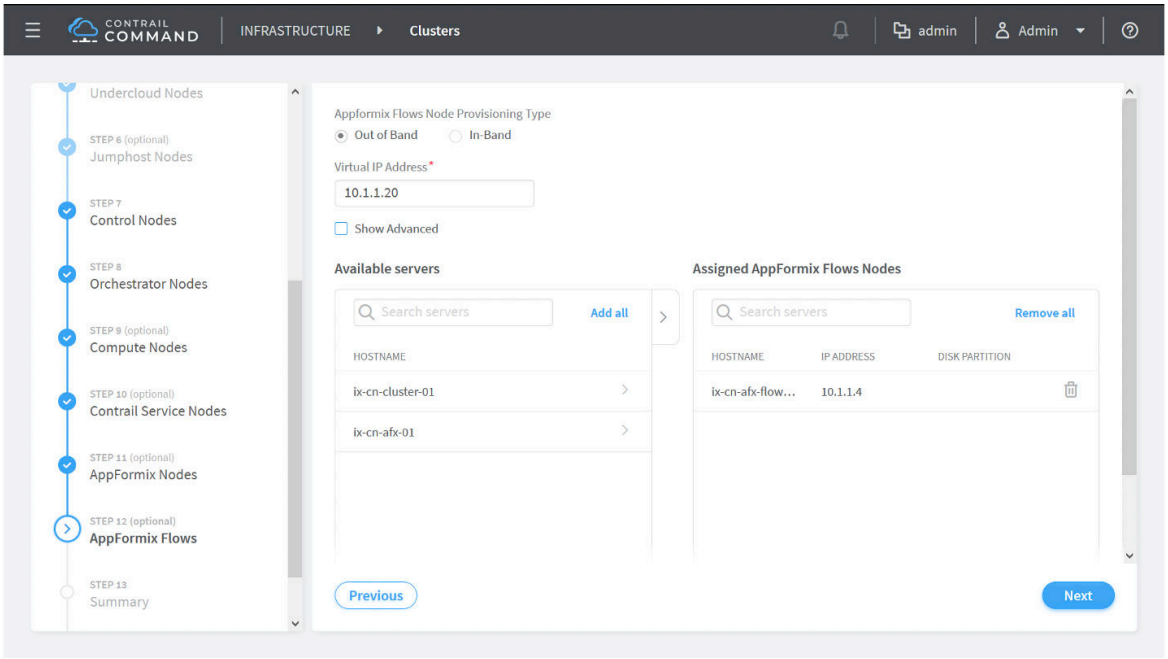
     Value: 6479

Figure 15: AppFormix Flows Configuration Parameters



Click the **>** icon next to the server or VM in the **Available servers** box to move it into the **Assigned AppFormix Flows Nodes** box.

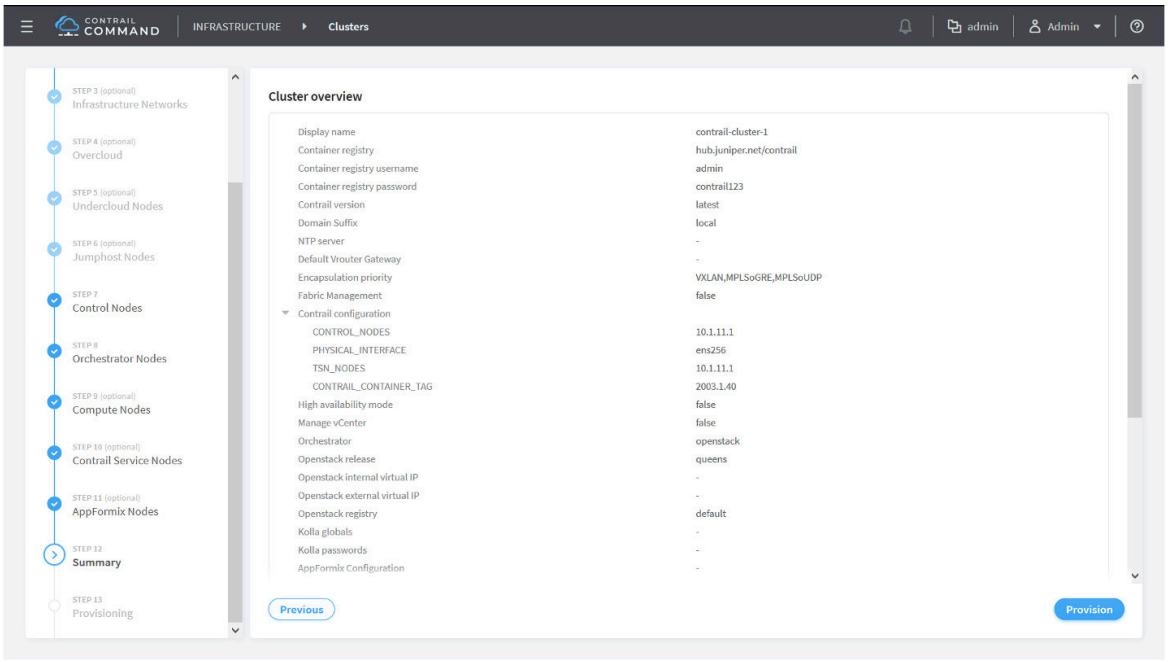
Figure 16: Available servers



Click the **Next** button to progress to the **Summary** page.

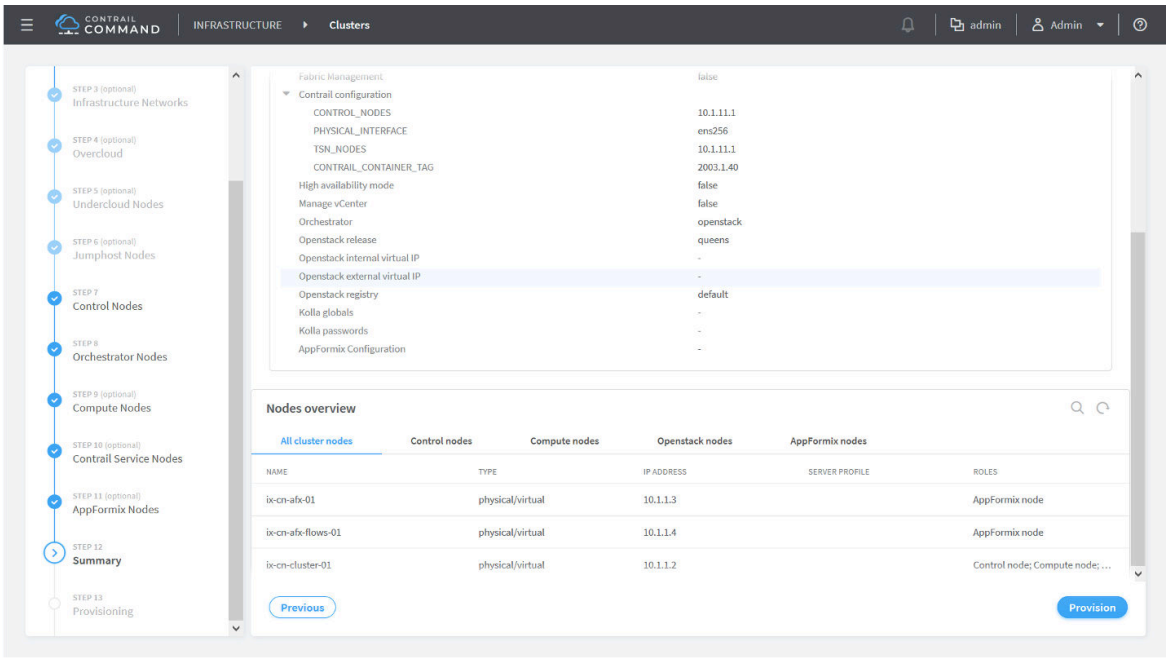
17. Review your settings in the **Cluster overview** screen.

Figure 17: Cluster Overview



Click any tab in the **Nodes Overview** box to review any configuration.

Figure 18: Nodes Overview



Click the **Provision** button after verifying your settings to provision the cluster.

The cluster provisioning process begins. This provisioning process time varies by environment and deployment. It has routinely taken 90 minutes or more in our testing environments.

18. (Optional) Monitor the provisioning process by logging onto the Contrail Command server and entering the **docker exec contrail-command tail /var/log/contrail/deploy.log** command.
19. When the provisioning process completes, click the **Proceed to Login** option.

    You are taken to the Contrail Command login screen.
20. Login to Contrail Command from the web browser.

    Enter the following values:

    - **Select Cluster**: Select a Contrail Cluster from the dropdown menu. The cluster is presented in the *<cluster-name>-<string>* format. The *<cluster-name>* options should include the cluster that you just created and should match the cluster name assigned in step 10 of this procedure.
    - **Username**: Enter the username credential to access Contrail Command. This username was set in the `command_servers.yml` file configured in step 5 of the "How to Install Contrail Command" on page 24 procedure.
    - **Password**: Enter the password credential to access Contrail Command. This password was set in the `command_servers.yml` file configured in step 5 of the "How to Install Contrail Command" on page 24 procedure.

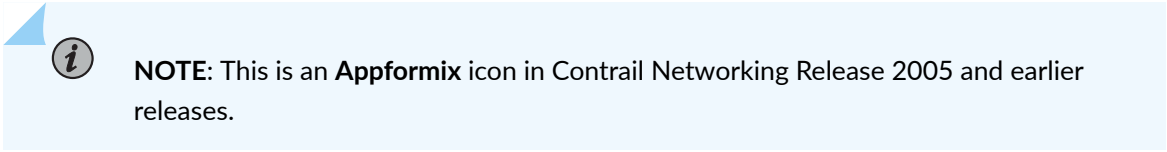
- **Domain**: You can often leave this field blank. Contrail Command logs into the **default_domain**—the default domain for all orchestration platforms supported by Contrail Command except Canonical Openstack—when the **Domain** field is empty.

  If you are logging into a cluster that includes Canonical Openstack as it's orchestration platform, you can enter *admin_domain*—the default domain name for Canonical Openstack—in the **Domain** field if your default domain name was not manually changed.

  You can enter the personalized domain name of your cloud network's orchestration platform in the **Domain** field if you've changed the default domain name.

See "How to Login to Contrail Command" on page 62 for additional information on logging into Contrail Command.

21. (Optional. Contrail Insights only) Click the **Contrail Insights** icon on the bottom-left hand corner of the Contrail Command page to open Contrail Insights.



NOTE: This is an **Appformix** icon in Contrail Networking Release 2005 and earlier releases.

If you are not accessing Contrail Command through the fabric network, you might also have to configure an External IP address to access Contrail Insights externally. Navigate to **Infrastructure > Advanced Options > Endpoints** and locate **insights** in the **Prefixes** list. Click the **Edit** button—the pencil icon—and change the **Public URL** field to a usable external IP address.

Contrail Insights Flows is integrated into Contrail Command. See Contrail Insights Flows in Contrail Command.

Sample command_servers.yml Files for Installing Contrail Command

IN THIS SECTION

Minimal command_servers.yml file

The following sample file has the minimum configuration that you need when you install Contrail Command.

CAUTION: For security purposes, we strongly recommend creating unique username and password combinations in your environment. Username and password combinations are provided in this example for illustrative purposes only.

```
---
# Required for Appformix and Appformix Flows installations in Release 2003 and earlier
user_command_volumes:
- /opt/software/appformix:/opt/software/appformix
- /opt/software/xflow:/opt/software/xflow
command_servers:
    server1:
        ip: <IP Address>  # IP address of server where you want to install Contrail Command
        connection: ssh
        ssh_user: root
        ssh_pass: <contrail command server password>
        sudo_pass: <contrail command server root password>
        ntpserver: <NTP Server address>

        registry_insecure: false
        container_registry: hub.juniper.net/contrail
        container_tag: <container_tag>
        container_registry_username: <registry username>
        container_registry_password: <registry password>
        config_dir: /etc/contrail

        contrail_config:
            database:
                type: postgres
                dialect: postgres
                password: contrail123
            keystone:
                assignment:
                    data:
                      users:
                        admin:
                          password: contrail123
            insecure: true
            client:
              password: contrail123
```

Complete command_servers.yml File

The following sample file has an exhaustive list of configurations and supporting parameters that you can use when you install Contrail Command.

CAUTION: For security purposes, we strongly recommend creating unique username and password combinations in your environment. Username and password combinations are provided in this example for illustrative purposes only.

```
---
# Required for Appformix and Appformix Flows installations in Release 2003 and earlier
user_command_volumes:
- /opt/software/appformix:/opt/software/appformix
- /opt/software/xflow:/opt/software/xflow

# User defined volumes
#user_command_volumes:
#  - /var/tmp/contrail:/var/tmp/contrail

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

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

        # contrail command container configurations given here go to /etc/contrail/contrail.yml
        contrail_config:
```

```
# Database configuration. PostgreSQL supported
database:
    type: postgres
    dialect: postgres
    host: localhost
    user: root
    password: contrail123
    name: contrail_test
    # Max Open Connections for DB Server
    max_open_conn: 100
    connection_retries: 10
    retry_period: 3s

# Log Level
log_level: debug

# Cache configuration
cache:
  enabled: true
  timeout: 10s
  max_history: 100000
  rdbms:
    enabled: true

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

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

  # Enable GRPC or not
  enable_grpc: false

  # Static file config
  # key: URL path
```

```
    # value: file path. (absolute path recommended in production)
    static_files:
        /: /usr/share/contrail/public

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

    notify_etcd: false

    # VNC Replication
    enable_vnc_replication: true

# Keystone configuration
keystone:
    local: true
    assignment:
        type: static
        data:
          domains:
            default: &default
              id: default
              name: default
          projects:
            admin: &admin
              id: admin
              name: admin
              domain: *default
            demo: &demo
              id: demo
              name: demo
              domain: *default
          users:
            admin:
              id: admin
              name: Admin
              domain: *default
              password: contrail123
              email: admin@juniper.nets
              roles:
```

```
                - id: admin
                  name: admin
                  project: *admin
              bob:
                id: bob
                name: Bob
                domain: *default
                password: bob_password
                email: bob@juniper.net
                roles:
                - id: Member
                  name: Member
                  project: *demo
    store:
        type: memory
        expire: 36000
    insecure: true
    authurl: https://localhost:9091/keystone/v3

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

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

watcher:
  enabled: false
  storage: json

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

compilation:
```

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

              # Plugin configuration
              plugin:
                  handlers:
                      create_handler: 'HandleCreate'
                      update_handler: 'HandleUpdate'
                      delete_handler: 'HandleDelete'

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

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

Disaster Recovery and Troubleshooting

SUMMARY

This section lists commonly seen errors and failure scenarios and procedures to fix them.

IN THIS SECTION

Problem

Description

Recovering the Galera Cluster Upon Server Shutdown—In an OpenStack HA setup provisioned using Kolla and OpenStack Rocky, if you shut down all the servers at the same time and bring them up later, the Galera cluster fails.

Solution

To recover the Galera cluster, follow these steps:

1. Edit the **/etc/kolla/mariadb/galera.cnf** file to remove the `wsrep` address on one of the controllers as shown here.

```
wsrep_cluster_address = gcomm://
#wsrep_cluster_address = gcomm://10.x.x.8:4567,10.x.x.10:4567,10.x.x.11:4567
```

NOTE: If all the controllers are shut down in the managed scenario at the same time, you must select the controller that was shut down last.

2. Docker start mariadb on the controller on which you edited the file.
3. Wait for a couple of minutes, ensure that the mariadb container is not restarting, and then Docker start mariadb on the remaining controllers.
4. Restore the **/etc/kolla/mariadb/galera.cnf** file changes and restart the mariadb container on the previously selected controller.

Problem

Description

Containers from Private Registry Not Accessible—You might have a situation in which containers that are pulled from a private registry named CONTAINER_REGISTRY are not accessible.

Solution

To resolve, check to ensure that REGISTRY_PRIVATE_INSECURE is set to **True**.

RELATED DOCUMENTATION

Video: Using Contrail Command to Install Contrail Networking 2005 and Contrail Insights

Server Requirements and Supported Platforms | **16**

How to Login to Contrail Command

Contrail Command is a single pane-of-glass GUI that configures and monitors Contrail Networking. You can login to Contrail Command using these instructions.

To login to Contrail Command:

Before you begin:

- Install Contrail Command. See "How to Install Contrail Command and Provision Your Contrail Cluster" on page 22.

1. Open a web browser and enter `https://<Contrail-Command-Server-IP-Address>:<port-number>` as the URL.

   The default port number is **9091** in most environments. The port number can be reset using the **address:** field in the **server:** hierarchy within the **command_servers.yml** file.

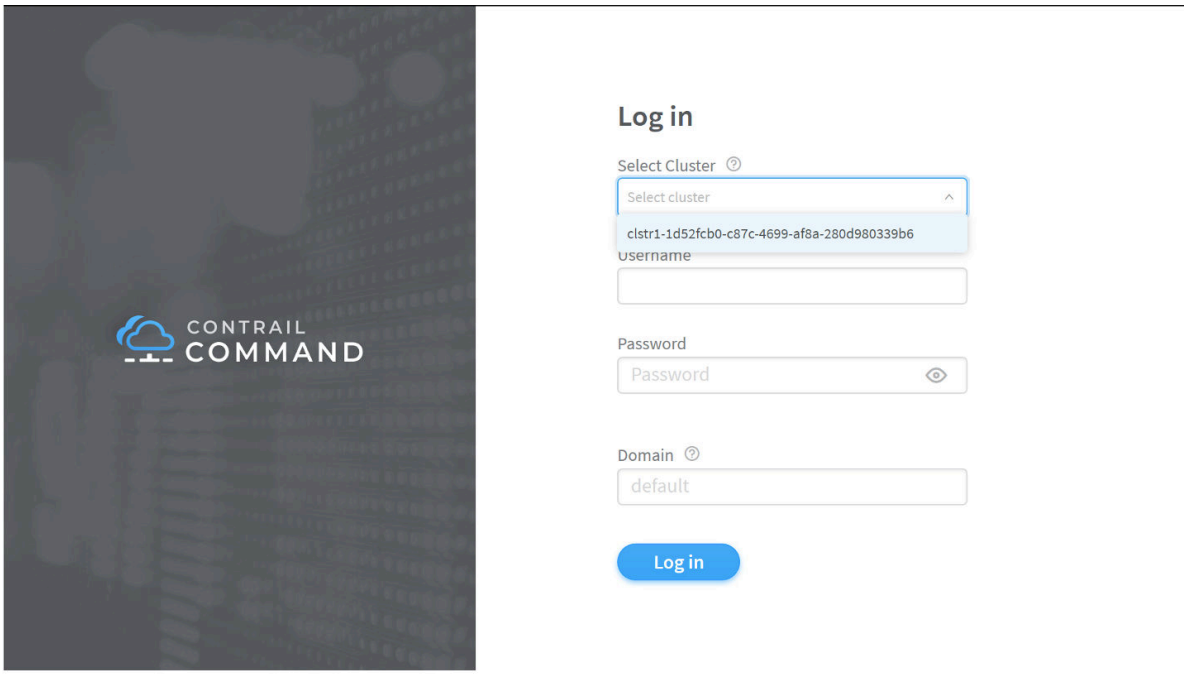
   The Contrail Command home screen appears.

2. Select your Contrail cluster from the **Select Cluster** drop-down menu. Leave this field blank if you are logging into Contrail Command to create a cluster.

   Your cluster is presented within the drop-down menu in the *<cluster-name>-<string>* format. Your *cluster-name* was defined during the cluster creation process and the *string* is a randomly-generated character string.

   In Figure 19 on page 63, the *cluster-name* is **clstr1**.

Figure 19: Contrail Command Login Homepage—Select Cluster



3. Enter the username and password credentials in the **Username** and **Password** fields. The username and password credentials are set in the `command_servers.yml` file during the Contrail Command installation. See "How to Install Contrail Command and Provision Your Contrail Cluster" on page 22.

   For security purposes, we strongly recommend creating unique username and password combinations in your environment. If you didn't change your password in the `command_servers.yml` file, however, the default username to access Contrail Command in most deployments is **admin** and the password is **contrail123**.

4. Enter the domain name of the cluster in the **Domain** field.

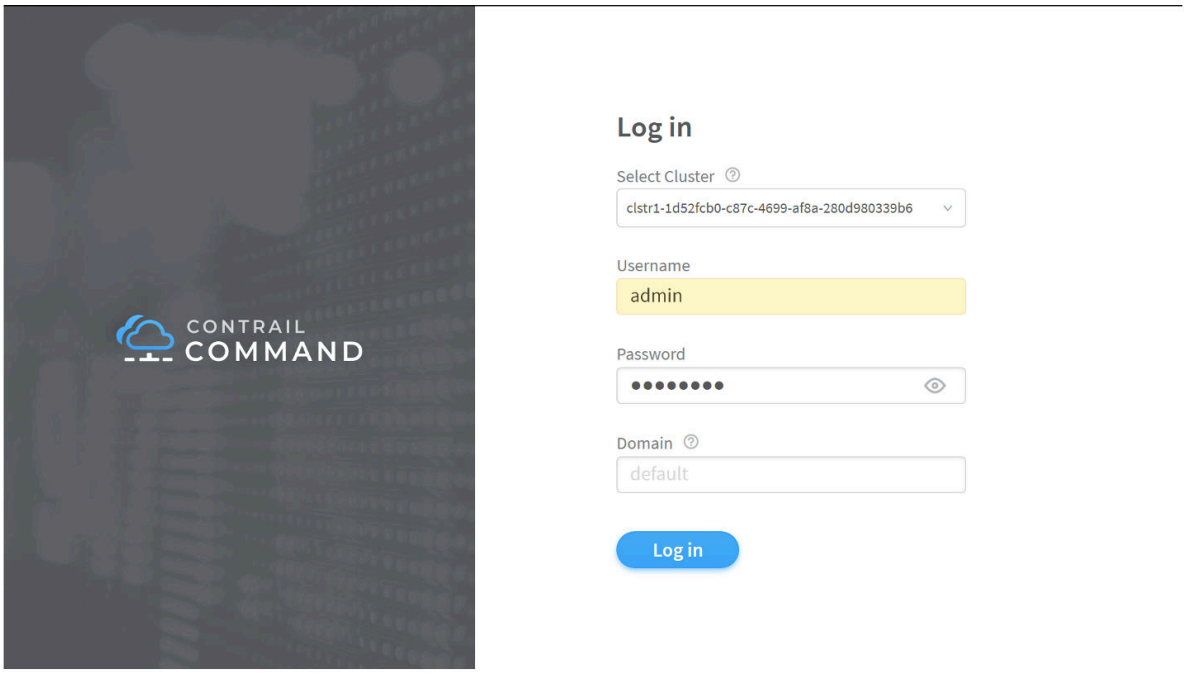
   You can often leave this field blank. Contrail Command logs into the **default_domain**—the default domain for all orchestration platforms supported by Contrail Command except Canonical Openstack—when the **Domain** field is empty.

   You may have to enter a domain in the following use cases:

   - Canonical Openstack orchestration. If you are logging into a cluster that includes Canonical Openstack as it's orchestration platform, you can enter *admin_domain*—the default domain name for Canonical Openstack—in the **Domain** field if your default domain name was not manually changed.

   - You have manually changed the domain name in your cloud network's orchestration platform. You can enter the personalized domain name of your cloud network's orchestration platform in the **Domain** field if you've changed the default domain name.

Figure 20: Contrail Command Login Page Example



5. Press the **Log in** button to enter Contrail Command.

RELATED DOCUMENTATION

- How to Install Contrail Command and Provision Your Contrail Cluster | 22
- Video: Using Contrail Command to Install Contrail Networking 2005 and Contrail Insights
- Contrail Networking Installation and Upgrade Guide

Navigating the Contrail Command UI

IN THIS SECTION

Contrail Networking Release 2003 introduces a redesigned Contrail Command UI. This topic describes how to navigate the UI, some of the new and improved features that the UI offers, and the supported browsers to install Contrail Command.

Using the Get Started with Contrail Enterprise Multicloud Panel

Starting with Contrail Networking Release 2003, you can use the *Get Started with Contrail Enterprise Multicloud* panel in Contrail Command. This *Get Started* panel provides a user-friendly walkthrough of initial Contrail Command configuration tasks. The panel includes **Begin** buttons that allow for quick task initiation and a dynamic tracking mechanism that tracks task progress.

The *Get Started* panel appears automatically when Contrail Command is initially accessed and can always be opened by selecting the **Get Started with Contrail** option in the **?** help menu. If you choose to close the panel, it remains closed within Contrail Command—including across login sessions—unless you choose to open the panel by selecting the **Get Started with Contrail** option.

Figure 21 on page 66 shows the *Get Started with Contrail Enterprise Multicloud* panel home screen.

Figure 21: Get Started with Contrail Enterprise Multicloud Home Screen

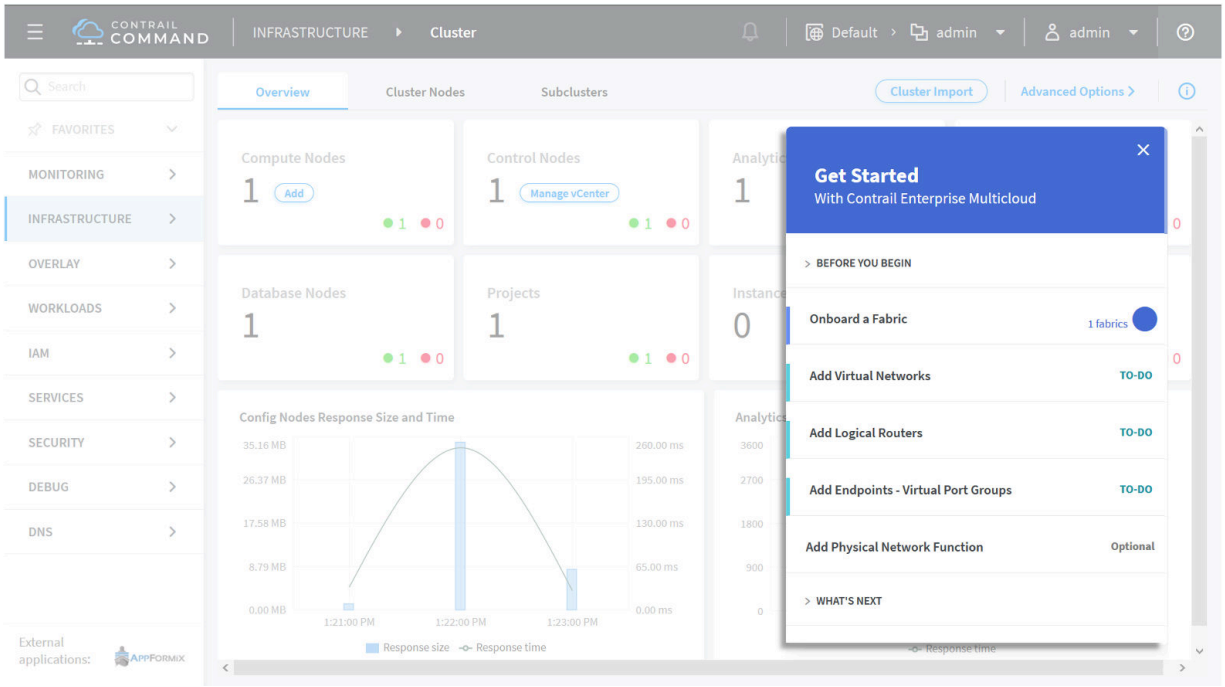
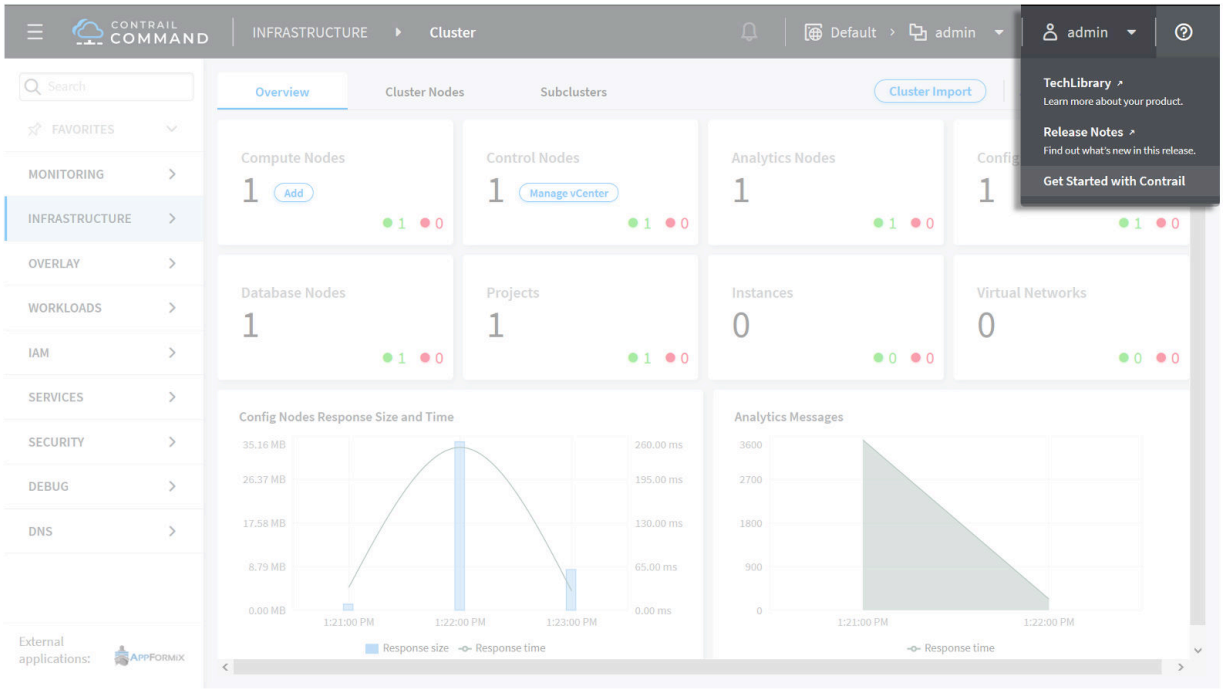


Figure 22 on page 67 shows how to open the *Get Started with Contrail Enterprise Multicloud* panel within Contrail Command.

Figure 22: Get Started with Contrail Option



The *Get Started* panel has the following limitations:

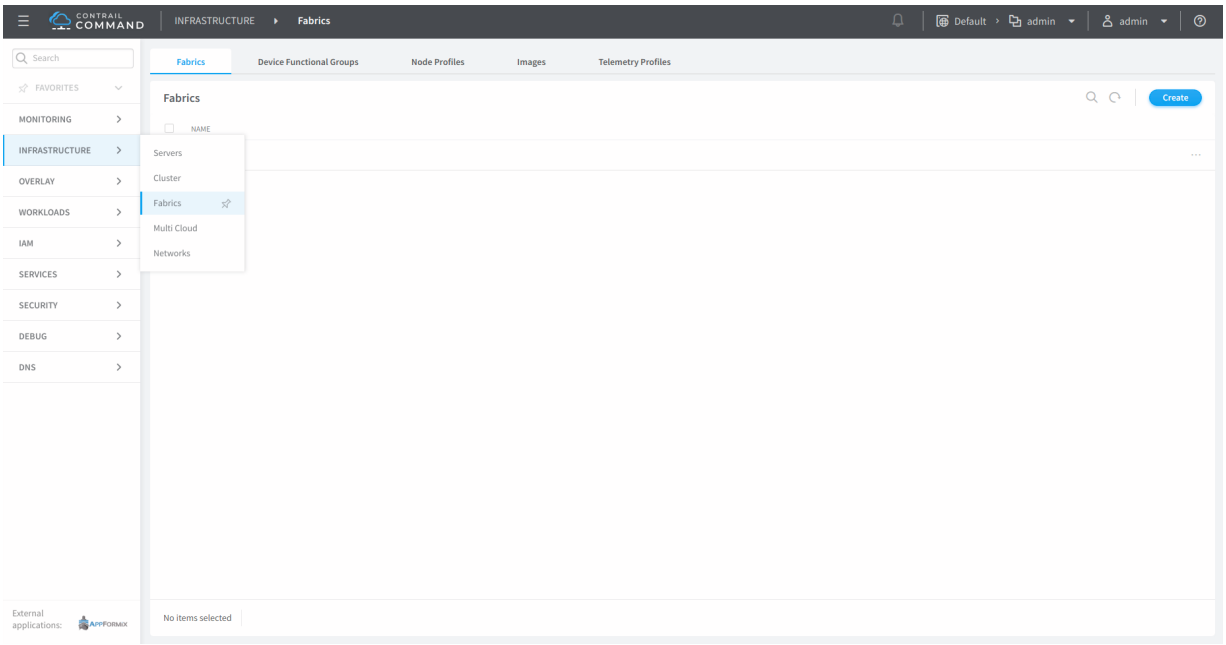
- if you switch between web browsers or login using incognito mode, the task status within the panel is lost.
- If you clear your web browser cache or delete your web browser's cookies, the task status within the panel is lost.
- If you access Contrail Command from a different web browser or login using incognito mode, the panel opens by default even if it had previously been closed.
- If you clear your web browser cache or delete your web browser's cookies, the panel opens by default even if it had previously been closed.

Navigating to pages using the side panel

All menu options are available in a side panel on the left of the UI. The side panel displays the main menu categories available in Contrail Command. Mouse over each category to view the corresponding second tier menu options. These second tier menu options correspond to pages and you can click the menu option to open the page.

The category as well as the page name can be viewed in the bread crumbs visible on the top banner. You can click on page names in the bread crumbs to navigate to the page, but the category name is not clickable.

Figure 23: Contrail Command Side Panel



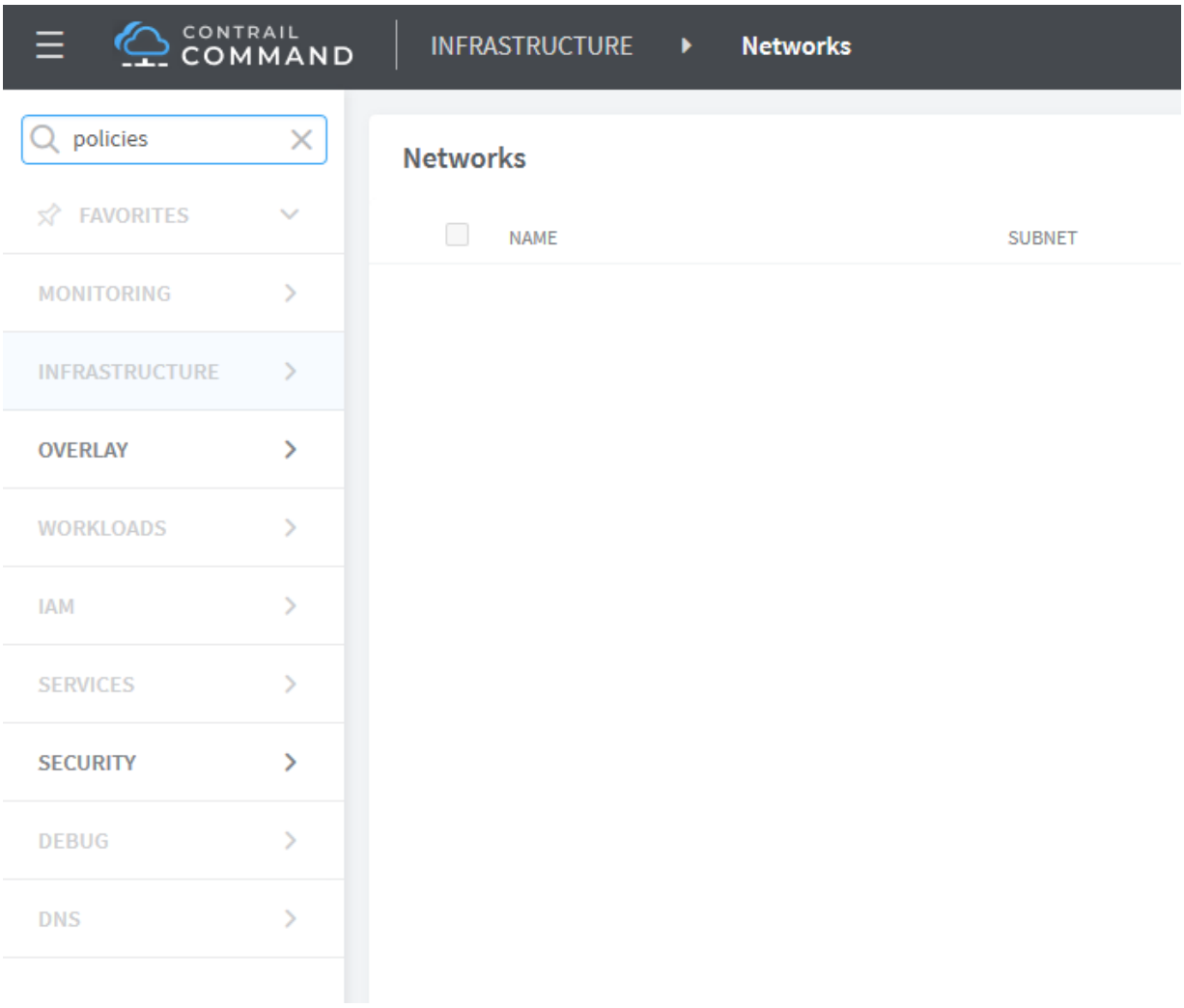
Hiding the side panel

You can also hide the side panel from view. To hide the side panel, click the toggle button (≡) on the top left of the banner on any page in Contrail Command. To view the side panel, click the toggle button (≡) again.

Search functionality

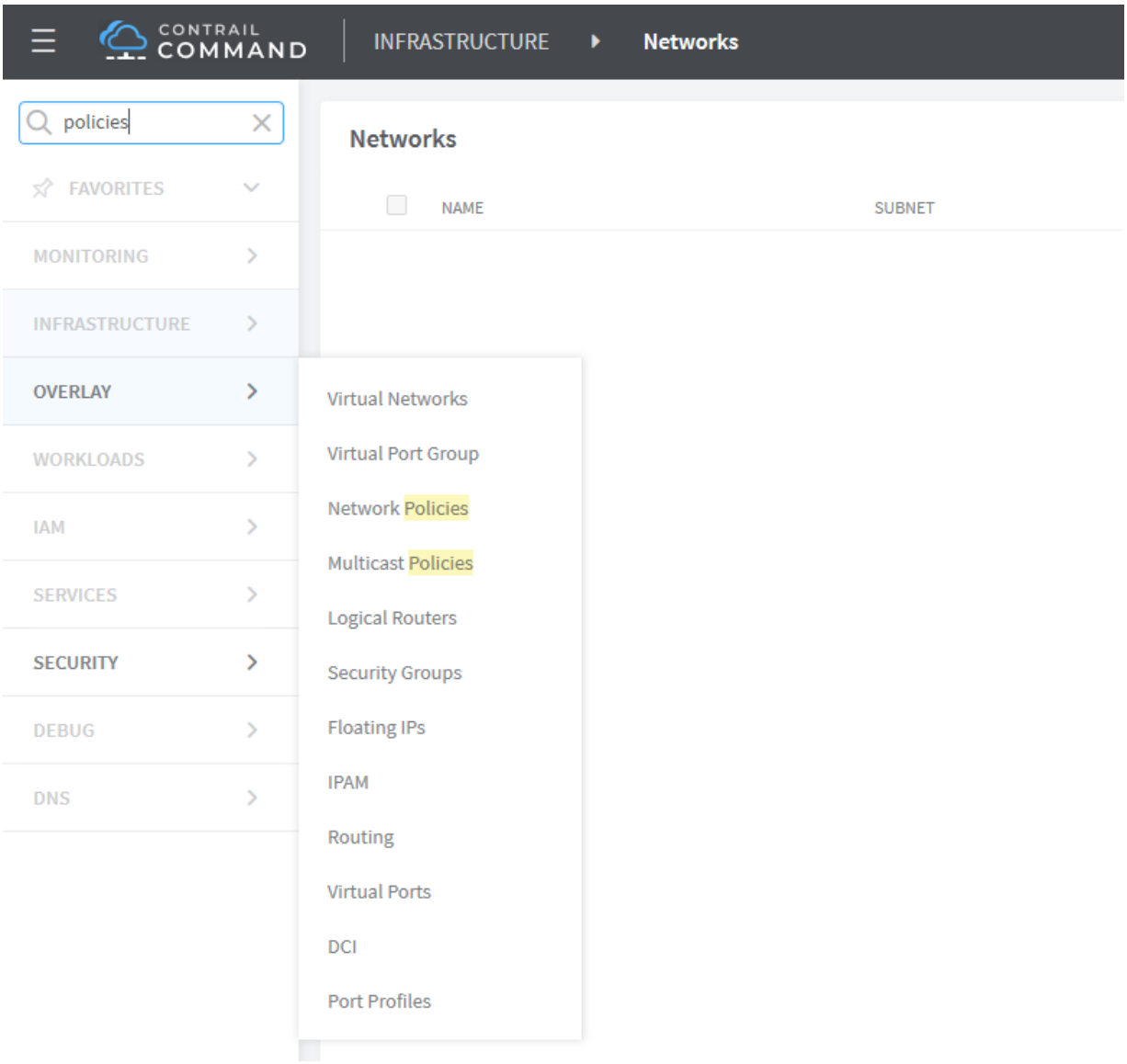
If you are unaware of the navigation path to any page, you can use the search functionality to search for the page. The updated Contrail Command UI has a search text box at the top of the left side panel. When you enter a string in the search text box, all the categories containing matching terms are highlighted in a bold font. All other categories are greyed out.

Figure 24: Contrail Command Search Function–Relevant Categories



Mouse over each highlighted category to view the corresponding second tier menu options. While all the menu options of the category are displayed, the pages matching the search input are highlighted in yellow.

Figure 25: Contrail Command Search Function–Relevant Pages

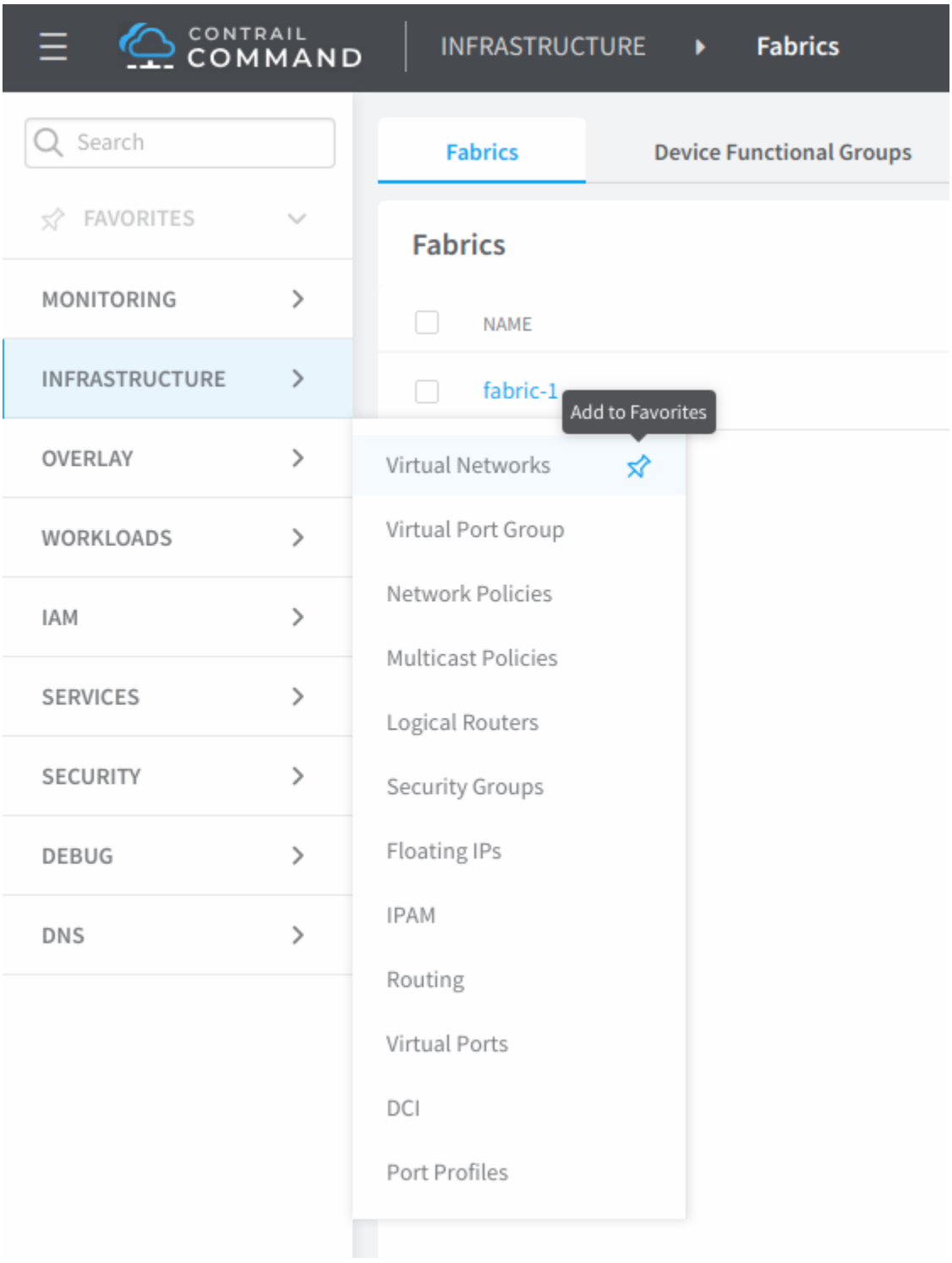


Pinning favorite pages

You can pin frequently visited pages to the favorites category. The left side panel has a **Favorites** category under the search text box and you can add second tier menu options to this category. Initially this category is empty and is grayed out.

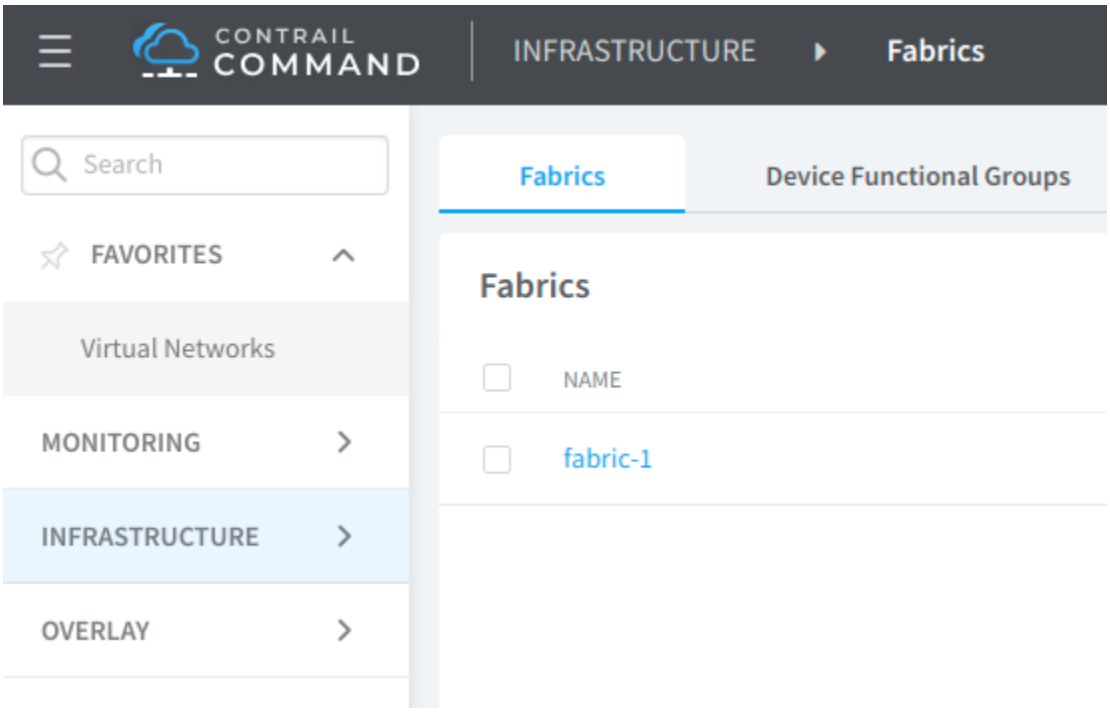
- Adding to **Favorites** – To add a page to favorites, mouse over a category in the side panel to display the second tier menu options. Mouse over the menu option and click the pin icon in-line with the page name to add to favorites.

Figure 26: Contrail Command Favorites Function–Add Page



Once pages are added, the **Favorites** category gets enabled and the pinned page is displayed underneath it.

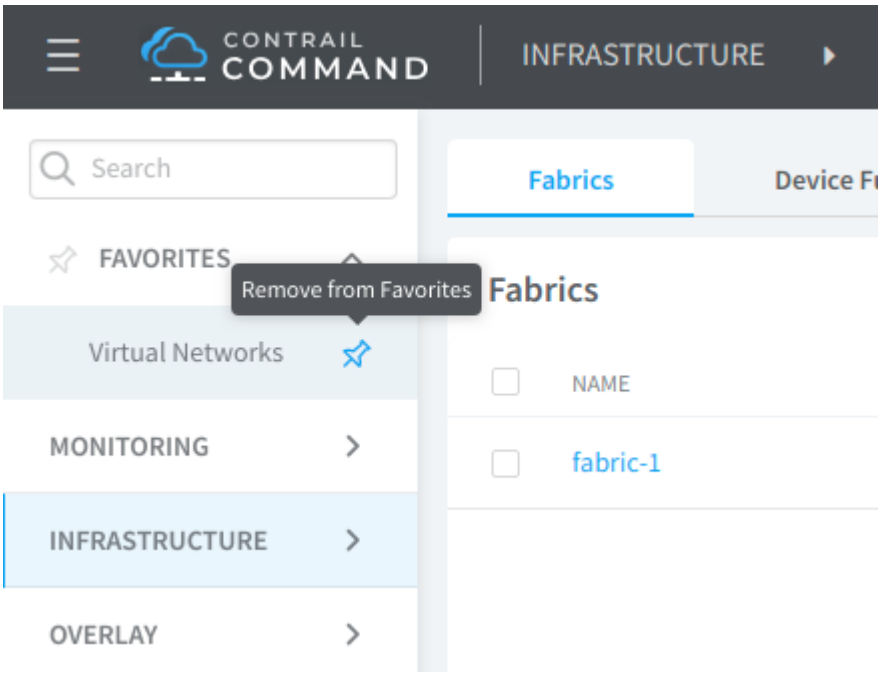
Figure 27: Contrail Command Favorites Function



NOTE: Pinned favorite pages are stored in the Web browser cache in local storage. The existing favorite-page list disappears if you switch between Web browsers, or if you log in under the incognito mode, or if you clear Web browser cache and cookies.

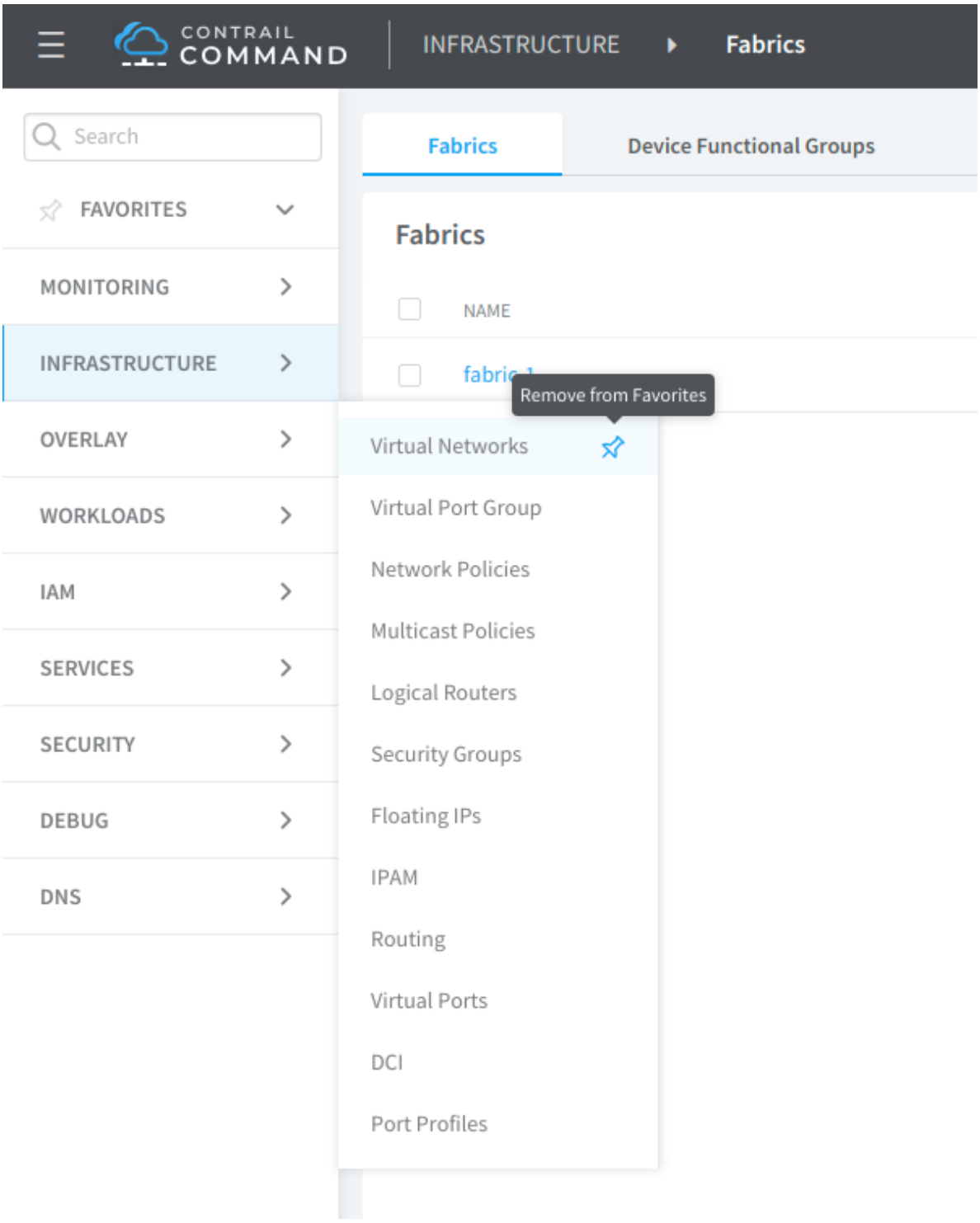
- Deleting from **Favorites** - You can delete pages from the favorites category in any one of the following two ways. You can click the enabled pin icon available in-line with the pinned page in the favorites category to remove the page.

Figure 28: Contrail Command Favorites Function–Remove Page In-line



Alternatively, mouse over the corresponding category of the pinned page to display all the second tier menu options. Mouse over the pinned page and click the enabled pin icon in-line with the page name to remove it from the favorites category.

Figure 29: Contrail Command Favorites Function–Remove Page in Panel



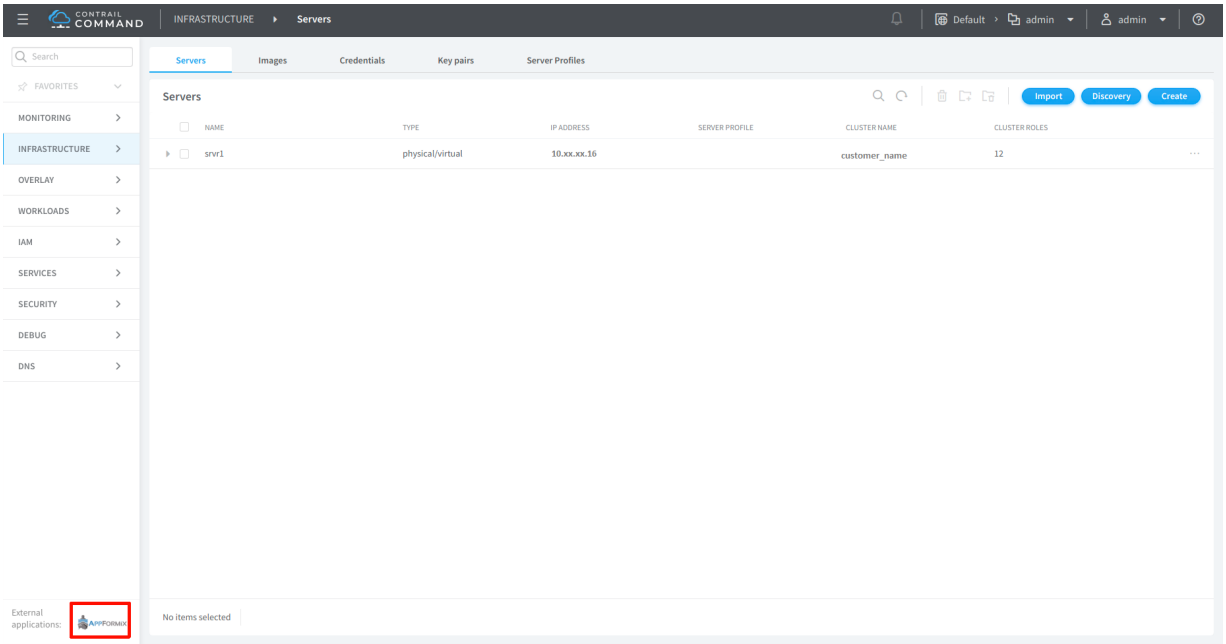
You can also collapse and expand the **Favorites** category by clicking the arrow icon (∧ or ∨) next to the category name.

Opening external applications

To open integrated external applications such as Contrail Insights, click the application name in the footer of the side panel on the left.

NOTE: If no external applications are available, the footer in the side panel is not visible.

Figure 30: Contrail Command Accessing External Applications

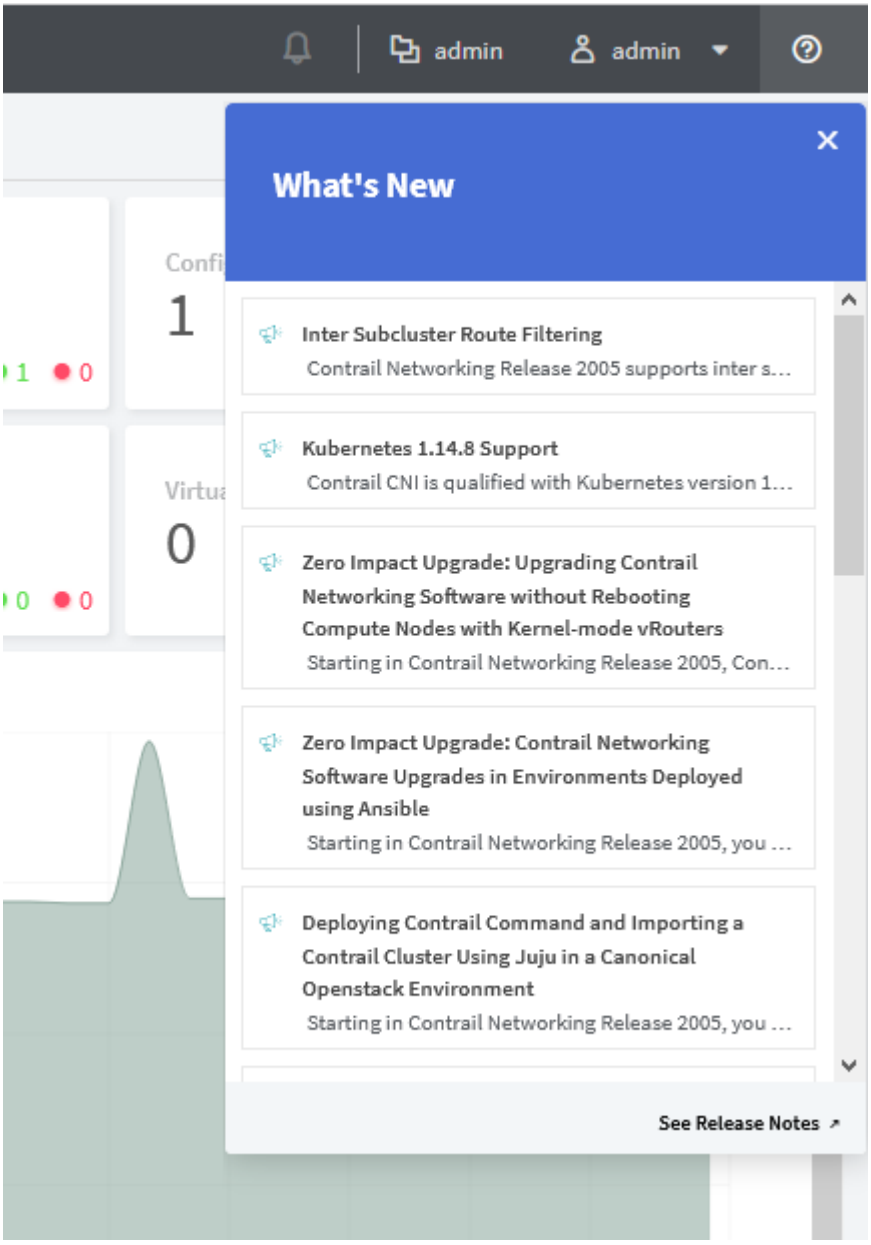


Using the What's New Panel

Starting with Contrail Networking Release 2005, you can use the *What's New* panel within Contrail Command to gather a summary list of the new Contrail Networking features in your Contrail Networking release. The *What's New* panel provides a high-level description of each new feature and a *See Release Notes* option that takes you to the Contrail Networking Release Notes for additional feature information.

You can access the *What's New* panel by selecting the **What's New** option in the **?** help menu.

Figure 31: What's New Panel



Supported Browsers for Installing Contrail Command

Use the following supported browsers to install Contrail Command.

Contrail Networking Release	Supported Browsers and Versions

2008	• Chrome 80, 81, 83, 84 • Firefox 68, 75, 76, 77, 78 • Microsoft Edge 80, 81, 83, 84 • Opera 66, 67, 68, 69 • Safari 12, 12.1, 13, 13.1
2005	• Chrome 78, 79, 80, 81 • Firefox 68, 72, 73, 74, 75 • Microsoft Edge 18, 79, 80, 81 • Opera 65, 66, 67, 68 • Safari 12, 12.1, 13, 13.1
2003	• Chrome 77, 78, 79, 80 • Firefox 68, 71, 72, 73, 74 • Microsoft Edge 17, 18, 79, 80 • Opera 63, 64, 65, 66 • Safari 12, 12.1, 13, 13.1

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
2003	Contrail Networking Release 2003 introduces a redesigned Contrail Command UI.

Installing a Contrail Cluster using Contrail Command and instances.yml

Contrail Networking supports deploying Contrail cluster using Contrail Command and the **instances.yml** file. A YAML file provides a concise format for specifying the instance settings.

We recommend installing Contrail Command and deploying your Contrail cluster from Contrail Command in most Contrail Networking deployments. See "How to Install Contrail Command and Provision Your Contrail Cluster" on page 22. You should only use the procedure in this document if you have a strong reason to not use the recommended procedure.

System Requirements

- A VM or physical server with:
  - 4 vCPUs
  - 32 GB RAM
  - 100 GB disk
- Internet access to and from the physical server, hereafter referred to as the Contrail Command server
- (Recommended) x86 server with CentOS 7.6 as the base OS to install Contrail Command

For a list of supported platforms for all Contrail Networking releases, see Contrail Networking Supported Platforms List.

NOTE: Contrail Release 5.1 does not support Contrail Insights deployment from command line with Contrail Cluster instances.yml file.

Before you begin

`docker-py` Python module is superseded by `docker` Python module. You must remove `docker-py` and `docker` Python packages from all the nodes where you want to install the Contrail Command UI.

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

Configuration

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

1. Install Docker to pull *contrail-command-deployer* container. This package is necessary to automate the deployment of Contrail Command software.

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

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

yum install -y docker-ce-18.03.1.ce

systemctl start docker
```

2. Download the contrail-command-deployer Docker container image from hub.juniper.net. To download these containers and for access to hub.juniper.net, refer to the *Access to Contrail Registry* topic on the Contrail software download page. Allow Docker to connect to the private secure registry.

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

   Pull contrail-command-deployer container from the private secure registry.

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

   Example, for container_tag: 5.1.0-0.38, use the following command:

   ```
   docker pull hub.juniper.net/contrail/contrail-command-deployer:5.1.0-0.38
   ```

3. Edit the input configuration **instances.yml** file. See "No Link Title" on page 80 for a sample **instances.yml** file.

4. Start the contrail_command_deployer container to deploy the Contrail Command (UI) server and provision Contrail Cluster using the **instances.yml** file provided.

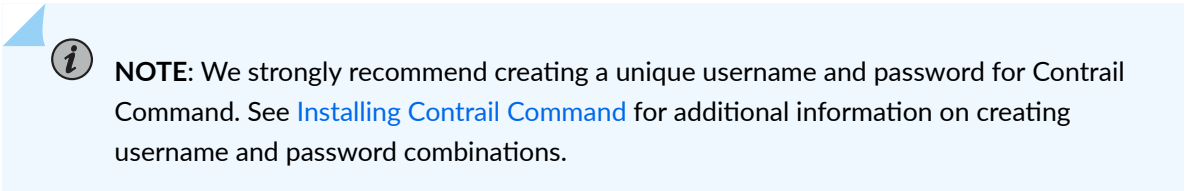
   ```
   docker run -td --net host -e action=provision_cluster -v <ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE>:/
   command_servers.yml -v < ABSOLUTE_PATH_TO_INSTANCES_FILE>:/instances.yml --privileged --name
   contrail_command_deployer hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
  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

   The `contrail_command` and `contrail_psql` Contrail Command containers will be deployed. Contrail Cluster is also provisioned using the given **instances.yml** file.

5. (Optional) Track the progress of 4.

   ```
   docker logs -f contrail_command_deployer
   ```

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



NOTE: We strongly recommend creating a unique username and password for Contrail Command. See Installing Contrail Command for additional information on creating username and password combinations.

NOTE: Enable subscription on all the RedHat nodes.

```
        sudo subscription-manager register --username <USERNAME> --password <PASSWORD>
        sudo subscription-manager attach --pool pool_id

        sudo subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-7-server-rh-
        common-rpms --enable=rhel-ha-for-rhel-7-server-rpms --enable=rhel-7-server-extras-rpms
```

Sample instances.yml File

```
global_configuration:
  CONTAINER_REGISTRY: hub.juniper.net/contrail
  CONTAINER_REGISTRY_USERNAME: < container_registry_username >
  CONTAINER_REGISTRY_PASSWORD: < container_registry_password >
provider_config:
  bms:
    ssh_pwd: <Pwd>
    ssh_user: root
    ntpserver: <NTP Server>
    domainsuffix: local
instances:
  bms1:
    provider: bms
    ip: <BMS IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      vrouter:
      openstack:
      openstack_compute:
  bms2:
    provider: bms
    ip: <BMS2 IP>
    roles:
      openstack:
  bms3:
    provider: bms
```

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

```
    ip: <BMS8 IP>
    roles:
      vrouter:
        # Add following line for TSN Compute Node
        TSN_EVPN_MODE: True
      openstack_compute:
contrail_configuration:
  CLOUD_ORCHESTRATOR: openstack
  CONTRAIL_VERSION: latest or <contrail_container_tag>

  RABBITMQ_NODE_PORT: 5673
  KEYSTONE_AUTH_PUBLIC_PORT: 5005
  VROUTER_GATEWAY: <Gateway IP>
  ENCAP_PRIORITY: VXLAN,MPLSoUDP,MPLSoGRE
  AUTH_MODE: keystone
  KEYSTONE_AUTH_HOST: <Internal VIP>
  KEYSTONE_AUTH_URL_VERSION: /v3
  CONTROLLER_NODES: < list of mgmt. ip of control nodes >
  CONTROL_NODES: <list of control-data ip of control nodes>
  OPENSTACK_VERSION: queens
kolla_config:
  kolla_globals:
    openstack_release: queens
    kolla_internal_vip_address: <Internal VIP>
    kolla_external_vip_address: <External VIP>
    openstack_release: queens
    enable_haproxy: "no"      ("no" by default, set "yes" to enable)
    enable_ironic: "no"       ("no" by default, set "yes" to enable)
    enable_swift: "no"        ("no" by default, set "yes" to enable)
    keystone_public_port: 5005
    swift_disk_partition_size = 10GB
    keepalived_virtual_router_id: <Value between 0-255>
  kolla_passwords:
    keystone_admin_password: <Keystone Admin Password>
```

NOTE: This representative instances.yaml file configures non-default Keystone ports by setting the *keystone_public_port:* and *KEYSTONE_AUTH_PUBLIC_PORT:*.

RELATED DOCUMENTATION

Installing Contrail Command

Installing a Contrail Cluster using Contrail Command

Importing Contrail Cluster Data using Contrail Command | 86

Importing Contrail Cluster Data using RedHat Director

You can use this document to deploy Contrail Command and import an existing cluster into Contrail Command with a single procedure in environments that are using Redhat Director.

If you want to perform this procedure in an environment that is using Contrail Networking but is not using Redhat Director, see "Importing Contrail Cluster Data using Contrail Command" on page 86.

Prerequisites

This document makes the following assumptions about your environment:

- You are running Contrail Networking Release 21.4.L1 or later with RHEL 8.4.
- You have a VM or a BareMetal server available for running Contrail Command, the contrail-command VM.
- Your Contrail Command node is registered with a RedHat subscription and you have a content lock on RHEL-8.4.
- You have installed podman on the machine running Contrail Command.

  You can install podman by entering the **yum install podman** command.
- Contrail Command has access to the RedHat provision network and a Keystone connection.
- Your Contrail Command node is configured with the proper DNS address in the /etc/resolv.conf file.

  Example:

```
[root@command-vm-rcompute tmp]# cat /etc/resolv.conf
search 5b6s1.local
nameserver 192.168.24.252
```

- Your Contrail Command VM has an updated keystone access IP address in the /etc/hosts file.

Example:

```
[root@command-vm-rcompute tmp]# cat /etc/hosts
10.2.0.99 overcloud.5c7.local
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
```

Import the Contrail Cluster

Perform the following steps to import the Contrail Cluster data.

1. Create the command_servers.yml file.

   Example:

```
[root@command-vm-rcompute ~]# cat command_servers.yml
command_servers:
    server1:
        ip: 10.87.86.111
        connection: ssh
        ssh_user: root
        ssh_pass: password
        sudo_pass: password
        ntpserver: 10.84.5.100
        registry_insecure: true
        container_registry: svl-artifactory.juniper.net/contrail-nightly
        container_tag: 21.4.L1.185
        contrail_config:
            database:
              type: postgres
              dialect: postgres
              password: password
            keystone:
                assignment:
                    data:
                      users:
                        admin:
                          password: keystone-password
            insecure: true
            client:
              password: password
```

2. Import the Contrail cluster by entering this command from the Contrail Command node:

```
podman run -td --net host -e orchestrator=tripleo -e undercloud=Undercloud-IP-address  -e
undercloud_user=stack
-e undercloud_password=Undercloud-password  -e action=import_cluster -v /root/
command_servers.yml:/command_servers.yml --privileged --name
contrail_command_deployer Container-Tag
```

You can obtain the *Container-tag* for your version of Contrail Networking at README Access to Contrail Registry 21XX.

If you need to access the Contrail private secure registry, e-mail **contrail-registry@juniper.net** to obtain credentials.

Example:

```
podman run -td --net host -e orchestrator=tripleo -e undercloud=192.168.24.1  -e
undercloud_user=stack -e undercloud_password=contrail
-e action=import_cluster -v /root/command_servers.yml:/command_servers.yml --privileged --
name contrail_command_deployer
enterprise-hub.juniper.net/contrail-container-prod/contrail-command-deployer:21.4.L1.185

[root@command-vm-rcompute ~]# podman images
REPOSITORY
TAG          IMAGE ID      CREATED        SIZE
enterprise-hub.juniper.net/contrail-container-prod/contrail-command-deployer
21.4.L1.185  d500a3307c7f  4 weeks ago    1.13 GB
```

3. Enter the **docker ps** command to confirm that Contrail Command and other Contrail containers are up and running.

```
[root@command-vm-rcompute ~]# docker ps
CONTAINER ID  IMAGE
COMMAND                  CREATED      STATUS     PORTS               NAMES
8ce9ae465361  svl-artifactory.juniper.net/contrail-nightly/contrail-command:21.4.L1.184 "/bin/
commandappserv…"   4 weeks ago  Up 4 weeks contrail_command
6af5bd0bf432  circleci/postgres:10.3-alpine
"docker-entrypoint.s…"   4 weeks ago  Up 4 weeks contrail_psql
```

Importing Contrail Cluster Data using Contrail Command

Contrail Networking supports importing of Contrail Cluster data to Contrail Command provisioned using one of the following applications - OpenStack, Kubernetes, VMware vCenter, and TripleO.

Before you begin

`docker-py` Python module is superseded by `docker` Python module. You must remove `docker-py` and `docker` Python packages from all the nodes where you want to install the Contrail Command UI.

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

System Requirements

- A VM or physical server with:
  - 4 vCPUs
  - 32 GB RAM
  - 100 GB storage
- Internet access to and from the physical server, which is the Contrail Command server.
- (Recommended) x86 server with CentOS 7.6 as the base OS to install Contrail Command.

For a list of supported platforms for all Contrail Networking releases, see Contrail Networking Supported Platforms List.

Access *Container Tags* are located at README Access to Contrail Registry 21XX.

If you need access to Contrail docker private secure registry, e-mail **contrail-registry@juniper.net** for Contrail container registry credentials.

Configuration

Perform the following steps to import Contrail Cluster data.

1. Install Docker to pull *contrail-command-deployer* container. This package is necessary to automate the deployment of Contrail Command software.

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

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

   yum install -y docker-ce-18.03.1.ce

   systemctl start docker
   ```

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

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

   Pull contrail-command-deployer container from the private secure registry.

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

   Example, for container_tag:5.1.0-0.38, use the following command:

   ```
   docker pull hub.juniper.net/contrail/contrail-command-deployer:5.1.0-0.38
   ```

3. Get the **command_servers.yml** file that was used to bring the Contrail Command server up and the configuration file that was used to provision the Contrail Cluster.

   NOTE: "For OpenShift orchestrator use the ose-install file instead of instances.yml file.

4. Start the `contrail-command-deployer` container to deploy the Contrail Command (UI) server and import Contrail Cluster data to Contrail Command (UI) server using the Cluster configuration file provided.

   - Import Contrail-Cluster provisioned using a supported orchestrator (OpenStack/Kubernetes/OpenShift/vCenter/Mesos).

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

     To use the following supported orchestrators, replace `<YOUR_ORCHESTRATOR>` in the command with the options given below.

     - For OpenStack, use `openstack`.
     - For Kubernetes, use `kubernetes`.
     - For Red Hat OpenShift, use `openshift`.

       NOTE: You must use **ose-install** file instead of **instances.yml** file.

     - For VMware vCenter, use `vcenter`.
     - For Mesos, use `mesos`.

   - Import Contrail-Cluster provisioned using OSPDirector/TripleO Life Cycle Manager for RedHat OpenStack Orchestration.

Prerequisites:

- *IP_ADDRESS_OF_UNDERCLOUD_NODE* is an Undercloud node IP that must be reachable from the *contrail-command-deployer* node. You must be able to SSH to Undercloud node from the *contrail-command-deployer* node.
- *External VIP* is an Overcloud VIP where OpenStack and Contrail public endpoints are available. *External VIP* must be reachable from Contrail Command node.
- DNS host name for Overcloud external VIP must be resolvable on Contrail Command node. Add the entry in the **/etc/hosts** file.

```
docker run -td --net host -e orchestrator=tripleo -e action=import_cluster -e
undercloud=<IP_ADDRESS_OF_UNDERCLOUD_NODE> -e undercloud_user=<user-name> -e
undercloud_password=<STACK_USER_PASSWORD_FOR_SSH_TO_UNDERCLOUD> -v <
ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE>:/command_servers.yml --privileged --name contrail_command_deployer
hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
```

NOTE: *undercloud_user* is root by default. Otherwise, the undercloud username is stack in most of the cases.

- Contrail command server must have access to External VIP network to communicate with the configured endpoints.

  Run the following commands:

  ```
  ovs-vsctl add-port br0 vlan<externalNetworkVlanID> tag=<externalNetworkVlanID>  -- set
  interface vlan<externalNetworkVlanID>  type=internal
  ip link set dev vlan<externalNetworkVlanID>  up
  ip addr add <externalNetworkGatewayIP>/<subnetMask> dev vlan<externalNetworkVlanID>
  ```

- If you have used domain name for the external VIP, add the entry in the **/etc/hosts** file.

  Run the following commands:

  ```
  docker exec -it contrail_command bash
  vi /etc/hosts
  <externalVIP> <externalVIP'sDomainName>
  ```

Sample instances.yml file

```
global_configuration:
  CONTAINER_REGISTRY: hub.juniper.net/contrail
  CONTAINER_REGISTRY_USERNAME: < container_registry_username >
  CONTAINER_REGISTRY_PASSWORD: < container_registry_password >
provider_config:
  bms:
    ssh_pwd: <Pwd>
    ssh_user: root
    ntpserver: <NTP Server>
    domainsuffix: local
instances:
  bms1:
    provider: bms
    ip: <BMS1 IP>
    roles:
      openstack:
  bms2:
    provider: bms
    ip: <BMS2 IP>
    roles:
      openstack:
  bms3:
    provider: bms
    ip: <BMS3 IP>
    roles:
      openstack:
  bms4:
    provider: bms
    ip: <BMS4 IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
  bms5:
    provider: bms
    ip: <BMS5 IP>
    roles:
```

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

```
    CONTROL_NODES: <list of control-data ip of control nodes>
    OPENSTACK_VERSION: queens
  kolla_config:
    kolla_globals:
      openstack_release: queens
      kolla_internal_vip_address: <Internal VIP>
      kolla_external_vip_address: <External VIP>
      openstack_release: queens
      enable_haproxy: "no"      ("no" by default, set "yes" to enable)
      enable_ironic: "no"       ("no" by default, set "yes" to enable)
      enable_swift: "no"        ("no" by default, set "yes" to enable)
      keepalived_virtual_router_id: <Value between 0-255>
    kolla_passwords:
      keystone_admin_password: <Keystone Admin Password>
```

RELATED DOCUMENTATION

- Installing Contrail Command
- Installing a Contrail Cluster Using Contrail Command
- Installing a Contrail Cluster using Contrail Command and instances.yml | 78

Adding a New Compute Node to Existing Contrail Cluster Using Contrail Command

You can add or remove a new node from an existing containerized Contrail cluster.

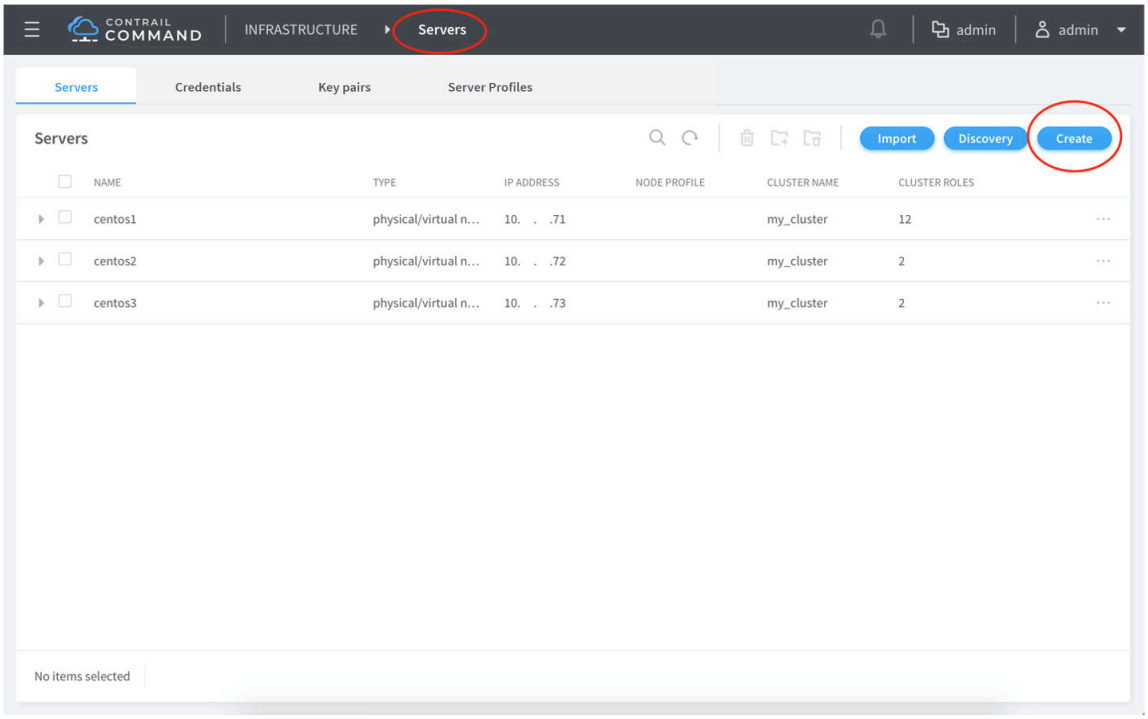
To add a new compute node to an existing Contrail OpenStack cluster:

1. Login to Contrail Command UI as a super user using credentials *admin* for username and *contrail123* for password.

   The default credentials for Contrail Command are *admin* for username and *contrail123* for password. We strongly recommend creating a unique username and password combination. See Installing Contrail Command.

2. Click **Servers**.

   a. Click **Create**.

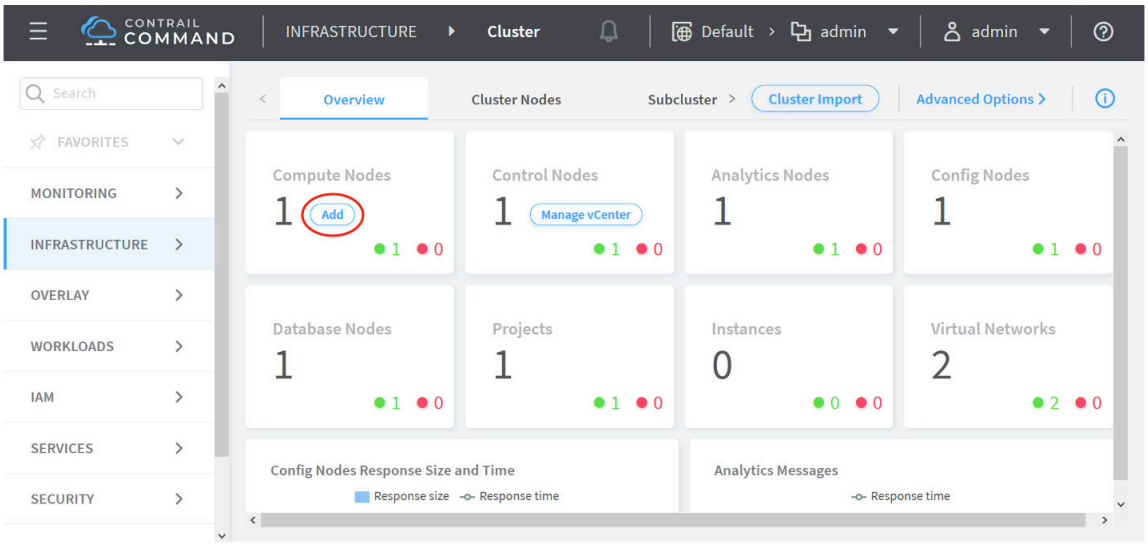


b. Enter the required details.

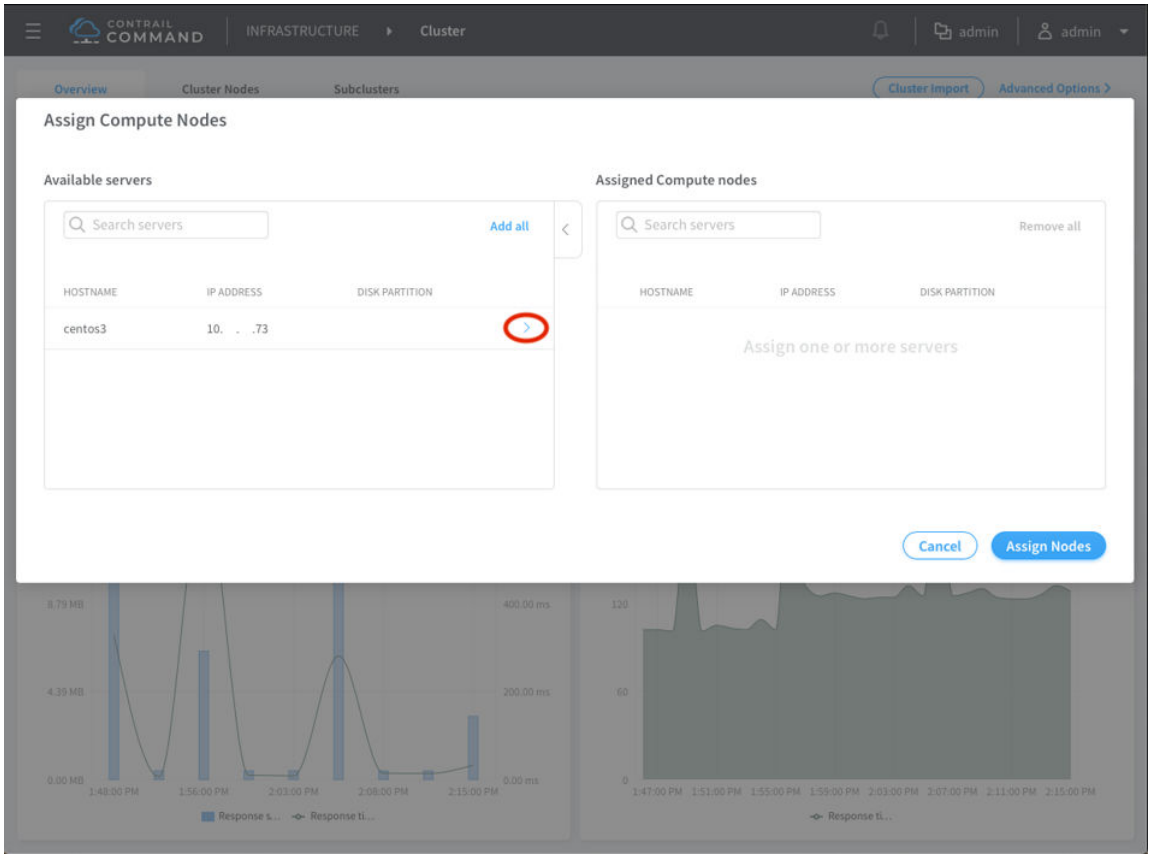
c. Click **Create**.

3. Click **Cluster**.

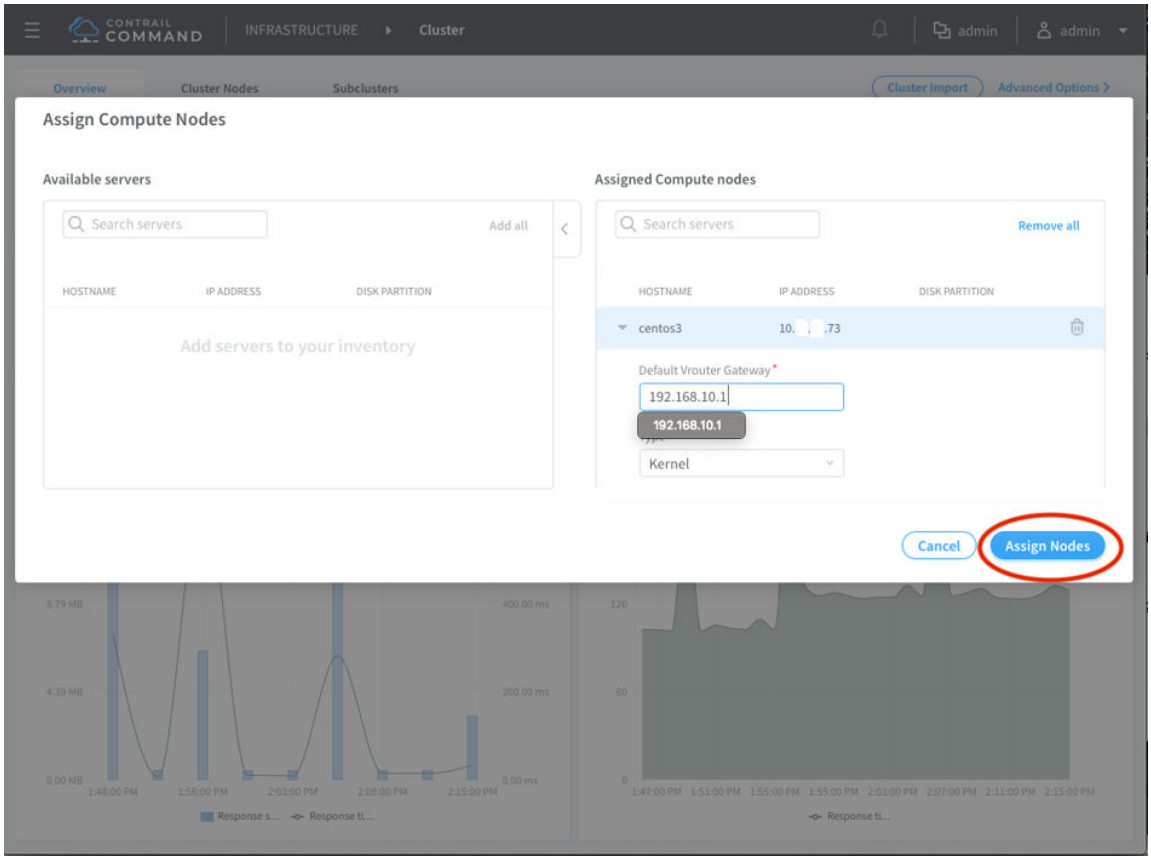
a. Click **Add** under **Compute Nodes**.



b. Select the required server from **Available Servers** list.



c. Click **Assign Nodes**.



Perform the following steps to remove a compute node from an existing Contrail OpenStack cluster.

NOTE: Workloads on the deleted computes must be removed before removing the compute node from the cluster.

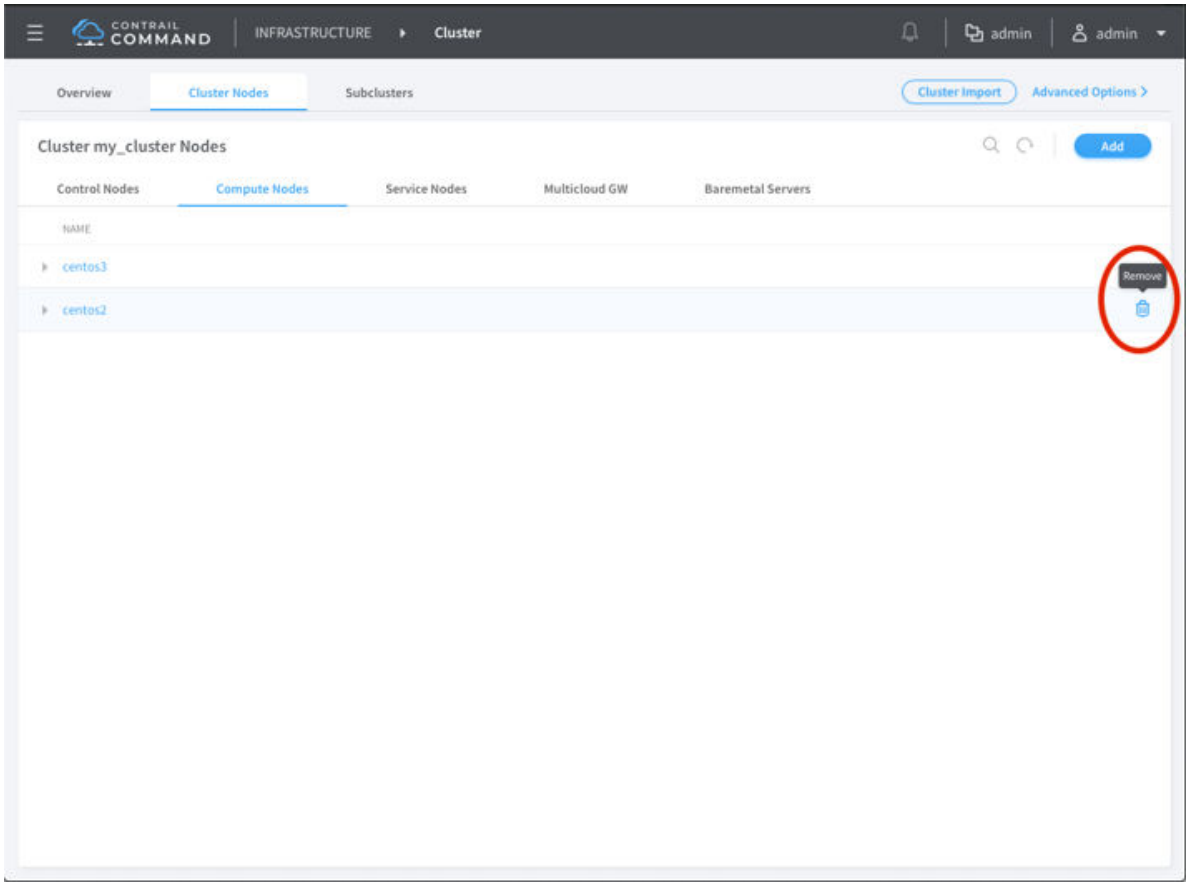
1. Login to Contrail Command UI as a super user using credentials *admin* for username and *contrail123* for password.

   The default credentials for Contrail Command are *admin* for username and *contrail123* for password. We strongly recommend creating a unique username and password combination for security purposes. See Installing Contrail Command.

2. Click **Cluster**.

3. Click **Compute Nodes**.

4. Remove the required compute node.



You can also add a compute node to existing Contrail cluster using **instances.yaml** file. For details, refer to How to Add a New Compute Node to an Existing Contrail Cluster Using the instances.yaml File.

How to Deploy Contrail Command and Import a Cluster Using Juju

IN THIS SECTION

You can use this document to deploy Contrail Command and import an existing cluster into Contrail Command using Juju with a single procedure. This procedure can be applied in environments using Canonical Openstack or environments that are running Juju and using Kubernetes for orchestration.

If you are already running Contrail Command in a Canonical Openstack environment and want to import a cluster, see "Importing a Canonical Openstack Deployment Into Contrail Command" on page 109.

Overview: Deploying Contrail Command with a Contrail Cluster Using Juju

Starting in Contrail Release 2005, you can deploy Contrail Command and import a cluster using Juju in a Canonical Openstack environment.

Starting in Contrail Release 2008, you can deploy Contrail Command and import a cluster using Juju in an environment using Kubernetes orchestration.

This document makes the following assumptions about your initial environment:

- Juju is already running in your environment, and your environment is either a Canonical Openstack deployment or a deployment using Kubernetes orchestration.
- Contrail Networking Release 2005 or later is running if you are operating a Canonical Openstack deployment.

  Contrail Networking Release 2008 or later is running if you are operating an environment using Kubernetes orchestration.

  See Contrail Networking Supported Platforms for information on the supported software components for any Contrail Networking release.
- A Juju controller is configured and reachable.
- Contrail Command is not running.

Preparing the SSL Certificate Authority (CA) for the Deployment

A base64-encoded SSL Certificate Authority (CA) for the Juju controller is required to deploy Contrail Command with an existing cluster in a Canonical Openstack or Kubernetes environment.

There are multiple ways to generate a base64-encoded SSL CA. You can use this procedure or a more familiar procedure to generate your base64-encoded SSL CA.

To create a base64-encoded SSL CA:

1. From the Juju jumphost, enter the **juju show-controller** command and locate the certificate output in the **ca-cert:** hierarchy.

```
$ juju show-controller
  jc5-cloud:
      details:
          ...<output removed for readability>...
          ca-cert: |
          -----BEGIN CERTIFICATE-----
           MIIErTCCAxWgAwIBAgIVAKRPIub8Q7imJ2+T2U8AK4thOss7MA0GCSqGSIb3DQEB
           CwUAMG4xDTALBgNVBAoTBGp1anUxLjAsBgNVBAMMJWp1anUtZ2VuZXJhdGVkIENB
           IGZvciBtb2RlbCAianVqdS1jYSIxLTArBgNVBAUTJDI0ZDJjODg0LTllYWYtNDU2
           Ni04NTA0LWJkZGYxZWJiYTgzYjAeFw0yMDA0MTUwMzE2MzdaFw0zMDA0MjIwMzE2
           MzVaMG4xDTALBgNVBAoTBGp1anUxLjAsBgNVBAMMJWp1anUtZ2VuZXJhdGVkIENB
           IGZvciBtb2RlbCAianVqdS1jYSIxLTArBgNVBAUTJDI0ZDJjODg0LTllYWYtNDU2
           Ni04NTA0LWJkZGYxZWJiYTgzYjCCAaIwDQYJKoZIhvcNAQEBBQADggGPADCCAYoC
           ggGBAL/7d3JtNcHW6ue6yOeKvOlSDhxgGs4vYLDO0QzlIMyW39+BytB4XY+05EBg
           A5JKfYV+u8xXL0meLvh+4yE87cwROObsT1WYFCDFVTiGSeSN3w+2UJxHWwuAubDl7
           zfAKnGgIzq/KZJJimxa6Yuqw5isCxffu3fQz+H5UlSpLCpFxvAq38VjrW7FnjEm1
           c4fFlBf07LUOqBxSIS0gxarO1DQE2IQv4mfIAFvJgT/5UKJYuGEX3NH9DerYqjJa
           NchyGMkXgyBj3YVec8bFE4+erDMISBvJHBMwyx74PTDQys+KlfNXptup5FH/FwBb
           9ZRBAD99c0f0VW6moNxoAkKhrGVZt1w7CxwvgRZnWUezthwoHI8yFqBvkT+lq6Nd
           jvLEv1DQ+3zmMfhz/emRD1DOQQfn3mQhSk40NdO3kw/B8bHOIXmgIgNbv48g0Ac7
           /hQO02moDxrLkCZNN0fVgOKvonDjbSo5YNCH/7fleacmQN3Mug3wXp9kYh7rKDHw
           6pkQQwIDAQABo0IwQDAOBgNVHQ8BAf8EBAMCAqQwDwYDVR0TAQH/BAUwAwEB/zAd
           BgNVHQ4EFgQUcGE6bMiGsQQyiDYKBl+txAfeFAkwDQYJKoZIhvcNAQELBQADggGB
           AG7pivQhJVNSCbG+9jVy3owhg/POnp2sewD1t8BMOkPTPgAa/37vrp4KSPdNXKZz
           hnFzBXkL8jBUP0qg2Vfy9iqlgXNdVAdb4Ijk44OhlwWNGUiZwl2nNbvnUL7NnTeh
           jqZaIb6Oe2y1ByNrQweVMO85qdrYJCelf9Wh9fYdtofx4TyOMg+ZqPqmvTRO8yTx
           KOupywxmezbjhEaaILXo9kouU4UV2gAIdYiHfvsbTaLkWbYeNgvvE5WAan8HuQqb
           YVnvxggIN45UgEgqGUHEgcj9tHgssfbnX3f2sCbOJkXL2cv7D+wK7hvUCS5tKS6H
           6O7OoXxfimFBdSZQuuqhqyiMYafnRo48Q2oCyQn1Q+g/qG+GYxmujIigoiYS1srV
           mIUaJQUGHtgXvyZGJFIvQiAzImQCylq1iyz77Da3myDRX0i0dauu5MACn5i9cgu9
           W7/MD2xR3kKMAY3b4y+pP7CKbEJ6UDswLyAQUkwPyeLi1r82vGh6CasinnGaUhk+
           zg==
           -----END CERTIFICATE-----
          ...<additional output removed for readability>...
```

2. Copy the contents of the SSL CA into the cert.pem file.

   Copy and paste options vary by user interface. The SSL CA content—all highlighted text from step 1 starting at the beginning of the -----**BEGIN CERTIFICATE**----- line and ending at the end of the -----**END CERTIFICATE**----- line—should be the only content in the cert.pem file.

Confirm that leading white spaces are not added to the SSL CA after copying the SSL CA into the cert.pem file. These leading white spaces are introduced by some user interfaces—often at the start of new lines—and will cause the SSL CA certification to be unusable. If leading whitespaces are added to the SSL CA after it is copied into the cert.pem file, manually delete the whitespaces before proceeding to the next step.

3. Generate the cert.pem file into base64-encoded output.

   You can generate the cert.pem file into base64-encoded output without saving the file contents by entering the following command:

   ```
   cat cert.pem | base64
   ```

   You can also generate the base-64 encoded output and save the SSL CA contents into a separate file.

   In this example, the base64-encoded output is generated and a new file containing the output—**cert.pem.b64**—is saved.

   ```
   cat cert.pem | base64 > "cert.pem.b64"
   ```

   The SSL CA in the **cert.pem.b64** file is now a base64-encoded SSL CA.

   The base64-encoded SSL CA will be entered as the *juju-CA-certificate* variable in "Deploy Contrail Command and Import a Contrail Cluster Using Juju" on page 98.

Deploy Contrail Command and Import a Contrail Cluster Using Juju

To deploy Contrail Command and import a Contrail cluster into Contrail Command:

1. From the Juju jumphost, deploy Contrail Command using one of the following command strings:

   ```
   juju deploy cs:~juniper-os-software/contrail-command --constraints tags=<machine-tag> --
   config docker-registry=<registry-directory> --config image-tag=<image-tag>
   ```

   ```
   juju deploy cs:~juniper-os-software/contrail-command --to <machine-name> --config docker-
   registry=<registry-directory> --config image-tag=<image-tag>
   ```

   where:

   - *machine-name*—the name of the machine instance in Juju that will host Contrail Command.

     The IP address of this machine—which can be obtained by entering the **juju status** command—is used to access Contrail Command from a web browser after the installation is complete.

- *registry-directory*—the directory path to the Contrail Networking registry.

  This *registry-directory* path can be obtained from Juniper Networks. Contact mailto:contrail-registry@juniper.net for information on accessing the Juniper registry.

- *image-tag*—the image tag for your target Contrail release.

  The image tag is used to identify your Contrail Networking image within the registry. You can retrieve the image tag for any Contrail Release 21xx image from README Access to Contrail Registry 21XX.

2. Create a juju relation between the Contrail Command charm and the Contrail Controller charm:

```
juju add-relation contrail-command contrail-controller
```

3. Import the Contrail cluster into Contrail command:

   a. Create a **config.yaml** file with the following parameters:

```
$ cat config.yaml
juju-controller: juju-controller-ip
juju-controller-password: password
juju-ca-cert: |
    juju-CA-certificate
juju-model-id: juju-model-id
juju-controller-user: juju-controller-user
```

   The command variables:

   - *juju-controller-ip*—The IP address of the Juju controller.

     You can retrieve the *juju-controller-ip* from the **juju show-controller** command output:

```
username@contrail-ci:~$ juju show-controller
  jc5-cloud:
      details:
          ...<output removed for readability>...
          api-endpoints: [10.102.72.40:17070]
          ...<output removed for readability>...
```

   - *password*—The password for Juju controller access.

     You can set the password for Juju controller access using the **juju change-user-password** command.

- *juju-CA-certificate*—The base64-encoded SSL Certificate Authority (CA) for the Juju controller.

  The *juju-CA-certificate* is the base64-encoded SSL CA created in "Preparing the SSL Certificate Authority (CA) for the Deployment" on page 96.

  See "Example: Config.YML File for Deploying Contrail Command with a Cluster Using Juju" on page 102 for a sample *juju-CA-certificate* entry.

- *juju-model-id*—The universally unique identifier (UUID) assigned to the model environment that includes the Contrail Networking cluster..

  You can retrieve the *juju-model-id* from the **juju show-controller** command output:

```
$ juju show-controller
jc5-cloud:
      ...<output removed for readability>...
      models:
          default:
          model-uuid: 4a62e0b0-bcfe-4b35-8db7-48e55f439217
      ...<output removed for readability>...
```

- *juju_controller_user*—(Optional) The username of the user with Juju controller access.

  The **admin** username is used by default if no user with Juju controller access is configured.

See "Example: Config.YML File for Deploying Contrail Command with a Cluster Using Juju" on page 102 for a sample **config.yaml** configuration for this deployment.

b. Save the **config.yaml** file.

c. Import the Contrail cluster with the parameters defined in the **config.yaml** file:

```
juju run-action contrail-command/0 import-cluster --params config.yaml
Action queued with id: 1
```

d. Check the cluster import status.

   You can check the import status by entering the **juju show-action-status *action-ID*** and **juju show-action-output *action-ID* | grep result** commands.

   The *action-ID* is assigned immediately after entering the **juju run-action** command in the previous step.

The cluster import is complete when the **status** field output in the **juju show-action-status** *action-ID* command shows **completed**, or when the **result** field in the **juju show-action-output** *action-ID* **| grep result** indicates **Success**.

Examples:

```
juju show-action-status 1
 actions:
 - action: import-cluster
 completed at: "2020-04-03 12:49:55"
 id: "60"
 status: completed
 unit: contrail-command/19
```

```
juju show-action-output 1 | grep result
 results:
     result: Success
```

4. Login to Contrail Command by opening a web browser and entering `https://<juju-machine-ip-address>:<port-number>` as the URL.

   The *<juju-machine-ip-address>* is the IP address of the machine hosting Contrail command that was specified in 1. You can retrieve the IP address using the **juju status** command:

   NOTE: Some **juju status** output removed for readability.

```
juju status
Unit                 Workload  Agent  Machine Public address
contrail-command/0*  active    idle   3       10.0.12.40
```

   The *port-number* typically defaults to 9091 or 8079. You can, however, configure a unique port number for your environment using the **command_servers.yml** file.

   Enter the following values after the Contrail Command homescreen appears:

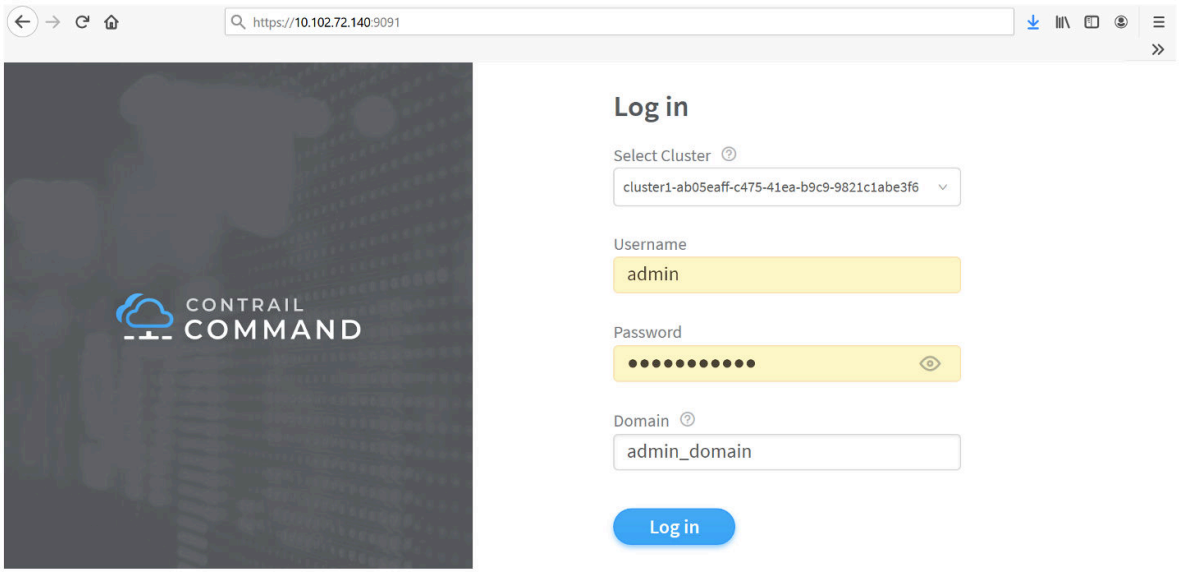
   - **Select Cluster**: Select a Contrail Cluster from the dropdown menu. The cluster is presented in the *<cluster-name>-<string>* format.
   - **Username**: Enter the username of the Juju keystone user.
   - **Password**: Enter the password of the Juju keystone user.

- **Domain**: If you are running Juju in a Canonical Openstack environment, enter **admin_domain**—the default domain name for Canonical Openstack— if you haven't established a unique domain in Canonical Openstack. Enter the name of your domain if you have created a unique domain.

  If you are running Juju in a Kubernetes environment, you can leave this field blank unless you've established a unique domain name in Kubernetes. Enter the name of your domain if you have created a unique domain.

Figure 32 on page 102 illustrates an example Contrail Command login to complete this procedure.

Figure 32: Contrail Command Login Example—Cluster in Environment using Canonical Openstack



See "How to Login to Contrail Command" on page 62 for additional information on logging into Contrail Command.

Example: Config.YML File for Deploying Contrail Command with a Cluster Using Juju

This sample **config.yml** file provides a representative example of a configuration that could be used to deploy Contrail Command with Contrail clusters in an environment running Juju.

See "Deploy Contrail Command and Import a Contrail Cluster Using Juju" on page 98 for step-by-step procedures to create this **config.yml** file and "Preparing the SSL Certificate Authority (CA) for the Deployment" on page 96 for instructions on generating the *juju-ca-cert* in the required base64-encoded format.

This sample **config.yml** file does not contain the **juju-controller-user:** field to specify a user with Juju controller access, so the default **admin** username is used.

CAUTION: The password **password** is used in this example for illustrative purposes only. We strongly recommend creating a unique password that meets your organization's security requirements for your environment.

```
$ cat config.yaml
  juju-controller: 10.102.72.40
  juju-ca-cert: |
    LS0tLS9CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUVyVENDQXhXZ0F3SUJBZ0lWQUtSUEl1YjhR
    N2ltSjIrVDJVOEFLNHRoT3NzN01BMEdDU3FHU0liM0RRRUIKQ3dVQU1HNHhEVEFMQmdOVkJBb1RC
    R3AxYW5VeExqQXNCZ05WQkFNTUpXcDFhblV0WjJWdVpYSmhkR1ZrSUVOQgpJR1p2Y2lCdGIyUmxi
    Q0FpYW5WcWRTMWpZU0l7TFRBckJnTlZCQVVUSkRJMFpESmpPRGcwTFRsbFlXWXRORFUyCk5pMDRO
    VEEwTFdKa1pHWXhaV0ppWVRnellqQWVGdzB5TURBME1UVXdNekUyTXpkYUZ3MHpNREEwTWpJd016
    RTIKTXpWYU1HNHhEVEFMQmdOVkJBb1RCR3AxYW5VeExqQXNCZ05WQkFNTUpXcDFhblV0WjJWdVpY
    SmhkR1ZrSUVOQgrJR1p2Y2lCdGIyUmxiQ0FpYW5WcWRTMWpZU0l4TFRBckJnTlZCQVVUSkRJMFpE
    SmpPRGcwTFRsaFlXWXRORFUyCk5pMDROVEEwTFdKa1pHWXhaV0ppWVRnellqQ0NBYUl3RFFZSktv
    WklodmNOQVFFQkJRQURnZ0dQQURDQ0FZb0MKZ2dHQkFMLzdkM0p0TmNIVzZ1ZTZ5T2VLdk9sU0Ro
    eGdHczR2WUxETzBRemxJTXlXMzkrQnl0QjRYWSswNUVCZwpBNUpLZllWK3U4eFhMMG1lTHZoKzR5
    RTg3Y3dST2JwVDFXWUZDREZWVGlHU2VTTjN3KzJVSnhIV3d1QXViRGw3CnpmQUtuR2dJenEvS1pK
    SmlteGE2WXVxdzVpc0N4ZmZ1M2ZReitINVVsU3BMQ3BGeHZBcTM4VmpyVzdGbmpFbTEKYzRmRmxC
    ZjA3TFVPcUJ4U0lTMGd4YXJPMURRRTJJUXY0bWZJQUZ2SmdULzVVS0pZdUdFWDNOSDlEZXJZcWpK
    YQpOY2h5R01rWGd5QmozWVZlYzhiRkU0K2VyRE1JU0J2SkhCTXd5eDc0UFREUXlzK0tsZk5YcHR1
    cDVGSC9Gd0JiCjlaUkJBRDk5YzBmMFZXNm1vTnhvQWtLaHJHVlp0MXc3Q3h3dmdSWm5XVWV6dGh3
    b0hJOHlGcUJ2a1QrbHE2TmQKanZMRXYxRFErM3ptTWZoei9lbVJEMURPUVFmbjNtUWhTazQwTmRA
    M2t3L0I4YkhPSVhtZ0lnTmJ2NDhnMEFjNwovaFFPMDJtb0R4ckxrQ1pOTjBmVmdPS3ZvbkRqYlNv
    NVlOQ0gvN2ZsZWFjbVFOM011ZzN3WHA5a1loN3JLREh3CjZwa1FRd0lEQVFBQm8wSXdRREFPQmdO
    VkhROEJBZjhFQkFNQ0FxUXdEd1lEVlIwVEFRSC9CQVV3QXdFQi96QWQKQmdOVkhRNEVGZ1FVY0dF
    NmJNaUdzUVF5aURZS0JpK3R4QWZlRkFrd0RRWUpLb1pJaHZjTkFRUxCUUFEZ2dHQgpBRzdwaXZR
    aEpWTlNDYkcrOWpWeTNad2hnL1BPbnAyc2V3RDF0OEJNT2tQVFBnQWEvMzd2cnA0S1NQZE5YS1p6
    CmhuRnpCWGtMOGpCVVAwcWcyVmZ5OWlxbGdYTmRWQWRiNElqazQ0T2hsd1dOR1VpWndsMm5OYnZu
    VUw3Tm5UZWgKanFaYUliNk9lMnkxQnlOclF3ZVZNTzg1cWRyWUpDZWxmOVdoOWZZZHRvZng0VHlP
    TWcrWnFQcW12VFJPOHlUeApLT3VweXd4bWV6YmpoRWFhSUxYbzlrb3VVNFVWMmdBSWRZaUhmdnNi
    VGFMa1diWWVOZ3Z2RTVXQWFuOEh1UXFiCllWbnZ4Z2dJTjQ1VWdFZ3FHVUhFZ2NqOXRIZ3NzZmJu
    WDNmMnNDYk9Ka1hMMmN2N0Qrd0s3aHZVQ1M1dEtTNkgKNk83T29YeGZpbUZCZFNaUXV1cWhxeWlN
    WWFmblJvNDhRMm9DeRFuMVErZy9xRytHWXhtdWpJaWdvaVlTMXNyVgptSVVhSlFVR0h0Z1h2eVpH
    SkZJdlFpQXpJbVFDeWxxMWl5ejc3RGEzbXlEUlgwaTBkYXV1NU1BQ241aTljZ3U5Clc3L01EMnhS
    M2tLTUFZM2I0eStwUDdDS2JFSjZVRHN3THlBUVVrd1B5ZUxpMXI4MnZHaDZDYXNpbm5HYVVoaysK
    eac9PQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
juju-model-id: 4a62e0b0-bcfe-4b35-8db7-48e55f439217
juju-controller-password: password
```

Prerequisites for Contrail Insights and Contrail Insights Flow

Contrail Networking Release 2011 supports installing Contrail Insights and Contrail Insights Flows on a Juju cluster after Contrail Networking and Contrail Command are installed. The following prerequisites apply.

`docker`, `python2.7`, `python-pip` must be installed on the Contrail Insights node and Contrail Insights Flows node.

To install the Docker engine, you need the 64-bit version of one of these Ubuntu versions:

- Ubuntu Groovy 20.10
- Ubuntu Focal 20.04 (LTS)
- Ubuntu Bionic 18.04 (LTS)
- Ubuntu Xenial 16.04 (LTS)

Docker Engine is supported on x86_64 (or amd64), armhf, and arm64 architectures. For more information, see https://docs.docker.com/engine/install/ubuntu/.

To install `python 2.7` and `python-pip` run the following commands:

```
sudo apt install python2.7
sudo apt install python-pip
```

If you are running the playbooks as `root` user then this step can be skipped. As a non-root user (for example, “ubuntu”), the user “ubuntu” needs access to the docker user group. The following command adds the user to the docker group:

```
sudo usermod -aG docker ubuntu
```

For more information, see *Contrail Insights Installation for OpenStack in HA*.

Contrail Insights Installation for Ubuntu Focal

Contrail Insights Release 3.3.5 supports Ubuntu 20.04 (Focal).

Software Requirements

- docker-ce : `5:19.03.9~3-0~ubuntu-focal`

NOTE: Python 2 is not installed by default with Ubuntu 20.04 (Focal).

Follow these steps before you install Contrail Insights.

1. Install `python` and `python-pip` on the Contrail Insights Controller nodes, and on the host(s) that the Contrail Insights Agent runs on.

```
sudo apt-get install -y python python3-pip
sudo apt install python-is-python3
```

2. In `group_vars/all`, set `appformix_ansible_python3_interpreter_enabled` to true.

```
appformix_ansible_python3_interpreter_enabled: true
```

3. Run the `iptables` rule to access port 9000.

```
iptables -t filter -A IN_public_allow -p tcp --dport 9000 -j ACCEPT
```

NOTE: Ignore any errors that may arise if `IN_public_allow` does not exist.

After you have completed these steps, you can install Contrail Insights.

Install Contrail Insights on the Juju Cluster after Contrail Command is Installed

NOTE: Appformix and Appformix Flows were renamed Contrail Insights and Contrail Insights Flows. The Appformix naming conventions still appear during product usage, including within these directory names.

To install Contrail Insights on the Juju Cluster:

1. Copy the Contrail Insights and Contrail Insights Flows installation directories to the `/opt/software/appformix/` and `/opt/software/xflow` directories inside the Contrail Command container, if not already present.

```
docker run -v /opt/software/appformix:/opt/software/appformix svl-artifactory.juniper.net/
contrail-nightly/appformix/appformix/contrail-insights-ansible:<Contrail Insights Version>

docker run -v /opt/software/flow:/opt/software/flow svl-artifactory.juniper.net/contrail-
nightly/appformix/flows/contrail-insights-flows-ansible:<Contrail Insights Flow Version>
```

For example `<Contrail Insights Version>` = `3.3.0-a8`.

2. Create the following two inventory files:

```
docker exec -it contrail_command bash
vi /opt/software/appformix/inventory/group_vars/all
vi /opt/software/appformix/inventory/hosts
```

3. Run the following commands to install Contrail Insights in HA mode:

```
cd /usr/share/contrail/appformix-ansible-deployer/appformix/
. venv/bin/activate

cd /opt/software/appformix/
ansible-playbook -i inventory --skip-tags=install_docker contrail-insights-ansible/
appformix_openstack_ha.yml -v
```

Install Contrail Insights Flows on the Juju Cluster after Contrail Insights is Installed

Disclaimer: Official installation method for installation is using the Contrail-Command UI. `contrail-ansible-deployer` installs all packages needed for Contrail Insights and Contrail Insights Flows. `appformix-ansible-deployer` creates inventory files for the installation. There are many variables set in the inventory files for specific releases, so setting them manually is prone to errors.

To install Contrail Insights Flows on the Juju Cluster:

1. Log in to the `contrail-command` container:

```
docker exec -it contrail_command bash
```

2. Run the following two commands:

```
cd /usr/share/contrail/appformix-ansible-deployer/xflow
source venv/bin/activate
```

3. Run one of the following commands dependent on your Contrail Networking Release version.

   If you are running a Contrail Networking Release later than 2005:

```
bash deploy_contrail_insights_flows.sh <path-to-instances-yml>/instances.yml --cluster-id
<cluster_id>
```

If you are running a Contrail Networking Release earlier than 2005:

```
bash deploy_xflow.sh <path-to-instances-yml>/instances.yml
```

If you are running a Contrail Networking Release earlier than 2005, add the following snippet to the end of the existing `instances.yml` before running the `deploy_contrail_insights_flows.sh` or `deploy_xflow.sh`.

Example `instances.yml` snippet for in-band configuration:

```
global_configuration:
  CONTAINER_REGISTRY: hub.juniper.net/contrail
  CONTAINER_REGISTRY_USERNAME: < container_registry_username >
  CONTAINER_REGISTRY_PASSWORD: < container_registry_password >
provider_config:
  bms:
    ssh_pwd: <Root Pwd>
    ssh_user: root
    ntpserver: <NTP Server>
    domainsuffix: local
instances:  < under existing hierarchy >
  a7s33:
    ip: 10.84.30.201
    provider: bms
    roles:
      appformix_flows:
        telemetry_in_band_interface_name: enp4s0f0
xflow_configuration:
  clickhouse_retention_period_secs: 7200
  loadbalancer_collector_vip: 30.1.1.3
  telemetry_in_band_cidr: 30.1.1.0/24
  loadbalancer_management_vip: 10.84.30.195
  telemetry_in_band_vlan_id: 11
```

Example `instances.yml` snippet for out-of-band configuration:

```
global_configuration:
  CONTAINER_REGISTRY: hub.juniper.net/contrail
  CONTAINER_REGISTRY_USERNAME: < container_registry_username >
  CONTAINER_REGISTRY_PASSWORD: < container_registry_password >
provider_config:
  bms:
```

```
    ssh_pwd: <Root Pwd>
    ssh_user: root
    ntpserver: <NTP Server>
    domainsuffix: local
instances:  < under existing hierarchy >
  a7s33:
    ip: 10.84.30.201
    provider: bms
    roles:
      appformix_flows:
xflow_configuration:
  clickhouse_retention_period_secs: 7200
  loadbalancer_collector_vip: 10.84.30.195
```

4. Add the collector nodes:

```
bash deploy_insights_flows.sh <instance_file> --skip-provision --cluster-id <cluster_id>
```

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
2011	Contrail Networking Release 2011 supports installing Contrail Insights and Contrail Insights Flows on a Juju cluster after Contrail Networking and Contrail Command are installed.
2008	Starting in Contrail Release 2008, you can deploy Contrail Command and import a cluster using Juju in an environment using Kubernetes orchestration.
2005	Starting in Contrail Release 2005, you can deploy Contrail Command and import a cluster using Juju in a Canonical Openstack environment.

RELATED DOCUMENTATION

How to Login to Contrail Command | 62

Importing a Canonical Openstack Deployment Into Contrail Command | 109

Importing a Canonical Openstack Deployment Into Contrail Command

IN THIS SECTION

This document provides the steps needed to import a Canonical Openstack deployment into Contrail Command.

This procedure assumes that Contrail Command is already running in your Contrail Networking environment that is using Canonical Openstack as it's orchestration platform. See "How to Deploy Contrail Command and Import a Cluster Using Juju" on page 95 if you'd like to deploy Contrail Command and import the Contrail cluster into Contrail Command in an environment using Contrail Networking and Canonical Openstack.

Overview: Canonical Openstack Deployment into Contrail Command

Starting in Contrail Networking Release 2003, Canonical Openstack deployments can be managed using Contrail Command.

This document provides the steps needed to import a Canonical Openstack deployment into Contrail Command. Contrail Command can be used to manage the Canonical Openstack deployment after this procedure is complete.

This document makes the following assumptions about your environment:

- A Canonical Openstack deployment managed by Contrail Networking is already operational.
- Contrail Command is running in your environment. See Installing Contrail Command.
- Contrail Command has access to the Juju jumphost and the Juju cluster.

Canonical Openstack is imported into Contrail Command using Juju in this procedure.

Importing Canonical Openstack Into Contrail Command

To import Canonical Openstack into Contrail Cloud:

1. Install and start the Docker Engine.

There are multiple ways to perform this step. In this example, Docker Community Edition version 18.03 is installed using `yum install` and `yum-config-manager` commands and started using the `systemctl start docker` command.

```
yum install -y yum-utils device-mapper-persistent-data lvm2
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
yum install -y docker-ce-18.03.1.ce
systemctl start docker
```

2. Retrieve the `contrail-command-deployer` Docker image by logging into *hub.juniper.net* and entering the `docker pull` command.

```
docker login hub.juniper.net --username <container_registry_username> --password
<container_registry_password>
docker pull hub.juniper.net/contrail/contrail-command-deployer:<container_tag>
```

where *<container_tag>* is the container tag for the Contrail Command (UI) container deployment for the release that you are installing.

The *<container_tag>* for any Contrail Release 21xx image can be found in README Access to Contrail Registry 21XX.

3. Update the `config.yml` configuration file on the Contrail Command server.

The configuration of the `config.yml` file is unique to your environment and complete documentation describing all `config.yml` configuration options is beyond the scope of this document.

The following configuration parameters must be present in the `config.yml` file to support Canonical Openstack in Contrail Command:

- *ntpserver: <NTP_IP>*.

  The *NTP_IP* variable is the IP address of the NTP server.

- *vrouter_gateway: <VROUTER_GATEWAY_IP>*

  The *VROUTER_GATEWAY_IP* variable is the IP address of the vRouter gateway. The *vrouter_gateway:* parameter can be left empty, but it must be present.

- *container_registry: <CONTAINER_REGISTRY>*

  The *CONTAINER_REGISTRY* variable is the path to the container registry. The *CONTAINER_REGISTRY* is *hub.juniper.net/contrail* in most deployments.

- *container_tag: <COMMAND_BUILD_TAG>*

The *COMMAND_BUILD_TAG* variable is the Contrail Command (UI) container deployment for the release that you are installing. For any Contrail Release 21xx image, you can retrieve this value from README Access to Contrail Registry 21XX.

- *contrail_container_tag: <CONTRAIL_BUILD_TAG>*

  The *CONTRAIL_BUILD_TAG* variable is the Contrail build container for the release that you are installing. For any Contrail Release 21xx image, you can retrieve this value from README Access to Contrail Registry 21XX.

4. Run the Contrail Command deployer.

```
docker run -t --net host -e action=import_cluster -e orchestrator=juju -e
juju_controller=<juju_controller>
[-e juju_model=<juju_model_name> ]
[-e juju_controller_user=<juju_controller_user>]
[-e juju_controller_password=<juju_controller_password>]
[-e delete_db=<delete_db>]
[-e persist_rules=<persist-rules>]
-v <config_file>:/cluster_config.yml --privileged --name contrail_command_deployer <CCD_image>
```

In the following example, Contrail Command is deployed from the Juju jump host at 172.31.40.101.

```
docker run -td --net host --privileged -e action=import_cluster -e orchestrator=juju -e
juju_model=controller -e juju_controller=172.31.40.101 -e juju_controller_user=ubuntu -e
juju_controller_password=password -v /home/ubuntu/contrail-command-deployer/config.yaml:/
cluster_config.yml -v /root/.ssh:/root/.ssh contrail-command-deployer:latest
```

The command variables:

- *juju_controller*—(Required) The IP address of the Juju jump host. We define the Juju jump host in this context as the device that has installed the Juju CLI and is being used to run Juju commands. The Contrail Command server must have access to the Juju jump host at this IP address.
- *config_file*—(Required) The path to the configuration file. This configuration file was created in the previous step of this procedure.
- *CCD_image*—(Required) The Contrail Command deployer image.
- *delete_db*—(Optional) Specifies whether the PostgreSQL database is deleted during the process. The PostgreSQL database is deleted by default. Enter *no* in this field if you do not want the PostgreSQL database deleted.
- *persist-rules*—(Optional) Specify whether IP rules remain persistent across reboots.

- *juju_model_name*—(Optional) The name of the Juju model. The name can be retrieved by entering the *juju show-models* command.
- *juju_controller_user*—(Optional) The username of the Juju user on the Juju jump server.
- *juju_controller_password*—(Optional) The password for the Juju user on the Juju jumpbox. This password is used if no SSH keys have been installed.

5. (Optional) Track the progress of step 4.

```
docker logs -f contrail_command_deployer
```

6. Verify that the Contrail Command containers are running:

```
[root@centos254 ~]# docker ps -a
CONTAINER ID  IMAGE    <trimmed>   STATUS  <trimmed>  NAMES
2e62e778aa91  hub.juniper.net/...  Up      <trimmed>  contrail_command
c8442860e462  circleci/postgre...  Up      <trimmed>  contrail_psql
57a666e93d1a  hub.juniper.net/...  Exited  <trimmed> contrail_command_deployer
```

The `contrail_command` container is the GUI and the `contrail_psql` container is the database. Both containers should have a STATUS of `Up`.

The `contrail-command-deployer` container should have a STATUS of `Exited` because it exits when the installation is complete.

7. Open a web browser and enter *https://<Contrail-Command-Server-IP-Address>:8079* as a URL. The Contrail Command home screen appears.

   Choose token from the drop-down menu. Enter the username and password combination to Juju as the credentials, and use **admin_domain** as the domain.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
2003	Starting in Contrail Networking Release 2003, Canonical Openstack deployments can be managed using Contrail Command.

RELATED DOCUMENTATION

Installing Contrail Command

CHAPTER 4

Upgrading Contrail Software

IN THIS CHAPTER

Upgrading Contrail Networking using Contrail Command

Use the following procedure to upgrade Contrail Networking using Contrail Command.

NOTE: Before performing any upgrade, install Docker serially over containers. However, you can upgrade computes in parallel to Docker via script. After upgrading each docker host, verify the status of contrail and services. Do not proceed with upgrade on next hosts until all the services of contrail-status reports are running properly.

Use the following script to stop the running containers, upgrade the docker, and bring containers back:

```
docker ps --format '{{.Names}}' > running_containers
for CONTAINER in $(cat running_containers); do sudo docker stop $CONTAINER; done
yum install -y docker-ce-20.10.9 docker-ce-cli-20.10.9 docker-ce-rootless-
extras-20.10.9
for CONTAINER in $(cat running_containers); do sudo docker start $CONTAINER; done
```

The procedure supports incremental model and you can use it to upgrade from Contrail Networking Release *N-1* to *N*.

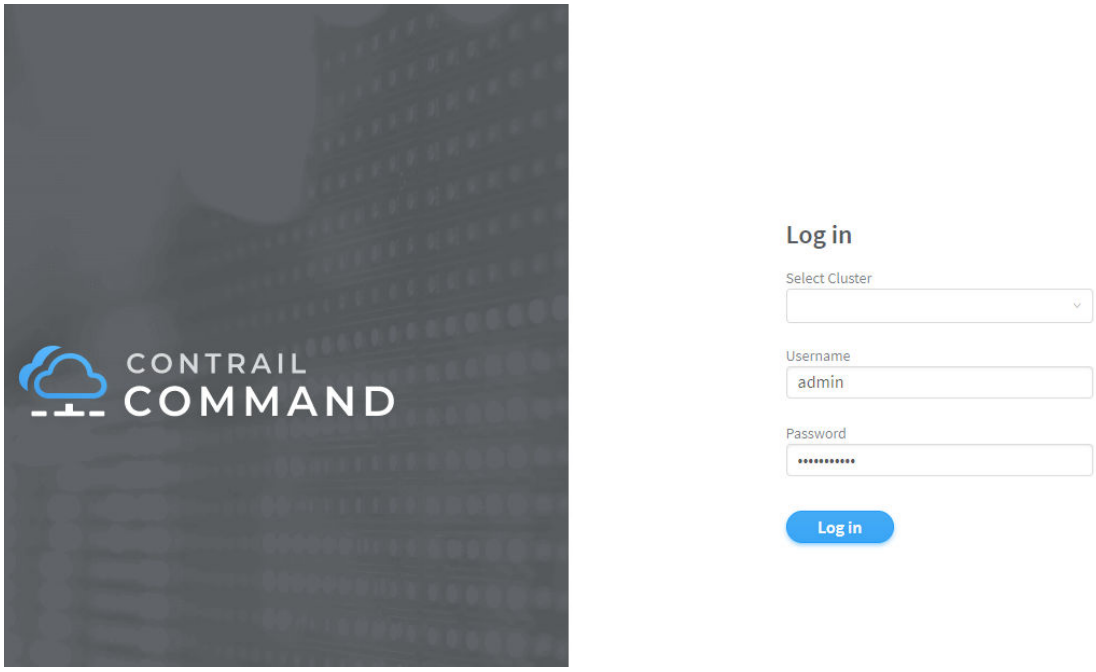
1. Create snapshots of your current configurations and upgrade Contrail Command. See "Upgrading Contrail Command using Backup Restore Procedure" on page 117.
2. Login to Contrail Command without logging into a Contrail cluster.

   In most environments, Contrail Command is accessed by entering `https://<Contrail-Command-Server-IP-Address>:9091` in your web browser..

   Leave the **Select Cluster** field blank. If the field is populated, use the drag-down menu to delete the pre-populated cluster selection.

   Enter your username and password credentials and click **Log in**

   If you have any issues logging into Contrail Command, see "How to Login to Contrail Command" on page 62.

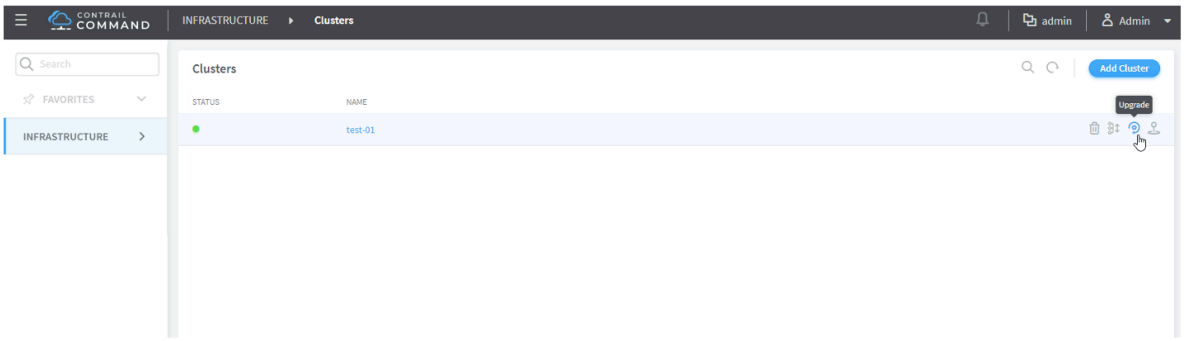


3. Click on **Clusters**.

   You will see the list of all the available clusters with the status.

4. Select the cluster you want to upgrade.

   Hover your mouse over ellipsis next to the cluster and click on **Upgrade**.



5. Enter **Contrail Version**, **Container Registry**, **Container Registry Username**, **Container Registry Password**.

   Contrail Version depicts the current installed Contrail version. You must update the value to the desired version number.

   The values for **Container Registry**, **Container Registry Username**, and **Container Registry Password** are pre-populated based on the values used during initial Contrail deployment.

   Click on **Contrail Configuration**.

   Add **CONTRAIL_CONTAINER_TAG**.

Access *CONTRAIL_CONTAINER_TAG* values for your targeted upgrade fromREADME Access to Contrail Registry 21XX.

6. If you have Contrail Insights and Contrail Insights Flows installed in the cluster:

   NOTE: Appformix and Appformix Flows were renamed Contrail Insights and Contrail Insights Flows. The Appformix naming conventions still appear during product usage, including within these directory names.

   - *Release 2005 or later*: You do not need to provide appropriate versions of Contrail Insights and Contrail Insights packages in the **/opt/software/appformix** and **/opt/software/xflow** directories. This step is no longer required starting in Release 2005.
   - *Release 2003 or earlier*:

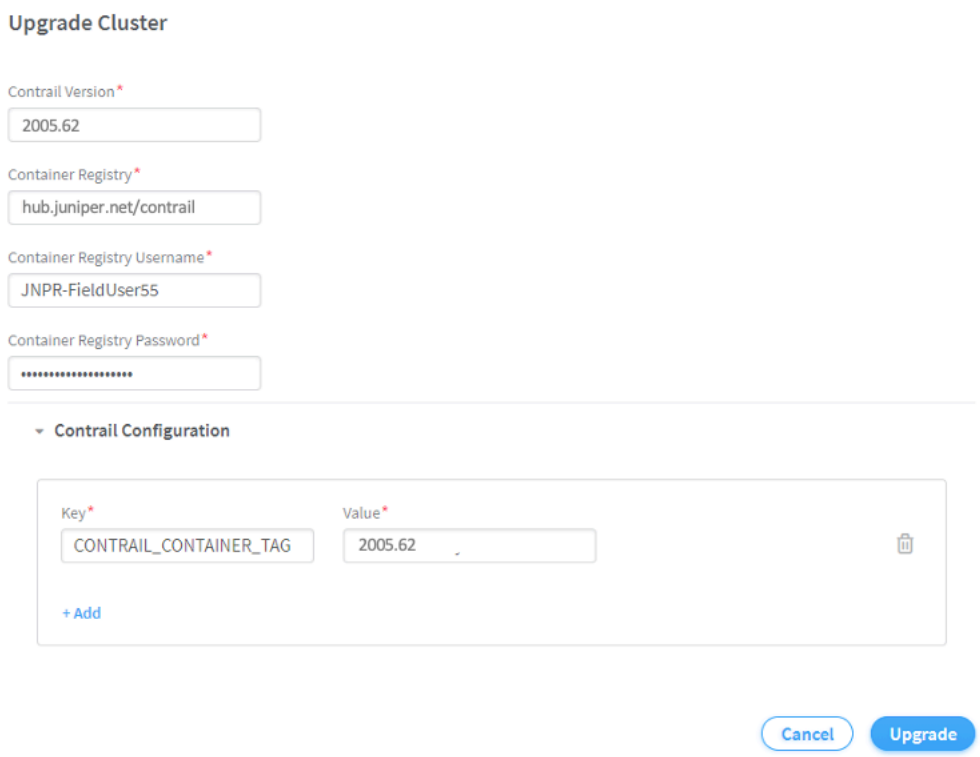
     Provide appropriate versions of Contrail Insights and Contrail Insights packages in the **/opt/software/appformix** and **/opt/software/xflow** directories on the Contrail Command server. For more details, refer to Installing Contrail Insights and Contrail Insights Flows using Contrail Command.

   Skip this step if you are not using Contrail Insights or Contrail Insights Flows.

7. Click on **Upgrade**.

   Figure 33 on page 117 provides a representative illustration of a user completing the **Upgrade Cluster** workflow to upgrade to Contrail Networking Release 2005.

Figure 33: Upgrade Cluster Workflow Example



RELATED DOCUMENTATION

- Installing Contrail Command

Upgrading Contrail Command using Backup Restore Procedure

You cannot use the SQL data with the new version of Contrail Command container if the database schema changes while upgrading the Contrail Command container.

You can resolve the issue by:

1. Back up SQL database in **yaml format db dump**.

   Run the following **docker exec contrail_command** command on the Contrail Command node to backup the DB.

   Contrail Networking Release 2005 or later:

```
docker exec contrail_command commandutil convert --intype rdbms --outtype yaml --out /etc/contrail/db.yml -
c /etc/contrail/command-app-server.yml; mkdir ~/backups; mv /etc/contrail/db.yml ~/backups/
```

Contrail Networking Release 2003 or earlier:

```
docker exec contrail_command contrailutil convert --intype rdbms --outtype yaml --out /etc/contrail/db.yml -
c /etc/contrail/contrail.yml; mkdir ~/backups; mv /etc/contrail/db.yml ~/backups/
```

2. Upgrade the Contrail Command container.

   Specify the desired version of Contrail Command container *(container_tag)* in the deployer input file (**command_servers.yml**) and deploy playbook.

   You must use *PostgreSQL* in the **command_servers.yml** file.

   The step depends on how you have deployed the Contrail Command.

   - Contrail Command is deployed using docker installation:

     ```
     docker run -td --net host -v <ABSOLUTE_PATH_OF_COMMAND_SERVERS_FILE>:/command_servers.yml --privileged --
     name contrail_command_deployer_<contrail_container_tag> hub.juniper.net/contrail/contrail-command-
     deployer:<<contrail_container_tag>
     ```

   - Contrail Command is deployed through juju-charms:

     ```
     juju config contrail-command image-tag=<contrail_container_tag>
     ```

     After entering this command, enter the **juju config contrail-command image-tag** command to ensure the Contrail Command container is associated with the new image tag.

     If the command output displays the old image tag, wait several minutes then retry the **juju config contrail-command image-tag** command.

     If the command output displays the new image tag, proceed to the next step. If the command output continues to display the old image tag, re-enter the **juju config contrail-command image-tag=*contrail_container_tag*** to upgrade the container.

   The *contrail_container_tag* for any Contrail Release 21 software can be obtained from README Access to Contrail Registry 21XX.

3. This step depends on your Contrail Networking release.

   Contrail Networking Release 2005 or later:

   Migrate the **yaml formatted db dump** to the new database schema:

   ```
   docker exec contrail_command mkdir /root/backups
   docker cp /root/backups/db.yml contrail_command:/root/backups/
   docker exec contrail_command commandutil migrate --in /root/backups/db.yml --out /root/
   backups/db_migrated.yml
   ```

Contrail Networking Release 2003 or earlier:

Modify the yaml-formatted db dump by adding or removing the fields per the new database schema.

4. Restore the modified **yaml formatted db dump** to the SQL database.

Contrail Networking Release 2005 or later:

```
docker exec contrail_command commandutil convert --intype yaml --in /root/backups/
db_migrated.yml --outtype rdbms -c /etc/contrail/command-app-server.yml
```

Contrail Networking Release 2003 or earlier:

```
docker exec contrail_command mkdir /root/backups docker cp /root/backups/db.yml
contrail_command:/root/backups/
docker exec contrail_command contrailutil convert --intype yaml --in ~/backups/db.yml --
outtype rdbms -c /etc/contrail/contrail.yml
```

NOTE: If the restore procedure fails because of schema mismatch, repeat Step 3 and Step 4 with incremental db dump changes.

Fast Forward Upgrade: Updating Contrail Networking 1912.L4 and Red Hat OpenStack 13 to Contrail Networking 21.4.L2 and Red Hat Openstack 16.2

IN THIS SECTION

This document provides the steps needed to update an environment that is running Contrail Networking Release 1912.L4 and Red Hat OpenStack 13 to an environment running Contrail Networking 21.4.L2 and Red Hat Openstack 16.2.

The procedure provides a fast forward upgrade (FFU). AN FFU upgrades the Red Hat OpenStack software from it's currently running version to a version that is several releases later. An FFU provides an opportunity to upgrade to OpenStack versions that are considered long life versions and upgrade when the next long life version is available. This document shows how to perform this FFU for Red Hat Openstack while also upgrading Contrail Networking.

When to Use This Procedure

The procedure in this document has been validated for the following Contrail Networking upgrade scenarios:

Table 4: Validated Upgrade Scenarios

Starting Releases	Target Releases
Contrail Networking Release 1912.L4 Red Hat Openstack 13	Contrail Networking Release 21.4.L2 Red Hat Openstack 16.2.4

Preparing for the Upgrade and Upgrading the Undercloud

This document will frequently reference the Red Hat Openstack document for performing the Red Hat Openstack upgrade. See Framework for Upgrades (13 to 16.2)

To upgrade the undercloud, see the Planning and preparation for an in-place upgrade through the Configuring the overcloud for a Leapp upgrade sections of the Framework for Upgrades (13 to 16.2) guide.

Upgrading the Overcloud

Before you begin, we strongly recommend becoming familiar with processes to efficiently upgrade the overcloud. See Speeding up an overcloud upgrade in the Framework for Upgrades (13 to 16.2) guide.

1. (Only needed if undercloud is used as a container registry) Create a Contrail container file and upload the container file.
   a. Create a Contrail container file using the Heat template tools and import the existing registry files into the Contrail container file:

```
cd ~/tf-heat-templates/tools/contrail
./import_contrail_container.sh -f container_outputfile -r registry -t tag [-i insecure] [-
u username] [-p password] [-c certificate path]
```

   Examples:

To import the files from a password protected public registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r enterprise-hub.jnpr.net/
contrail-container-prod/contrail-base -u USERNAME -p PASSWORD -t 1234
```

To import the files from a private secure registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r device.example.net:5443 -c
http://device.example.net/pub/device.example.net.crt -t 1234
```

To import the files from a private insecure registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r 10.0.0.1:5443 -i 1 -t 1234
```

b. From the undercloud, upload the container file:

```
openstack overcloud container image upload --config-file /tmp/contrail_container
```

2. Clone the Tungsten Fabric deployment repository:

> **WARNING**: Review the files in the Github repository at https://github.com/tungstenfabric/tf-deployment-test.git before entering this command to clone the files. In most scenarios, you will need to update some file parameters for your environment after cloning the repository.

```
git clone https://github.com/tungstenfabric/tf-deployment-test.git
```

Note the following key directory locations in the repository for a Contrail Networking upgrade:

- Tungsten fabric scripts location: *tf-deployment-test/tree/master/rhosp/ffu_ziu_13_16/tf_specific*
- Red Hat-specific steps are available in this directory: *tf-deployment-test/tree/master/rhosp/ffu_ziu_13_16/redhat_ffu_steps*

3. Prepare the templates to upgrade all of the overcloud nodes.

Update the services.yaml file with the '21.4.L2' tag . Be aware of the following while performing this step:

- Name the overcloud nodes using predictable NIC names. See the Using predictable NIC names for overcloud nodes section of the Framework for Upgrades (13 to 16.2) guide from Red Hat.

```
ansible-playbook -i inventory.yaml tf-deployment-test/rhosp/ffu_ziu_13_16/redhat_files/
playbook-nics.yaml
```

After entering this command, follow the instructions in Speeding up an overcloud upgrade to upgrade all overcloud nodes. All overcloud nodes should be upgraded in this step, including Openstack nodes, Contrail controller nodes, and compute nodes.

4. For compute nodes, also perform these steps:

   a. Migrate the existing workloads. See the Migrating virtual machine instances between Compute nodes document from Red Hat.

   b. Be aware of the following potential problems and the workarounds if you run into them:

      - CEM-29946: A kernel upgrade fails during the compute node reboot. Containers are stuck in the creating state.

        Workaround: Reboot the node. After rebooting the node, add the following snippet to the *~/tripleo-heat-templates/upgrades-environment.yaml* file:

```
UpgradeInitCommand: |
    {% if 'Compute' in group_names %}
       sudo grubby --update-kernel=/boot/vmlinuz-$(uname -r) --
args="default_hugepagesz=1GB hugepagesz=1G hugepages=20"
    {% endif %}
```

        NOTE: This snippet assumes you are not using custom roles.
        If you are using custom roles, please replace **Compute** in line 2 of this snippet with your custom name.

      - CEM-29947: A compute node using DPDK fails during an FFU upgrade.

        Workaround: Add the following snippet to the *~/tripleo-heat-templates/upgrades-environment.yaml* file:

```
UpgradeInitCommand: |
    {% if 'ContrailDpdk' in group_names %}
      sudo grubby --update-kernel=/boot/vmlinuz-$(uname -r) --args="intel_iommu=on
iommu=pt"
```

```
    sudo grub2-mkconfig -o /etc/grub2.cfg
{% endif %}
```

NOTE: This snippet assumes your are not using custom roles.

If you are using custom roles, please replace **ContrailDpdk** in line 2 of this snippet with your custom name.

c. Run the overcloud system upgrade prepare scripts.

To obtain the scripts from Github, see the following URLs:

- https://github.com/tungstenfabric/tf-deployment-test/blob/master/rhosp/ffu_ziu_13_16/tf_specific/run_overcloud_system_upgrade_prepare.sh
- https://github.com/tungstenfabric/tf-deployment-test/blob/master/rhosp/ffu_ziu_13_16/tf_specific/run_overcloud_system_upgrade_run.sh

d. Setup the vhost interfaces on the Contrail compute nodes by executing the playbook-nics-vhost0.yaml playbook.

```
ansible-playbook -i inventory.yaml -l overcloud_ContrailDpdk tf-deployment-test/rhosp/
ffu_ziu_13_16/tf_specific/playbook-nics-vhost0.yaml
ansible-playbook -i inventory.yaml -l overcloud_Compute tf-deployment-test/rhosp/
ffu_ziu_13_16/tf_specific/playbook-nics-vhost0.yaml
```

e. Upgrade the compute nodes in parallel:

WARNING: Execute these two commands as listed in this step. Do not execute a single command - for instance, **openstack overcloud upgrade run --limit compute0** - as suggested in the Red Hat documentation to compute the compute node upgrade.

```
openstack overcloud upgrade run --yes --stack overcloud --tags system_upgrade_prepare --
limit name-of-compute
openstack overcloud upgrade run --yes --stack overcloud --tags system_upgrade_run --limit
name-of-compute
```

Follow the steps in Upgrading Compute nodes in parallel to complete this step.

5. Starting with the Synchronizing the overcloud stack step from the Framework for Upgrades (13 to 16.2) Red Hat document, complete the upgrade.

How to Perform a Zero Impact Contrail Networking Upgrade using the Ansible Deployer

Before you begin:

- The target release for this upgrade must be Contrail Release 2005 or later.
- You can use this procedure to incrementally upgrade to the next Contrail Networking release only. For instance, if you are running Contrail Networking Release 2003 and want to upgrade to the next Contrail Release—which is Contrail Networking Release 2005—you can use this procedure to perform the upgrade.

  This procedure is not validated for upgrades between releases that are two or more releases apart. For instance, it could not be used to upgrade from Contrail Networking Release 2002 to Contrail Networking Release 2005.

  For a list of Contrail Networking releases in a table that illustrates Contrail Networking release order, see Contrail Networking Supported Platforms.
- The Contrail Ansible Deployer container can only be used in CentOS environments.
- Take snapshots of your current configurations before you proceed with the upgrade process. For details, refer to "How to Backup and Restore Contrail Databases in JSON Format" on page 243.

Starting in Contrail Networking Release 2005, you can perform a Zero Impact Upgrade (ZIU) of Contrail Networking using the Contrail Ansible Deployer container. The Contrail Ansible Deployer container image can be loaded from the Juniper Networks Contrail Container Registry hosted at *hub.juniper.net/contrail*.

Use the procedure in this document to perform a Zero Impact Upgrade (ZIU) of Contrail Networking using the Contrail Ansible Deployer container. This ZIU allows Contrail Networking to upgrade while sustaining minimal network downtime.

This procedure illustrates how to perform a ZIU using the Ansible deployer container. It includes a representative example of the steps being performed to upgrade from Contrail Networking Release 2005 to Release 2008.

To perform the ZIU using the Ansible deployer:

1. Pull the *contrail-ansible-deployer* file for the target upgrade release. This procedure is typically performed from a Contrail controller running in your environment, but it can also be performed from a separate server which has network connectivity to the deployment that is being upgraded.

   This procedure shows you how to load a 2008 image from the Juniper Networks Contrail Container Registry. You can, however, also change the values to load the file from a private registry.

The Juniper Networks Contrail Container Registry is hosted at *hub.juniper.net/contrail*. If you need the credentials to access the registry, email mailto:contrail-registry@juniper.net.

Enter the following commands to pull the *contrail-ansible-deployer* file from the registry:

```
sudo docker login -u <username> -p <password> hub.juniper.net
sudo docker pull hub.juniper.net/contrail/contrail-kolla-ansible-
deployer:2008.<contrail_container_tag>
```

where:

- *username*—username to access the registry. Email mailto:contrail-registry@juniper.net if you need to obtain *username* and *password* credentials.
- *password*—password to access the registry. Email mailto:contrail-registry@juniper.net if you need to obtain *username* and *password* credentials.
- *contrail_container_tag*—the container tag ID for your target Contrail Networking release. The *contrail_container_tag* for any Contrail Release 21 software can be obtained from README Access to Contrail Registry 21XX.

2. Start the Contrail Ansible Deployer:

```
docker run -t --net host -d --privileged --name contrail-kolla-ansible-deployer
hub.juniper.net/contrail/contrail-kolla-ansible-deployer:2008.<contrail_container_tag>
```

3. Navigate to the *instances.yaml* file and open it for editing.

   The *instances.yaml* file was used to initially deploy the setup. The *instances.yaml* can be loaded into the Contrail Ansible Deployer and edited to supported the target upgrade version.

   Contrail Release 2008 Target Upgrade Example using VI as the editor.

```
docker cp instances.yaml contrail-kolla-ansible-deployer:/root/contrail-ansible-deployer/
config/instances.yaml
docker exec -it contrail-kolla-ansible-deployer bash
cd /root/contrail-ansible-deployer/config/
vi instances.yaml
```

4. Update the *CONTRAIL_CONTAINER_TAG* to the desired version tag in the **instances.yaml** file from the existing deployment. The *CONTRAIL_CONTAINER_TAG* variable is in the *contrail_configuration:* hierarchy within the **instances.yaml** file.

   The *CONTRAIL_CONTAINER_TAG* for any Contrail Release 21 software can be obtained from README Access to Contrail Registry 21XX.

Here is an example instances.yml file configuration:

```
contrail_configuration:
  CONTRAIL_CONTAINER_TAG: "2008.121"
  CONFIG_DATABASE_NODEMGR__DEFAULTS__minimum_diskGB: "2"
  DATABASE_NODEMGR__DEFAULTS__minimum_diskGB: "2"
  JVM_EXTRA_OPTS: "-Xms1g -Xmx2g"
  VROUTER_ENCRYPTION: FALSE
  LOG_LEVEL: SYS_DEBUG
  CLOUD_ORCHESTRATOR: kubernetes
```

5. Navigate to a path from where you want to trigger the upgrade playbook.

```
cd /root/contrail-ansible-deployer
```

6. Upgrade the control plane by running the ziu.yml playbook file from inside the contrail ansible deployer container.

   - For Contrail Networking Release 2005 to Contrail Networking Release 2008:

     Upgrade the control plane by running the **ziu.yml** playbook file.

     ```
     sudo -E ansible-playbook -v -e orchestrator=openstack -e config_file=instances.yaml playbooks/ziu.yml
     ```

   - For Contrail Networking Release 2011 and later:

     Upgrade the control plane by running the controller stage of **ziu.yml** playbook file.

     ```
     sudo -E ansible-playbook -v -e stage=controller -e orchestrator=openstack -e config_file=config/
     instances.yaml playbooks/ziu.yml
     ```

7. Upgrade the Openstack plugin by running the install_openstack.yml playbook file.

   - For Contrail Networking Release 2005 to Contrail Networking Release 2008:

     ```
     sudo -E ansible-playbook -v -e orchestrator=openstack -e config_file=instances.yaml playbooks/
     install_openstack.yml
     ```

   - For Contrail Networking Release 2011 and later:

     ```
     sudo -E ansible-playbook -v -e stage=openstack -e orchestrator=openstack -e config_file=config/
     instances.yaml playbooks/ziu.yml
     ```

8. Enter the `contrail-status` command to monitor upgrade status. Ensure all pods reach the *running* state and all services reach the *active* state.

   This `contrail-status` command provides this output after a successful upgrade:

NOTE: Some output fields and data have been removed for readability.

```
                            Original
Pod             Service     Name                                  State
                 redis       contrail-external-redis               running
                 rsyslogd                                          running
analytics        api         contrail-analytics-api                running
analytics        collector   contrail-analytics-collector          running
analytics        nodemgr     contrail-nodemgr                      running
analytics        provisioner contrail-provisioner                  running
analytics-alarm  alarm-gen   contrail-analytics-alarm-gen          running
analytics-alarm  kafka       contrail-external-kafka               running
analytics-alarm  nodemgr     contrail-nodemgr                      running
analytics-alarm  provisioner contrail-provisioner                  running
analytics-snmp   nodemgr     contrail-nodemgr                      running
analytics-snmp   provisioner contrail-provisioner                  running
analytics-snmp   snmp-collector contrail-analytics-snmp-collector  running
analytics-snmp   topology    contrail-analytics-snmp-topology      running
config           api         contrail-controller-config-api        running
config           device-manager contrail-controller-config-devicemgr running
config           dnsmasq     contrail-controller-config-dnsmasq    running
config           nodemgr     contrail-nodemgr                      running
config           provisioner contrail-provisioner                  running
config           schema      contrail-controller-config-schema     running
config           stats       contrail-controller-config-stats      running
config           svc-monitor contrail-controller-config-svcmonitor running
config-database  cassandra   contrail-external-cassandra           running
<trimmed>

vrouter kernel module is PRESENT
== Contrail control ==
control: active
nodemgr: active
named: active
dns: active

== Contrail analytics-alarm ==
nodemgr: active
kafka: active
```

```
alarm-gen: active

== Contrail kubernetes ==
kube-manager: active

== Contrail database ==
nodemgr: active
query-engine: active
cassandra: active

== Contrail analytics ==
nodemgr: active
api: active
collector: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
cassandra: active

== Contrail webui ==
web: active
job: active

== Contrail vrouter ==
nodemgr: active
agent: active

== Contrail analytics-snmp ==
snmp-collector: active
nodemgr: active
topology: active

== Contrail config ==
svc-monitor: active
nodemgr: active
device-manager: active
api: active
schema: active
```

9. Migrate workloads VM from one group of compute nodes. Leave them uncommented in the instances.yaml file. Comment other computes not ready to upgrade in instances.yaml.

10. Upgrade compute nodes.

    - For Contrail Networking Release 2005 to Contrail Networking Release 2008:

      Run the **install_contrail.yml** playbook file to upgrade the compute nodes that were uncommented in the **instances.yaml** file. Only the compute nodes that were left uncommented in 9 are upgraded to the target release in this step.

      ```
      sudo -E ansible-playbook -v -e orchestrator=openstack -e config_file=instances.yaml playbooks/
      install_contrail.yml
      ```

    - For Contrail Networking Release 2011 and later:

      Run the compute stage of **ziu.yml** playbook file to upgrade the compute nodes that were uncommented in the **instances.yaml** file. Only the compute nodes that were left uncommented in 9 are upgraded to the target release in this step.

      ```
      sudo -E ansible-playbook -v -e stage=compute -e orchestrator=openstack -e config_file=config/
      instances.yaml playbooks/ziu.yml
      ```

11. Repeat Steps 9 and 10 until all compute nodes are upgraded.

You can access the Ansible playbook logs of the upgrade at `/var/log/ansible.log`.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
2005	Starting in Contrail Networking Release 2005, you can perform a Zero Impact Upgrade (ZIU) of Contrail Networking using the Contrail Ansible Deployer container.

RELATED DOCUMENTATION

Installing a Contrail Cluster using Contrail Command and instances.yml | 78

Upgrading Contrail Networking using Contrail Command | 114

Updating Contrail Networking Release 21.4 with Openstack 16.2 to Contrail Networking Release 21.4.L1 with Openstack 16.2.3 using Zero Impact Upgrade Process

IN THIS SECTION

This document provides the steps needed to update a Contrail Networking deployment that is using Red Hat Openstack 16.2 as it's orchestration platform. The procedure provides a zero impact upgrade (ZIU) with minimal disruption to network operations. This procedure upgrades Contrail Networking and the Red Hat Openstack versions.

You have the option to perform a procedure that upgrades the Contrail Networking version without upgrading OpenStack. For information on that procedure, see "Updating Contrail Networking Containers Without Updating OpenStack" on page 139.

When to Use This Procedure

This procedure is used to upgrade Contrail Networking when it is running in environments using RHOSP version 16.2 (RHOSP16.2).

You use this procedure to update Contrail Networking and OpenStack in the same procedure. You have the option to perform a procedure that upgrades the Contrail Networking version without upgrading OpenStack. For information on that procedure, see "Updating Contrail Networking Containers Without Updating OpenStack" on page 139.

The procedure in this document has been validated for the following Contrail Networking upgrade scenarios:

Table 5: Contrail Networking with RHOSP16.2.x Validated Upgrade Scenarios

Starting Releases	Target Releases
Contrail Release: 21.4 Openstack Release: 16.2 Contrail Release: 21.4.L1 Openstack Release: 16.2.3	Contrail Release: 21.4.L1 Openstack Release: 16.2.3 Contrail Release: 21.4.L2 Openstack Release: 16.2.4

NOTE: You can upgrade from Contrail Release: 21.4.L1 Openstack Release: 16.2.3 to Contrail Release: 21.4.L2 Openstack Release: 16.2.4 using same upgrade procedure given for Contrail Release: 21.4 Openstack Release: 16.2 to Contrail Release: 21.4.L1 Openstack Release: 16.2.3. However, do remember to update tags in accordance with the release.

For a similar procedure to upgrade Contrail Networking in Openstack 16.1 environments, see Updating Contrail Networking using the Zero Impact Upgrade Process in an Environment using Red Hat Openstack 16.1.

Prerequisites

This document makes the following assumptions about your environment:

- A Contrail Networking deployment using Red Hat Openstack version 16.2 (RHOSP16.2) as the orchestration platform is already operational.
- The overcloud nodes in the RHOSP16.2 environment have an enabled Red Hat Enterprise Linux (RHEL) subscription.
- If you are updating Red Hat Openstack simultaneously with Contrail Networking, we assume that the undercloud node is updated to the latest minor version and that new overcloud images are prepared for an upgrade. See the Upgrading the Undercloud section of the Keeping Red Hat OpenStack Platform Updated guide from Red Hat.

  If the undercloud has been updated and a copy of the heat templates are used for the deployment, update the copy of the heat template from the Red Hat's core heat template collection at `/usr/share/openstack-tripleo-heat-templates`. See the Understanding Heat Templates document from Red Hat for information on this process.

Before You Begin

We recommend performing these procedures before starting the update:

- Backup your Contrail configuration database before starting this procedure. See How to Backup and Restore Contrail Databases in JSON Format in Openstack Environments Using the Openstack 16.1 Director Deployment.

- Each compute node agent will go down during this procedure, causing some compute node downtime. The estimated downtime for a compute node varies by environment, but typically took between 12 and 15 minutes in our testing environments.

  If you have compute nodes with workloads that cannot tolerate this downtime, consider migrating workloads or taking other steps to accommodate this downtime in your environment.

Updating Contrail Networking in an Environment using Red Hat Openstack 16.2

To update Contrail Networking in an environment that is using Red Hat Openstack 16.2 as the orchestration platform:

1. Prepare your container registry. The registry is often included in the undercloud, but it can also be a separate node.
2. Backup the Contrail TripleO Heat Templates. See Using the Contrail Heat Template.
3. Get the Contrail TripleO Heat Templates (Stable/Train branch) from https://github.com/Juniper/contrail-tripleo-heat-templates.

   Prepare the new tripleo-heat-templates with latest available software from Openstack and Contrail Networking.

   ```
   sudo cp -r /usr/share/openstack-tripleo-heat-templates/ tripleo-heat-templates
   sudo git clone https://github.com/tungstenfabric/tf-tripleo-heat-templates -b stable/train
   sudo cp -r tf-tripleo-heat-templates/* tripleo-heat-templates/
   ```

4. Update the parameter *ContrailImageTag* to the new version.

   The location of the *ContrailImageTag* variable varies by environment. In the most commonly-used environments, this variable is set in the *contrail-services.yaml* file.

   You can obtain the *ContrailImageTag* parameter from the README Access to Contrail Registry 21XX.

   NOTE: If you are using the undercloud as a registry, ensure the new contrail image is updated in undercloud before proceeding further.

5. Update the overcloud by entering the `openstack overcloud update prepare` command and include the files that were updated during the previous steps with the overcloud update.

Example:

```
(undercloud) [stack@uc-train ~]$ cat deploy-ipu-prepare.sh
python3 tripleo-heat-templates/tools/process-templates.py --clean \
-r roles_data_contrail_aio.yaml -p tripleo-heat-templates/
python3 tripleo-heat-templates/tools/process-templates.py \
-r roles_data_contrail_aio.yaml \
-p tripleo-heat-templates/
openstack overcloud update prepare --templates tripleo-heat-templates/ \
--stack overcloud --libvirt-type kvm \
--roles-file roles_data_contrail_aio.yaml \
-e ~/rhsm.yaml \
-e tripleo-heat-templates/environments/network-isolation.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e tripleo-heat-templates /environments/contrail/contrail-net.yaml \
-e tripleo-heat-templates /environments/contrail/contrail-plugins.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-tls.yaml \
-e tripleo-heat-templates /environments/ssl/tls-everywhere-endpoints-dns.yaml \
-e tripleo-heat-templates /environments/services/haproxy-public-tls-certmonger.yaml \
-e tripleo-heat-templates /environments/ssl/enable-internal-tls.yaml \
-e containers-prepare-parameter.yaml
```

6. Prepare the overcloud nodes that include Contrail containers for the update.

- Pull the images in the repository onto the overcloud nodes.

  There are multiple methods for performing this step. Commonly used methods for performing this operation include using the `podman pull` command for Docker containers and the `openstack overcloud container image upload` command for Openstack containers, or running the `tripleo-heat-templates/upload.containers.sh` and `tools/contrail/update_contrail_preparation.sh` scripts.

- (Not required in all setups) Provide export variables for the script if the predefined values aren't appropriate for your environment. The script location:

  ```
  ~/tripleo-heat-templates/tools/contrail/update_contrail_preparation.sh
  ```

  The following variables within the script are particularly significant for this upgrade:

  - `CONTRAIL_NEW_IMAGE_TAG`—The image tag of the target upgrade version of Contrail. The default value is `latest`.

    If needed, you can obtain this parameter for a specific image from the README Access to Contrail Registry 21XX.

> **NOTE**: Some older deployments use the `CONTRAIL_IMAGE_TAG` variable in place of the `CONTRAIL_NEW_IMAGE_TAG` variable. Both variables are recognized by the `update_contrail_preparation.sh` script and perform the same function.

- `SSH_USER`—The SSH username for logging into overcloud nodes. The default value is `heat-admin`.
- `SSH_OPTIONS`—Custom SSH option values.

  The default SSH options for your environment are typically pre-defined. You are typically only changing this value if you want to customize your update.
- `STOP_CONTAINERS`—The list of containers that must be stopped before the upgrade can proceed. The default value is `contrail_config_api contrail_analytics_api`.

- Run the script:

> **CAUTION**: Contrail services stop working when the script starts running.

```
~/tripleo-heat-templates/tools/contrail/update_contrail_preparation.sh
```

7. Update the Contrail Controller nodes:

- Run the `openstack overcloud update run` command on the first Contrail controller and, if needed, on a Contrail Analytics node. The purpose of this step is to update one Contrail Controller and one Contrail Analytics node to support the environment so the other Contrail Controllers and analytics nodes can be updated without incurring additional downtime.

  Example:

```
openstack overcloud update run --limit overcloud-contrailcontroller-0
```

  Ensure that the contrail status is `ok` on overcloud-contrailcontroller-0 before proceeding.

  If the analytics and the analyticsdb nodes are on separate nodes, you may have to update the nodes individually:

```
openstack overcloud update run --limit overcloud-contrailcontroller-0
openstack overcloud update run --roles
ContrailAnalytics,ContrailAnalyticsDatabase
```

- After the upgrade, check the docker container status and versions for the Contrail Controllers and the Contrail Analytics and AnalyticsDB nodes.

```
podman ps -a
```

- Update the remaining Contrail Controller nodes:

  Example:

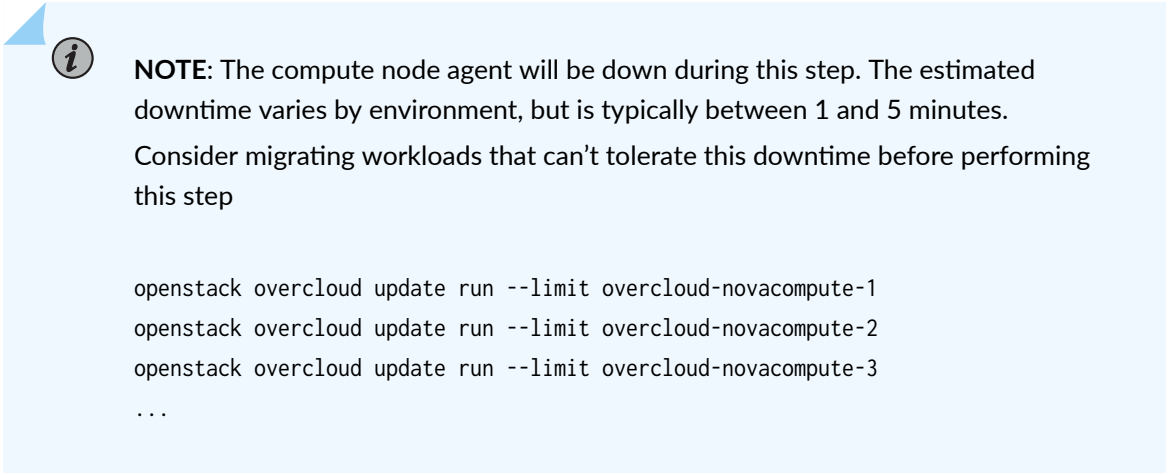
```
openstack overcloud update run --limit overcloud-contrailcontroller-1
openstack overcloud update run --limit overcloud-contrailcontroller-2
openstack overcloud update run --limit overcloud-contrailcontroller-3
...
```

8. Update the Openstack Controllers using the `openstack overcloud update run` commands:

   Example:

```
openstack overcloud update run --limit overcloud-controller-0
openstack overcloud update run --limit overcloud-controller-1
openstack overcloud update run --limit overcloud-controller-2
...
```

9. Individually update the compute nodes.



NOTE: The compute node agent will be down during this step. The estimated downtime varies by environment, but is typically between 1 and 5 minutes. Consider migrating workloads that can't tolerate this downtime before performing this step

```
openstack overcloud update run --limit overcloud-novacompute-1
openstack overcloud update run --limit overcloud-novacompute-2
openstack overcloud update run --limit overcloud-novacompute-3
...
```

Reboot your compute node to complete the update.

NOTE: A reboot is required to complete this procedure only if a kernel update is also needed. If you would like to avoid rebooting your compute node, check the log files in the `/var/log/yum.log` file to see if kernel packages were updated during the compute node update. A reboot is required only if kernel updates occurred as part of the compute node update procedure.

```
sudo reboot
```

Use the `contrail-status` command to monitor upgrade status. Ensure all pods reach the *running* state and all services reach the *active* state.

This `contrail-status` command provides output after a successful upgrade:

NOTE: Some output fields and data have been removed from this `contrail-status` command sample for readability.

```
Pod             Service        Original Name                    State
analytics       api            contrail-analytics-api           running
analytics       collector      contrail-analytics-collector     running
analytics       nodemgr        contrail-nodemgr                 running
analytics       provisioner    contrail-provisioner             running
analytics       redis          contrail-external-redis          running
analytics-alarm alarm-gen      contrail-analytics-alarm-gen     running
analytics-alarm kafka          contrail-external-kafka          running
analytics-alarm nodemgr        contrail-nodemgr                 running
analytics-alarm provisioner    contrail-provisioner             running
analytics-alarm zookeeper      contrail-external-zookeeper      running
analytics-snmp  nodemgr        contrail-nodemgr                 running
analytics-snmp  provisioner    contrail-provisioner             running
analytics-snmp  snmp-collector contrail-analytics-snmp-collector running
analytics-snmp  topology       contrail-analytics-snmp-topology  running
config          api            contrail-controller-config-api    running
<trimmed>

== Contrail control ==
control: active
nodemgr: active
named: active
```

```
dns: active

== Contrail analytics-alarm ==
nodemgr: active
kafka: active
alarm-gen: active

== Contrail database ==
nodemgr: active
query-engine: active
cassandra: active

== Contrail analytics ==
nodemgr: active
api: active
collector: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
cassandra: active

== Contrail webui ==
web: active
job: active

== Contrail analytics-snmp ==
snmp-collector: active
nodemgr: active
topology: active

== Contrail config ==
svc-monitor: active
nodemgr: active
device-manager: active
api: active
schema: active
```

10. Enter the `openstack overcloud update converge` command to finalize the update.

NOTE: The options used in the `openstack overcloud update converge` in this step will match the options used with the `openstack overcloud update prepare` command entered in 5.

```
(undercloud) [stack@uc-train ~]$ cat deploy-ipu-converge.sh
python3 tripleo-heat-templates/tools/process-templates.py --clean \
-r roles_data_contrail_aio.yaml -p tripleo-heat-templates/
python3 tripleo-heat-templates/tools/process-templates.py \
-r roles_data_contrail_aio.yaml \
-p tripleo-heat-templates/
openstack overcloud update converge --templates tripleo-heat-templates/ \
--stack overcloud --libvirt-type kvm \
--roles-file roles_data_contrail_aio.yaml \
-e ~/rhsm.yaml \
-e tripleo-heat-templates/environments/network-isolation.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e tripleo-heat-templates /environments/contrail/contrail-net.yaml \
-e tripleo-heat-templates /environments/contrail/contrail-plugins.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-tls.yaml \
-e tripleo-heat-templates /environments/ssl/tls-everywhere-endpoints-dns.yaml \
-e tripleo-heat-templates /environments/services/haproxy-public-tls-certmonger.yaml \
-e tripleo-heat-templates /environments/ssl/enable-internal-tls.yaml \
-e containers-prepare-parameter.yaml
```

Monitor screen messages indicating *SUCCESS* to confirm that the updates made in this step are successful.

RELATED DOCUMENTATION

Installing Contrail with OpenStack by Using Juju Charms | 596

Updating Contrail Networking Containers Without Updating OpenStack

IN THIS SECTION

In Contrail Networking Release 21.4, you have the option to upgrade your Contrail containers without updating OpenStack. This document provides the steps to complete this upgrade.

NOTE: This upgrade procedure is not applicable to Contrail Networking Release 21.4.L1 and thereafter.

If you want to upgrade Contrail Networking and OpenStack in the same procedure, see "Updating Contrail Networking Release 21.4 with Openstack 16.2 to Contrail Networking Release 21.4.L1 with Openstack 16.2.3 using the Zero Impact Upgrade Process" on page 130.

When to Use This Procedure

This procedure is used to upgrade Contrail Networking when it is running in environments using RHOSP version 16.2 (RHOSP16.2). RHOSP16.2 is not updated during this procedure.

The procedure in this document has been validated for the following Contrail Networking upgrade scenarios:

Table 6: Contrail Networking with RHOSP 16.2 Validated Upgrade Scenarios

Starting Releases	Target Releases
Contrail Networking: 21.4 Red Hat OpenStack: 16.2 Red Hat Enterprise Linux (RHEL): 8.4	Contrail Networking: 21.4.L1 Red Hat OpenStack: 16.2 Red Hat Enterprise Linux (RHEL): 8.4

Prerequisites

This document makes the following assumptions about your environment:

- A Contrail Networking deployment using Red Hat Openstack version 16.2 (RHOSP16.2) as the orchestration platform is already operational.
- The overcloud nodes in the RHOSP16.2 environment have an enabled Red Hat Enterprise Linux (RHEL) subscription.
- Your environment is running Contrail Networking Release 21.4 and upgrading to Contrail Networking Release 21.4.L1 or later.

Before You Begin

Before starting the update, it is recommended to perform the following procedures.

- Backup your Contrail configuration database before starting this procedure. See How to Backup and Restore Contrail Databases in JSON Format in Openstack Environments Using the Openstack 16.1 Director Deployment.
- Each compute node agent will go down during this procedure, causing some compute node downtime. The estimated downtime for a compute node varies by environment, but typically took between 12 and 15 minutes in our testing environments.

  If you have compute nodes with workloads that cannot tolerate this downtime, consider migrating workloads or taking other steps to accommodate this downtime in your environment.

Updating Contrail Networking

To update the Contrail Networking version in an environment that is using Red Hat Openstack 16.2 without updating Red Hat OpenStack:

1. Register your nodes with the Contrail Networking Release 21.4 satellite key.
2. Prepare your container registry. The registry is often included in the undercloud, but it can also be a separate node.
3. Backup the Contrail TripleO Heat Templates. See Using the Contrail Heat Template.
4. Get the Contrail TripleO Heat Templates (stable/queens branch) from https://github.com/Juniper/contrail-tripleo-heat-templates.

Prepare the new tripleo-heat-templates with latest available software from Openstack and Contrail Networking.

```
sudo cp -r /usr/share/openstack-tripleo-heat-templates/ tripleo-heat-templates
sudo git clone https://github.com/tungstenfabric/tf-tripleo-heat-templates -b stable/queens
sudo cp -r tf-tripleo-heat-templates/* tripleo-heat-templates/
```

5. Update the parameter *ContrailImageTag* to the new version.

   The location of the *ContrailImageTag* variable varies by environment. In the most commonly-used environments, this variable is set in the *contrail-services.yaml* file.

   You can obtain the *ContrailImageTag* parameter from the README Access to Contrail Registry 21XX.

   NOTE: If you are using the undercloud as a registry, ensure the new contrail image is updated in undercloud before proceeding further.

6. Update the overcloud by entering the `openstack overcloud update prepare` command and include the files that were updated during the previous steps with the overcloud update.

   Example:

```
(overcloud) [stack@5cosp16 ~]$ cat prepare.sh
python3 ~/tripleo-heat-templates/tools/process-templates.py --clean -r ~/tripleo-heat-
templates/roles_data_contrail_aio.yaml -p ~/tripleo-heat-templates/
python3 ~/tripleo-heat-templates/tools/process-templates.py -r ~/tripleo-heat-templates/
roles_data_contrail_aio.yaml -p ~/tripleo-heat-templates/

openstack overcloud update prepare  --templates ~/tripleo-heat-templates/ --stack overcloud
--libvirt-type kvm --roles-file
~/tripleo-heat-templates/roles_data_contrail_aio.yaml -e ~/rhsm.yaml -e ~/tripleo-heat-
templates/hostname-map.yaml -e
~/tripleo-heat-templates/environments/network-isolation.yaml -e ~/tripleo-heat-templates/
environments/contrail/contrail-services.yaml -e
~/tripleo-heat-templates/environments/contrail/contrail-net.yaml -e ~/tripleo-heat-
templates/environments/contrail/contrail-plugins.yaml -e
~/tripleo-heat-templates/environments/contrail/contrail-tls.yaml -e ~/tripleo-heat-
templates/environments/ssl/tls-everywhere-endpoints-dns.yaml -e
~/tripleo-heat-templates/environments/services/haproxy-public-tls-certmonger.yaml -e
~/tripleo-heat-templates/environments/ssl/enable-internal-tls.yaml -e /home/stack/
containers-prepare-parameter.yaml
```

7. If you haven't already migrated the workloads, migrate the workloads now.
8. Upgrade the Contrail controllers:

```
openstack overcloud update run -limit contrail-controller-vm-name
```

Example:

```
openstack overcloud update run --limit overcloud-contrailcontroller-0-5c5s
openstack overcloud update run --limit overcloud-contrailcontroller-1-5c5s
openstack overcloud update run --limit overcloud-contrailcontroller-2-5c5s
```

9. Upgrade the Openstack controller node:

```
openstack overcloud update run -limit openstack-node-name
```

10. Upgrade the Contrail compute nodes:

```
openstack overcloud update run -limit contrail-compute-node-name
```

11. Enter the **openstack overcloud update converge** command.

Example:

```
(overcloud) [stack@5cosp16 ~]$ cat converge.sh
python3 ~/tripleo-heat-templates/tools/process-templates.py --clean -r ~/tripleo-heat-
templates/roles_data_contrail_aio.yaml -p ~/tripleo-heat-templates/
python3 ~/tripleo-heat-templates/tools/process-templates.py -r ~/tripleo-heat-templates/
roles_data_contrail_aio.yaml -p ~/tripleo-heat-templates/

openstack overcloud update converge --templates ~/tripleo-heat-templates/ --stack overcloud
--libvirt-type kvm --roles-file
~/tripleo-heat-templates/roles_data_contrail_aio.yaml -e ~/rhsm.yaml -e ~/tripleo-heat-
templates/hostname-map.yaml -e
~/tripleo-heat-templates/environments/network-isolation.yaml -e ~/tripleo-heat-templates/
environments/contrail/contrail-services.yaml -e
~/tripleo-heat-templates/environments/contrail/contrail-net.yaml -e ~/tripleo-heat-
templates/environments/contrail/contrail-plugins.yaml -e
~/tripleo-heat-templates/environments/contrail/contrail-tls.yaml -e ~/tripleo-heat-
templates/environments/ssl/tls-everywhere-endpoints-dns.yaml -e
~/tripleo-heat-templates/environments/services/haproxy-public-tls-certmonger.yaml -e
```

```
~/tripleo-heat-templates/environments/ssl/enable-internal-tls.yaml-e /home/stack/containers-
prepare-parameter.yaml
```

The Contrail containers should be up and running after this convergence operation is completed.

12. Confirm that the Contrail containers are up and running.

    There are many methods of performing this step. In this provided example, the **contrail-status** and **podman ps** commands are entered on a compute node to verify the Contrail containers on a compute node:

```
[heat-admin@overcloud-contraildpdk-0-new ~]$ sudo su
[root@overcloud-contraildpdk-0-new heat-admin]# contrail-status
Pod      Service      Original Name                Original Version    State
Id            Status
vrouter  agent        contrail-vrouter-agent       ubi-train-21.4-224  running
349cd9d6ed6b  Up 16 hours ago
vrouter  agent-dpdk   contrail-vrouter-agent-dpdk  ubi-train-21.4-224  running
cf73460aaa4f  Up 16 hours ago
vrouter  nodemgr      contrail-nodemgr             ubi-train-21.4-224  running
3820b61bc50a  Up 15 hours ago
vrouter  provisioner  contrail-provisioner         ubi-train-21.4-224  running
66cc9f00f912  Up 16 hours ago

vrouter DPDK module is PRESENT
== Contrail vrouter ==
nodemgr: active
agent: active

[root@overcloud-contraildpdk-0-new heat-admin]# podman ps
CONTAINER ID
IMAGE
                                  COMMAND               CREATED       STATUS
PORTS   NAMES
c2519bb82c66  5cosp16.ctlplane.5c5s.local:8787/5c5s45-satellite-rhosp-16-rhel-8-product-
containers-rhosp-rhel8_openstack-nova-libvirt:16.2  kolla_start           25 hours ago  Up
16 hours ago          nova_virtlogd
720ba7851089  5cosp16.ctlplane.5c5s.local:8787/5c5s45-satellite-rhosp-16-rhel-8-product-
containers-rhosp-rhel8_openstack-nova-libvirt:16.2  kolla_start           25 hours ago  Up
16 hours ago          nova_libvirt
48105e151d32  5cosp16.ctlplane.5c5s.local:8787/5c5s45-satellite-rhosp-16-rhel-8-product-
containers-rhosp-rhel8_openstack-iscsid:16.2        kolla_start           25 hours ago  Up
16 hours ago          iscsid
e6eb82a63826  5cosp16.ctlplane.5c5s.local:8787/5c5s45-satellite-rhosp-16-rhel-8-product-
```

```
containers-rhosp-rhel8_openstack-cron:16.2          kolla_start           25 hours ago  Up
16 hours ago          logrotate_crond
86d989d8cbc1  5cosp16.ctlplane.5c5s.local:8787/5c5s45-satellite-rhosp-16-rhel-8-product-
containers-rhosp-rhel8_openstack-nova-compute:16.2  kolla_start           25 hours ago  Up
16 hours ago          nova_migration_target
691b92fde276  5cosp16.ctlplane.5c5s.local:8787/5c5s45-satellite-rhosp-16-rhel-8-product-
containers-rhosp-rhel8_openstack-nova-compute:16.2  kolla_start           25 hours ago  Up
16 hours ago          nova_compute
349cd9d6ed6b  192.168.24.1:8787/contrail-nightly/contrail-vrouter-
agent:21.4.L1.224                                                       /usr/bin/
contrail...  17 hours ago  Up 16 hours ago         contrail_vrouter_agent
3820b61bc50a  192.168.24.1:8787/contrail-nightly/contrail-
nodemgr:21.4.L1.224                                                                 /bin/sh -
c /usr/b...  17 hours ago  Up 15 hours ago         contrail_vrouter_agent_nodemgr
66cc9f00f912  192.168.24.1:8787/contrail-nightly/contrail-
provisioner:21.4.L1.224                                                             /usr/bin/
tail -f ...  17 hours ago  Up 16 hours ago         contrail_vrouter_agent_provisioner
cf73460aaa4f  192.168.24.1:8787/contrail-nightly/contrail-vrouter-agent-
dpdk:21.4.L1.224                                                  /usr/bin/contrail...
16 hours ago  Up 16 hours ago          contrail-vrouter-agent-dpdk
[root@overcloud-contraildpdk-0-new heat-admin]#
```

RELATED DOCUMENTATION

Installing Contrail with OpenStack by Using Juju Charms | 596

Updating Contrail Networking using the Zero Impact Upgrade Process in an Environment using Red Hat Openstack 16.1

IN THIS SECTION

This document provides the steps needed to update a Contrail Networking deployment that is using Red Hat Openstack 16.1 as it's orchestration platform. The procedure provides a zero impact upgrade (ZIU) with minimal disruption to network operations.

If you are using Contrail Networking in an environment that is using a Red Hat Openstack 13-based release, see Updating Contrail Networking using the Zero Impact Upgrade Process in an Environment using Red Hat Openstack 13.

When to Use This Procedure

This procedure is used to upgrade Contrail Networking when it is running in environments using RHOSP version 16.1 (RHOSP16.1).

The procedure in this document has been validated for the following Contrail Networking upgrade scenarios:

Table 7: Contrail Networking with RHOSP16.1 Validated Upgrade Scenarios

Starting Contrail Networking Release	Target Upgraded Contrail Networking Release
2011.L1 or any later 2011.L release	2011.L2 or any later 2011.L release
2011.L1 or any later 2011.L release	Any 21.3 Release
Any 21.3 Release	Any 21.4 Release

A different procedure is followed for upgrading Contrail Networking in environments using Red Hat Openstack 13. See Updating Contrail Networking using the Zero Impact Upgrade Process in an Environment using Red Hat Openstack 13.

Prerequisites

This document makes the following assumptions about your environment:

- A Contrail Networking deployment using Red Hat Openstack version 16.1 (RHOSP16.1) as the orchestration platform is already operational.
- The overcloud nodes in the RHOSP16.1 environment have an enabled Red Hat Enterprise Linux (RHEL) subscription.
- Your environment is running Contrail Release 2011.L1 and upgrading to Contrail Release 2011.L2 or later.

- If you are updating Red Hat Openstack simultaneously with Contrail Networking, we assume that the undercloud node is updated to the latest minor version and that new overcloud images are prepared for an upgrade. See the Upgrading the Undercloud section of the Keeping Red Hat OpenStack Platform Updated guide from Red Hat.

  If the undercloud has been updated and a copy of the heat templates are used for the deployment, update the copy of the heat template from the Red Hat's core heat template collection at `/usr/share/openstack-tripleo-heat-templates`. See the Understanding Heat Templates document from Red Hat for information on this process.

Before You Begin

We recommend performing these procedures before starting the update:

- Backup your Contrail configuration database before starting this procedure. See How to Backup and Restore Contrail Databases in JSON Format in Openstack Environments Using the Openstack 16.1 Director Deployment.

- Each compute node agent will go down during this procedure, causing some compute node downtime. The estimated downtime for a compute node varies by environment, but typically took between 12 and 15 minutes in our testing environments.

  If you have compute nodes with workloads that cannot tolerate this downtime, consider migrating workloads or taking other steps to accommodate this downtime in your environment.

Updating Contrail Networking in an Environment using Red Hat Openstack 16.1

To update Contrail Networking in an environment that is using Red Hat Openstack 16.1 as the orchestration platform:

1. Prepare your container registry. The registry is often included in the undercloud, but it can also be a separate node.
2. Backup the Contrail TripleO Heat Templates. See Using the Contrail Heat Template.
3. Get the Contrail TripleO Heat Templates (Stable/Train branch) from https://github.com/Juniper/contrail-tripleo-heat-templates.

   Prepare the new tripleo-heat-templates with latest available software from Openstack and Contrail Networking.

   ```
   sudo cp -r /usr/share/openstack-tripleo-heat-templates/ tripleo-heat-templates
   sudo git clone https://github.com/tungstenfabric/tf-tripleo-heat-templates -b stable/train
   sudo cp -r tf-tripleo-heat-templates/* tripleo-heat-templates/
   ```

4. Update the parameter *ContrailImageTag* to the new version.

The location of the *ContrailImageTag* variable varies by environment. In the most commonly-used environments, this variable is set in the *contrail-services.yaml* file.

You can obtain the *ContrailImageTag* parameter from the README Access to Contrail Registry 21XX.

NOTE: If you are using the undercloud as a registry, ensure the new contrail image is updated in undercloud before proceeding further.

5. Update the overcloud by entering the `openstack overcloud update prepare` command and include the files that were updated during the previous steps with the overcloud update.

   Example:

```
(undercloud) [stack@uc-train ~]$ cat deploy-ipu-prepare.sh
python3 tripleo-heat-templates/tools/process-templates.py --clean \
-r roles_data_contrail_aio.yaml -p tripleo-heat-templates/
python3 tripleo-heat-templates/tools/process-templates.py \
-r roles_data_contrail_aio.yaml \
-p tripleo-heat-templates/
openstack overcloud update prepare --templates tripleo-heat-templates/ \
--stack overcloud --libvirt-type kvm \
--roles-file roles_data_contrail_aio.yaml \
-e ~/rhsm.yaml \
-e tripleo-heat-templates/environments/network-isolation.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e tripleo-heat-templates /environments/contrail/contrail-net.yaml \
-e tripleo-heat-templates /environments/contrail/contrail-plugins.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-tls.yaml \
-e tripleo-heat-templates /environments/ssl/tls-everywhere-endpoints-dns.yaml \
-e tripleo-heat-templates /environments/services/haproxy-public-tls-certmonger.yaml \
-e tripleo-heat-templates /environments/ssl/enable-internal-tls.yaml \
-e containers-prepare-parameter.yaml
```

6. Prepare the overcloud nodes that include Contrail containers for the update.

   - Pull the images in the repository onto the overcloud nodes.

     There are multiple methods for performing this step. Commonly used methods for performing this operation include using the `podman pull` command for Docker containers and the `openstack overcloud container image upload` command for Openstack containers, or running the `tripleo-heat-templates/upload.containers.sh` and `tools/contrail/update_contrail_preparation.sh` scripts.

- (Not required in all setups) Provide export variables for the script if the predefined values aren't appropriate for your environment. The script location:

```
~/tripleo-heat-templates/tools/contrail/update_contrail_preparation.sh
```

The following variables within the script are particularly significant for this upgrade:

- CONTRAIL_NEW_IMAGE_TAG—The image tag of the target upgrade version of Contrail. The default value is latest.

  If needed, you can obtain this parameter for a specific image from the README Access to Contrail Registry 21XX.

  NOTE: Some older deployments use the CONTRAIL_IMAGE_TAG variable in place of the CONTRAIL_NEW_IMAGE_TAG variable. Both variables are recognized by the update_contrail_preparation.sh script and perform the same function.

- SSH_USER—The SSH username for logging into overcloud nodes. The default value is heat-admin.

- SSH_OPTIONS—Custom SSH option values.

  The default SSH options for your environment are typically pre-defined. You are typically only changing this value if you want to customize your update.

- STOP_CONTAINERS—The list of containers that must be stopped before the upgrade can proceed. The default value is contrail_config_api contrail_analytics_api.

- Run the script:

  CAUTION: Contrail services stop working when the script starts running.

```
~/tripleo-heat-templates/tools/contrail/update_contrail_preparation.sh
```

7. Update the Contrail Controller nodes:

   - Run the openstack overcloud update run command on the first Contrail controller and, if needed, on a Contrail Analytics node. The purpose of this step is to update one Contrail Controller and one Contrail Analytics node to support the environment so the other Contrail Controllers and analytics nodes can be updated without incurring additional downtime.

Example:

```
openstack overcloud update run --limit overcloud-contrailcontroller-0
```

Ensure that the contrail status is `ok` on overcloud-contrailcontroller-0 before proceeding.

If the analytics and the analyticsdb nodes are on separate nodes, you may have to update the nodes individually:

```
openstack overcloud update run --limit overcloud-contrailcontroller-0
openstack overcloud update run --roles
ContrailAnalytics,ContrailAnalyticsDatabase
```

- After the upgrade, check the docker container status and versions for the Contrail Controllers and the Contrail Analytics and AnalyticsDB nodes.

```
podman ps -a
```

- Update the remaining Contrail Controller nodes:

  Example:

```
openstack overcloud update run --limit overcloud-contrailcontroller-1
openstack overcloud update run --limit overcloud-contrailcontroller-2
openstack overcloud update run --limit overcloud-contrailcontroller-3
...
```

8. Update the Openstack Controllers using the `openstack overcloud update run` commands:

   Example:

```
openstack overcloud update run --limit overcloud-controller-0
openstack overcloud update run --limit overcloud-controller-1
openstack overcloud update run --limit overcloud-controller-2
...
```

9. Individually update the compute nodes.

NOTE: The compute node agent will be down during this step. The estimated downtime varies by environment, but is typically between 1 and 5 minutes. Consider migrating workloads that can't tolerate this downtime before performing this step

```
openstack overcloud update run --limit overcloud-novacompute-1
openstack overcloud update run --limit overcloud-novacompute-2
openstack overcloud update run --limit overcloud-novacompute-3
...
```

Reboot your compute node to complete the update.

NOTE: A reboot is required to complete this procedure only if a kernel update is also needed. If you would like to avoid rebooting your compute node, check the log files in the `/var/log/yum.log` file to see if kernel packages were updated during the compute node update. A reboot is required only if kernel updates occurred as part of the compute node update procedure.

```
sudo reboot
```

Use the `contrail-status` command to monitor upgrade status. Ensure all pods reach the *running* state and all services reach the *active* state.

This `contrail-status` command provides output after a successful upgrade:

NOTE: Some output fields and data have been removed from this `contrail-status` command sample for readability.

```
Pod             Service       Original Name                   State
analytics       api           contrail-analytics-api          running
analytics       collector     contrail-analytics-collector    running
analytics       nodemgr       contrail-nodemgr                running
analytics       provisioner   contrail-provisioner            running
analytics       redis         contrail-external-redis         running
analytics-alarm alarm-gen     contrail-analytics-alarm-gen    running
```

```
analytics-alarm kafka          contrail-external-kafka           running
analytics-alarm nodemgr        contrail-nodemgr                  running
analytics-alarm provisioner    contrail-provisioner              running
analytics-alarm zookeeper      contrail-external-zookeeper       running
analytics-snmp  nodemgr        contrail-nodemgr                  running
analytics-snmp  provisioner    contrail-provisioner              running
analytics-snmp  snmp-collector contrail-analytics-snmp-collector running
analytics-snmp  topology       contrail-analytics-snmp-topology  running
config          api            contrail-controller-config-api    running
<trimmed>

== Contrail control ==
control: active
nodemgr: active
named: active
dns: active

== Contrail analytics-alarm ==
nodemgr: active
kafka: active
alarm-gen: active

== Contrail database ==
nodemgr: active
query-engine: active
cassandra: active

== Contrail analytics ==
nodemgr: active
api: active
collector: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
cassandra: active

== Contrail webui ==
web: active
job: active

== Contrail analytics-snmp ==
```

```
snmp-collector: active
nodemgr: active
topology: active

== Contrail config ==
svc-monitor: active
nodemgr: active
device-manager: active
api: active
schema: active
```

10. Enter the `openstack overcloud update converge` command to finalize the update.

NOTE: The options used in the `openstack overcloud update converge` in this step will match the options used with the `openstack overcloud update prepare` command entered in 5.

```
(undercloud) [stack@uc-train ~]$ cat deploy-ipu-converge.sh
python3 tripleo-heat-templates/tools/process-templates.py --clean \
-r roles_data_contrail_aio.yaml -p tripleo-heat-templates/
python3 tripleo-heat-templates/tools/process-templates.py \
-r roles_data_contrail_aio.yaml \
-p tripleo-heat-templates/
openstack overcloud update converge --templates tripleo-heat-templates/ \
--stack overcloud --libvirt-type kvm \
--roles-file roles_data_contrail_aio.yaml \
-e ~/rhsm.yaml \
-e tripleo-heat-templates/environments/network-isolation.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e tripleo-heat-templates /environments/contrail/contrail-net.yaml \
-e tripleo-heat-templates /environments/contrail/contrail-plugins.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-tls.yaml \
-e tripleo-heat-templates /environments/ssl/tls-everywhere-endpoints-dns.yaml \
-e tripleo-heat-templates /environments/services/haproxy-public-tls-certmonger.yaml \
-e tripleo-heat-templates /environments/ssl/enable-internal-tls.yaml \
-e containers-prepare-parameter.yaml
```

Monitor screen messages indicating *SUCCESS* to confirm that the updates made in this step are successful.

RELATED DOCUMENTATION

Installing Contrail with OpenStack by Using Juju Charms | 596

Updating Contrail Networking using Zero Impact Upgrade Procedure in a Canonical Openstack Multi-model Deployment with Juju Charms

IN THIS SECTION

This document provides the steps needed to update a Contrail Networking deployment that is using Canonical Openstack Multi-model as it's orchestration platform. The procedure utilizes Juju charms and provides a zero impact upgrade (ZIU) with minimal disruption to network operations.

Prerequisites

This document makes the following assumptions about your environment:

- A Contrail Networking deployment using Canonical Openstack Multi-model as the orchestration platform is already operational.
- Juju charms for Contrail services are active in your environment, and the Contrail Networking controller has access to the Juju jumphost and the Juju cluster.

When to Use This Procedure

This procedure is used to upgrade Contrail Networking when it is running in environments using Canonical Openstack Multi-model.

You can use this procedure to incrementally upgrade to the next main Contrail Networking release only. This procedure is not validated for upgrades between releases that are two or more releases apart.

The procedure in this document has been validated for the following Contrail Networking upgrade scenarios:

Table 8: Contrail Networking with Canonical Openstack Multi-model Validated Upgrade Scenarios

Starting Contrail Networking Release	Target Upgraded Contrail Networking Release
21.4.L2 Release	21.4.L3 Release

Recommendations

We strongly recommend performing the following tasks before starting this procedure:

- Backup your current environment.
- Enable huge pages for the kernel-mode vRouter.

  Huge pages in Contrail Networking are used primarily to allocate flow and bridge table memory within the vRouter. Huge pages for kernel-mode vRouters provide enough flow and bridge table memory to avoid compute node reboots to complete future Contrail Networking software upgrades.

  We recommend allotting 2GB of memory—either using the default 1024x2MB huge page size setting or the 2x1GB size setting—for huge pages. Other huge page size settings should only be set by expert users in specialized circumstances.

  A compute node reboot is required to initially enable huge pages. Future compute node upgrades can happen without reboots after huge pages are enabled. The 1024x2MB huge page setting is configured by default starting in Contrail Networking Release 2005, but is not active in any compute node until the compute node is rebooted to enable the setting.

  2GB and 1MB huge page size settings cannot be enabled simultaneously in environments using Juju.

  The huge page configurations can be changed by entering one of the following commands:

  - Enable 1024 2MB huge pages (Default): **juju config contrail-agent kernel-hugepages-2m=1024**

    Disable 2MB huge pages (empty value): **juju config contrail-agent kernel-hugepages-2m=“”**

  - Enable 2 1GB huge pages: **juju config contrail-agent kernel-hugepages-1g=2**

    Disable 1GB huge pages (default. empty value): **juju config contrail-agent kernel-hugepages-1g=“”**

    NOTE: 1GB huge page settings can only be specified at initial deployment; you cannot modify the setting in active deployments. The 1GB huge page setting can also not be completely disabled after being activated on a compute node. Be sure that you want to use 1GB huge page settings on your compute node before enabling the setting.

Updating Contrail Networking in a Canonical Openstack Multi-model Deployment Using Juju Charms

Before updating Contrail Networking in a Canonical Openstack Multi-model Deployment, you are required to prepare the multi-model setup.

Preparing Multi-model Setup

You can initiate the creation of multi-model setup during Juju bootstrap and specify the multi-model name with --add-model parameter. Let the initial model be called as 'openstack'. Perform the following steps to prepare the multi-model setup:

1. Add 'openstack' model for Kubernetes.

```
juju add-model openstack
```

2. Deploy Contrail-Openstack bundle for openstack model.

```
juju -m openstack deploy ./openstack_bundle.yaml
```

3. Add 'kubernetes' model for Kubernetes.

```
juju add-model k8s
```

4. Deploy k8s bundle.

```
juju -m k8s deploy ./k8s_bundle.yaml
```

5. Add offer for contrail-controller and keystone (if you are using keystone authorization for k8s).

```
juju -m k8s offer openstack.tf-controller:contrail-controller
juju -m k8s offer openstack.keystone:identity-credentials
```

If easyrsa is implemented in Openstack model, then run the following command:

```
juju -m k8s offer openstack.easyrsa:client
```

6. Add relations to contrail-openstack.

```
juju -m k8s add-relation tf-kubernetes-master openstack.tf-controller
juju -m k8s add-relation tf-agent openstack.tf-controller
juju -m k8s add-relation kubernetes-master openstack.keystone
```

If easyrsa is implemented in Openstack model, then run the following commands:

```
juju -m k8s add-relation tf-kubernetes-master openstack.easyrsa
juju -m k8s add-relation tf-agent openstack.easyrsa
juju -m k8s add-relation kubernetes-master openstack.easyrsa
juju -m k8s add-relation etcd openstack.easyrsa
```

Zero Impact Upgrade Procedure for Multi-model Deployment

To update Contrail Networking in an environment that is using Canonical Openstack Multi-model as the orchestration platform:

1. From the Juju jumphost, enter the `run-action` command to place all control plane services—Contrail Controller, Contrail Analytics, and Contrail AnalyticsDB—into maintenance mode in preparation for the upgrade.

```
juju run-action contrail-controller/leader upgrade-ziu
```

> **NOTE**: The **--wait** option is not required to complete this step, but is recommended to ensure this procedure completes without interfering with the procedures in the next step.

Wait for all charms to move to the *maintenance* status. You can check the status of all charms by entering the **juju status** command.

2. Upgrade all charms. See the Upgrading applications document from Juju.
3. Update the image tags on both modals with new-tag in Juju. The update applies to Contrail Analytics, Contrail AnalyticsDB, Contrail Agent, Contrail Openstack and Contrail Controller services.

```
juju switch openstack
juju config contrail-analytics image-tag=<new tag>
juju config contrail-analyticsdb image-tag=<new tag>
```

```
juju config contrail-agent image-tag=<new tag>
juju config contrail-openstack image-tag=<new tag>
juju config contrail-controller image-tag=<new tag>

juju switch k8s
juju config contrail-kubernetes-master image-tag=<new tag>
juju config contrail-kubernetes-node image-tag=<new tag>
juju config contrail-agent image-tag=<new tag>
```

The upgrade process starts. It automatically passes six stages.

```
Stage 0. Pull new images.
Stage 1. Stops config containers.
Stage 2. Start config containers with new tag.
Stage 3. Restart control containers one-by-one.
Stage 4. Restart database containers one-by-one.
Stage 5. Restart contrail-agent container.
```

4. After updating the image tags, wait for all services to complete stage 5 of the ZIU upgrade process workflow. The wait time for this step varies by environment, but often takes 40 minutes.

   Enter the `juju status` command and review the **Workload** and **Message** field outputs to monitor progress. The update is complete when all services are in the maintenance state—the **Workload** field output is `maintenance`—and each individual service has completed stage 5 of the ZIU upgrade—illustrated by the `ziu is in progress - stage/done = 5/5` output in the **Message** field.

   A sample output of an in-progress update that has not completed the image tag update process. The **Message** field illustrates that the ZIU processes have not completed stage 5 of the upgrade.

```
juju status
Unit                      Workload     Agent      Machine  Public address
Ports                     Message
contrail-analytics/0*     maintenance  idle       3        10.0.12.20      8081/
tcp                  ziu is in progress - stage/done = 4/4
contrail-analytics/1      maintenance  idle       4        10.0.12.21      8081/
tcp                  ziu is in progress - stage/done = 4/4
contrail-analytics/2      maintenance  idle       5        10.0.12.22      8081/
tcp                  ziu is in progress - stage/done = 4/4
contrail-analyticsdb/0*   maintenance  idle       3
10.0.12.20                               ziu is in progress - stage/done = 4/4
contrail-analyticsdb/1    maintenance  idle       4
10.0.12.21                               ziu is in progress - stage/done = 4/3
contrail-analyticsdb/2    maintenance  idle       5
```

```
10.0.12.22                              ziu is in progress - stage/done = 4/3
contrail-controller/0*     maintenance  idle       3
10.0.12.20                              ziu is in progress - stage/done = 4/4
  ntp/3                    active       idle               10.0.12.20      123/
udp                   chrony: Ready
contrail-controller/1      maintenance  executing  4
10.0.12.21                              ziu is in progress - stage/done = 4/3
  ntp/2                    active       idle               10.0.12.21      123/
udp                   chrony: Ready
contrail-controller/2      maintenance  idle       5
10.0.12.22                              ziu is in progress - stage/done = 4/3
  ntp/4                    active       idle               10.0.12.22      123/
udp                   chrony: Ready
contrail-keystone-auth/0*  active       idle       3/lxd/0
10.0.12.121                             Unit is ready
contrail-keystone-auth/0* active      idle      Unit is ready
```

5. Upgrade every Contrail agent on each individual compute node:

```
juju run-action contrail-agent/0 upgrade
juju run-action contrail-agent/1 upgrade
```

Wait for each compute node and CSN node upgrade to finish. The wait time for this step varies by environment, but typically takes around 5 minutes to complete per node.

6. If huge pages are not enabled for your vRouter, log into each individual compute node and reboot to complete the procedure.

NOTE: A compute node reboot is required to initially enable huge pages. If huge pages have been configured in Juju without a compute node reboot, you can also use this reboot to enable huge pages. You can avoid rebooting the compute node during future software upgrades after this initial reboot.

1024x2MB huge page support is configured by default starting in Contrail Networking Release 2005, which is also the first Contrail Networking release that supports huge pages. If you are upgrading to Release 2005 for the first time, a compute node reboot is always required because huge pages could not have been previously enabled.

This reboot also enables the default 1024x2MB huge page configuration unless you change the huge page configuration in Release 2005 or later.

```
sudo reboot
```

This step can be skipped if huge pages are enabled.

RELATED DOCUMENTATION

Installing Contrail with OpenStack by Using Juju Charms | 596

Updating Contrail Networking using Zero Impact Upgrade Procedure in a Canonical Openstack Deployment with Juju Charms

IN THIS SECTION

This document provides the steps needed to update a Contrail Networking deployment that is using Canonical Openstack as it's orchestration platform. The procedure utilizes Juju charms and provides a zero impact upgrade (ZIU) with minimal disruption to network operations.

Prerequisites

This document makes the following assumptions about your environment:

- A Contrail Networking deployment using Canonical Openstack as the orchestration platform is already operational.
- Juju charms for Contrail services are active in your environment, and the Contrail Networking controller has access to the Juju jumphost and the Juju cluster.

When to Use This Procedure

This procedure is used to upgrade Contrail Networking when it is running in environments using Canonical Openstack.

You can use this procedure to incrementally upgrade to the next main Contrail Networking release only. This procedure is not validated for upgrades between releases that are two or more releases apart.

The procedure in this document has been validated for the following Contrail Networking upgrade scenarios:

Table 9: Contrail Networking with Canonical Openstack Validated Upgrade Scenarios

Starting Contrail Networking Release	Target Upgraded Contrail Networking Release
Any 1912.L Release	Any 1912.L Release
1912 or any 1912.L Release	2003
2003	2005
2005	2008
2008	2011
2011	Any 2011.L Release
Any 2011.L Release	Any 2011.L Release
Any 2011 or 2011.L Release	Any 21.3 Release
Any 21.3 Release	Any 21.4 Release
2011 Release	21.4.L2 Release
2011 Release	21.4.L3 Release
2011.L5 Release	Any 21.4 Release

Recommendations

We strongly recommend performing the following tasks before starting this procedure:

- Backup your current environment.
- Enable huge pages for the kernel-mode vRouter.

  Starting in Contrail Networking Release 2005, you can enable huge pages in the kernel-mode vRouter to avoid future compute node reboots during upgrades. Huge pages in Contrail Networking are used primarily to allocate flow and bridge table memory within the vRouter. Huge pages for kernel-mode vRouters provide enough flow and bridge table memory to avoid compute node reboots to complete future Contrail Networking software upgrades.

  We recommend allotting 2GB of memory—either using the default 1024x2MB huge page size setting or the 2x1GB size setting—for huge pages. Other huge page size settings should only be set by expert users in specialized circumstances.

  A compute node reboot is required to initially enable huge pages. Future compute node upgrades can happen without reboots after huge pages are enabled. The 1024x2MB huge page setting is configured by default starting in Contrail Networking Release 2005, but is not active in any compute node until the compute node is rebooted to enable the setting.

  2GB and 1MB huge page size settings cannot be enabled simultaneously in environments using Juju.

  The huge page configurations can be changed by entering one of the following commands:

  - Enable 1024 2MB huge pages (Default): **juju config contrail-agent kernel-hugepages-2m=1024**

    Disable 2MB huge pages (empty value): **juju config contrail-agent kernel-hugepages-2m=““**

  - Enable 2 1GB huge pages: **juju config contrail-agent kernel-hugepages-1g=2**

    Disable 1GB huge pages (default. empty value): **juju config contrail-agent kernel-hugepages-1g=““**

    NOTE: 1GB huge page settings can only be specified at initial deployment; you cannot modify the setting in active deployments. The 1GB huge page setting can also not be completely disabled after being activated on a compute node. Be sure that you want to use 1GB huge page settings on your compute node before enabling the setting.

Updating Contrail Networking in a Canonical Openstack Deployment Using Juju Charms

To update Contrail Networking in an environment that is using Canonical Openstack as the orchestration platform:

1. From the Juju jumphost, enter the `run-action` command to place all control plane services—Contrail Controller, Contrail Analytics, & Contrail AnalyticsDB—into maintenance mode in preparation for the upgrade.

```
juju run-action --wait contrail-controller/leader upgrade-ziu
```

> **NOTE**: The **--wait** option is not required to complete this step, but is recommended to ensure this procedure completes without interfering with the procedures in the next step.

   Wait for all charms to move to the *maintenance* status. You can check the status of all charms by entering the **juju status** command.

2. Upgrade all charms. See the Upgrade Juju document from Juju.
3. Update the image tags in Juju for the Contrail Analytics, Contrail AnalyticsDB, Contrail Agent, and Contrail Openstack services.

```
juju config contrail-analytics image-tag=master-latest
juju config contrail-analyticsdb image-tag=master-latest
juju config contrail-agent image-tag=master-latest
juju config contrail-openstack image-tag=master-latest
```

   If a Contrail Service node (CSN) is part of the cluster, also update the image tags in Juju for the Contrail Service node.

```
juju config contrail-agent-csn image-tag=master-latest
```

4. Update the image tag in Juju for the Contrail Controller service:

```
juju config contrail-controller image-tag=master-latest
```

5. After updating the image tags, wait for all services to complete stage 5 of the ZIU upgrade process workflow. The wait time for this step varies by environment, but often takes 30 to 90 minutes.

   Enter the `juju status` command and review the **Workload** and **Message** field outputs to monitor progress. The update is complete when all services are in the maintenance state—the **Workload** field output is `maintenance`—and each individual service has completed stage 5 of the ZIU upgrade—illustrated by the `ziu is in progress - stage/done = 5/5` output in the **Message** field.

A sample output of an in-progress update that has not completed the image tag update process. The **Message** field illustrates that the ZIU processes have not completed stage 5 of the upgrade.

NOTE: Some **juju status** output fields removed for readability.

```
juju status
Unit                      Workload    Agent     Message
contrail-analytics/0*     maintenance idle      ziu is in progress - stage/done = 4/4
contrail-analytics/1      maintenance idle      ziu is in progress - stage/done = 4/4
contrail-analytics/2      maintenance idle      ziu is in progress - stage/done = 4/4
contrail-analyticsdb/0*   maintenance idle      ziu is in progress - stage/done = 4/4
contrail-analyticsdb/1    maintenance idle      ziu is in progress - stage/done = 4/3
contrail-analyticsdb/2    maintenance idle      ziu is in progress - stage/done = 4/3
contrail-controller/0*    maintenance idle      ziu is in progress - stage/done = 4/4
  ntp/3                   active      idle      chrony: Ready
contrail-controller/1     maintenance executing ziu is in progress - stage/done = 4/3
  ntp/2                   active      idle      chrony: Ready
contrail-controller/2     maintenance idle      ziu is in progress - stage/done = 4/3
  ntp/4                   active      idle      chrony: Ready
contrail-keystone-auth/0* active      idle      Unit is ready
```

A sample output of an update that has completed the image tag update process on all services. The **Workload field** is `maintenance` for all services and the **Message** field explains that stage 5 of the ZIU process is done.

NOTE: Some **juju status** output fields removed for readability.

```
juju status
Unit                      Workload     Agent Message
contrail-analytics/0*     maintenance  idle  ziu is in progress - stage/done = 5/5
contrail-analytics/1      maintenance  idle  ziu is in progress - stage/done = 5/5
contrail-analytics/2      maintenance  idle  ziu is in progress - stage/done = 5/5
contrail-analyticsdb/0*   maintenance  idle  ziu is in progress - stage/done = 5/5
contrail-analyticsdb/1    maintenance  idle  ziu is in progress - stage/done = 5/5
contrail-analyticsdb/2    maintenance  idle  ziu is in progress - stage/done = 5/5
contrail-controller/0*    maintenance  idle  ziu is in progress - stage/done = 5/5
  ntp/3                   active       idle  chrony: Ready
contrail-controller/1     maintenance  idle  ziu is in progress - stage/done = 5/5
  ntp/2                   active       idle  chrony: Ready
```

```
contrail-controller/2     maintenance  idle  ziu is in progress - stage/done = 5/5
  ntp/4                   active       idle  chrony: Ready
contrail-keystone-auth/0* active       idle  Unit is ready
glance/0*                 active       idle  Unit is ready
haproxy/0*                active       idle  Unit is ready
  keepalived/2            active       idle  VIP ready
haproxy/1                 active       idle  Unit is ready
  keepalived/0*           active       idle  VIP ready
haproxy/2                 active       idle  Unit is ready
  keepalived/1            active       idle  VIP ready
heat/0*                   active       idle  Unit is ready
  contrail-openstack/3    active       idle  Unit is ready
keystone/0*               active       idle  Unit is ready
mysql/0*                  active       idle  Unit is ready
neutron-api/0*            active       idle  Unit is ready
  contrail-openstack/2    active       idle  Unit is ready
nova-cloud-controller/0*  active       idle  Unit is ready
nova-compute/0*           active       idle  Unit is ready
```

6. Upgrade every Contrail agent on each individual compute node:

```
juju run-action contrail-agent/0 upgrade
juju run-action contrail-agent/1 upgrade
juju run-action contrail-agent/2 upgrade
...
```

If Contrail Service nodes (CSNs) are part of the cluster, also upgrade every Contrail CSN agent:

```
juju run-action contrail-agent-csn/0 upgrade
...
```

Wait for each compute node and CSN node upgrade to finish. The wait time for this step varies by environment, but typically takes around 10 minutes to complete per node.

7. If huge pages are not enabled for your vRouter, log into each individual compute node and reboot to complete the procedure.

NOTE: A compute node reboot is required to initially enable huge pages. If huge pages have been configured in Juju without a compute node reboot, you can also use this

reboot to enable huge pages. You can avoid rebooting the compute node during future software upgrades after this initial reboot.

1024x2MB huge page support is configured by default starting in Contrail Networking Release 2005, which is also the first Contrail Networking release that supports huge pages. If you are upgrading to Release 2005 for the first time, a compute node reboot is always required because huge pages could not have been previously enabled.

This reboot also enables the default 1024x2MB huge page configuration unless you change the huge page configuration in Release 2005 or later.

```
sudo reboot
```

This step can be skipped if huge pages are enabled.

RELATED DOCUMENTATION

Installing Contrail with OpenStack by Using Juju Charms | 596

Upgrading Contrail Networking Release 1912.L2 with RHOSP13 to Contrail Networking Release 2011.L3 with RHOSP16.1

The goal of this topic is to provide a combined procedure to upgrade Red Hat OpenStack Platform (RHOSP) from RHOSP 13 to RHOSP 16.1 by leveraging Red Hat Fast Forward Upgrade (FFU) procedure while simultaneously upgrading Contrail Networking from Release 1912.L2 to Release 2011.L3. This procedure leverages the speeding up an overcloud upgrade process from RHOSP.

Follow *chapter 2—Planning and preparation for an in-place upgrade through chapter 8.3— Copying the Leapp data to the overcloud nodes* of FRAMEWORK FOR UPGRADES (13 TO 16.1) procedure.

Before upgrading overcloud, refer Chapter 20—Speeding Up An Overcloud Upgrade.

To upgrade an overcloud:

1. Create a Contrail container file to upload the Contrail container image to undercloud, if the undercloud has used a registry.

```
cd ~/tf-heat-templates/tools/contrail
./import_contrail_container.sh -f container_outputfile -r registry -t tag [-i insecure] [-u
username] [-p password] [-c certificate path]
```

For example:

```
Import from password protected public registry:
./import_contrail_container.sh -f /tmp/contrail_container -r hub.juniper.net/contrail -u
USERNAME -p PASSWORD -t 1234
```

```
Import from private secure registry:
./import_contrail_container.sh -f /tmp/contrail_container -r device.example.net:5443 -c
http://device.example.net/pub/device.example.net.crt -t 1234
```

```
Import from private insecure registry:
./import_contrail_container.sh -f /tmp/contrail_container -r 10.0.0.1:5443 -i 1 -t 1234
```

Run the following command from the undercloud to upload the Contrail container image.

```
openstack overcloud container image upload --config-file /tmp/contrail_container
```

2. Clone the tungsten fabric deployment code.

```
git clone https://github.com/tungstenfabric/tf-deployment-test.git
```

NOTE: Tungsten fabric scripts are located at **- tf-deployment-test/tree/master/rhosp/ffu_ziu_13_16/tf_specific**.

3. Follow *chapter 2—Planning and preparation for an in-place upgrade through chapter 8.3— Copying the Leapp data to the overcloud nodes* of FRAMEWORK FOR UPGRADES (13 TO 16.1) procedure to upgrade the undercloud and prepare the environment.
4. Use predictable NIC names for overcloud nodes.

   Follow Chapter 8.4—Using predictable NIC names for overcloud nodes.

```
ansible-playbook -i inventory.yaml tf-deployment-test/rhosp/ffu_ziu_13_16/redhat_files/
playbook-nics.yaml
ansible-playbook -i inventory.yaml tf-deployment-test/rhosp/ffu_ziu_13_16/redhat_files/
playbook-nics.yaml
```

Follow *Chapter 20.1—Running the overcloud upgrade preparation and Chapter 20.2—Upgrading the control plane nodes* of RedHat OpenStack Speeding Up an Overcloud Upgrade procedure to upgrade openstack and other overcloud nodes including contrail controllers.

5. Run the `playbook-nics-vhost0.yaml` playbook to setup the vhost interface on Contrail compute nodes.

```
ansible-playbook -i inventory.yaml -l overcloud_Compute tf-deployment-test/rhosp/
ffu_ziu_13_16/tf_specific/playbook-nics-vhost0.yaml

ansible-playbook -i inventory.yaml -l overcloud_ContrailDpdk tf-deployment-test/rhosp/
ffu_ziu_13_16/tf_specific/playbook-nics-vhost0.yaml
```

> **NOTE**: Stop or migrate the workloads running on the compute batch that you are going to upgrade.

Follow *Chapter 20.3—Upgrading Compute nodes in parallel* of RedHat OpenStack Speeding Up an Overcloud Upgrade procedure.

Instead of executing a single command for compute system upgrade, you can use a separate upgrade command with the `system_upgrade_prepare` and `system_upgrade_run` tags.

```
openstack overcloud upgrade run --yes --stack overcloud --tags system_upgrade_prepare --limit
<name-of-compute>

openstack overcloud upgrade run --yes --stack overcloud --tags system_upgrade_run --limit
<name-of-compute>
```

6. Follow *Chapter 20.4—Synchronizing the overcloud stack* and later chapters of RedHat OpenStack Speeding Up an Overcloud Upgrade procedure to complete the overcloud upgrade.

Upgrading Contrail Networking Release 1912.L4 or 2011.L3 with RHOSP 13 or RHOSP 16.1 to Contrail Networking Release 21.4 with RHOSP 16.2

IN THIS SECTION

The goal of this topic is to provide a combined procedure to upgrade Red Hat OpenStack Platform (RHOSP) from RHOSP 13 or RHOSP 16.1 to RHOSP 16.2 by leveraging Red Hat Fast Forward Upgrade (FFU) procedure while simultaneously upgrading Contrail Networking from Release 1912.L4 or 2011.L3 to Release 21.4.

The downtime will be reduced by not requiring extra server reboots in addition to the ones that the RHOSP FFU procedure already requires for Kernel/RHEL upgrades.

Before upgrading overcloud, refer Chapter 20—Speeding up an overcloud upgrade.

Refer to Framework for Upgrades (13 to 16.2) documentation for details on RHOSP 13 to RHOSP 16.2 Fast Forward Upgrade (FFU) procedure of OpenStack Platform environment from one long life version to the next long life version.

Refer to Keeping Red Hat OpenStack Platform Updated documentation for details on RHOSP 16.1 to RHOSP 16.2 to perform minor updates of Red Hat OpenStack Platform.

When to Use This Procedure

The procedure in this document has been validated for the following Contrail Networking upgrade scenarios:

Table 10: Validated Upgrade Scenarios

Current Version	Target Version
RHOSP 13	RHOSP 16.2
RHOSP 16.1	RHOSP 16.2
Contrail Networking Release 1912.L4	Contrail Networking Release 21.4
Contrail Networking Release 2011.L3	Contrail Networking Release 21.4

Prerequisites

This document makes the following assumptions about your environment:

- A Contrail Networking deployment using Red Hat OpenStack version 13 (RHOSP 13) or RHOSP 16.1 as the orchestration platform is already operational.
- The overcloud nodes in the RHOSP 13 or RHOSP 16.1 environment have an enabled Red Hat Enterprise Linux (RHEL) subscription.
- Your environment is running Contrail Release 1912.L4 or 2011.L3 and upgrading to Contrail Release 21.4.
- If you are updating Red Hat OpenStack simultaneously with Contrail Networking, we assume that the undercloud node is updated to the latest minor version and that new overcloud images are prepared for an upgrade. See the *Chapter 2—Updating the undercloud* section of the Keeping Red Hat OpenStack Platform Updated guide from Red Hat. If the undercloud has been updated and a copy of the heat templates are used for the deployment, update the copy of the heat template from the Red Hat's core heat template collection at **/usr/share/openstack-tripleo-heat-templates**. For more information on this process, see the Understanding Heat Templates chapter from Red Hat.
- Per Red Hat OpenStack support guidelines, do not change IP addresses during this upgrade.
- New Contrail Control plane is deployed on K8s/OpenShift cluster with a self-signed root CA:
  - You should generate a self-signed root CA and a key: k8s-root-ca.pem and k8s-root-ca-key.pem.
  - You must deploy Contrail with the generated self-signed root CA and key as a Contrail root CA.

    Example of input environment variables for deploying new Contrail Control plane on K8s/ OpenShift cluster with a self-signed root CA:

```
export SSL_CACERT=$(cat ~/k8s-root-ca.pem)
export SSL_CAKEY=$(cat ~/k8s-root-ca-key.pem)
... other actions to deploy from tf-operator ...
```

- Use CA bundle, if RHOSP uses IPA for the certificate management cluster. Example of how to prepare CA bundle and use as Contrail root CA:

```
cat k8s-root-ca.pem /etc/ipa/ca.crt  > ca-bundle.pem
export SSL_CACERT=$(cat ~/ca-bundle.pem)
export SSL_CAKEY=$(cat ~/k8s-root-ca-key.pem)
... other actions to deploy from tf-operator ...
```

Before You Begin

We recommend performing these procedures before you start the update:

- Backup your Contrail configuration database before starting this procedure, see:
  - "How to Backup and Restore Contrail Databases in JSON Format in Openstack Environments Using the Openstack 13 or Ansible Deployers" on page 243
  - "How to Backup and Restore Contrail Databases in JSON Format in Openstack Environments Using the Openstack 16.1 Director Deployment" on page 231
- Each compute node agent will go down during this procedure, causing some compute node downtime. The estimated downtime for a compute node varies by environment, but typically took between 12 and 15 minutes in our testing environments.

  If you have compute nodes with workloads that cannot tolerate this downtime, consider migrating workloads or taking other steps to accommodate this downtime in your environment.
- Obtain *ContrailImageTag* from:
  - README Access to Contrail Registry 19XX
  - README Access to Contrail Registry 20XX
  - README Access to Contrail Registry 21XX

Upgrade Contrail Networking Release 1912.L4 or 2011.L3 with RHOSP 13 or RHOSP 16.1 to Contrail Networking Release 21.4 with RHOSP 16.2

To perform the upgrade:

1. Deploy new Contrail control plane on Kubernetes/Openshift cluster. See "Prerequisites" on page 169.
2. To prepare the cluster for ISSU update:
   a. Ensure that overcloud VIP FQDNs and all overcloud nodes FQDNs are resolvable on the selected ISSU node (**modify /etc/hosts**). For example:

```
cat /etc/hosts | grep overcloud
192.XXX.XX.200    overcloud.dev.localdomain
192.XXX.XX.105   overcloud-contrailcontroller-0.dev.localdomain
... IMPORTANT: all FQDNs of all overcloud nodes (all networks) ...
```

> **NOTE**: Overcloud node FQDNs can be taken from **/etc/hosts** of one of the overcloud node.

b. Distribute the self signed root CA of the Kubernetes cluster on Contrail Controller nodes as trusted CA and switch Contrail RabbitMQ to use the CA bundle.

   i. Make a CA bundle file.

```
cat /etc/ipa/ca.crt k8s-root-ca.pem > ca-bundle.pem
```

   ii. Prepare the environment file **ca-bundle.yaml**.

```
# create file
cat <<EOF > ca-bundle.yaml
resource_registry:
  OS::TripleO::NodeTLSCAData: tripleo-heat-templates/puppet/extraconfig/tls/ca-
inject.yaml
parameter_defaults:
  ContrailCaCertFile: "/etc/pki/ca-trust/source/anchors/contrail-ca-cert.pem"
  SSLRootCertificatePath: "/etc/pki/ca-trust/source/anchors/contrail-ca-cert.pem"
  SSLRootCertificate: |
EOF
# append cert data
cat ca-bundle.pem | while read l ; do
  echo "    $l" >> ca-bundle.yaml
done
# check
cat ca-bundle.yaml
```

   iii. Distribute the CA bundle. Add a new environment file and make Contrail services to use the CA bundle to run the deploy command. For example:

```
# note: ca-bundle.yaml must be after contrail-tls.yaml as it overrides params
openstack overcloud deploy \
  <all env files used before for deploy> \
  -e ca-bundle.yaml
```

3. Start the ISSU sync data:

a. Stop service Device Manager, Schema Transformer and SVC Monitor, edit manager's manifest, and add devicemanager, schematransformer, and svcmonitor containers to the containers section.

```
kubectl -n tf edit manager
```

> **NOTE**: Ensure that the config pod is restarted and the removed containers are not listed in the containers section.

b. Select one master node to run the ISSU scripts and then label the master node (for example, node1).

```
kubectl label nodes node1 contrail-issu=""
```

c. Collect data from the operator environment. For example:

```
kubectl -n tf exec -it config1-config-statefulset-0 \
-c api -- bash -c 'cat /etc/contrailconfigmaps/api.0.$POD_IP'
```

d. Prepare **issu-configmap**.

```
mkdir -p issu-configmap
cp tf-tripleo-heat-templates/tools/contrail/issu/issu.* \
  tf-tripleo-heat-templates/tools/contrail/issu/*.sh \
  issu-configmap/
```

e. Modify **issu.env** file according to your environment and create a configmap.

```
kubectl create configmap -n tf issu-configmap \
  --from-file=issu.env=issu-configmap/issu.env \
  --from-file=issu.conf=issu-configmap/issu.conf \
  --from-file=issu_node_pair.sh=issu-configmap/issu_node_pair.sh
```

f. Prepare jobs.

```
sudo mkdir -p /var/log/contrail/issu
mkdir -p issu-jobs
```

```
cp tf-tripleo-heat-templates/tools/contrail/issu/*.yaml \
  issu-jobs/
```

Modify the yaml files if you need to customize your configurations.

> **NOTE**: Modify images to use already pulled images on K8s nodes. This is because, the ISSU jobs do not pull images and rely on images used to deploy the K8s cluster.

g. Run a pair job to pair control nodes between the old and new clusters.

```
kubectl apply -f issu-jobs/issu-pair-add-job.yaml
```

Wait till the job is completed and log is validated.

```
kubectl get pods -n tf -w | grep issu
cat /var/log/contrail/issu/issu-pair-add-job.log
....
INFO: provision_control.py --host_name node-10-100-0-147.localdomain ... exit with code 0
```

h. Pre-synchronize data to the new cluster and run the presync job.

```
kubectl apply -f issu-jobs/issu-presync-job.yaml
```

Wait till the job is completed and log is validated.

```
kubectl get pods -n tf -w | grep issu
cat /var/log/contrail/issu/issu-presync-job.log
...
Done syncing Configdb uuid
Done syncing bgp keyspace
Done syncing useragent keyspace
Done syncing svc-monitor keyspace
Done syncing dm keyspace
```

i. Run the sync data between clusters.

i. Run the sync job.

```
kubectl apply -f issu-jobs/issu-sync-job.yaml
```

ii. Check that the sync job is running normally.

```
cat /var/log/contrail/issu/issu-sync-job.log
...
Config Sync initiated...
Config Sync done...
Started runtime sync...
Start Compute upgrade...
```

NOTE: At this point switch to main instruction and follow RHOSP update/ upgrade procedure.

j. Return to the main update workflow and update the cluster.

4. Upgrade the undercloud:

   a. For RHOSP 13 to RHOSP 16.2, follow *Chapter 4—Preparing for the undercloud upgrade* to *Chapter 6—Upgrading director* of Framework for Upgrades (13 to 16.2).

   b. For RHOSP 16.1 to RHOSP 16.2, follow *Chapter 2—Updating the undercloud* of Keeping Red Hat OpenStack Platform Updated.

5. Upgrade the overcloud:

   a. For RHOSP 13 to RHOSP 16.2, follow *Chapter 7—Initial steps for overcloud preparation* through *Step 17.3—OpenStack overcloud external-upgrade run* of Framework for Upgrades (13 to 16.2).

   b. For RHOSP 16.1 to RHOSP 16.2, follow *Step 3.1—Running the overcloud update preparation* through *Step 3.9—Performing online database updates* of Keeping Red Hat OpenStack Platform Updated.

6. Stop the ISSU sync job, finalize sync data, and unpair control nodes:

   a. Stop the ISSU sync job.

```
kubectl delete -f issu-jobs/issu-sync-job.yaml
# wait till it deleted
kubectl get pods -n tf -w | grep issu
```

b. Finalize the sync and run the sync job.

```
kubectl apply -f issu-jobs/issu-postsync-job.yaml
```

c. Check that the sync job is running normally.

```
kubectl get pods -n tf -w | grep issu
cat /var/log/contrail/issu/issu-postsync-job.log
```

d. Finalize the sync of ZK data and run the sync job.

```
kubectl apply -f issu-jobs/issu-post-zk-sync-job.yaml
```

e. Check that the sync job is running normally.

```
cat /var/log/contrail/issu/issu-post-zk-sync-job.log
```

f. Run the unpair job to delete pairing of control nodes between the old and new clusters.

```
kubectl apply -f issu-jobs/issu-pair-del-job.yaml
```

Wait till the job is completed and log is validated.

```
kubectl get pods -n tf -w | grep issu
cat /var/log/contrail/issu/issu-pair-del-job.log
....
INFO: operation finished successfully
...
```

g. Start service Device Manager, Schema Transformer and SVC Monitor, edit manager's manifest, and add devicemanager, schematransformer, and svcmonitor containers to the containers section.

```
kubectl -n tf edit manager
```

NOTE: Ensure that the config pod is restarted and the removed containers are not listed in the containers section.

Example of the edited config section:

```
config:
        metadata:
          labels:
            tf_cluster: cluster1
          name: config1
        spec:
          commonConfiguration:
            nodeSelector:
              node-role.kubernetes.io/master: ""
          serviceConfiguration:
            containers:
            - image: tungstenfabric/contrail-controller-config-devicemgr:latest
              name: devicemanager
            - image: tungstenfabric/contrail-controller-config-schema:latest
              name: schematransformer
            - image: tungstenfabric/contrail-controller-config-svcmonitor:latest
              name: servicemonitor
            - image: tungstenfabric/contrail-controller-config-api:latest
              name: api
            - image: tungstenfabric/contrail-controller-config-dnsmasq:latest
              name: dnsmasq
            - image: tungstenfabric/contrail-nodemgr:latest
              name: nodemanager
            - image: tungstenfabric/contrail-node-init:latest
              name: nodeinit
            - image: tungstenfabric/contrail-provisioner:latest
              name: provisioner
```

h. Remove the label from the ISSU node and **issu-config configmap**.

```
# ajdust to use your node name
kubectl label nodes node1 contrail-issu-
kubectl delete configmap -n tf issu-configmap
```

i. Return to the main update workflow.

7. Continue with the overcloud upgrade process:

   a. For RHOSP 13 to RHOSP 16.2, follow *Step 17.4—OpenStack overcloud upgrade converge* of Framework for Upgrades (13 to 16.2).

   b. For RHOSP 16.1 to RHOSP 16.2, follow *Step 3.10—Finalizing the update* of Keeping Red Hat OpenStack Platform Updated.

8. Remove the old Contrail Control plane nodes.

```
# set counts to 0
# (note - use your file where counts are defined)
sed -i misc_opts.yaml -e 's/ContrailControllerCount: .*/ContrailControllerCount: 0/'
sed -i misc_opts.yaml -e 's/ContrailAnalyticsCount: .*/ContrailAnalyticsCount: 0/'
sed -i misc_opts.yaml -e 's/ContrailAnalyticsDatabaseCount: .*/
ContrailAnalyticsDatabaseCount: 0/'

# delete nodes from overcloud
openstack overcloud node delete --yes --stack overcloud \
  overcloud-contrailcontroller-0 \
  .... other nodes ...
```

9. (Optional) Perform the post-upgrade tasks.

   a. For RHOSP 13 to RHOSP 16.2, follow *Chapter 25—Performing post-upgrade actions* of Framework for Upgrades (13 to 16.2).

   b. For RHOSP 16.1 to RHOSP 16.2, follow *Chapter 4—Rebooting the overcloud* through *Step 4.3—Rebooting Compute nodes* of Keeping Red Hat OpenStack Platform Updated.

RELATED DOCUMENTATION

How to Upgrade Contrail Networking Through Kubernetes and/or Red Hat OpenShift | 222

Deploying Red Hat Openstack with Contrail Control Plane Managed by Tungsten Fabric Operator | 227

Upgrading Contrail Networking until 21.4.L2 using the Ansible Deployer In-Service Software Upgrade Procedure in OpenStack Environments

IN THIS SECTION

When to Use This Procedure?

NOTE: Before performing any upgrade procedure, install Docker serially over containers. However, you can upgrade computes in parallel to Docker via script. After upgrading each docker host, verify the status of contrail and services. Do not proceed with upgrade on next hosts until all the services of contrail-status reports are running properly.

We recommend using the Zero Impact Upgrade (ZIU) procedures to upgrade Contrail Networking with minimal network disruption in most environments using Openstack orchestration.

To perform a ZIU upgrade, follow the instructions in How to Perform a Zero Impact Contrail Networking Upgrade using the Ansible Deployer. If you are running Red Hat Openstack 13 or 16.1, see Updating Contrail Networking using the Zero Impact Upgrade Process in an Environment using Red Hat Openstack 13 or Updating Contrail Networking using the Zero Impact Upgrade Process in an Environment using Red Hat Openstack 16.1.

The procedure in this document also provides a method of upgrading Contrail Networking with minimal network disruption using the Ansible deployer in environments using Openstack orchestration.

The procedure in this document has been validated to upgrade Contrail Networking from Release 3.2 or later to Release 5.0 or later. The starting Contrail release for this upgrade can be any Contrail Networking Release after Release 3.2, including all Contrail Networking 4, 5, 19, 20, 21 upto 21.4.L1

releases. The target release for this upgrade can be any Contrail Networking Release after Release 5.0, including all Contrail Networking 5, 19, 20, 21 until 21.4.L2 releases.

Table 11: Contrail Networking Validated Upgrade Scenarios

Starting Contrail Networking Release	Target Upgraded Contrail Networking Release
Release 3.2 or Later	Any Release 5
Any Release 4	Any Release 19
Any Release 5	Any Release 20
Any Release 19	Any Release 21 until 21.4.L2
Any Release 20	
Any Release 21 prior to 21.4.L2	

Contrail In-Service Software Upgrade (ISSU) Overview

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

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

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

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

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

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

Prerequisites

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

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

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

Preparing the Contrail System for the Ansible Deployer ISSU Procedure

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

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

   NOTE: You must provide keystone authorization information for setup.

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

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

1. To deploy the new controller, download **contrail-ansible-deployer-*release-tag*.tgz** onto your provisioning host from Juniper Networks.

2. The new controller file **config/instances.yaml** appears as follows, with actual values in place of the variables as shown in the example:

```
provider_config:
 bms:
   domainsuffix: local
   ssh_user: user
   ssh_pwd: password
instances:
 server1:
  ip: controller 1 ip
  provider: bms
  roles:
   analytics: null
   analytics_database: null
   config: null
   config_database: null
   control: null
   webui: null
contrail_configuration:
 CONTROLLER_NODES: controller ip-s from api/mgmt network
 CONTROL_NODES: controller ip-s from ctrl/data network
 AUTH_MODE: keystone
 KEYSTONE_AUTH_ADMIN_TENANT: old controller's admin's tenant
 KEYSTONE_AUTH_ADMIN_USER: old controller's admin's user name
 KEYSTONE_AUTH_ADMIN_PASSWORD: password for admin user
 KEYSTONE_AUTH_HOST: keystone host/ip of old controller
 KEYSTONE_AUTH_URL_VERSION: "/v3"
 KEYSTONE_AUTH_USER_DOMAIN_NAME: user's domain in case of keystone v3
 KEYSTONE_AUTH_PROJECT_DOMAIN_NAME: project's domain in case of keystone v3
 RABBITMQ_NODE_PORT: 5673
 IPFABRIC_SERVICE_HOST: metadata service host/ip of old controller
 AAA_MODE: cloud-admin
 METADATA_PROXY_SECRET: secret phrase that is used in old controller
kolla_config:
 kolla_globals:
   kolla_internal_vip_address: keystone host/ip of old controller
   kolla_external_vip_address: keystone host/ip of old controller
```

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

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

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

Provisioning Control Nodes and Performing Synchronization Steps

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

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

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

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

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

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

```
docker exec -it $config_api_image /bin/bash -c "LOG_LEVEL=SYS_NOTICE source /common.sh ;
python /opt/contrail/utils/provision_control.py --host_name hostname of old control node
```

```
--host_ip IP of old control node --api_server_ip $(hostname -i)
 --api_server_port 8082 --oper add --router_asn 64512 --ibgp_auto_mesh \$AUTH_PARAMS"
```

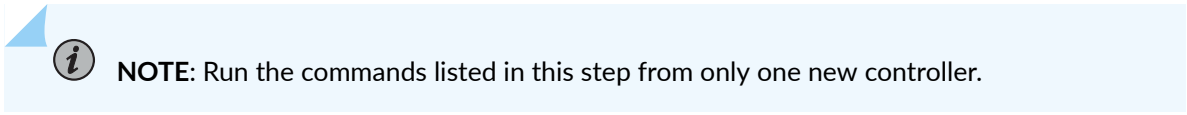
3. Pair the new control nodes in the old cluster with similar commands (the specific syntax depends on the deployment method of the old cluster), again substituting actual values where indicated.

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

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

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

5. For Kolla/Juju setup, perform the following steps to delete the contrail-device-manager queue from Contrail rabbitmq after the contrail-device-manager container is stopped.



   a. Enter the Contrail rabbitmq container.

```
docker exec -it config_database_rabbitmq_1 bash
```

   b. Find the name of the contrail-device-manager queue.

```
rabbitmqctl list_queues | grep -F device_manager
```

   c. Delete the contrail-device-manager queue.

```
rabbitmqctl delete_queue <device_manager.queue>
```

For RHOSP setup, perform the following steps to delete the contrail-device-manager queue from Contrail rabbitmq after the contrail-device-manager container is stopped.

a. Enter the Contrail rabbitmq container.

```
podman exec -it contrail_config_rabbitmq bash
```

b. Find the name of the contrail-device-manager queue.

```
rabbitmqctl list_queues | grep -F device_manager
```

c. Delete the contrail-device-manager queue.

```
rabbitmqctl delete_queue <device_manager.queue>
```

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

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

The configuration appears as follows and can be stored locally:

```
[DEFAULTS]
# details about oldrabbit
old_rabbit_user = contrail
old_rabbit_password = ab86245f4f3640a29b700def9e194f72
old_rabbit_q_name = vnc-config.issu-queue
old_rabbit_vhost = contrail
old_rabbit_port = 5672
old_rabbit_address_list = ip-addresses
# details about new rabbit
# new_rabbit_user = rabbitmq
# new_rabbit_password = password
# new_rabbit_ha_mode =
new_rabbit_q_name = vnc-config.issu-queue
new_rabbit_vhost = /
new_rabbit_port = 5673
```

```
new_rabbit_address_list = ip-addresses
# details about other old/new services
old_cassandra_user = controller
old_cassandra_password = 04dc0540b796492fad6f7cbdcfb18762
old_cassandra_address_list = ip-address:9161
old_zookeeper_address_list = ip-address:2181
new_cassandra_address_list = ip-address:9161 ip-address:9161 ip-address:9161
new_zookeeper_address_list = ip-address:2181
# details about new controller nodes
new_api_info = {"ip-address": [("root"), ("password")], "ip-address": [("root"), ("password")],
"ip-address": [("root"), ("password")]}
```

1. Detect the config-api image ID.

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

2. Run the pre-synchronization.

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

3. Run the run-synchronization.

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

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

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

Transferring the Compute Nodes into the New Cluster

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

NOTE: Before transferring the Compute Nodes into a new cluster, make sure that docker was successfully updated.

1. Select the compute node(s) for transferring into the new cluster. This selects the virtual machines of the compute node(s).
2. Migrate the Virtual Machines (VMs) manually from one compute node to another. The steps are as follows:

   NOTE: This procedure is useful when live migration cannot be done.

   a. Identify the VM to migrate and its host. Run the following command to stop VM.

   ```
   openstack server stop <vm-uuid>
   ```

   b. Identify the VM disk image location on the source compute node where the VM instance was launched, Usually, the disk image location is:

   ```
   /var/lib/docker/volumes/nova_compute/_data/instances/<vm-UUID>
   ```

   c. Copy this directory to the destination compute node.

   d. On the destination compute node, run the following command to change the permission of this directory:

   ```
   chown -R 42436:42436 /var/lib/docker/volumes/nova_compute/_data/instances/<vm-UUID>
   ```

   e. Update the nova database of this instance to new host.

   ```
   docker exec -it mariadb bash
   mysql -u <username> -p <password> nova
   update instances set node='new host fqname', host='new hostname' where uuid='<VM-UUID>'
   ```

Example:

```
(mariadb)[mysql@nodem1 /]$ mysql -u root -p contrail123 nova
MariaDB [nova]> update instances
    -> set node='nodem2.englab.juniper.net', host='nodem2'
    -> where uuid='b7178be6-d4da-4074-9124-d246fa3a2105'
    -> ;
```

f. Run the following command to start the VM

```
openstack server start <vm-uuid>
```

3. For Contrail Release 3.x, remove Contrail from the node(s) as follows:
   - Stop the vrouter-agent service.
   - Remove the `vhost0` interface.
   - Switch the physical interface down, then up.
   - Remove the vrouter.ko module from the kernel.
4. For Contrail Release 4.x and later, remove Contrail from the node(s) as follows:
   - Stop the agent container.
   - Restore the physical interface.
5. Add the required node(s) to **instances.yml** with the roles `vrouter` and `openstack_legacy_compute`.
6. Run the Contrail ansible deployer to deploy the new vrouter and to configure the old compute service.
7. All new compute nodes will have:
   - The collector setting pointed to the new Contrail cluster
   - The Control/DNS nodes pointed to the new Contrail cluster
   - The config-api setting in **vnc_api_lib.ini** pointed to the new Contrail cluster
8. (Optional) Run a test workload on transferred nodes to ensure the new vrouter-agent works correctly.

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

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

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

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

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

For other kind of deployments, remove the vrouter-agent and vrouter-agent-nodemgr containers, and disable vhost0 interface.

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

```
instances:
 server10:
  ip: compute 10 ip
  provider: bms
  roles:
```

```
  vrouter:
   MAINTENANCE_MODE: TRUE
   VROUTER_ENCRYPTION: FALSE
  openstack_compute_legacy: null
```

Make sure that **instances.yaml** nodes definition includes only the compute nodes you want to upgrade. All other nodes should be commented out.

2. Run the ansible playbooks.

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

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

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

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

```
docker restart control_control_1
```

5. After upgrading the compute nodes, XMPP goes down due to SSLhandshake issue. Example:

```
vrouter kernel module is PRESENT
== Contrail vrouter ==
nodemgr: active
agent: initializing (XMPP:control-node:10.10.10.1, XMPP:dns-server:10.10.10.1 connection down,
No Configuration for self)
```

The steps to bring up XMPP are as follows:

a. Copy the following two files from new control node to upgraded compute node:

```
/etc/contrail/ssl/private/server-privkey.pem
/etc/contrail/ssl/certs/server.pem
```

b. Restart the VRouter agent of upgraded compute node.

```
docker restart vrouter_vrouter-agent_1
```

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

Finalizing the Contrail Ansible Deployer ISSU Process

Finalize the Contrail ansible deployer ISSU as follows:

1. Stop the issu-run-sync container.

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

2. Run the post synchronization commands.

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

3. Run the following commands on all the new controller nodes.

```
docker-compose -f /etc/contrail/config/docker-compose.yaml restart api
docker-compose -f /etc/contrail/config/docker-compose.yaml up -d
```

4. Restart the container.

```
docker restart config_api_1
```

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

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

During this step, only the compute nodes should be included in the instances.yaml, and other nodes should be commented out.

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

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

NOTE: Currently, the previous step works only with hostname and not with IP Address.

7. Run the following commands from any controller node.

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

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

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

9. Navigate to the following path in old controller:

```
[root@nodem1 ~]# cd /etc/kolla/neutron-server/
[root@nodem1 neutron-server]# pwd
/etc/kolla/neutron-server
[root@nodem1 neutron-server]# cat ContrailPlugin.ini
[APISERVER]
api_server_port = 8082
api_server_ip = <old_controller_ip>
multi_tenancy = True
contrail_extensions =
ipam:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_ipam.NeutronPluginContrail
Ipa
m,policy:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_policy.NeutronPluginCo
ntr
ailPolicy,routetable:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_vpc.Neutro
nPluginContrailVpc
,contrail:None,service-interface:None,vf-binding:None
[COLLECTOR]
analytics_api_ip = <old_controller_ip>
analytics_api_port = 8081
[keystone_authtoken]
auth_host = <old_controller_ip>
auth_port = 5000
auth_protocol = http
admin_user = admin
admin_password = password
admin_tenant_name = admin
insecure = True
region_name = RegionOne
```

10. Modify the api_server_ip and analytics_api_ip addresses with the new controller IP addresses.

```
[root@nodem1 neutron-server]# pwd
/etc/kolla/neutron-server
[root@nodem1 neutron-server]# cat ContrailPlugin.ini
[APISERVER]
api_server_port = 8082
api_server_ip = <new_controller_ip>
multi_tenancy = True
contrail_extensions =
ipam:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_ipam.NeutronPluginContrail
Ipa
m,policy:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_policy.NeutronPluginCo
ntr
ailPolicy,routetable:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_vpc.Neutro
nPluginContrailVpc
,contrail:None,service-interface:None,vf-binding:None

[COLLECTOR]
analytics_api_ip = <new_controller_ip>
analytics_api_port = 8081

[keystone_authtoken]
auth_host = <keystone_ip_addr>
auth_port = 5000
auth_protocol = http
admin_user = admin
admin_password = password
admin_tenant_name = admin
insecure = True
region_name = RegionOne
```

11. Restart the neutron-server container in old controller.

```
[root@nodem1]# docker restart neutron_server
```

12. Go to neutron_server container in the old control node. Verify whether the ContrailPlugin.ini file contains new controller IP's or not. It should contain new controller IP's.

```
[root@nodem1 ~]# docker exec -it neutron_server bash
(neutron-server)[neutron@nodem1 /]$ cd /etc/neutron/plugins/opencontrail
(neutron-server)[neutron@nodem1 /etc/neutron/plugins/opencontrail]$ cat ContrailPlugin.ini
[APISERVER]
api_server_port = 8082
api_server_ip = <new_controller_ip>
multi_tenancy = True
contrail_extensions =
ipam:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_ipam.NeutronPluginContrail
Ipa
m,policy:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_policy.NeutronPluginCo
ntr
ailPolicy,routetable:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_vpc.Neutro
nPluginContrailVpc
,contrail:None,service-interface:None,vf-binding:None
[COLLECTOR]
analytics_api_ip = <new_controller_ip>
analytics_api_port = 8081
[keystone_authtoken]
auth_host = <keystone_ip_addr>
auth_port = 5000
auth_protocol = http
admin_user = admin
admin_password = password
admin_tenant_name = admin
insecure = True
region_name = RegionOne
```

13. The heat configuration needs the same changes. Locate the parameter `[clients_contrail]/api_server` and change it to point to the list of the new config-api IP addresses.

Troubleshooting link-loop in Release 21.4.L2

The ansible deployer of Contrail Networking Release 21.4.L2 introduces link-loop in the `/var/log/contrail` directory present in the contrail config nodes. This happens every time the Contrail Networking Release 21.4.L2 ansible deployer is started. Re-running ansible deployer playbooks fails due to mentioned recursion. This issue is resolved in Contrail Networking Release 21.4.L3. However, for Contrail Networking Release 21.4.L2, it requires a manual intervention to follow the given workaround.

Workaround: Manually remove the incorrect symlink from all contrail config nodes:

```
sudo unlink /var/log/contrail/config-database-rabbitmq/config-database-rabbitmq
```

Upgrading Contrail Networking to Release 21.4.L3 using Ansible Deployer in Service Software Upgrade Procedure in OpenStack Environment

IN THIS SECTION

When to Use This Procedure?

NOTE: Before performing any upgrade procedure, install Docker serially over containers. However, you can upgrade computes in parallel to Docker via script. After upgrading each docker host, verify the status of contrail and services. Do not proceed with upgrade on next hosts until all the services of contrail-status reports are running properly.

Use the following script to stop the running containers, upgrade the docker, and bring containers back:

```
docker ps --format '{{.Names}}' > running_containers
for CONTAINER in $(cat running_containers); do sudo docker stop $CONTAINER; done
yum install -y docker-ce-20.10.9 docker-ce-cli-20.10.9 docker-ce-rootless-
```

```
        extras-20.10.9
        for CONTAINER in $(cat running_containers); do sudo docker start $CONTAINER; done
```

It is recommended to use Zero Impact Upgrade (ZIU) procedures to upgrade Contrail Networking with minimal network disruption in most environments using Openstack orchestration.

To perform a ZIU upgrade, follow the instructions in How to Perform a Zero Impact Contrail Networking Upgrade using the Ansible Deployer. If you are running Red Hat Openstack 13 or 16.1, see Updating Contrail Networking using the Zero Impact Upgrade Process in an Environment using Red Hat Openstack 13 or Updating Contrail Networking using the Zero Impact Upgrade Process in an Environment using Red Hat Openstack 16.1.

The procedure in this document also provides a method of upgrading Contrail Networking with minimal network disruption using the Ansible deployer in environments using Openstack orchestration.

The procedure in this document has been validated to upgrade Contrail Networking from Release 3.2 or later to Release 5.0 or later. The starting Contrail release for this upgrade can be any Contrail Networking Release after Release 3.2, including all Contrail Networking 4, 5, 19, 20, and 21 releases. The target release for this upgrade can be any Contrail Networking Release after Release 5.0, including all Contrail Networking 5, 19, 20, and 21 releases.

Table 12: Contrail Networking Validated Upgrade Scenarios

Starting Contrail Networking Release	Target Upgraded Contrail Networking Release
Release 3.2 or Later	Any Release 5
Any Release 4	Any Release 19
Any Release 5	Any Release 20
Any Release 19	Any Release 21
Any Release 20	
Any Release 21	

Contrail In-Service Software Upgrade (ISSU) Overview

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

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

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

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

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

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

Prerequisites

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

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

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

Preparing the Contrail System for the Ansible Deployer ISSU Procedure

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

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

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

NOTE: You must provide keystone authorization information for setup.

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

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

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

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

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

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

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

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

Provisioning Control Nodes and Performing Synchronization Steps

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

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

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

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

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

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

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

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

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

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

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

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

5. Perform the following steps to delete the contrail-device-manager queue from Contrail rabbitmq after the contrail-device-manager container is stopped.

NOTE: Run the commands listed in this step from only one new controller.

a. Enter the Contrail rabbitmq container.

```
docker exec -it config_rabbitmq_rabbitmq_1 bash
```

b. Find the name of the contrail-device-manager queue.

```
rabbitmqctl list_queues | grep -F device_manager | grep $(hostname) | grep -v ztp
```

c. Delete the contrail-device-manager queue.

```
rabbitmqctl delete_queue <device_manager.queue>
```

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

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

The configuration appears as follows and can be stored locally:

```
[DEFAULTS]
# details about oldrabbit
old_rabbit_user = contrail
old_rabbit_password = ab86245f4f3640a29b700def9e194f72
old_rabbit_q_name = vnc-config.issu-queue
old_rabbit_vhost = contrail
old_rabbit_port = 5672
old_rabbit_address_list = ip-addresses
# details about new rabbit
# new_rabbit_user = rabbitmq
# new_rabbit_password = password
# new_rabbit_ha_mode =
new_rabbit_q_name = vnc-config.issu-queue
new_rabbit_vhost = /
new_rabbit_port = 5673
new_rabbit_address_list = ip-addresses
# details about other old/new services
```

```
old_cassandra_user = controller
old_cassandra_password = 04dc0540b796492fad6f7cbdcfb18762
old_cassandra_address_list = ip-address:9161
old_zookeeper_address_list = ip-address:2181
new_cassandra_address_list = ip-address:9161 ip-address:9161 ip-address:9161
new_zookeeper_address_list = ip-address:2181
# details about new controller nodes
new_api_info = {"ip-address": [("root"), ("password")], "ip-address": [("root"), ("password")],
"ip-address": [("root"), ("password")]}
```

1. Detect the config-api image ID.

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

2. Run the pre-synchronization.

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

3. Run the run-synchronization.

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

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

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

Transferring the Compute Nodes into the New Cluster

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

NOTE: Before transferring the Compute Nodes into a new cluster, make sure that docker was successfully updated.

1. Select the compute node(s) for transferring into the new cluster. This selects the virtual machines of the compute node(s).
2. Migrate the Virtual Machines (VMs) manually from one compute node to another. The steps are as follows:

   NOTE: This procedure is useful when live migration cannot be done.

   a. Identify the VM to migrate and its host. Run the following command to stop VM.

   ```
   openstack server stop <vm-uuid>
   ```

   b. Identify the VM disk image location on the source compute node where the VM instance was launched, Usually, the disk image location is:

   ```
   /var/lib/docker/volumes/nova_compute/_data/instances/<vm-UUID>
   ```

   c. Copy this directory to the destination compute node.

   d. On the destination compute node, run the following command to change the permission of this directory:

   ```
   chown -R 42436:42436 /var/lib/docker/volumes/nova_compute/_data/instances/<vm-UUID>
   ```

   e. Update the nova database of this instance to new host.

   ```
   docker exec -it mariadb bash
   mysql -u <username> -p <password> nova
   update instances set node='new host fqname', host='new hostname' where uuid='<VM-UUID>'
   ```

Example:

```
(mariadb)[mysql@nodem1 /]$ mysql -u root -p contrail123 nova
MariaDB [nova]> update instances
    -> set node='nodem2.englab.juniper.net', host='nodem2'
    -> where uuid='b7178be6-d4da-4074-9124-d246fa3a2105'
    -> ;
```

f. Run the following command to start the VM

```
openstack server start <vm-uuid>
```

3. For Contrail Release 3.x, remove Contrail from the node(s) as follows:

   - Stop the vrouter-agent service.
   - Remove the `vhost0` interface.
   - Switch the physical interface down, then up.
   - Remove the vrouter.ko module from the kernel.

4. For Contrail Release 4.x and later, remove Contrail from the node(s) as follows:

   - Stop the agent container.
   - Restore the physical interface.

5. Update docker.

```
docker ps --format '{{.Names}}' > running_containers
for CONTAINER in $(cat running_containers); do sudo docker stop $CONTAINER; done
yum install -y docker-ce-20.10.9 docker-ce-cli-20.10.9 docker-ce-rootless-extras-20.10.9
for CONTAINER in $(cat running_containers); do sudo docker start $CONTAINER; done
rm running_containers
```

6. Remove vrouter_vrouter-agent_1 and vrouter_nodemgr_1.

```
docker rm -f vrouter_vrouter-agent_1
docker rm -f vrouter_nodemgr_1
```

7. Stop vhost0.

```
Ifdown vhost0
```

8. Add the required node(s) to **instances.yml** with the roles `vrouter` and `openstack_legacy_compute`.
9. Run the Contrail ansible deployer to deploy the new vrouter and to configure the old compute service.
10. All new compute nodes will have:
    - The collector setting pointed to the new Contrail cluster
    - The Control/DNS nodes pointed to the new Contrail cluster
    - The config-api setting in **vnc_api_lib.ini** pointed to the new Contrail cluster
11. (Optional) Run a test workload on transferred nodes to ensure the new vrouter-agent works correctly.

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

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

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

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

```
# stop services
systemctl stop contrail-vrouter-nodemgr
systemctl stop contrail-vrouter-agent
# remove packages
apt-get purge -y contrail*
# restore original interfaces definition
cd /etc/network/interfaces.d/
```

```
cp 50-cloud-init.cfg.save 50-cloud-init.cfg
rm vrouter.cfg
# restart networking
systemctl restart networking.service
# remove old kernel module
rmmod vrouter
# maybe you need to restore default route
ip route add 0.0.0.0/0 via 10.0.10.1 dev ens3
```

For other kind of deployments remove the vrouter-agent and vrouter-agent-nodemgr containers, and disable vhost0 interface.

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

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

Make sure that **instances.yaml** nodes definition includes only the compute nodes you want to upgrade. All other nodes should be commented out.

2. Run the ansible playbooks.

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

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

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

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

```
docker restart control_control_1
```

5. After upgrading the compute nodes, XMPP goes down due to SSLhandshake issue. Example:

```
vrouter kernel module is PRESENT
== Contrail vrouter ==
nodemgr: active
agent: initializing (XMPP:control-node:10.10.10.1, XMPP:dns-server:10.10.10.1 connection down,
No Configuration for self)
```

   The steps to bring up XMPP are as follows:

   a. Copy the following two files from new control node to upgraded compute node:

```
/etc/contrail/ssl/private/server-privkey.pem
/etc/contrail/ssl/certs/server.pem
```

   b. Restart the VRouter agent of upgraded compute node.

```
docker restart vrouter_vrouter-agent_1
```

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

Finalizing the Contrail Ansible Deployer ISSU Process

Finalize the Contrail ansible deployer ISSU as follows:

1. Stop the issu-run-sync container.

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

2. Run the post synchronization commands.

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

3. Run the following commands on all the new controller nodes.

```
docker-compose -f /etc/contrail/config/docker-compose.yaml restart api
docker-compose -f /etc/contrail/config/docker-compose.yaml up -d
```

4. Restart the container.

```
docker-compose -f /etc/contrail/config/docker-compose.yaml restart API
docker-compose -f /etc/contrail/config/docker-compose.yaml up -d
```

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

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

During this step, only compute nodes should be included in the instances.yaml, and other nodes should be commented out.

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

```
[DEFAULTS]
db_host_info={"ip-address": "node-ip-address or hostname", "ip-address": "node-ip-address
```

```
or hostname", "ip-address": "node-ip-address or hostname"}
config_host_info={"ip-address": "node-ip-address or hostname", "ip-address": "node-ip-
address or hostname", "ip-address": "node-ip-address or hostname"}
analytics_host_info={"ip-address": "node-ip-address or hostname", "ip-address": "node-ip-
address or hostname", "ip-address": "node-ip-address or hostname"}
control_host_info={"ip-address": "node-ip-address or hostname", "ip-address": "node-ip-
address or hostname", "ip-address": "node-ip-address or hostname"}
admin_password = <admin password>
admin_tenant_name = <admin tenant>
admin_user = <admin username>
api_server_ip= <any IP of new config-api controller>
api_server_port=8082
```

NOTE: Currently, the previous step works only with hostname and not with IP Address.

7. Run the following commands from any controller node.

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

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

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

9. Navigate to the following path in old controller:

```
[root@nodem1 ~]# cd /etc/kolla/neutron-server/
[root@nodem1 neutron-server]# pwd
/etc/kolla/neutron-server
[root@nodem1 neutron-server]# cat ContrailPlugin.ini
[APISERVER]
api_server_port = 8082
api_server_ip = <old_controller_ip>
multi_tenancy = True
contrail_extensions =
ipam:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_ipam.NeutronPluginContrail
Ipa
```

```
m,policy:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_policy.NeutronPluginCo
ntr
ailPolicy,routetable:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_vpc.Neutro
nPluginContrailVpc
,contrail:None,service-interface:None,vf-binding:None
[COLLECTOR]
analytics_api_ip = <old_controller_ip>
analytics_api_port = 8081
[keystone_authtoken]
auth_host = <old_controller_ip>
auth_port = 5000
auth_protocol = http
admin_user = admin
admin_password = password
admin_tenant_name = admin
insecure = True
region_name = RegionOne
```

10. Modify the api_server_ip and analytics_api_ip addresses with the new controller IP addresses.

```
[root@nodem1 neutron-server]# pwd
/etc/kolla/neutron-server
[root@nodem1 neutron-server]# cat ContrailPlugin.ini
[APISERVER]
api_server_port = 8082
api_server_ip = <new_controller_ip>
multi_tenancy = True
contrail_extensions =
ipam:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_ipam.NeutronPluginContrail
Ipa
m,policy:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_policy.NeutronPluginCo
ntr
ailPolicy,routetable:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_vpc.Neutro
nPluginContrailVpc
,contrail:None,service-interface:None,vf-binding:None

[COLLECTOR]
analytics_api_ip = <new_controller_ip>
analytics_api_port = 8081

[keystone_authtoken]
auth_host = <keystone_ip_addr>
```

```
auth_port = 5000
auth_protocol = http
admin_user = admin
admin_password = password
admin_tenant_name = admin
insecure = True
region_name = RegionOne
```

11. Restart the neutron-server container in old controller.

```
[root@nodem1]# docker restart neutron_server
```

12. Go to neutron_server container in the old control node. Verify whether the ContrailPlugin.ini file contains new controller IP's or not. It should contain new controller IP's.

```
[root@nodem1 ~]# docker exec -it neutron_server bash
(neutron-server)[neutron@nodem1 /]$ cd /etc/neutron/plugins/opencontrail
(neutron-server)[neutron@nodem1 /etc/neutron/plugins/opencontrail]$ cat ContrailPlugin.ini
[APISERVER]
api_server_port = 8082
api_server_ip = <new_controller_ip>
multi_tenancy = True
contrail_extensions =
ipam:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_ipam.NeutronPluginContrail
Ipa
m,policy:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_policy.NeutronPluginCo
ntr
ailPolicy,routetable:neutron_plugin_contrail.plugins.opencontrail.contrail_plugin_vpc.Neutro
nPluginContrailVpc
,contrail:None,service-interface:None,vf-binding:None
[COLLECTOR]
analytics_api_ip = <new_controller_ip>
analytics_api_port = 8081
[keystone_authtoken]
auth_host = <keystone_ip_addr>
auth_port = 5000
auth_protocol = http
admin_user = admin
admin_password = password
admin_tenant_name = admin
```

```
insecure = True
region_name = RegionOne
```

13. The heat configuration needs the same changes. Locate the parameter `[clients_contrail]/api_server` and change it to point to the list of the new config-api IP addresses.

14. To resync the database:

    a. Login to the zookeeper container.

    ```
    docker exec -it config_database_zookeeper_1 bash
    ```

    b. Go to the bin directory.

    ```
    cd bin
    ```

    c. Connect to zookeeper.

    ```
    zkCli.sh -server <controller-ip>:2181
    ```

    d. Delete the lock on zookeeper container.

    ```
    delete /vnc_api_server_locks/dbe-resync-complete
    ```

    e. Quit and exit from the zookeeper container.

    f. Restart the Config API server container and wait for the Contrail status to be up.

    ```
    docker restart config_api_1
    ```

    g. Restart the Control container.

    ```
    docker restart control_control_1
    ```

Troubleshooting link-loop in Release 21.4.L2

The ansible deployer of Contrail Networking Release 21.4.L2 introduces link-loop in the `/var/log/contrail` directory present in the contrail config nodes. This happens every time the Contrail Networking Release 21.4.L2 ansible deployer is started. Re-running ansible deployer playbooks fails due to mentioned recursion. This issue is resolved in Contrail Networking Release 21.4.L3. However, for Contrail Networking Release 21.4.L2, it requires a manual intervention to follow the given workaround.

Workaround: Manually remove the incorrect symlink from all contrail config nodes:

```
sudo unlink /var/log/contrail/config-database-rabbitmq/config-database-rabbitmq
```

Contrail In-Service Software Upgrade from Releases 21.4 L2 and 21.4 L3 to 21.4 L4 using Ansible Deployer

IN THIS SECTION

Contrail In-Service Software Upgrade (ISSU) Overview

If your installed version is Contrail Release 21.4 L2 or L3, you can perform an in-service software upgrade (ISSU) to upgrade to Contrail Release 21.4 L4 using the Ansible deployer. In performing the ISSU, the Contrail controller cluster is upgraded side-by-side with a parallel setup, and the compute nodes are upgraded in place.

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

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

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

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

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

Prerequisites

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

- A previous version of Contrail installed, no earlier than Release 21.4 L2.
- There are OpenStack controller and compute nodes, and Contrail nodes.
- OpenStack needs to have been installed from packages.
- Contrail and OpenStack should be installed on different nodes.

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

Preparing the Contrail System for the Ansible Deployer ISSU Procedure

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

1. Deploy Contrail Release 21.4 L4 using the Contrail ansible deployer, but make sure to include only the following Contrail controller services:
    - Config
    - Control
    - Analytics

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

Note:
You must provide keystone authorization information for setup.

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

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

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

```
provider_config:
 bms:
   domainsuffix: local
   ssh_user: user
   ssh_pwd: password
instances:
 server1:
  ip: controller 1 ip
  provider: bms
  roles:
   analytics: null
   analytics_database: null
   config: null
   config_database: null
   control: null
   webui: null
contrail_configuration:
 CONTROLLER_NODES: controller ip-s from api/mgmt network
 CONTROL_NODES: controller ip-s from ctrl/data network
 AUTH_MODE: keystone
 KEYSTONE_AUTH_ADMIN_TENANT: old cloud's admin's tenant
 KEYSTONE_AUTH_ADMIN_USER: old cloud's admin's user name
 KEYSTONE_AUTH_ADMIN_PASSWORD: password for admin user
 KEYSTONE_AUTH_HOST: keystone host/ip of old cloud
 KEYSTONE_AUTH_URL_VERSION: "/v3"
 KEYSTONE_AUTH_USER_DOMAIN_NAME: user's domain in case of keystone v3
 KEYSTONE_AUTH_PROJECT_DOMAIN_NAME: project's domain in case of keystone v3
```

```
RABBITMQ_NODE_PORT: 5673
IPFABRIC_SERVICE_HOST: metadata service host/ip of old cloud
AAA_MODE: cloud-admin
METADATA_PROXY_SECRET: secret phrase that is used in old cloud
kolla_config:
kolla_globals:
  kolla_internal_vip_address: keystone host/ip of old cloud
  kolla_external_vip_address: keystone host/ip of old cloud
```

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

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

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

Provisioning Control Nodes and Performing Synchronization Steps

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

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

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

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

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

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

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

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

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

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

```
docker stop config_devicemgr-1
docker stop config_schema-1
docker stop config_svcmonitor-1
```

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

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

The configuration appears as follows and can be stored locally:

```
[DEFAULTS]
# details about oldrabbit
old_rabbit_user = contrail
old_rabbit_password = ab86245f4f3640a29b700def9e194f72
old_rabbit_q_name = vnc-config.issu-queue
old_rabbit_vhost = contrail
old_rabbit_port = 5672
old_rabbit_address_list = ip-addresses
# details about new rabbit
# new_rabbit_user = rabbitmq
# new_rabbit_password = password
# new_rabbit_ha_mode =
new_rabbit_q_name = vnc-config.issu-queue
new_rabbit_vhost = /
new_rabbit_port = 5673
new_rabbit_address_list = ip-addresses
# details about other old/new services
```

```
old_alter_table = false
new_alter_table = true
old_cassandra_user = controller
old_cassandra_password = 04dc0540b796492fad6f7cbdcfb18762
old_cassandra_address_list = ip-address:9161
old_zookeeper_address_list = ip-address:2181
new_cassandra_address_list = ip-address:9161 ip-address:9161 ip-address:9161
new_zookeeper_address_list = ip-address:2181
# details about new controller nodes
new_api_info = {"ip-address": [("root"), ("password")], "ip-address": [("root"), ("password")], "ip-
address": [("root"), ("password")]}
```

1. Detect the config-api image ID.

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

2. Run the pre-synchronization.

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

3. Run the run-synchronization.

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

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

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

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

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

Transferring the Compute Nodes into the New Cluster

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

1. Select the compute node(s) for transferring into the new cluster.

2. Move all workloads from the node(s) to other compute nodes. You also have the option to terminate workloads as appropriate.
3. For Contrail Release 3.x, remove Contrail from the node(s) as follows:
   - Stop the vrouter-agent service.
   - Remove the `vhost0` interface.
   - Switch the physical interface down, then up.
   - Remove the vrouter.ko module from the kernel.
4. For Contrail Release 4.x, remove Contrail from the node(s) as follows:
   - Stop the agent container.
   - Restore the physical interface.
5. Add the required node(s) to instances.yml with the roles `vrouter` and `openstack_legacy_compute`.
6. Run the Contrail ansible deployer to deploy the new vrouter and to configure the old compute service.
7. All new compute nodes will have:
   - The collector setting pointed to the new Contrail cluster
   - The Control/DNS nodes pointed to the new Contrail cluster
   - The config-api setting in vnc_api_lib.ini pointed to the new Contrail cluster
8. (Optional) Run a test workload on transferred nodes to ensure the new vrouter-agent works correctly.

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

1. Move the workload from the compute node.
2. Stop the Contrail Release 21.4 L4 containers.
3. Ensure the network configuration has been successfully reverted.
4. Deploy the previous version of Contrail using the deployment method for that version.

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

Note:

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

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

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

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

2. Run the ansible playbooks.

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

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

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

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

```
docker restart control_control_1
```

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

Finalizing the Contrail Ansible Deployer ISSU Process

Finalize the Contrail ansible deployer ISSU as follows:

1. Stop the issu-run-sync container.

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

2. Run the post synchronization commands.

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

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

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

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

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

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

5. Run the following commands from any controller node.

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

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

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

7. All configurations for the neutron-api must be edited to have the parameter `api_server_ip` point to the list of new config-api IP addresses. Locate ContrailPlugin.ini (or other file that contains this parameter) and change the IP addresses to the list of new config-api IP addresses.

8. The heat configuration needs the same changes. Locate the parameter `[clients_contrail]/api_server` and change it to point to the list of the new config-api IP addresses.

How to Upgrade Contrail Networking Through Kubernetes and/or Red Hat OpenShift

Starting in Contrail Networking Release 21.3, you can update Contrail Networking through Kubernetes and/or Red Hat OpenShift.

You can use this procedure to update Contrail Networking deployed by the Tungsten Fabric (TF) Operator.

To update Contrail Networking:

1. Update manifests with the new container tag.

```
export CONTRAIL_CONTAINER_TAG=<new tag>
./tf-operator/contrib/render_manifests.sh
```

> **NOTE**: Only CONTRAIL_CONTAINER_TAG must have a new tag. The render manifest must be done with all the same exported environment variables used during the initial deployment.

2. Update the tf-operator deployment.

```
kubectl apply -k ./tf-operator/deploy/kustomize/operator/templates/
```

3. Wait and ensure that the tf-operator deployment is updated.

```
kubectl -n tf get pods -w | grep tf-operator
```

4. Update Contrail Networking resources.

```
kubectl apply -k ./tf-operator/deploy/kustomize/contrail/templates/
```

5. Wait and ensure that the Contrail Control plane pods are updated.

```
kubectl -n tf get pods -w
```

6. Use the contrail-status tool to check the Contrail Networking status on all the master nodes.

```
$ contrail-status
Must show that all services are active:
== Contrail control ==
control: active
nodemgr: active
named: active
dns: active

== Contrail analytics-alarm ==
nodemgr: active
kafka: active
```

```
alarm-gen: active

== Contrail kubernetes ==
kube-manager: backup

== Contrail database ==
nodemgr: active
query-engine: active
cassandra: active

== Contrail analytics ==
nodemgr: active
api: active
collector: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
cassandra: active

== Contrail webui ==
web: active
job: active

== Contrail vrouter ==
nodemgr: active
agent: active

== Contrail analytics-snmp ==
snmp-collector: active
nodemgr: active
topology: active

== Contrail config ==
svc-monitor: backup
nodemgr: active
device-manager: backup
api: active
schema: backup
```

7. Upgrade the Contrail vRouter components (one-by-one or by groups).

- Choose a node to upgrade and obtain the vRouter daemon name for the node.

```
kubectl describe node <node name>
```

- Delete the vRouter pod resource by specifying the name of the pod you want to delete.

```
kubectl -n tf delete pod <vrouter1-vrouter-daemonset-xxxxx>
```

- Wait until the new daemon set is run by kubernetes on a node.

  Use `kubectl get pods < >` commad.

```
kubectl get pods -n tf | grep "vrouter1-vrouter-daemonset"
vrouter1-vrouter-daemonset-77cnz   3/3     Running   0    51m
vrouter1-vrouter-daemonset-7rlvf   3/3     Running   0    87m
vrouter1-vrouter-daemonset-jrzfm   3/3     Running   0    82m
vrouter1-vrouter-daemonset-jvhmj   3/3     Running   0    85m
vrouter1-vrouter-daemonset-v4brl   3/3     Running   0    52m
```

  The status is showing Running for all the vRouter daemon sets. The number of daemon set entries depends on the cluster size (that is number of master nodes and worker nodes).

  You can also verify the status of a particular daemon set. Obtain the new vrouter-daemonset from the `kubectl describe node <node name>` command. Check the status of that particular daemon set using the `kubectl get pods -n tf | grep "vrouter1-vrouter-daemonset-XXX"` command.

8. Verify the vRouter agent status by using the `contrail-status` command on the node.

   Control/Master nodes

```
$ contrail-status
Must show that all services are active:
== Contrail control ==
control: active
nodemgr: active
named: active
dns: active

== Contrail analytics-alarm ==
nodemgr: active
kafka: active
alarm-gen: active
```

```
== Contrail kubernetes ==
kube-manager: backup

== Contrail database ==
nodemgr: active
query-engine: active
cassandra: active

== Contrail analytics ==
nodemgr: active
api: active
collector: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
cassandra: active

== Contrail webui ==
web: active
job: active

== Contrail vrouter ==
nodemgr: active
agent: active

== Contrail analytics-snmp ==
snmp-collector: active
nodemgr: active
topology: active

== Contrail config ==
svc-monitor: backup
nodemgr: active
device-manager: backup
api: active
schema: backup
```

Worker Nodes

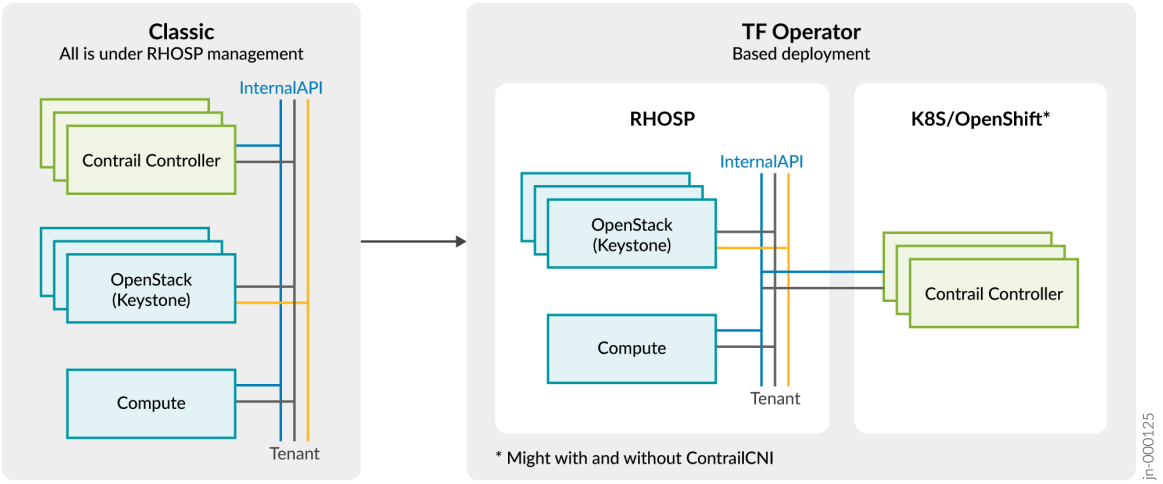
```
vrouter kernel module is PRESENT
== Contrail vrouter ==
nodemgr: active
agent: active
```

Deploying Red Hat Openstack with Contrail Control Plane Managed by Tungsten Fabric Operator

This document provides steps needed to use the Operator Framework for Contrail Networking deployment that is using Red Hat Openstack version 16.1 (RHOSP16.1) as it's orchestration platform.

Figure 34 on page 227 shows the difference between the classic deployment of RHOSP16.1 and the Tungsten Fabric (TF) Operator based deployment.

Figure 34: Difference Between RHOSP16.1 Classic Deployment and TF-Operator



- RHOSP does not deploy ContrailController, ContrailAnalytics, and ContrailAnalyticsDatabase Overcloud roles.
- Compute nodes are deployed and managed by RHOSP.
- Contrail Control plane is deployed separately on top of Kubernetes or OpenShift cluster.

- Contrail Control plane nodes require access to RHOSP networks such as Internal API and Tenant. Administrator must configure the Internal API and Tenant, if Kubernetes or OpenShift cluster is deployed outside of the RHOSP networks.

To deploy RHOSP based on TF-Operator:

1. Deploy Kubernetes or OpenShift cluster with Contrail Control plane without the keystone options.
   a. Generate a self-signed CA certificate and key and provide their content in the environmental variables.

      ```
      export SSL_CAKEY=$(cat ca_key.pem)
      export SSL_CACERT=$(cat ca.pem)
      ```

      If Red Hat IdM is used for RHOSP, the RHOSP must bundle its own CA and IPA CA.

      ```
      export SSL_CACERT=$(cat ca.crt.pem /etc/ipa/ca.crt)
      ```

   b. To know more about Kubernetes procedure, see here.

   c. To know more about OpenShift procedure, see here.
2. Deploy RHOSP16 without the Contrail Control plane roles.

   Follow the deployment procedure provided here.

   a. Do not create virtual machines (VMs) for Control Plane and skip any related steps.

   b. For Transport Layer Security (TLS), use the self-signed certificates. The Kubernetes or OpenShift is not integrated with Red Hat IdM.

      Use the generated CA and SSH key in Step1 to prepare **environments/contrail/contrail-tls.yaml**.

   c. Set the counters to zero for ContrailController, Analytics, and database roles.

      For example, ContrailControllerCount: 0.

   d. Provide the heat parameters addresses to Contrail Control plane deployed by Kubernetes or OpenShift.

      ```
      ExtraHostFileEntries:
      'ip1 <FQDN K8S master1> <Short name master1>'
      ```

```
'ip2 <FQDN K8S master2> <Short name master2>'
'ip3 <FQDN K8S master3> <Short name master3>'
```

```
ExternalContrailConfigIPs: <ip1>,<ip2>,<ip3>
ExternalContrailControlIPs: <ip1>,<ip2>,<ip3>
ExternalContrailAnalyticsIPs: <ip1>,<ip2>,<ip3>
```

```
ControllerExtraConfig:
    contrail_internal_api_ssl: True
ComputeExtraConfig:
    contrail_internal_api_ssl: True
# add contrail_internal_api_ssl for all other roles if any
```

3. Ensure that Contrail Control plane deployed by Kubernetes or OpenShift has connectivity to RHOSP InternalAPI and tenant networks.

```
# Assuming that OpenShift network name is 'ocp' with CIDR 192.168.123.0/24,
# Allow forwarding from RHOSP network to OpenShift network
sudo iptables -I LIBVIRT_FWI 1 -i prov-20 -d 192.168.123.0/24 -o ocp -j ACCEPT
sudo iptables -I LIBVIRT_FWI 1 -i mgmt-20 -d 192.168.123.0/24 -o ocp -j ACCEPT
# Allow forwarding from OpenShift network to RHOSP network
sudo iptables -I LIBVIRT_FWI 1 -i ocp -d 192.168.20.0/24 -o mgmt-20 -j ACCEPT
sudo iptables -I LIBVIRT_FWI 1 -i ocp -d 192.168.21.0/24 -o prov-20 -j ACCEPT
```

4. Ensure that Contrail Control plane nodes deployed by Kubernetes or OpenShift are able to resolve RHOSP FQDNs for Internal API and Control Plane networks. For example, add names to /etc/hosts on Contrail Control plane nodes.

```
cat << EOF | sudo tee -a /etc/hosts
192.168.21.200 overcloud.ctlplane.dev.localdomain
    192.168.22.200 overcloud.internalapi.dev.localdomain
    192.168.21.200 overcloud.dev.localdomain
    EOF
```

5. Connect Contrail Control plane to RHOSP keystone.

```
# Adjust options according to overcloudrc
export AUTH_MODE='keystone'
```

```
export IPFABRIC_SERVICE_HOST='192.168.21.200'
export KEYSTONE_AUTH_HOST='192.168.21.200'
export KEYSTONE_AUTH_PROTO='http'
export KEYSTONE_AUTH_ADMIN_PASSWORD='qwe123QWE'
export KEYSTONE_AUTH_REGION_NAME='regionOne'
./tf-operator/contrib/render_manifests.sh
oc apply -k ./tf-operator/deploy/kustomize/contrail/templates/
```

6. Wait until contrail-status shows active for Control plane and for RHOSP computes.

CHAPTER 5

Backup and Restore Contrail Software

IN THIS CHAPTER

How to Backup and Restore Contrail Databases in JSON Format in Openstack Environments Using the Openstack 16.1 Director Deployment

IN THIS SECTION

This document shows how to backup and restore the Contrail databases—Cassandra and Zookeeper—in JSON format when Contrail Networking is running in Openstack-orchestrated environments that were deployed using the RedHat Openstack 16.1 director deployment.

If you are deploying Contrail Networking in an Openstack-orchestrated environment that was deployed using an Openstack 13-based or Ansible deployer, see How to Backup and Restore Contrail Databases in JSON Format in Openstack Environments Using the Openstack 13 or Ansible Deployer.

Contrail Networking is initially supported in Openstack environments using the Openstack 16.1 director deployment in Contrail Networking Release 2008. See Contrail Networking Supported Platforms for a matrix of Contrail Networking release support within orchestration platforms and deployers.

Before You Begin

The backup and restore procedure must be completed for nodes running the same Contrail Networking release. The procedure is used to backup the Contrail Networking databases only; it does not include instructions for backing up orchestration system databases.

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

Simple Database Backup in JSON Format

This procedure provides a simple database backup in JSON format. This procedure is performed using the `db_json_exim.py` script located inside the config-api container in `/usr/lib/python2.7/site-packages/cfgm_common` on the controller node.

To perform this database backup:

1. From a controller node, ensure the `db_json_exim.py` script is available:

```
(overcloud) [user@overcloud-contrailcontroller-0 heat-admin]# podman exec -it
contrail_config_api bash
(config-api)[user@overcloud-contrailcontroller-0 /]$ ls /usr/lib/python2.7/site-packages/
cfgm_common/db_json_exim.py
/usr/lib/python2.7/site-packages/cfgm_common/db_json_exim.py
```

2. Log into one of the contrail controller nodes. Create the **/tmp/db-dump** directory on any of the contrail controller node hosts.

```
mkdir /tmp/db-dump
```

NOTE: **/tmp/db-dump** is a user-defined directory name. You can assign any name to a directory.

3. On the same contrail controller node, copy the `contrail-api.conf` file from the container to the host.

```
podman cp contrail_config_api:/etc/contrail/contrail-api-0.conf /tmp/db-dump/contrail-
api.conf
```

The Cassandra database instance on any contrail controller node includes the complete Cassandra database for all contrail controller nodes in the cluster. Steps 2 and 3, therefore, only need to be performed on one contrail controller node.

4. On all Contrail controller nodes, stop the following Contrail configuration services:

```
systemctl stop tripleo_contrail_config_svc_monitor.service
systemctl stop tripleo_contrail_config_device_manager.service
systemctl stop tripleo_contrail_config_schema.service
systemctl stop tripleo_contrail_config_api.service
systemctl stop tripleo_contrail_config_nodemgr.service
systemctl stop tripleo_contrail_config_database_nodemgr.service
```

This step must be performed on each individual controller node in the cluster.

5. On all nodes hosting Contrail Analytics containers, stop the following analytics services:

```
systemctl  stop tripleo_contrail_analytics_kafka.service
systemctl stop tripleo_contrail_analytics_snmp_nodemgr.service
systemctl stop tripleo_contrail_analytics_alarmgen.service
systemctl stop tripleo_contrail_analytics_alarm_nodemgr.service
systemctl stop tripleo_contrail_analytics_topology.service
systemctl stop tripleo_contrail_analytics_collector.service
systemctl stop tripleo_contrail_analytics_nodemgr.service
systemctl stop tripleo_contrail_analytics_snmp_collector.service
systemctl stop tripleo_contrail_analytics_api.service
```

This step must be performed on each individual analytics node in the cluster.

6. Return to the contrail controller node where you performed steps 2 and 3.

Use the **podman images** command to list the name or ID of the *config api* image.

```
podman images | grep config-api
```

Example:

```
(overcloud) [user@overcloud-contrailcontroller-0 db-dump]# podman images | grep config-api
192.168.24.1:8787/contrail/contrail-controller-config-api
2011.L1.297   2dcd2feaeed5   2 months ago   876 MB
```

7. From the same contrail controller node, start the *config api* container by pointing the `entrypoint.sh` script to the **/bin/bash** directory then mapping **/tmp/db-dump** directory from the host to the **/tmp** directory inside the container. You perform this step to ensure that the API services are not started on the contrail controller node.

   Enter the *-v /etc/contrail/ssl:/etc/contrail/ssl:ro* command option when `cassandra_use_ssl` is used as the api-server configuration parameter to ensure TLS certificates are mounted to the Contrail SSL directory. This mounting ensures that the backup procedure succeeds in environments with endpoints that require TLS authentication.

   The *registry_name* and *container_tag* variables must match step "6" on page 233.

```
podman run --rm -it -v /tmp/db-dump/:/tmp:Z -v /etc/contrail/ssl:/etc/contrail/ssl:ro -
v /etc/ipa:/etc/ipa:ro --network host --entrypoint=/bin/bash registry_name/contrail-
controller-config-api:container_tag
```

Example:

```
podman run --rm -it -v /tmp/db-dump/:/tmp:Z -v /etc/contrail/ssl:/etc/contrail/ssl:ro -
v /etc/ipa:/etc/ipa:ro --network host --entrypoint=/bin/bash 192.168.24.1:8787/contrail-
nightly/contrail-controller-config-api:21.4.L3.441
```

NOTE: The cluster is TLS-enabled if TLS-related files are used during Contrail Networking deployment. The TLS-related files are part of the deployment script. These are custom files and can have any name. For example:

```
 ~/tripleo-heat-templates/environments/contrail/contrail-tls.yaml \
 ~/tripleo-heat-templates/environments/ssl/tls-everywhere-endpoints-dns.yaml \
 ~/tripleo-heat-templates/environments/services/haproxy-public-tls-
certmonger.yaml \
 ~/tripleo-heat-templates/environments/ssl/enable-internal-tls.yaml \
```

8. From the container created on the contrail controller node in Step "7" on page 234, use the `db_json_exim.py` script to backup data in JSON format. The db dump file will be saved in the **/tmp/db-dump/** directory on this contrail controller node. Example:

```
(config-api)[user@overcloud-contrailcontroller-0 /]$ cd /usr/lib/python2.7/site-packages/
cfgm_common
(config-api)[user@overcloud-contrailcontroller-0 /usr/lib/python2.7/site-packages/
cfgm_common]$ python db_json_exim.py --export-to /tmp/db-dump.json --api-conf /tmp/contrail-
api.conf
2021-06-30 19:47:27,120 INFO: Cassandra DB dumped
2021-06-30 19:47:28,878 INFO: Zookeeper DB dumped
2021-06-30 19:47:28,895 INFO: DB dump wrote to file /tmp/db-dump.json
```

   The Cassandra database instance on any contrail controller node includes the complete Cassandra database for all contrail controller nodes in the cluster. You, therefore, only need to perform step 4 through 6 from one of the contrail controller nodes.

9. (Optional. Recommended) From the same contrail controller node, enter the `cat /tmp/db-dump.json | python -m json.tool | less` command to view a more readable version of the file transfer.

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

10. From the same contrail controller node, exit out of the *config api* container. This will stop the container.

```
exit
```

11. On each contrail controller node, start the following configuration services:

```
systemctl start tripleo_contrail_config_svc_monitor.service
systemctl start tripleo_contrail_config_device_manager.service
systemctl start tripleo_contrail_config_schema.service
systemctl start tripleo_contrail_config_api.service
systemctl start tripleo_contrail_config_nodemgr.service
systemctl start tripleo_contrail_config_database_nodemgr.service
```

    This step must be performed on each individual contrail controller node.

12. On each analytics node, start the following analytics services:

```
systemctl  start tripleo_contrail_analytics_kafka.service
systemctl  start tripleo_contrail_analytics_snmp_nodemgr.service
systemctl  start tripleo_contrail_analytics_alarmgen.service
systemctl  start tripleo_contrail_analytics_alarm_nodemgr.service
systemctl  start tripleo_contrail_analytics_topology.service
systemctl  start tripleo_contrail_analytics_collector.service
systemctl  start tripleo_contrail_analytics_nodemgr.service
systemctl  start tripleo_contrail_analytics_snmp_collector.service
systemctl  start tripleo_contrail_analytics_api.service
```

This step must be performed on each individual analytics node.

13. On each contrail controller node, enter the `contrail-status` command to confirm that services are in the *active* or *running* states.

NOTE: Some command output and output fields are removed for readability. Output shown is from a single node hosting configuration and analytics services.

```
contrail-status
Pod             Service     Original Name                 State
analytics       api         contrail-analytics-api        running
analytics       collector   contrail-analytics-collector  running
analytics       nodemgr     contrail-nodemgr              running
analytics       provisioner contrail-provisioner          running
analytics       redis       contrail-external-redis       running
analytics-alarm alarm-gen   contrail-analytics-alarm-gen  running
analytics-alarm kafka       contrail-external-kafka       running
<some output removed for readability>

== Contrail control ==
control: active
nodemgr: active
named: active
dns: active

== Contrail analytics-alarm ==
nodemgr: active
kafka: active
```

```
alarm-gen: active

== Contrail database ==
nodemgr: active
query-engine: active
cassandra: active

== Contrail analytics ==
nodemgr: active
api: active
collector: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
cassandra: active

== Contrail webui ==
web: active
job: active

== Contrail analytics-snmp ==
snmp-collector: active
nodemgr: active
topology: active

== Contrail config ==
svc-monitor: active
nodemgr: active
device-manager: active
api: active
schema: active
```

Restore Database from the Backup in JSON Format

This procedure provides the steps to restore a system using the simple database backup JSON file that was created in "Simple Database Backup in JSON Format" on page 232.

To restore a system from a backup JSON file:

1. Copy the `contrail-api.conf` file from the container to the host on any one of the contrail controller nodes.

```
podman cp contrail_config_api:/etc/contrail/contrail-api-0.conf /tmp/db-dump/contrail-
api.conf
```

2. On all of the Contrail controller nodes, stop these configuration services:

```
systemctl stop tripleo_contrail_config_svc_monitor.service
systemctl stop tripleo_contrail_config_device_manager.service
systemctl stop tripleo_contrail_config_schema.service
systemctl stop tripleo_contrail_config_api.service
systemctl stop tripleo_contrail_config_nodemgr.service
systemctl stop tripleo_contrail_config_database_nodemgr.service
```

3. On all nodes hosting Contrail Analytics containers, stop the following services:

```
systemctl stop tripleo_contrail_analytics_kafka.service
systemctl stop tripleo_contrail_analytics_snmp_nodemgr.service
systemctl stop tripleo_contrail_analytics_alarmgen.service
systemctl stop tripleo_contrail_analytics_alarm_nodemgr.service
systemctl stop tripleo_contrail_analytics_topology.service
systemctl stop tripleo_contrail_analytics_collector.service
systemctl stop tripleo_contrail_analytics_nodemgr.service
systemctl stop tripleo_contrail_analytics_snmp_collector.service
systemctl stop tripleo_contrail_analytics_api.service
```

4. Stop the Cassandra service on all the `config-db` controllers.

```
systemctl stop tripleo_contrail_config_database.service
```

5. Stop the Zookeeper service on all the contrail controllers.

```
systemctl stop tripleo_contrail_config_zookeeper.service
```

6. Stop the Zookeeper service on all the contrail-analytics controllers.

```
systemctl stop tripleo_contrail_analytics_zookeeper.service
```

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

```
cd /var/lib/contrail/config_zookeeper
cp -aR version-2/ zookper-bkp.save
```

8. Delete the Zookeeper data directory contents on all the controllers.

```
rm -rf version-2
```

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

```
cd /var/lib/contrail/config_cassandra
cp -aR data/ Cassandra_data-save
```

10. Delete the Cassandra data directory contents on all controllers.

```
rm -rf data/
```

11. Start the Zookeeper service on all the contrail controllers.

```
systemctl start tripleo_contrail_config_zookeeper.service
```

12. Start the Zookeeper service on all the contrail-analytics controllers.

```
systemctl start tripleo_contrail_analytics_zookeeper.service
```

13. Start the Cassandra service on all of the controllers.

```
systemctl start tripleo_contrail_config_database.service
```

14. Use the **podman images** command to list the name or ID of the *config api* image.

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

Example:

```
user@overcloud-contrailcontroller-0 heat-admin]# podman image ls | grep config-a
192.168.24.1:8787/contrail/contrail-controller-config-api
2011.L1.297    2dcd2feaeed5    1 months ago    876 MB
```

15. Run a new podman container using the name or ID of the `config_api` image on the same contrail controller node.

    Enter the *-v /etc/contrail/ssl:/etc/contrail/ssl:ro* command option when `cassandra_use_ssl` is used as api-server configuration parameter to ensure TLS certificates are mounted to the Contrail SSL directory. This mounting ensures that this backup procedure succeeds in environments with endpoints that require TLS authentication.

    Use the *registry_name* and *container_tag* from the output of the step "14" on page 240.

```
podman run --rm -it -v /tmp/db-dump/:/tmp:Z -v /etc/contrail/ssl:/etc/contrail/ssl:ro --
network host --entrypoint=/bin/bash <registry_name>/contrail-controller-config-
api:<container tag>
```

Example:

```
podman run --rm -it -v /tmp/db-dump/:/tmp:Z -v /etc/contrail/ssl:/etc/contrail/ssl:ro --
network host --entrypoint=/bin/bash 192.168.24.1:8787/contrail/contrail-controller-config-
api:2011.L1.297
```

16. Restore the data in the new running container on the same contrail controller node.

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

Example:

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

```
2021-07-06 17:22:17,157 INFO: DB dump file loaded
2021-07-06 17:23:12,227  INFO: Cassandra DB restored
2021-07-06 17:23:14,236  INFO: Zookeeper DB restored
```

17. Exit out of the *config api* container. This will stop the container.

```
exit
```

18. Start config services on all of the controllers:

```
systemctl start tripleo_contrail_config_svc_monitor.service
systemctl start tripleo_contrail_config_device_manager.service
systemctl start tripleo_contrail_config_schema.service
systemctl start tripleo_contrail_config_api.service
systemctl start tripleo_contrail_config_nodemgr.service
systemctl start tripleo_contrail_config_database_nodemgr.service
```

19. Start services on all of the analytics nodes:

```
systemctl  start tripleo_contrail_analytics_kafka.service
systemctl  start tripleo_contrail_analytics_snmp_nodemgr.service
systemctl  start tripleo_contrail_analytics_alarmgen.service
systemctl  start tripleo_contrail_analytics_alarm_nodemgr.service
systemctl  start tripleo_contrail_analytics_topology.service
systemctl start tripleo_contrail_analytics_collector.service
systemctl  start tripleo_contrail_analytics_nodemgr.service
systemctl  start tripleo_contrail_analytics_snmp_collector.service
systemctl  start tripleo_contrail_analytics_api.service
```

20. Enter the `contrail-status` command on each contrail controller node and, when applicable, on each analytics node to confirm that services are in the *active* or *running* states.

NOTE: Output shown for a contrail controller node. Some command output and output fields are removed for readability.

```
contrail-status
Pod     Service         Original Name                          State
```

```
config  api             contrail-controller-config-api        running
config  device-manager  contrail-controller-config-devicemgr  running
config  dnsmasq         contrail-controller-config-dnsmasq    running
config  nodemgr         contrail-nodemgr                      running
config  provisioner     contrail-provisioner                  running
config  schema          contrail-controller-config-schema     running
config  stats           contrail-controller-config-stats      running
<some output removed for readability>

== Contrail control ==
control: active
nodemgr: active
named: active
dns: active


== Contrail database ==
nodemgr: active
query-engine: active
cassandra: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
cassandra: active

== Contrail webui ==
web: active
job: active

== Contrail config ==
svc-monitor: active
nodemgr: active
device-manager: active
api: active
schema: active
```

How to Backup and Restore Contrail Databases in JSON Format in Openstack Environments Using the Openstack 13 or Ansible Deployers

IN THIS SECTION

This document shows how to backup and restore the Contrail databases—Cassandra and Zookeeper—in JSON format when Contrail Networking is running in Openstack-orchestrated environments that were deployed using an Openstack 13-based or Ansible deployer. For a list of Openstack-orchestrated environments that are deployed using Openstack 13-based or Ansible deployers, see the Contrail Networking Supported Platforms matrix.

If you are deploying Contrail Networking in an Openstack-orchestrated environment that was deployed using an Openstack 16-based deployer, see How to Backup and Restore Contrail Databases in JSON Format in Openstack Environments Using the Openstack 16.1 Deployer.

Before You Begin

The backup and restore procedure must be completed for nodes running the same Contrail Networking release. The procedure is used to backup the Contrail Networking databases only; it does not include instructions for backing up orchestration system databases.

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

Simple Database Backup in JSON Format

This procedure provides a simple database backup in JSON format. This procedure is performed using the `db_json_exim.py` script located in the `/usr/lib/python2.7/site-packages/cfgm_common` on the controller node.

To perform this database backup:

1. Log into one of the contrail controller nodes. Create the **/tmp/db-dump** directory on any of the contrail controller node hosts.

```
mkdir /tmp/db-dump
```

2. On the same contrail controller node, copy the `contrail-api.conf` file from the container to the host.

   Ansible Deployer:

```
docker cp config_api_1:/etc/contrail/contrail-api.conf /tmp/db-dump/
```

   Red Hat Openstack Deployer:

```
docker cp contrail_config_api:/etc/contrail/contrail-api.conf /tmp/db-dump/
```

   The Cassandra database instance on any contrail controller node includes the complete Cassandra database for all contrail controller nodes in the cluster. Steps 1 and 2, therefore, only need to be performed on one contrail controller node.

3. Stop the following docker configuration services on all of the contrail controller nodes.

   Ansible Deployer:

```
docker stop config_svcmonitor_1
docker stop config_devicemgr_1
docker stop config_schema_1
docker stop config_api_1
```

   Red Hat Openstack Deployer:

```
docker stop contrail_config_svc_monitor
docker stop contrail_config_device_manager
```

```
docker stop contrail_config_schema
docker stop contrail_config_api
```

This step must be performed on each individual contrail controller node in the cluster.

4. Return to the contrail controller node where you performed steps 1 and 2.

   List the docker image to find the name or ID of the *config api* image.

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

   Example:

   ```
   docker image ls | grep config-api
   hub.juniper.net/contrail/contrail-controller-config-api 1909.30-ocata c9d757252a0c  4
   months ago  583MB
   ```

5. From the same contrail controller node, start the *config api* container pointing the `entrypoint.sh` script to **/bin/bash** and mapping **/tmp/db-dump** from the host to the **/tmp** directory inside the container. You perform this step to ensure that the API services are not started on the contrail controller node.

   Enter the *-v /etc/contrail/ssl:/etc/contrail/ssl:ro* command option when `cassandra_use_ssl` is used as api-server configuration parameter to ensure TLS certificates are mounted to the Contrail SSL directory. This mounting ensures that the backup procedure succeeds in environments with endpoints that require TLS authentication.

   The *registry_name* and *container_tag* variables must match step "4" on page 245.

   ```
   docker run --rm -it -v /tmp/db-dump/:/tmp/ -v /etc/contrail/ssl:/etc/contrail/ssl:ro --
   network host --entrypoint=/bin/bash <registry_name>/contrail-controller-
   config_api:<container_tag>
   ```

   Example:

   ```
   docker run --rm -it -v /tmp/db-dump/:/tmp/ -v /etc/contrail/ssl:/etc/contrail/ssl:ro --
   network host --entrypoint=/bin/bash hub.juniper.net/contrail/contrail-controller-config-
   api:1909.30-ocata
   ```

6. From the docker container created on the contrail controller node in Step "5" on page 245, use the `db_json_exim.py` script to backup data in JSON format.. The db dump file will be saved in the **/tmp/db-dump/** on this contrail controller node.

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

The Cassandra database instance on any contrail controller node includes the complete Cassandra database for all contrail controller nodes in the cluster. You, therefore, only need to perform step 4 through 6 from one of the contrail controller nodes.

7. (Optional. Recommended) From the same contrail controller node, enter the `cat /tmp/db-dump.json | python -m json.tool | less` command to view a more readable version of the file transfer.

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

8. From the same contrail controller node, exit out of the *config api* container. This will stop the container.

```
exit
```

9. Start the following configuration services on all of the contrail controller nodes.

Ansible Deployer:

```
docker start config_api_1
docker start config_schema_1
docker start config_svcmonitor_1
docker start config_devicemgr_1
```

Red Hat Openstack Deployer:

```
docker start contrail_config_api
docker start contrail_config_schema
docker start contrail_config_svc_monitor
docker start contrail_config_device_manager
```

This step must be performed on each individual contrail controller node.

10. On each contrail controller node, enter the `contrail-status` command to confirm that services are in the *active* or *running* states.

NOTE: Some command output and output fields are removed for readability. Output shown is from a node hosting configuration and analytics services.

```
contrail-status
Pod             Service     Original Name                  State
analytics       api         contrail-analytics-api         running
analytics       collector   contrail-analytics-collector   running
analytics       nodemgr     contrail-nodemgr               running
analytics       provisioner contrail-provisioner           running
analytics       redis       contrail-external-redis        running
analytics-alarm alarm-gen   contrail-analytics-alarm-gen   running
analytics-alarm kafka       contrail-external-kafka        running
<some output removed for readability>

== Contrail control ==
control: active
nodemgr: active
named: active
dns: active

== Contrail analytics-alarm ==
nodemgr: active
kafka: active
alarm-gen: active

== Contrail database ==
nodemgr: active
query-engine: active
cassandra: active

== Contrail analytics ==
nodemgr: active
api: active
collector: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
```

```
rabbitmq: active
cassandra: active

== Contrail webui ==
web: active
job: active

== Contrail analytics-snmp ==
snmp-collector: active
nodemgr: active
topology: active

== Contrail config ==
svc-monitor: active
nodemgr: active
device-manager: active
api: active
schema: active
```

Examples: Simple Database Backups in JSON Format

These examples illustrate the process for creating a simple database backup in JSON format in both an Ansible deployer environment and a Red Hat Openstack deployer environment.

In each example, a cluster with three config nodes—**control_config1**, **control_config2**, and **control_config3**—is backed up. All tasks that need to be performed on a single config nodes are performed on **control-config1**. The tasks must be performed in the shown order.

Ansible Deployer Environment:

```
## control_config1 ##
mkdir /tmp/db-dump
docker cp config_api_1:/etc/contrail/contrail-api.conf /tmp/db-dump/
docker stop config_svcmonitor_1
docker stop config_devicemgr_1
docker stop config_schema_1
docker stop config_api_1

## control_config2 ##
docker stop config_svcmonitor_1
docker stop config_devicemgr_1
```

```
docker stop config_schema_1
docker stop config_api_1

## control_config3 ##
docker stop config_svcmonitor_1
docker stop config_devicemgr_1
docker stop config_schema_1
docker stop config_api_1

## control_config1 ##
docker run --rm -it -v /tmp/db-dump/:/tmp/ -v /etc/contrail/ssl:/etc/contrail/ssl:ro --network
host --entrypoint=/bin/bash hub.juniper.net/contrail/contrail-controller-config-api:1909.30-ocata
cd /usr/lib/python2.7/site-packages/cfgm_common
python db_json_exim.py --export-to /tmp/db-dump.json --api-conf /tmp/contrail-api.conf
cat /tmp/db-dump.json | python -m json.tool | less
exit
docker start config_api_1
docker start config_schema_1
docker start config_svcmonitor_1
docker start config_devicemgr_1
contrail-status

## control_config2 ##
docker start config_api_1
docker start config_schema_1
docker start config_svcmonitor_1
docker start config_devicemgr_1
contrail-status

## control_config3 ##
docker start config_api_1
docker start config_schema_1
docker start config_svcmonitor_1
docker start config_devicemgr_1
contrail-status
```

Red Hat Openstack Deployer Environment:

```
## control_config1 ##
mkdir /tmp/db-dump
docker cp contrail_config_api:/etc/contrail/contrail-api.conf /tmp/db-dump/
docker stop contrail_config_svc_monitor
```

```
docker stop contrail_config_device_manager
docker stop contrail_config_schema
docker stop contrail_config_api

## control_config2 ##
docker stop contrail_config_svc_monitor
docker stop contrail_config_device_manager
docker stop contrail_config_schema
docker stop contrail_config_api

## control_config3 ##
docker stop contrail_config_svc_monitor
docker stop contrail_config_device_manager
docker stop contrail_config_schema
docker stop contrail_config_api

## control_config1 ##
docker run --rm -it -v /tmp/db-dump/:/tmp/ -v /etc/contrail/ssl:/etc/contrail/ssl:ro --network
host --entrypoint=/bin/bash hub.juniper.net/contrail/contrail-controller-config-api:1909.30-ocata
cd /usr/lib/python2.7/site-packages/cfgm_common
python db_json_exim.py --export-to /tmp/db-dump.json --api-conf /tmp/contrail-api.conf
cat /tmp/db-dump.json | python -m json.tool | less
exit
docker start contrail_config_api
docker start contrail_config_schema
docker start contrail_config_svc_monitor
docker start contrail_config_device_manager
contrail-status

## control_config2 ##
docker start contrail_config_api
docker start contrail_config_schema
docker start contrail_config_svc_monitor
docker start contrail_config_device_manager
contrail-status

## control_config3 ##
docker start contrail_config_api
docker start contrail_config_schema
docker start contrail_config_svc_monitor
docker start contrail_config_device_manager
contrail-status
```

Restore Database from the Backup in JSON Format

This procedure provides the steps to restore a system using the simple database backup JSON file that was created in "Simple Database Backup in JSON Format" on page 244.

To restore a system from a backup JSON file:

1. Copy the `contrail-api.conf` file from the container to the host on any one of the config nodes.

   Ansible Deployer:

   ```
   docker cp config_api_1:/etc/contrail/contrail-api.conf /tmp/db-dump/
   ```

   Red Hat Openstack Deployer:

   ```
   docker cp contrail_config_api:/etc/contrail/contrail-api.conf /tmp/db-dump/
   ```

2. Stop the configuration services on all of the controllers.

   Ansible Deployer:

   ```
   docker stop config_svcmonitor_1
   docker stop config_devicemgr_1
   docker stop config_schema_1
   docker stop config_api_1
   docker stop config_nodemgr_1
   docker stop config_database_nodemgr_1
   docker stop analytics_snmp_snmp-collector_1
   docker stop analytics_snmp_topology_1
   docker stop analytics_alarm_alarm-gen_1
   docker stop analytics_api_1
   docker stop analytics_collector_1
   docker stop analytics_alarm_kafka_1
   ```

   Red Hat Openstack Deployer—Node hosting Contrail Config containers:

   ```
   docker stop contrail_config_svc_monitor
   docker stop contrail_config_device_manager
   docker stop contrail_config_schema
   docker stop contrail_config_api
   ```

```
docker stop contrail_config_nodemgr
docker stop contrail_config_database_nodemgr
```

Red Hat Openstack Deployer—Node hosting Contrail Analytics containers:

```
docker stop contrail_analytics_snmp_collector
docker stop contrail_analytics_topology
docker stop contrail_analytics_alarmgen
docker stop contrail_analytics_api
docker stop contrail_analytics_collector
docker stop contrail_analytics_kafka
```

3. Stop the Cassandra service on all the `config-db` controllers.

   Ansible Deployer:

   ```
   docker stop config_database_cassandra_1
   ```

   Red Hat Openstack Deployer:

   ```
   docker stop contrail_config_database
   ```

4. Stop the Zookeeper service on all controllers.

   Ansible Deployer:

   ```
   docker stop config_database_zookeeper_1
   ```

   Red Hat Openstack Deployer:

   ```
   docker stop contrail_config_zookeeper
   ```

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

   Ansible Deployer:

   ```
   cd /var/lib/docker/volumes/config_database_config_zookeeper/
   cp -R _data/version-2/ version-2-save
   ```

Red Hat Openstack Deployer:

```
cd /var/lib/docker/volumes/config_zookeeper/
cp -R _data/version-2/ version-2-save
```

6. Delete the Zookeeper data directory contents on all the controllers.

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

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

Ansible Deployer:

```
cd /var/lib/docker/volumes/config_database_config_cassandra/
cp -R _data/ Cassandra_data-save
```

Red Hat Openstack Deployer:

```
cd /var/lib/docker/volumes/config_cassandra/
cp -R _data/ Cassandra_data-save
```

8. Delete the Cassandra data directory contents on all controllers.

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

9. Start the Zookeeper service on all the controllers.

Ansible Deployer:

```
docker start config_database_zookeeper_1
```

Red Hat Openstack Deployer:

```
docker start contrail_config_zookeeper
```

10. Start the Cassandra service on all the controllers.

Ansible Deployer:

```
docker start config_database_cassandra_1
```

Red Hat Openstack Deployer:

```
docker start contrail_config_database
```

11. List docker image to find the name or ID of the `config-api` image on the config node.

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

Example:

```
docker image ls | grep config-api
hub.juniper.net/contrail/contrail-controller-config-api 1909.30-ocata c9d757252a0c  4
months ago  583MB
```

12. Run a new docker container using the name or ID of the `config_api` image on the same config node.

Enter the *-v /etc/contrail/ssl:/etc/contrail/ssl:ro* command option when `cassandra_use_ssl` is used as api-server configuration parameter to ensure TLS certificates are mounted to the Contrail SSL directory. This mounting ensures that this backup procedure succeeds in environments with endpoints that require TLS authentication.

Use the *registry_name* and *container_tag* from the output of the step "11" on page 254.

```
docker run --rm -it -v /tmp/db-dump/:/tmp/ -v /etc/contrail/ssl:/etc/contrail/ssl:ro --
network host --entrypoint=/bin/bash <registry_name>/contrail-controller-
config_api:<container tag>
```

Example

```
docker run --rm -it -v /tmp/db-dump/:/tmp/ -v /etc/contrail/ssl:/etc/contrail/ssl:ro --
network host --entrypoint=/bin/bash hub.juniper.net/contrail/contrail-controller-config-
api:1909.30-ocata
```

13. Restore the data in new running docker on the same config node.

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

14. Exit out of the *config api* container. This will stop the container.

```
exit
```

15. Start config services on all the controllers.

Ansible Deployer:

```
docker start config_svcmonitor_1
docker start config_devicemgr_1
docker start config_schema_1
docker start config_api_1
docker start config_nodemgr_1
docker start config_database_nodemgr_1
docker start analytics_snmp_snmp-collector_1
docker start analytics_snmp_topology_1
docker start analytics_alarm_alarm-gen_1
docker start analytics_api_1
docker start analytics_collector_1
docker start analytics_alarm_kafka_1
```

Red Hat Openstack Deployer—Node hosting Contrail Config containers:

```
docker start contrail_config_svc_monitor
docker start contrail_config_device_manager
docker start contrail_config_schema
docker start contrail_config_api
docker start contrail_config_nodemgr
docker start contrail_config_database_nodemgr
```

Red Hat Openstack Deployer—Node hosting Contrail Analytics containers:

```
docker start contrail_analytics_snmp_collector
docker start contrail_analytics_topology
```

```
docker start contrail_analytics_alarmgen
docker start contrail_analytics_api
docker start contrail_analytics_collector
docker start contrail_analytics_kafka
```

16. Enter the `contrail-status` command on each configuration node and, when applicable, on each analytics node to confirm that services are in the *active* or *running* states.

NOTE: Output shown for a config node. Some command output and output fields are removed for readability.

```
contrail-status
Pod     Service         Original Name                          State
config  api             contrail-controller-config-api         running
config  device-manager  contrail-controller-config-devicemgr   running
config  dnsmasq         contrail-controller-config-dnsmasq     running
config  nodemgr         contrail-nodemgr                       running
config  provisioner     contrail-provisioner                   running
config  schema          contrail-controller-config-schema      running
config  stats           contrail-controller-config-stats       running
<some output removed for readability>

== Contrail control ==
control: active
nodemgr: active
named: active
dns: active


== Contrail database ==
nodemgr: active
query-engine: active
cassandra: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
cassandra: active
```

```
== Contrail webui ==
web: active
job: active

== Contrail config ==
svc-monitor: active
nodemgr: active
device-manager: active
api: active
schema: active
```

Example: How to Restore a Database Using the JSON Backup (Ansible Deployer Environment)

This example shows how to restore the databases for three controllers connected to the Contrail Configuration database (config-db). This example assumes a JSON backup file of the databases was previously created using the instructions provided in "Simple Database Backup in JSON Format" on page 244.The network was deployed using Ansible and the three controllers—nodec53, nodec54, and nodec55—have separate IP addresses.

```
## Make db-dump directory. Copy contrail-api.conf to db-dump directory. ##
root@nodec54 ~]# mkdir /tmp/db-dump
root@nodec54 ~]# docker cp config_api_1:/etc/contrail/contrail-api.conf /tmp/db-dump/

## Stop Configuration Services on All Controllers ##
[root@nodec53 ~]# docker stop config_schema_1
[root@nodec53 ~]# docker stop config_api_1
[root@nodec53 ~]# docker stop config_svcmonitor_1
[root@nodec53 ~]# docker stop config_devicemgr_1
[root@nodec53 ~]# docker stop config_nodemgr_1
[root@nodec53 ~]# docker stop config_database_nodemgr_1
[root@nodec53 ~]# docker stop analytics_snmp_snmp-collector_1
[root@nodec53 ~]# docker stop analytics_snmp_topology_1
[root@nodec53 ~]# docker stop analytics_alarm_alarm-gen_1
[root@nodec53 ~]# docker stop analytics_api_1
[root@nodec53 ~]# docker stop analytics_collector_1
[root@nodec53 ~]# docker stop analytics_alarm_kafka_1

[root@nodec54 ~]# # docker stop config_schema_1
[root@nodec54 ~]# docker stop config_api_1
[root@nodec54 ~]# docker stop config_svcmonitor_1
```

```
[root@nodec54 ~]# docker stop config_devicemgr_1
[root@nodec54 ~]# docker stop config_nodemgr_1
[root@nodec54 ~]# docker stop config_database_nodemgr_1
[root@nodec54 ~]# docker stop analytics_snmp_snmp-collector_1
[root@nodec54 ~]# docker stop analytics_snmp_topology_1
[root@nodec54 ~]# docker stop analytics_alarm_alarm-gen_1
[root@nodec54 ~]# docker stop analytics_api_1
[root@nodec54 ~]# docker stop analytics_collector_1
[root@nodec54 ~]# docker stop analytics_alarm_kafka_1

[root@nodec55 ~]# docker stop config_schema_1
[root@nodec55 ~]# docker stop config_api_1
[root@nodec55 ~]# docker stop config_svcmonitor_1
[root@nodec55 ~]# docker stop config_devicemgr_1
[root@nodec55 ~]# docker stop config_nodemgr_1
[root@nodec55 ~]# docker stop config_database_nodemgr_1
[root@nodec55 ~]# docker stop analytics_snmp_snmp-collector_1
[root@nodec55 ~]# docker stop analytics_snmp_topology_1
[root@nodec55 ~]# docker stop analytics_alarm_alarm-gen_1
[root@nodec55 ~]# docker stop analytics_api_1
[root@nodec55 ~]# docker stop analytics_collector_1
[root@nodec55 ~]# docker stop analytics_alarm_kafka_1

## Stop Cassandra ##
[root@nodec53 ~]# docker stop config_database_cassandra_1
[root@nodec54 ~]# docker stop config_database_cassandra_1
[root@nodec55 ~]# docker stop config_database_cassandra_1

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

## Backup Zookeeper Directories Before Deleting Zookeeper Data Directory Contents ##
[root@nodec53 _data]# cd /var/lib/docker/volumes/config_database_config_zookeeper/
[root@nodec53 config_database_config_zookeeper]# cp -R _data/version-2/ version-2-save
[root@nodec53 config_database_config_zookeeper]# rm -rf _data/version-2/*

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

[root@nodec55 _data]# cd /var/lib/docker/volumes/config_database_config_zookeeper/
```

```
[root@nodec55 config_database_config_zookeeper]# cp -R _data/version-2/ version-2-save
[root@nodec55 config_database_config_zookeeper]# rm -rf _data/version-2/*

## Backup Cassandra Directory Before Deleting Cassandra Data Directory Contents ##
[root@nodec53 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra/
[root@nodec53 config_database_config_cassandra]# cp -R _data/ Cassandra_data-save
[root@nodec53 config_database_config_cassandra]# rm -rf _data/*

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

[root@nodec55 ~]# cd /var/lib/docker/volumes/config_database_config_cassandra/
[root@nodec55 config_database_config_cassandra]# cp -R _data/ Cassandra_data-save
[root@nodec55 config_database_config_cassandra]# rm -rf _data/*

## Start Zookeeper ##
[root@nodec53 ~]# docker start config_database_zookeeper_1
[root@nodec54 ~]# docker start config_database_zookeeper_1
[root@nodec55 ~]# docker start config_database_zookeeper_1

## Start Cassandra ##
[root@nodec53 ~]# docker start config_database_cassandra_1
[root@nodec54 ~]# docker start config_database_cassandra_1
[root@nodec55 ~]# docker start config_database_cassandra_1

## Run Docker Image & Mount Contrail TLS Certificates to Contrail SSL Directory ##
[root@nodec54 ~]# docker image ls | grep config-api
hub.juniper.net/contrail/contrail-controller-config-api  1909.30-ocata c9d757252a0c  4 months
ago  583MB
[root@nodec54 ~]# docker run --rm -it -v /tmp/db-dump/:/tmp/ -v /etc/contrail/ssl:/etc/contrail/
ssl:ro --network host --entrypoint=/bin/bash hub.juniper.net/contrail/contrail-controller-config-
api:1909.30-ocata

## Restore Data in New Docker Containers ##
(config_api_1)[root@nodec54 /root]$ cd /usr/lib/python2.7/site-packages/cfgm_common/
(config_api_1)[root@nodec54 /usr/lib/python2.7/site-packages/cfgm_common]$ python
db_json_exim.py --import-from /tmp/db-dump.json --api-conf /tmp/contrail-api.conf

## Start Configuration Services ##
[root@nodec53 ~]# docker start config_schema_1
[root@nodec53 ~]# docker start config_svcmonitor_1
[root@nodec53 ~]# docker start config_devicemgr_1
```

```
[root@nodec53 ~]# docker start config_nodemgr_1
[root@nodec53 ~]# docker start config_database_nodemgr_1
[root@nodec53 ~]# docker start config_api_1
[root@nodec53 ~]# docker start analytics_snmp_snmp-collector_1
[root@nodec53 ~]# docker start analytics_snmp_topology_1
[root@nodec53 ~]# docker start analytics_alarm_alarm-gen_1
[root@nodec53 ~]# docker start analytics_api_1
[root@nodec53 ~]# docker start analytics_collector_1
[root@nodec53 ~]# docker start analytics_alarm_kafka_1

[root@nodec54 ~]# docker start config_schema_1
[root@nodec54 ~]# docker start config_svcmonitor_1
[root@nodec54 ~]# docker start config_devicemgr_1
[root@nodec54 ~]# docker start config_nodemgr_1
[root@nodec54 ~]# docker start config_database_nodemgr_1
[root@nodec54 ~]# docker start config_api_1
[root@nodec54 ~]# docker start analytics_snmp_snmp-collector_1
[root@nodec54 ~]# docker start analytics_snmp_topology_1
[root@nodec54 ~]# docker start analytics_alarm_alarm-gen_1
[root@nodec54 ~]# docker start analytics_api_1
[root@nodec54 ~]# docker start analytics_collector_1
[root@nodec54 ~]# docker start analytics_alarm_kafka_1

[root@nodec55 ~]# docker start config_schema_1
[root@nodec55 ~]# docker start config_svcmonitor_1
[root@nodec55 ~]# docker start config_devicemgr_1
[root@nodec55 ~]# docker start config_nodemgr_1
[root@nodec55 ~]# docker start config_database_nodemgr_1
[root@nodec55 ~]# docker start config_api_1
[root@nodec55 ~]# docker start analytics_snmp_snmp-collector_1
[root@nodec55 ~]# docker start analytics_snmp_topology_1
[root@nodec55 ~]# docker start analytics_alarm_alarm-gen_1
[root@nodec55 ~]# docker start analytics_api_1
[root@nodec55 ~]# docker start analytics_collector_1
[root@nodec55 ~]# docker start analytics_alarm_kafka_1

## Confirm Services are Active ##
[root@nodec53 ~]# contrail-status
[root@nodec54 ~]# contrail-status
[root@nodec55 ~]# contrail-status
```

Example: How to Restore a Database Using the JSON Backup (Red Hat Openstack Deployer Environment)

This example shows how to restore the databases from an environment that was deployed using Red Hat Openstack and includes three config nodes—*config1*, *config2*, and *config3*—connected to the Contrail Configuration database (config-db). All steps that need to be done from a single config node are performed from *config1*.

The environment also contains three analytics nodes—*analytics1*, *analytics2*, and *analytics3*—to provide analytics services.

This example assumes a JSON backup file of the databases was previously created using the instructions provided in "Simple Database Backup in JSON Format" on page 244.

```
## Make db-dump directory. Copy contrail-api.conf to db-dump directory. ##
[root@config1 ~]# mkdir /tmp/db-dump
[root@config1 ~]# docker cp config_api_1:/etc/contrail/contrail-api.conf /tmp/db-dump/

## Stop Configuration Services on All Config Nodes ##
[root@config1 ~]# docker stop contrail_config_svc_monitor
[root@config1 ~]# docker stop contrail_config_device_manager
[root@config1 ~]# docker stop contrail_config_schema
[root@config1 ~]# docker stop contrail_config_api
[root@config1 ~]# docker stop contrail_config_nodemgr
[root@config1 ~]# docker stop contrail_config_database_nodemgr

[root@config2 ~]# docker stop contrail_config_svc_monitor
[root@config2 ~]# docker stop contrail_config_device_manager
[root@config2 ~]# docker stop contrail_config_schema
[root@config2 ~]# docker stop contrail_config_api
[root@config2 ~]# docker stop contrail_config_nodemgr
[root@config2 ~]# docker stop contrail_config_database_nodemgr

[root@config3 ~]# docker stop contrail_config_svc_monitor
[root@config3 ~]# docker stop contrail_config_device_manager
[root@config3 ~]# docker stop contrail_config_schema
[root@config3 ~]# docker stop contrail_config_api
[root@config3 ~]# docker stop contrail_config_nodemgr
[root@config3 ~]# docker stop contrail_config_database_nodemgr

## Stop Analytics Services on All Analytics Nodes ##
[root@analytics1 ~]# docker stop contrail_analytics_snmp_collector
[root@analytics1 ~]# docker stop contrail_analytics_topology
```

```
[root@analytics1 ~]# docker stop contrail_analytics_alarmgen
[root@analytics1 ~]# docker stop contrail_analytics_api
[root@analytics1 ~]# docker stop contrail_analytics_collector
[root@analytics1 ~]# docker stop contrail_analytics_kafka

[root@analytics2 ~]# docker stop contrail_analytics_snmp_collector
[root@analytics2 ~]# docker stop contrail_analytics_topology
[root@analytics2 ~]# docker stop contrail_analytics_alarmgen
[root@analytics2 ~]# docker stop contrail_analytics_api
[root@analytics2 ~]# docker stop contrail_analytics_collector
[root@analytics2 ~]# docker stop contrail_analytics_kafka

[root@analytics3 ~]# docker stop contrail_analytics_snmp_collector
[root@analytics3 ~]# docker stop contrail_analytics_topology
[root@analytics3 ~]# docker stop contrail_analytics_alarmgen
[root@analytics3 ~]# docker stop contrail_analytics_api
[root@analytics3 ~]# docker stop contrail_analytics_collector
[root@analytics3 ~]# docker stop contrail_analytics_kafka

## Stop Cassandra ##
[root@config1 ~]# docker stop contrail_config_database
[root@config2 ~]# docker stop contrail_config_database
[root@config3 ~]# docker stop contrail_config_database

## Stop Zookeeper ##
[root@config1 ~]# docker stop contrail_config_zookeeper
[root@config2 ~]# docker stop contrail_config_zookeeper
[root@config3 ~]# docker stop contrail_config_zookeeper

## Backup Zookeeper Directories Before Deleting Zookeeper Data Directory Contents ##
[root@config1 _data]# cd /var/lib/docker/volumes/config_zookeeper/
[root@config1 config_zookeeper]# cp -R _data/version-2/ version-2-save
[root@config1 config_zookeeper]# rm -rf _data/version-2/*
[root@config2 _data]# cd /var/lib/docker/volumes/config_zookeeper/
[root@config2 config_zookeeper]# cp -R _data/version-2/ version-2-save
[root@config2 config_zookeeper]# rm -rf _data/version-2/*
[root@config3 _data]# cd /var/lib/docker/volumes/config_zookeeper/
[root@config3 config_zookeeper]# cp -R _data/version-2/ version-2-save
[root@config3 config_zookeeper]# rm -rf _data/version-2/*

## Backup Cassandra Directory Before Deleting Cassandra Data Directory Contents ##
[root@config1 ~]# cd /var/lib/docker/volumes/config_cassandra/
[root@config1 config_cassandra]# cp -R _data/ Cassandra_data-save
```

```
[root@config1 config_cassandra]# rm -rf _data/*

[root@config2 ~]# cd /var/lib/docker/volumes/config_cassandra/
[root@config2 config_cassandra]# cp -R _data/ Cassandra_data-save
[root@config2 config_cassandra]# rm -rf _data/*

[root@config3 ~]# cd /var/lib/docker/volumes/config_cassandra/
[root@config3 config_cassandra]# cp -R _data/ Cassandra_data-save
[root@config3 config_cassandra]# rm -rf _data/*

## Start Zookeeper ##
[root@config1 ~]# docker start contrail_config_zookeeper
[root@config2 ~]# docker start contrail_config_zookeeper
[root@config3 ~]# docker start contrail_config_zookeeper

## Start Cassandra ##
[root@config1 ~]# docker start contrail_config_database
[root@config2 ~]# docker start contrail_config_database
[root@config3 ~]# docker start contrail_config_database

## Run Docker Image & Mount Contrail TLS Certificates to Contrail SSL Directory ##
[root@config1 ~]# docker image ls | grep config-api
hub.juniper.net/contrail/contrail-controller-config-api  1909.30-ocata c9d757252a0c  4 months
ago  583MB
[root@config1 ~]# docker run --rm -it -v /tmp/db-dump/:/tmp/ -v /etc/contrail/ssl:/etc/contrail/
ssl:ro --network host --entrypoint=/bin/bash hub.juniper.net/contrail/contrail-controller-config-
api:1909.30-ocata

## Restore Data in New Docker Containers ##
(config_api_1)[root@config1 /root]$ cd /usr/lib/python2.7/site-packages/cfgm_common/
(config_api_1)[root@config1 /usr/lib/python2.7/site-packages/cfgm_common]$ python
db_json_exim.py --import-from /tmp/db-dump.json --api-conf /tmp/contrail-api.conf

## Start Configuration Services on All Config Nodes ##
[root@config1 ~]# docker start contrail_config_svc_monitor
[root@config1 ~]# docker start contrail_config_device_manager
[root@config1 ~]# docker start contrail_config_schema
[root@config1 ~]# docker start contrail_config_api
[root@config1 ~]# docker start contrail_config_nodemgr
[root@config1 ~]# docker start contrail_config_database_nodemgr

[root@config2 ~]# docker start contrail_config_svc_monitor
[root@config2 ~]# docker start contrail_config_device_manager
```

```
[root@config2 ~]# docker start contrail_config_schema
[root@config2 ~]# docker start contrail_config_api
[root@config2 ~]# docker start contrail_config_nodemgr
[root@config2 ~]# docker start contrail_config_database_nodemgr

[root@config3 ~]# docker start contrail_config_svc_monitor
[root@config3 ~]# docker start contrail_config_device_manager
[root@config3 ~]# docker start contrail_config_schema
[root@config3 ~]# docker start contrail_config_api
[root@config3 ~]# docker start contrail_config_nodemgr
[root@config3 ~]# docker start contrail_config_database_nodemgr

## Start Configuration Services on All Analytics Nodes ##
[root@analytics1 ~]# docker start contrail_analytics_snmp_collector
[root@analytics1 ~]# docker start contrail_analytics_topology
[root@analytics1 ~]# docker start contrail_analytics_alarmgen
[root@analytics1 ~]# docker start contrail_analytics_api
[root@analytics1 ~]# docker start contrail_analytics_collector
[root@analytics1 ~]# docker start contrail_analytics_kafka

[root@analytics2 ~]# docker start contrail_analytics_snmp_collector
[root@analytics2 ~]# docker start contrail_analytics_topology
[root@analytics2 ~]# docker start contrail_analytics_alarmgen
[root@analytics2 ~]# docker start contrail_analytics_api
[root@analytics2 ~]# docker start contrail_analytics_collector
[root@analytics2 ~]# docker start contrail_analytics_kafka

[root@analytics3 ~]# docker start contrail_analytics_snmp_collector
[root@analytics3 ~]# docker start contrail_analytics_topology
[root@analytics3 ~]# docker start contrail_analytics_alarmgen
[root@analytics3 ~]# docker start contrail_analytics_api
[root@analytics3 ~]# docker start contrail_analytics_collector
[root@analytics3 ~]# docker start contrail_analytics_kafka


## Confirm Services are Active ##
[root@config1 ~]# contrail-status
[root@config2 ~]# contrail-status
[root@config3 ~]# contrail-status

[root@analytics1 ~]# contrail-status
```

```
[root@analytics2 ~]# contrail-status
[root@analytics3 ~]# contrail-status
```

CHAPTER 6

Setting Up Contrail with Red Hat OpenStack 17.1

IN THIS CHAPTER

Understanding Red Hat OpenStack Platform Director 17.1

IN THIS SECTION

Red Hat OpenStack Platform Director

Starting with Contrail Networking Release 21.4.L4, Contrail Networking supports using Contrail with Red Hat OpenStack Platform 17.1. This chapter explains how to integrate a Contrail Networking Release 21.4.L4(.x) installation with Red Hat OpenStack Platform 17.1.

Red Hat OpenStack Platform provides an installer called the Red Hat OpenStack Platform director (RHOSPd or OSPd), which is a toolset based on the OpenStack project TripleO (OOO, OpenStack on OpenStack). TripleO is an open source project that uses features of OpenStack to deploy a fully functional, tenant-facing OpenStack environment.

Red Hat OpenStack Platform director can be used to deploy an RHOSP-based OpenStack environment integrated with Contrail Networking.

OSPd uses the concepts of undercloud and overcloud. OSPd sets up an undercloud, a single server running an operator-facing deployment that contains the OpenStack components needed to deploy and manage an overcloud, a tenant-facing deployment that hosts user workloads.

The overcloud is the deployed solution that can represent a cloud for any purpose, such as production, staging, test, and so on. The operator can select to deploy to their environment any of the available overcloud roles, such as controller, compute, and the like.

OSPd leverages existing core components of OpenStack including Nova, Ironic, Neutron, Heat, Glance, Metalsmith and Ceilometer to deploy OpenStack on bare metal hardware.

- Nova and Ironic are used in the undercloud to manage the bare metal instances that comprise the infrastructure for the overcloud.
- Neutron is used to provide a networking environment in which to deploy the overcloud.
- Glance stores machine images.
- Metalsmith is a project that uses keystone and ironic for baremetal provisioning.
- Ceilometer collects metrics about the overcloud.

For more information about OSPd architecture, see OSPd documentation.

Contrail Networking Roles

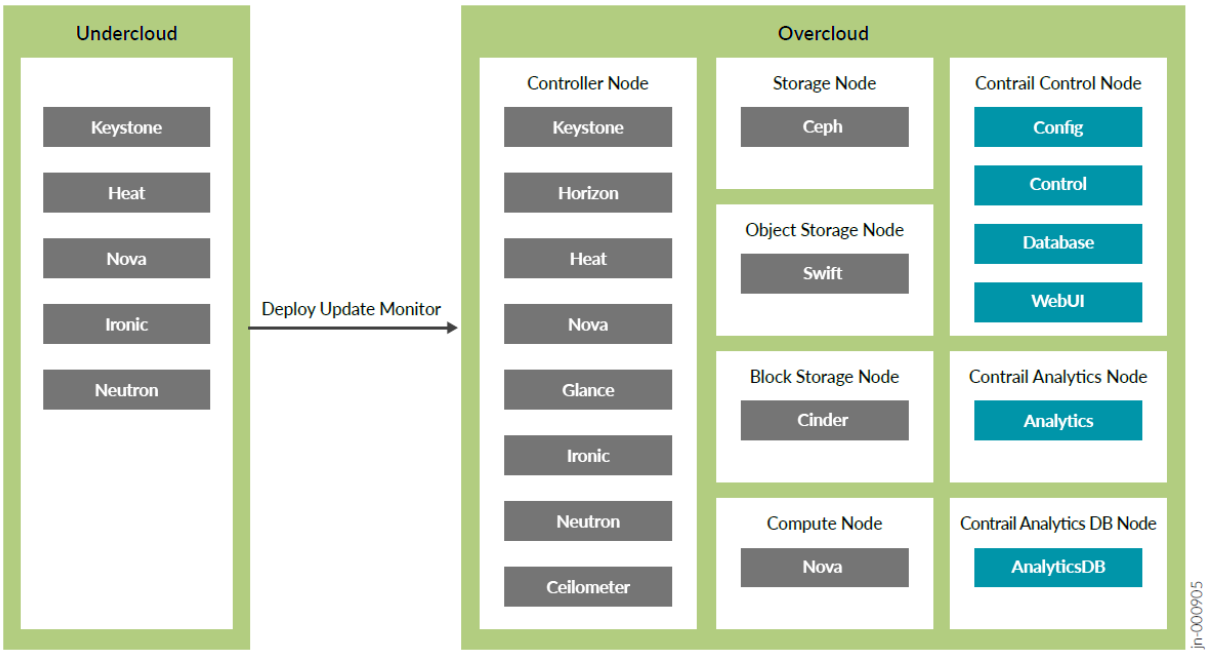
OSPd supports composable roles, which are groups of services that you define through Heat templates. Composable roles allow you to integrate Contrail Networking into the overcloud environment.

The following are the Contrail Networking roles used for integrating into the overcloud:

- Contrail Controller
- Contrail Analytics
- Contrail Analytics Database
- Contrail-TSN
- Contrail-DPDK

Figure 35 on page 268 shows the relationship and components of an undercloud and overcloud architecture for Contrail Networking.

Figure 35: Undercloud and Overcloud with Roles



RHVM and KVM Requirements

Starting in Contrail Networking Release 21.4.L4, Contrail Networking does not support Red Hat based Virtual Machine (RHVM). You must deploy overcloud components on bare metal hosts.

Undercloud Requirements

The undercloud is a single server that hosts the OpenStack Platform director, which is an OpenStack installation used to provision OpenStack on the overcloud.

See Updating the Undercloud section for the compute requirements of the undercloud.

Overcloud Requirements

After deploying the overcloud roles to the bare metal servers, deploy the compute nodes to the bare metal servers. Every overcloud node must support IPMI for booting up from the undercloud using PXE.

Ensure the following requirements are met for the Contrail Networking nodes per role.

- Non-high availability: A minimum of 4 overcloud nodes are needed for control plane roles for a non-high availability deployment:

    - 1x contrail-config (includes Contrail control)
    - 1x contrail-analytics
    - 1x contrail-analytics-database
    - 1x OpenStack controller
- High availability: A minimum of 12 overcloud nodes are needed for control plane roles for a high availability deployment:
    - 3x contrail-config (includes Contrail control)
    - 3x contrail-analytics
    - 3x contrail-analytics-database
    - 3x OpenStack controller

See Updating the Overcloud for the compute requirements of the overcloud.

Networking Requirements

As a minimum, the installation requires two networks:

- provisioning network - This is the private network that the undercloud uses to provision the overcloud.
- external network - This is the externally-routable network you use to access the undercloud and overcloud nodes.

Ensure the following requirements are met for the provisioning network:

- One NIC from every machine must be in the same broadcast domain of the provisioning network, and it should be the same NIC on each of the overcloud machines. For example, if you use the second NIC on the first overcloud machine, you should use the second NIC on each additional overcloud machine.

  During installation, these NICs are referenced by a single name across all overcloud machines.
- The provisioning network NIC should not be the same NIC that you are using for remote connectivity to the undercloud machine. During the undercloud installation, an Open vSwitch bridge will be created for Neutron, and the provisioning NIC will be bridged to the Open vSwitch bridge. Consequently, connectivity would be lost if the provisioning NIC was also used for remote connectivity to the undercloud machine.
- The provisioning NIC on the overcloud nodes must be untagged.

- You must have the MAC address of the NIC that will PXE boot the IPMI information for the machine on the provisioning network. The IPMI information will include such things as the IP address of the IPMI NIC and the IPMI username and password.
- All of the networks must be available to all of the Contrail Networking roles and computes.

While the provisioning and external networks are sufficient for basic applications, you should create additional networks in most overcloud environments to provide isolation for the different traffic types by assigning network traffic to specific network interfaces or bonds.

When isolated networks are configured, the OpenStack services are configured to use the isolated networks. If no isolated networks are configured, all services run on the provisioning network. If only some isolated networks are configured, traffic belonging to a network not configured runs on the provisioning network.

The following networks are typically deployed when using network isolation topology:

- Provisioning - used by the undercloud to provision the overcloud
- Internal API - used by OpenStack services to communicate with each other
- Tenant - used for tenant overlay data plane traffic (one network per tenant)
- Storage - used for storage data traffic
- Storage Management - used for storage control and management traffic
- External - provides external access to the undercloud and overcloud, including external access to the web UIs and public APIs
- Floating IP - provides floating IP access to the tenant network (can either be merged with external or can be a separate network)
- Management - provides access for system administration

Compatibility Matrix

The following combinations of Operating System/OpenStack/Deployer/Contrail Networking are supported:

Table 13: Compatibility Matrix

Operating System	OpenStack	Deployer	Contrail Networking
RHEL 8.2 or 8.4	OSP16.2	RHOSP16 Director	Contrail Networking 21.3

Table 13: Compatibility Matrix *(Continued)*

Operating System	OpenStack	Deployer	Contrail Networking
RHEL 8.4	OSP16.2	RHOSP16 Director	Contrail Networking 21.4
RHEL 8.4	OSP16.2.3	RHOSP16 Director	Contrail Networking 21.4.L1
RHEL 8.4	OSP16.2.4	RHOSP16 Director	Contrail Networking 21.4.L2
RHEL 8.4	OSP16.2.5	RHOSP16 Director	Contrail Networking 21.4.L3
RHEL 8.4	OSP16.2.6	RHOSP16 Director	Contrail Networking 21.4.L4.1
RHEL 9.2	OSP17.1.2	RHOSP17.1 Director	Contrail Networking 21.4.L4
RHEL 9.2	OSP17.1.3	RHOSP17.1 Director	Contrail Networking 21.4.L4.1

Installation Summary

The general installation procedure is as follows:

- Set up the infrastructure, which is the set of servers that host the undercloud and overcloud, including the networks that connect them together.
- Set up the undercloud, which is the OSPd application.
- Set up the overcloud, which is the set of services in the tenant-facing network. Contrail Networking is part of the overcloud.

For more information on installing and using the RHOSPd, see Red Hat documentation.

For information on installing Contrail Networking beyond what's in this documentation, see the README included in the Contrail Networking software package (**contrail-tripleo-heat-templates-RHOSP-17/README.md**).

Setting Up the Infrastructure (Contrail Networking Release 21.4.L4 or Later)

SUMMARY

Follow this topic to set up the infrastructure for Contrail Networking deployment in a RHOSP 17.1 environment when you are using Contrail Networking Release 21.4.L4 or later.

IN THIS SECTION

When to Use This Procedure

Before you can deploy your undercloud and overcloud, set up your node and network infrastructure.

Figure 36 on page 273 shows an example deployment consisting of three OpenStack Controller nodes, three Contrail Controller nodes, a single Compute node, and the Director where we run our undercloud, as follows:

- All nodes connect to the management (green) network. We use this network to SSH into the nodes and to allow the nodes to reach outside networks. The subnet for the management network is 10.102.70.0/24 and the gateway is 10.102.70.1 (not shown).This is an untagged network.
- All nodes connect to the control plane (blue) network. This network contains all the overcloud control plane networks, including the provisioning network that the director uses to run IPMI and provision the overcloud. The provisioning network is untagged but all the other control plane networks are tagged. The subnet for the provisioning network is 192.168.213.0/24.
- The Contrail Controller and Compute nodes connect to the tenant (orange) network. This is the data network that carries tenant traffic. This is an untagged network. The subnet for the tenant network is 192.168.33.0/24.
- The director has an IP address of 192.168.213.10/24 on the provisioning network and 10.102.70.10/24 on the management network.

We will use this deployment example to show you how to install Contrail.

Figure 36: Contrail Deployment Example

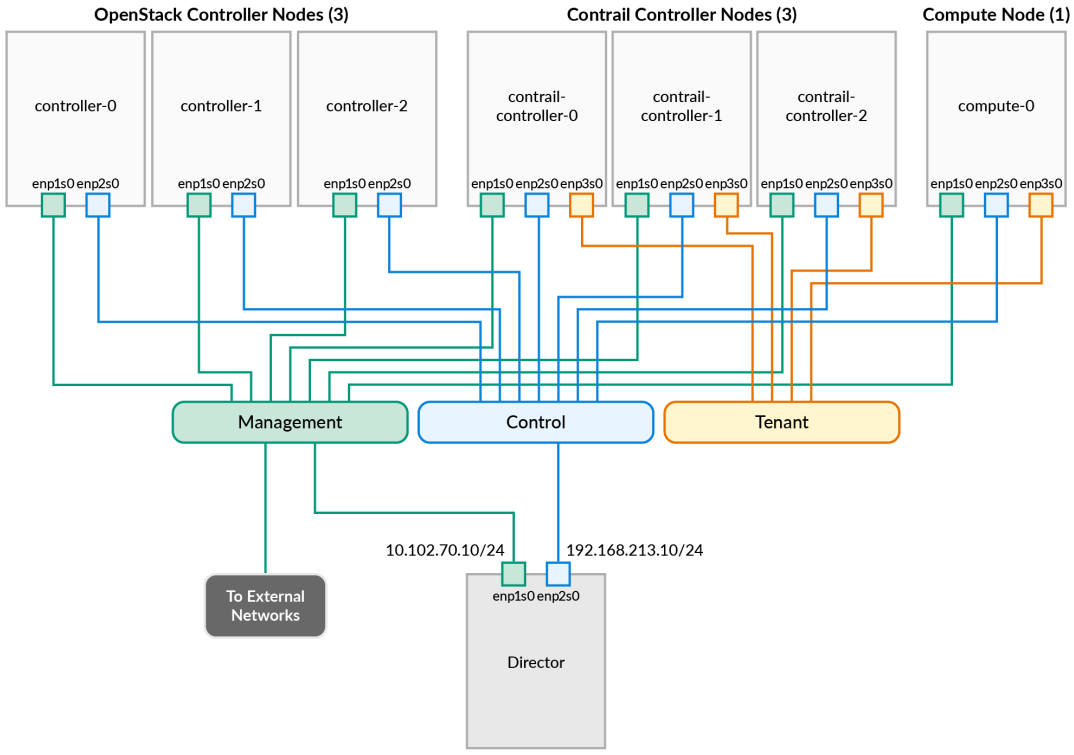
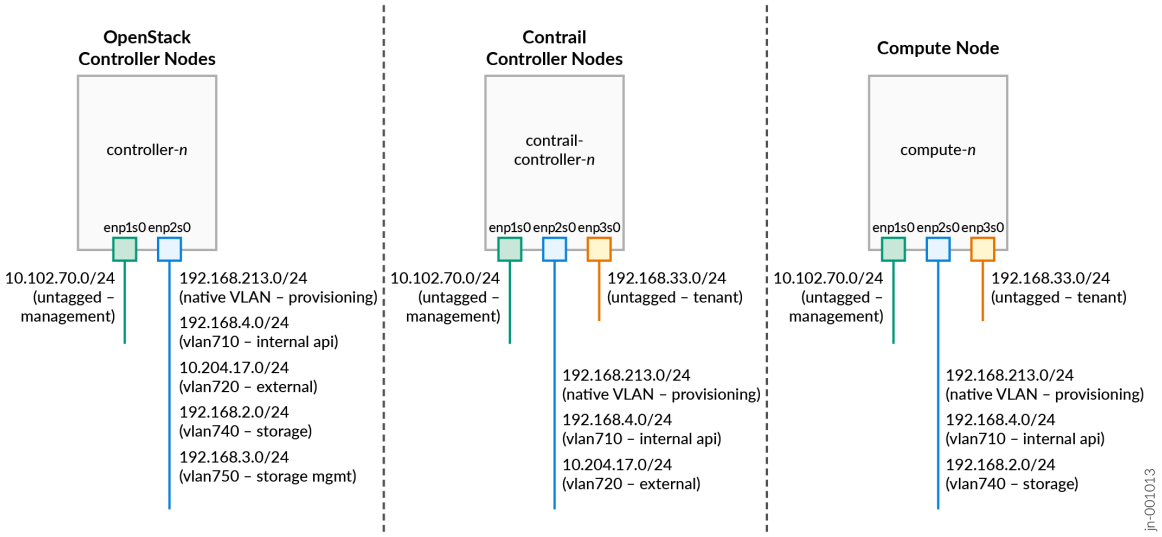


Figure 37 on page 274 shows the VLAN and IP subnet configuration for the control plane in our example.

- The OpenStack Controller nodes connect to the management, provisioning, internal api, external, storage, and storage management networks.
- The Contrail Controller nodes connect to the management, provisioning, internal api, external, and tenant networks.
- The Compute node connects to the management, provisioning, internal api, storage, and tenant networks.
- The Director (not shown) connects to the management and provisioning networks.

Figure 37: Control Plane VLANs



Set Up the Infrastructure

1. Set up the eight bare metal servers with interfaces as shown in Figure 36 on page 273.
2. Set up the management, control plane, and tenant networks (including VLANs) as shown in Figure 36 on page 273 and Figure 37 on page 274, including the 10.102.70.1/24 gateway on the management network.
3. Install a fresh RHEL 9.2 OS on the Director node.
4. Configure the Director for baseline functionality.
   - SSH root access
   - NTP
   - DNS
   - Static IP addresses on the enp1s0 interface (10.102.70.10/24) and enp2s0 interface (192.168.213.10/24)
   - Gateway (10.102.70.1) as the default route on the management network

Setting Up the Undercloud for RHOSP 17.1

SUMMARY

Follow this topic to set up the undercloud for a Contrail Networking deployment on RHOSP 17.1.

IN THIS SECTION

Prepare for Director Installation

Use this example procedure to prepare for RHOSP 17.1 director installation on a node running a freshly installed RHEL 9.2 OS.

NOTE: This procedure tailors the standard RHOSP 17.1 undercloud preparation procedure to our example. We provide this procedure purely for illustration and convenience. This procedure in no way supersedes the procedures in Red Hat documentation . Where conflicting information exists between this procedure and the documented Red Hat procedures, the Red Hat documentation prevails.

1. SSH into the undercloud node as **root**.
2. Create the **stack** user.
   a. Create the user and configure the password.

```
useradd stack
```

```
passwd stack
```

b. Add the user to the sudo user list.

```
echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack
```

```
chmod 0440 /etc/sudoers.d/stack
```

3. Switch to the new **stack** user and perform the remaining steps as this new **stack** user.

```
su - stack
```

4. Configure the hostname if you didn't already configure it during OS installation.

   For example, to set the hostname to `undercloud.contrail.lan`:

```
sudo hostnamectl set-hostname undercloud.contrail.lan
sudo hostnamectl set-hostname --transient undercloud.contrail.lan
```

5. Add an entry for this hostname to the **/etc/hosts** file.

   For example, add this line to **/etc/hosts** using sudo:

```
192.168.213.10 undercloud.contrail.lan undercloud
```

   where 192.168.213.10 is the IP address of the interface connecting the director to the provisioning network.

6. Set the locale to UTF-8.

```
export LC_ALL=en_US.UTF-8
```

```
echo "export LC_ALL=en_US.UTF-8" >> ~/.bashrc
```

7. Register the machine and update the packages.

   a. Register the machine.

```
sudo subscription-manager register
```

   Enter your Red Hat user name and password when prompted.

b. Set the release to RHEL 9.2.

```
sudo subscription-manager release --set=9.2
```

c. Disable all default repositories, and enable the required RHEL 9.2 repositories.

```
sudo subscription-manager repos --disable='*' --enable=rhel-9-for-x86_64-baseos-eus-rpms
--enable=rhel-9-for-x86_64-appstream-eus-rpms --enable=rhel-9-for-x86_64-
highavailability-eus-rpms --enable=fast-datapath-for-rhel-9-x86_64-rpms --
enable=openstack-17.1-for-rhel-9-x86_64-rpms --enable=rhceph-6-tools-for-rhel-9-x86_64-
rpms
```

NOTE: You might need to manually attach a pool before you can run this command if your license doesn't allow automatic entitlement. See Red Hat documentation .

d. Update the packages.

```
sudo dnf -y update
```

8. Reboot the machine.

```
sudo reboot
```

9. SSH back in as the **stack** user and install the Tripleo client.

```
sudo dnf install -y python3-tripleoclient
```

10. Prepare the **containers-prepare-parameter.yaml** file.

a. Generate the default **containers-prepare-parameter.yaml**.

```
openstack tripleo container image prepare default --local-push-destination --output-env-
file ~/containers-prepare-parameter.yaml
```

b. Configure the file to suit your deployment.

For this example, we simply add the `registry.redhat.io` login credentials to the `parameter_defaults` section and leave all other content as is.

```
parameter_defaults:
  ContainerImageRegistryCredentials:
    registry.redhat.io:
      <my_username>: <my_password>
```

where *<my_username>* and *<my_password>* are your Red Hat login credentials.

SEE ALSO

> https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/17.1/html/installing_and_managing_red_hat_openstack_platform_with_director/assembly_preparing-for-director-installation

Install the Director

Use this example procedure to install the director.

NOTE: This procedure tailors the standard RHOSP 17.1 undercloud installation procedure to our example. We provide this procedure purely for illustration and convenience. This procedure in no way supersedes the procedures in Red Hat documentation . Where conflicting information exists between this procedure and the documented Red Hat procedures, the Red Hat documentation prevails.

1. SSH into the undercloud node as **stack**.
2. Configure the undercloud.
   a. Copy the undercloud configuration file sample.

```
cp /usr/share/python-tripleoclient/undercloud.conf.sample ~/undercloud.conf
```

   b. Modify **undercloud.conf** to suit your deployment.

   Here is the configuration for our example:

```
[DEFAULT]

local_ip = 192.168.213.10/24
```

```
generate_service_certificate = false

ipxe_enabled = true

local_interface = enp2s0
subnets = ctlplane-subnet
undercloud_hostname = undercloud.contrail.lan
overcloud_domain_name = contrail.lan
enable_telemetry = false
enable_validations = false
clean_nodes = true
container_images_file = /home/stack/containers-prepare-parameter.yaml
inspection_enable_uefi = true
undercloud_ntp_servers = <my_ntp_server>
undercloud_nameservers = <my_dns_server>
enabled_hardware_types = ipmi,redfish,ilo,idrac,staging-ovirt

[ctlplane-subnet]

gateway = 192.168.213.10
cidr = 192.168.213.0/24
masquerade = true
dhcp_start = 192.168.213.150
dhcp_end = 192.168.213.200
inspection_iprange = 192.168.213.201,192.168.213.230
```

Table 14 on page 279 describes the main configuration parameters. For more information or for parameters not described, see Red Hat documentation.

Table 14: Main Undercloud Configuration Parameters

Parameter	Description	Setting
local_ip	IP address of interface attached to provisioning network.	192.168.213.10/24
local_interface	Name of interface attached to provisioning network.	enp2s0
container_images_file	Location of **containers-prepare-parameter.yaml** file.	/home/stack/containers-prepare-parameter.yaml

Table 14: Main Undercloud Configuration Parameters *(Continued)*

Parameter	Description	Setting
undercloud_ntp_servers	Name or IP address of your NTP server(s).	Specific to your deployment.
undercloud_nameservers	Name or IP address of your DNS server(s).	Specific to your deployment.
gateway	Gateway for the provisioning network. We're using the undercloud VM as the gateway.	192.168.213.10
cidr	The subnet mask for the provisioning network.	192.168.213.0/24
dhcp_start	The start of the range of IP addresses that the undercloud will assign to devices in the provisioning network.	192.168.213.150
dhcp_end	The end of the range of IP addresses that the undercloud will assign to devices in the provisioning network.	192.168.213.200
inspection_iprange	The range of IP addresses that the undercloud will assign during inspection. This range must be outside the regular DHCP assignment range.	192.168.213.201,192.168.213.230

3. Install the director.

```
openstack undercloud install
```

4. Confirm that the RHOSP service containers are running.

```
sudo podman ps -a --format "{{.Names}} {{.Status}}"
```

SEE ALSO

> https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/17.1/html/installing_and_managing_red_hat_openstack_platform_with_director/assembly_installing-director-on-the-undercloud

Obtain and Import the Base Overcloud Images

After you install the undercloud, obtain and import the base overcloud images to the undercloud. These are the kernel images that Red Hat provides for introspection and deployment of overcloud nodes.

NOTE: This procedure tailors the standard RHOSP 17.1 overcloud image import procedure to our example. We provide this procedure purely for illustration and convenience. This procedure in no way supersedes the procedures in Red Hat documentation . Where conflicting information exists between this procedure and the documented Red Hat procedures, the Red Hat documentation prevails.

1. SSH into the undercloud as the **stack** user.
2. Source the **stackrc** file.

```
source stackrc
```

3. Obtain the base overcloud images.

```
sudo dnf install -y rhosp-director-images-x86_64 rhosp-director-images-ipa-x86_64 rhosp-director-images-uefi-x86_64
```

This installs the images tarfile into the **/usr/share/rhosp-director-images** directory.

4. Extract the images to the **images** directory.

```
mkdir ~/images
cd ~/images
```

```
for i in /usr/share/rhosp-director-images/ironic-python-agent-latest.tar /usr/share/rhosp-director-images/overcloud-hardened-uefi-full-latest.tar; do tar -xvf $i; done
```

5. Import the images to the director so that the director can serve these images to the overcloud nodes as they come up.

```
openstack overcloud image upload --image-path /home/stack/images/
```

You can find the overcloud image in **/var/lib/ironic/images** and the introspection PXE images in **/var/lib/ironic/httpboot**.

You've installed the undercloud. You're now ready to create the overcloud.

SEE ALSO

https://docs.redhat.com/en/documentation/red_hat_openstack_platform/17.1/html/installing_and_managing_red_hat_openstack_platform_with_director/assembly_installing-director-on-the-undercloud#assembly_obtaining-images-for-overcloud-nodes

Setting Up the Overcloud for RHOSP 17.1

SUMMARY

Use this procedure to set up the overcloud for a Contrail Networking deployment on RHOSP 17.1.

IN THIS SECTION

Download Heat Templates

1. SSH into the undercloud as the **stack** user.

2. Source the stackrc undercloud credential file.

```
source ~/stackrc
```

3. Put all the heat templates into a new **~/tripleo-heat-templates** directory.

   a. Copy the Red Hat TripleO heat templates to the **~/tripleo-heat-templates** directory.

   ```
   cp -r /usr/share/openstack-tripleo-heat-templates/ ~/tripleo-heat-templates
   ```

   b. Download the Contrail Networking RHEL 9 + RHOSP17 OOO Heat Templates release 21.4.L4.1 from the Juniper Networks software download site and copy them to the same **~/tripleo-heat-templates** directory.

   ```
   wget <contrail-tripleo-heat-templates-RHOSP17-url> -O contrail-tripleo-heat-templates-
   RHOSP17-<version>.tgz
   ```

   ```
   tar -xzvf contrail-tripleo-heat-templates-RHOSP17-<version>.tgz
   ```

   ```
   cp -r contrail-tripleo-heat-templates-RHOSP-17/* ~/tripleo-heat-templates
   ```

Upload Container Images to the Undercloud Registry

Use this example procedure on the undercloud to upload overcloud container images to the undercloud registry.

1. SSH into the undercloud as the **stack** user.
2. Source the stackrc undercloud credential file.

```
source ~/stackrc
```

3. Create a **~/tripleo-heat-templates/environments/contrail/rhsm.yaml** file with your Red Hat credentials.

   Here is an example of the one we're using.

```
parameter_defaults:
  RhsmVars:
```

```
    rhsm_repos:
      - rhel-9-for-x86_64-baseos-eus-rpms
      - rhel-9-for-x86_64-appstream-eus-rpms
      - rhel-9-for-x86_64-highavailability-eus-rpms
      - openstack-17.1-for-rhel-9-x86_64-rpms
      - fast-datapath-for-rhel-9-x86_64-rpms
    rhsm_username: "YOUR_REDHAT_LOGIN"
    rhsm_password: "YOUR_REDHAT_PASSWORD"
    rhsm_org_id: "YOUR_REDHAT_ID"
    rhsm_pool_ids: "YOUR_REDHAT_POOL_ID"
    rhsm_release: "9.2"
```

4. Pull container images from remote registries and push them to the undercloud registry.

   OpenStack container images are available from the Red Hat registry and Contrail container images are available from the Juniper Networks registry.

   a. Pull the Red Hat OpenStack container images from `registry.redhat.io` and push them to the undercloud registry.

      i. Create the OpenStack container file.

      ```
      sudo openstack tripleo container image prepare \
        -e /home/stack/containers-prepare-parameter.yaml \
        -e /home/stack/tripleo-heat-templates/environments/contrail/rhsm.yaml \
        --output-env-file /home/stack/tripleo-heat-templates/environments/contrail/
      overcloud_containers.yaml
      ```

      This produces the **~/tripleo-heat-templates/environments/contrail/overcloud-containers.yaml** file.

      ii. Pull the container images and push them to the undercloud registry.

      ```
      sudo openstack overcloud container image upload \
        --config-file /home/stack/tripleo-heat-templates/environments/contrail/
      overcloud_containers.yaml
      ```

   b. Pull the Contrail container images from `enterprise-hub.juniper.net`.

Edit the **~/tripleo-heat-templates/tools/contrail/import_contrail_container.sh** with the appropriate push destination for your deployment. Look for the push destination line and change the IP address. For example:

```
 echo "  push_destination: 192.168.213.10:8787" >> ${output_file}
```

Pull the Contrail containers:

```
~/tripleo-heat-templates/tools/contrail/import_contrail_container.sh \
    -f /home/stack/tripleo-heat-templates/environments/contrail/contrail-containers.yaml \
    -r enterprise-hub.juniper.net/contrail-container-prod -t 21.4.L4.17.1 \
    -u <username> -p <password>
```

where *<username>* and *<password>* are your login credentials to the `enterprise-hub.juniper.net` registry.

c. Push the container images to the undercloud registry.

```
sudo openstack overcloud container image upload \
  --config-file /home/stack/tripleo-heat-templates/environments/contrail/contrail-
containers.yaml
```

d. List the uploaded containers.

```
openstack tripleo container image list
```

The container images are served from the **/var/lib/image-serve** directory.

Provision Overcloud Networks

1. SSH into the undercloud as the **stack** user.
2. Source the stackrc undercloud credential file.

```
source ~/stackrc
```

3. Create the network definition file **network_data.yaml** that describes our networks.

   Table 15 on page 286 shows the network configuration that corresponds to Figure 36 on page 273 and Figure 37 on page 274 in our example.

Table 15: Network Configuration

Network	VLAN	Subnet	Gateway
internal_api	710	192.168.4.0/24	192.168.4.1
management	–	10.102.70.0/24	10.102.70.1
storage	740	192.168.2.0/24	192.168.2.1
storage_mgmt	750	192.168.3.0/24	192.168.3.1
tenant	–	192.168.33.0/24	192.168.33.1
external	720	10.204.17.0/24	10.204.17.1

Here is the resulting **network_data.yaml**.

```
- name: InternalApi
  name_lower: internal_api
  vip: true
  subnets:
    internal_api_subnet:
      ip_subnet: '192.168.4/24'
      allocation_pools:
        - start: 192.168.4.50
          end: 192.168.4.99
      gateway_ip: '192.168.4.1'
      vlan: 710
- name: Management
  name_lower: management
  vip: false
  subnets:
    management_subnet:
      ip_subnet: '10.102.70.0/24'
      allocation_pools:
        - start: 10.102.70.50
          end: 10.102.70.99
      gateway_ip: '10.102.70.1'
- name: Storage
  name_lower: storage
```

```
  vip: false
  subnets:
    storage_subnet:
      ip_subnet: '192.168.2.0/24'
      allocation_pools:
        - start: 192.168.2.50
          end: 192.168.2.99
      gateway_ip: '192.168.2.1'
      vlan: 740
- name: StorageMgmt
  name_lower: storage_mgmt
  vip: false
  subnets:
    storage_mgmt_subnet:
      ip_subnet: '192.168.3.0/24'
      allocation_pools:
        - start: 192.168.3.50
          end: 192.168.3.99
      gateway_ip: '192.168.3.1'
      vlan: 750
- name: Tenant
  name_lower: tenant
  vip: false
  subnets:
    tenant_subnet:
      ip_subnet: '192.168.33.0/24'
      allocation_pools:
        - start: 192.168.33.50
          end: 192.168.33.99
      gateway_ip: '192.168.33.1'
- name: External
  name_lower: external
  vip: true
  subnets:
    external_subnet:
      ip_subnet: '10.204.17.0/24'
      allocation_pools:
        - start: 10.204.17.50
          end: 10.204.17.0.99
      gateway_ip: '10.204.17.0.1'
      vlan: 720
```

4. Place the network definition file in **~/tripleo-heat-templates/environments/contrail/network_data.yaml** and apply it.

```
openstack overcloud network provision \
   --templates /home/stack/tripleo-heat-templates \
   --output /home/stack/tripleo-heat-templates/environments/contrail/overcloud-networks-
deployed.yaml \
   --yes /home/stack/tripleo-heat-templates/environments/contrail/network_data.yaml
```

   This produces the **~/tripleo-heat-templates/environments/contrail/overcloud-networks-deployed.yaml** file.

5. Create the network VIP definition file **vip_data.yaml** that describes the network VIPs in our network. Here is the **vip_data.yaml** for our example.

```
- network: internal_api
  dns_name: overcloud
  ip_address: 192.168.4.100
- network: external
  dns_name: overcloud
  ip_address: 10.204.17.100
- network: ctlplane
  dns_name: overcloud
```

6. Place the VIP definition file in **~/tripleo-heat-templates/environments/contrail/vip_data.yaml** and apply it.

```
openstack overcloud network vip provision \
   --templates /home/stack/tripleo-heat-templates \
   --stack overcloud \
   --output /home/stack/tripleo-heat-templates/environments/contrail/overcloud-vip-
deployed.yaml \
   --yes /home/stack/tripleo-heat-templates/environments/contrail/vip_data.yaml
```

   This produces the **~/tripleo-heat-templates/environments/contrail/overcloud-vip-deployed.yaml** file.

7. Check the created networks and subnets.

```
openstack network list
```

```
openstack subnet list
```

Provision Bare Metal Overcloud Nodes

1. SSH into the undercloud as the **stack** user.
2. Source the stackrc undercloud credential file.

```
source ~/stackrc
```

3. Create the overcloud node definition file **~/nodes.json** that describes the nodes in our overcloud network.
   - Three OpenStack controller nodes (controller-0, controller-1, controller-2)
   - Three Contrail controller nodes (contrail-controller-0, contrail-controller-1, contrail-controller-2)
   - One Compute node (compute-0)

   This file is specific to your deployment. See Red Hat documentation for information on how to create and populate this file.
4. Import the nodes to the director.

```
openstack overcloud node import ~/nodes.json
```

   List the nodes to make sure they've been imported:

```
openstack baremetal node list
```

5. Introspect the nodes.

   Introspection collects hardware information from each node.

```
openstack overcloud node introspect --all-manageable --provide
```

After the introspection completes, all nodes change to an available state.

```
openstack baremetal node list
```

6. Create the **overcloud-baremetal-deploy.yaml** node definition file and set the node count for each role that you want to provision. Place the file in **~/tripleo-heat-templates/environments/contrail/overcloud-baremetal-deploy.yaml**.

   Here is the node definition file for our example:

```
- name: Controller
  count: 3
  ansible_playbooks:
  - playbook: /home/stack/configure-chrony-client.yml
  # Additional playbooks can be included here, as per following example of disk layout:
  - playbook: /usr/share/ansible/tripleo-playbooks/cli-overcloud-node-growvols.yaml
    extra_vars:
      role_growvols_args:
        default:
          /=20GB
          /tmp=1GB
          /var/log=15GB
          /var/log/audit=5GB
          /home=5GB
          /var=100GB
          /srv=100%
  defaults:
    networks:
    - network: management
    - network: internal_api
    - network: storage
    - network: storage_mgmt
    - network: external
    network_config:
      template: /home/stack/tripleo-heat-templates/environments/contrail/contrail-nic-
config-Controller.j2
  instances:
  - hostname: overcloud-controller-0
    name: controller-0
  - hostname: overcloud-controller-1
    name: controller-1
  - hostname: overcloud-controller-2
```

```
    name: controller-2

- name: ContrailController
  count: 3
  ansible_playbooks:
  - playbook: /home/stack/configure-chrony-client.yml
  - playbook: /usr/share/ansible/tripleo-playbooks/cli-overcloud-node-growvols.yaml
    extra_vars:
      role_growvols_args:
        default:
          /=20GB
          /tmp=1GB
          /var/log=15GB
          /var/log/audit=5GB
          /home=5GB
          /var=100%
  defaults:
    networks:
    - network: management
    - network: internal_api
    - network: tenant
    - network: external
    network_config:
      template: /home/stack/tripleo-heat-templates/environments/contrail/contrail-nic-
config-ContrailController.j2
  instances:
  - hostname: overcloud-contrailcontroller-0
    name: contrail-controller-0
    networks:
    - network: internal_api
      fixed_ip: 192.168.4.10
    - network: tenant
      fixed_ip: 192.168.33.10
  - hostname: overcloud-contrailcontroller-1
    name: contrail-controller-1
    networks:
    - network: internal_api
      fixed_ip: 192.168.4.11
    - network: tenant
      fixed_ip: 192.168.33.11
  - hostname: overcloud-contrailcontroller-2
    name: contrail-controller-2
    networks:
```

```
    - network: internal_api
      fixed_ip: 192.168.4.12
    - network: tenant
      fixed_ip: 192.168.33.12

- name: Compute
  count: 1
  ansible_playbooks:
  - playbook: /home/stack/configure-chrony-client.yml
  - playbook: /usr/share/ansible/tripleo-playbooks/cli-overcloud-node-growvols.yaml
    extra_vars:
      role_growvols_args:
        default:
          /=20GB
          /tmp=1GB
          /var/log=10GB
          /var/log/audit=3GB
          /home=5GB
          /var=100%
  defaults:
    networks:
    - network: management
    - network: internal_api
    - network: storage
    - network: tenant
    network_config:
      template: /home/stack/tripleo-heat-templates/environments/contrail/contrail-nic-
config-ComputeKernel.j2
  instances:
  - hostname: overcloud-novacompute-0
    name: compute-0
```

7. Provision the overcloud nodes.

```
openstack overcloud node provision \
   --templates /home/stack/tripleo-heat-templates \
   --stack overcloud \
   --output /home/stack/tripleo-heat-templates/environments/contrail/overcloud-baremetal-
deployed.yaml \
   --yes /home/stack/tripleo-heat-templates/environments/contrail/overcloud-baremetal-
deploy.yaml
```

This produces the **~/tripleo-heat-templates/environments/contrail/overcloud-baremetal-deployed.yaml** file.

8. Check that the node state is active.

```
openstack baremetal node list
```

9. Find the IP address of every overcloud node.

```
metalsmith list
```

10. Register each overcloud node.

    a. SSH into an overcloud node from the undercloud.

    ```
    ssh tripleo-admin@<overcloud-node-ip>
    ```

    where *<overcloud-node-ip>* is the IP address of one of the overcloud nodes.

    b. Register the node.

    ```
    sudo subscription-manager register
    ```

    Enter your Red Hat user name and password when prompted.

    c. Set the release to RHEL 9.2.

    ```
    sudo subscription-manager release --set=9.2
    ```

    d. Disable all default repositories, and enable the required RHEL 9.2 repositories.

    ```
    sudo subscription-manager repos --disable='*' --enable=rhel-9-for-x86_64-baseos-eus-rpms
    --enable=rhel-9-for-x86_64-appstream-eus-rpms --enable=rhel-9-for-x86_64-
    highavailability-eus-rpms --enable=fast-datapath-for-rhel-9-x86_64-rpms --
    enable=openstack-17.1-for-rhel-9-x86_64-rpms --enable=rhceph-6-tools-for-rhel-9-x86_64-
    rpms
    ```

> **NOTE**: You might need to manually attach a pool before you can run this command if your license doesn't allow automatic entitlement. See Red Hat documentation .

e. Repeat on all overcloud nodes.

11. Install the ansible community.general collection on every overcloud node.

a. SSH into an overcloud node from the undercloud.

```
ssh tripleo-admin@<overcloud-node-ip>
```

where *<overcloud-node-ip>* is the IP address of one of the overcloud nodes.

b. Install the ansible community.general collection as a sudo user and a regular user.

```
sudo ansible-galaxy collection install community.general
```

```
ansible-galaxy collection install community.general
```

c. Repeat on all overcloud nodes.

Configure Contrail

IN THIS SECTION

We provide various configuration files in the Contrail Networking Heat Templates package for you to customize your overcloud deployment. Table 16 on page 295 describes the files that you need to configure in our example.

Table 16: Configuration Files

File	Description
roles_data.yaml	Description of the different node roles
contrail-nic-config-Controller.j2	Description of the interfaces on the OpenStack Controllers
contrail-nic-config-ContrailController.j2	Description of the interfaces on the Contrail Controllers
contrail-nic-config-ComputeKernel.j2	Description of the interfaces on the Compute node running in kernel mode
contrail-net.yaml	Description of various Contrail network settings
contrail-services.yaml	Description of the Contrail services
contrail-plugins.yaml	Miscellaneous settings
NOTE: You can find samples of the above files under comparable names in the Contrail Networking Heat Templates package that you downloaded. Our example, however, assumes that you'll be using the filenames as shown above.	

Roles Configuration (roles_data.yaml)

The roles configuration file contains definitions for the various node roles. You can find sample Contrail roles files in **~/tripleo-heat-templates** and in **~/tripleo-heat-templates/roles**.

We define three roles in our example. Put all three roles into the **roles_data.yaml** file and place the file at **~/tripleo-heat-templates/environments/contrail/roles_data.yaml**.

OpenStack Controller Role

In our example, we have three nodes for the OpenStack controller. Each of these nodes connects to the Management, Internal API, Management, Storage, Storage Management, and External networks, as shown below:

```
##############################################################################
```

```
# Role: Controller                                                             #
###############################################################################
- name: Controller
  description: |
    Controller role that has all the controler services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 3
  tags:
    - primary
    - controller
  networks:
    Management:
      subnet: management_subnet
    InternalApi:
      subnet: internal_api_subnet
    Storage:
      subnet: storage_subnet
    StorageMgmt:
      subnet: storage_mgmt_subnet
    External:
      subnet: external_subnet
  default_route_networks: ['External']
  HostnameFormatDefault: '%stackname%-controller-%index%'
  RoleParametersDefault:
    NeutronPublicInterface: nic2
  uses_deprecated_params: True
  deprecated_param_extraconfig: 'controllerExtraConfig'
  deprecated_param_flavor: 'OvercloudControlFlavor'
  deprecated_param_image: 'controllerImage'
  deprecated_nic_config_name: 'controller.yaml'
  update_serial: 1
  ServicesDefault:
      <leave unchanged>
```

Contrail Controller Role

We also have three nodes for the Contrail controller. Each of these nodes connects to the Management, Internal API, External, and tenant networks, as shown below:

```


###############################################################################
```

```
# Role: ContrailController                                                    #
##############################################################################
- name: ContrailController
  description: |
    ContrailController role that has all the Contrail controler services loaded
    and handles config, control and webui functions
  CountDefault: 3
  tags:
    - primary
    - contrailcontroller
  networks:
    Management:
      subnet: management_subnet
    InternalApi:
      subnet: internal_api_subnet
    Tenant:
      subnet: tenant_subnet
    External:
      subnet: external_subnet
  HostnameFormatDefault: '%stackname%-contrailcontroller-%index%'
  RoleParametersDefault:
    NeutronPublicInterface: nic2
    tripleo_podman_tls_verify: false
  ServicesDefault:
      <leave unchanged>
```

Compute Node Role

We have one Compute node. The compute node connects to the Management, Internal API, Storage, and tenant networks, as shown below.

```
##############################################################################
# Role: Compute                                                              #
##############################################################################
- name: Compute
  description: |
    Basic Compute Node role
  CountDefault: 1
  tags:
    - compute
```

```
networks:
  Management:
    subnet: management_subnet
  InternalApi:
    subnet: internal_api_subnet
  Storage:
    subnet: storage_subnet
  Tenant:
    subnet: tenant_subnet
HostnameFormatDefault: '%stackname%-novacompute-%index%'
RoleParametersDefault:
  NeutronPublicInterface: nic2
  FsAioMaxNumber: 1048576
  TunedProfileName: "virtual-host"
  tripleo_podman_tls_verify: false
uses_deprecated_params: True
deprecated_param_image: 'NovaImage'
deprecated_param_extraconfig: 'NovaComputeExtraConfig'
deprecated_param_metadata: 'NovaComputeServerMetadata'
deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
deprecated_param_ips: 'NovaComputeIPs'
deprecated_server_resource_name: 'NovaCompute'
deprecated_nic_config_name: 'compute.yaml'
update_serial: 25
ServicesDefault:
  <leave unchanged>
```

Network Interface Configuration (*-nic-*.j2)

NIC configuration files describe the interfaces for each role. You can find sample NIC configuration files in **~/tripleo-heat-templates/network/config/contrail** and **~/tripleo-heat-templates/network/config/contrail/examples**.

NOTE: NIC configuration files are referred to in **overcloud-baremetal-deploy.yaml** . Be sure to name the configuration files properly and place them in **~/tripleo-heat-templates/environments/contrail** .

We define three NICs (one for each role) in our example as shown in Table 17 on page 299.

Table 17: NIC Mapping

Nodes	Interfaces	Networks
OpenStack Controller	enp1s0	management
	enp2s0	control plane • external • internal_api • storage_mgmt • storage
Contrail Controller	enp1s0	management
	enp2s0	control plane • external • internal_api
	enp3s0	tenant
Compute	enp1s0	management
	enp2s0	control plane • internal_api • storage
	enp3s0	tenant

OpenStack Controller NIC (contrail-nic-config-Controller.j2)

```
---
network_config:
- addresses:
  - ip_netmask: {{ management_ip }}/{{ management_subnet_cidr }}
  mtu: 1500
```

```
  name: enp1s0
  type: interface
  use_dhcp: false
  routes:
  - default: true
    next_hop: {{ management_gateway_ip }}
- addresses:
  - ip_netmask: {{ ctlplane_ip }}/{{ ctlplane_subnet_cidr }}
  dns_servers: {{ ctlplane_dns_nameservers }}
  mtu: 1500
  name: enp2s0
  type: interface
  use_dhcp: false
- addresses:
  - ip_netmask: {{ external_ip }}/{{ external_cidr }}
  device: enp2s0
  mtu: 1500
  type: vlan
  vlan_id: {{ external_vlan_id }}
- addresses:
  - ip_netmask: {{ internal_api_ip }}/{{ internal_api_cidr }}
  device: enp2s0
  mtu: 1500
  type: vlan
  vlan_id: {{ internal_api_vlan_id }}
- addresses:
  - ip_netmask: {{ storage_mgmt_ip }}/{{ storage_mgmt_cidr }}
  device: enp2s0
  mtu: 1500
  type: vlan
  vlan_id: {{ storage_mgmt_vlan_id }}
- addresses:
  - ip_netmask: {{ storage_ip }}/{{ storage_cidr }}
  device: enp2s0
  mtu: 1500
  type: vlan
  vlan_id: {{ storage_vlan_id }}
```

Contrail Controller NIC (contrail-nic-config-ContrailController.j2)

```
---
network_config:
- addresses:
  - ip_netmask: {{ management_ip }}/{{ management_subnet_cidr }}
  mtu: 1500
  name: enp1s0
  type: interface
  use_dhcp: false
  routes:
  - default: true
    next_hop: {{ management_gateway_ip }}
- addresses:
  - ip_netmask: {{ ctlplane_ip }}/{{ ctlplane_subnet_cidr }}
  dns_servers: {{ ctlplane_dns_nameservers }}
  mtu: 1500
  name: enp2s0
  type: interface
  use_dhcp: false
- addresses:
  - ip_netmask: {{ external_ip }}/{{ external_cidr }}
  device: enp2s0
  mtu: 1500
  type: vlan
  vlan_id: {{ external_vlan_id }}
- addresses:
  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}
  name: enp3s0
  mtu: 1500
  type: interface
  use_dhcp: false
- addresses:
  - ip_netmask: {{ internal_api_ip }}/{{ internal_api_cidr }}
  device: enp2s0
  mtu: 1500
  type: vlan
  vlan_id: {{ internal_api_vlan_id }}
```

Compute Node NIC (contrail-nic-config-Compute.j2)

```
---
network_config:
- addresses:
  - ip_netmask: {{ management_ip }}/{{ management_subnet_cidr }}
  mtu: 1500
  name: enp1s0
  type: interface
  use_dhcp: false
  routes:
  - default: true
    next_hop: {{ management_gateway_ip }}
- addresses:
  - ip_netmask: {{ ctlplane_ip }}/{{ ctlplane_subnet_cidr }}
  dns_servers: {{ ctlplane_dns_nameservers }}
  mtu: 1500
  name: enp2s0
  type: interface
  use_dhcp: false
- addresses:
  - ip_netmask: {{ internal_api_ip }}/{{ internal_api_cidr }}
  device: enp2s0
  mtu: 1500
  type: vlan
  vlan_id: {{ internal_api_vlan_id }}
- addresses:
  - ip_netmask: {{ storage_ip }}/{{ storage_cidr }}
  device: enp2s0
  mtu: 1500
  type: vlan
  vlan_id: {{ storage_vlan_id }}
- addresses:
  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}
  members:
  - name: enp3s0
    type: interface
    use_dhcp: false
  mtu: 1500
  name: vhost0
```

```
  type: contrail_vrouter
  use_dhcp: false
```

Network Parameter Configuration (contrail-net.yaml)

Customize Contrail network parameters by modifying the **contrail-net.yaml** file. We provide a sample at **~/tripleo-heat-templates/environments/contrail/contrail-net.yaml**. Look through that file for explanations of the parameters.

Here is the **contrail-net.yaml** for our example.

```
resource_registry:
   <leave unchanged>
parameter_defaults:
# Customize all these values to match the local environment
  TenantNetCidr: 192.168.33.0/24
  InternalApiNetCidr: 192.168.4.0/24
  ExternalNetCidr: 10.204.17.0/24
  StorageNetCidr: 192.168.2.0/24
  StorageMgmtNetCidr: 192.168.3.0/24
  ManagementNetCidr: 10.102.70.0/24
  # CIDR subnet mask length for provisioning network
  ControlPlaneSubnetCidr: '24'
  # Allocation pools
  ManagementAllocationPools: [{'start': '10.102.70.50', 'end': '10.102.70.99'}]
  TenantAllocationPools: [{'start': '192.168.33.50', 'end': '192.168.33.99'}]
  InternalApiAllocationPools: [{'start': '192.168.4.50', 'end': '192.168.4.99'}]
  ExternalAllocationPools: [{'start': '10.204.17.50', 'end': '10.204.17.99'}]
  StorageAllocationPools: [{'start': '192.168.2.50', 'end': '192.168.2.99'}]
  StorageMgmtAllocationPools: [{'start': '192.168.3.50', 'end': '192.168.3.99'}]
  # Routes
  ControlPlaneDefaultRoute: 192.168.213.1
  InternalApiDefaultRoute: 192.168.4.1
  ExternalInterfaceDefaultRoute: 10.204.17.1
  ManagementInterfaceDefaultRoute: 10.102.70.1
  # Vlans
  InternalApiNetworkVlanID: 710
  ExternalNetworkVlanID: 720
  StorageNetworkVlanID: 740
  StorageMgmtNetworkVlanID: 750
  # Services
```

```
  EC2MetadataIp: 192.168.213.10  # Generally the IP of the Undercloud
  DnsServers: ["YOUR_DNS_SERVER"]
  NtpServer: "YOUR_NTP_SERVER"
```

Contrail Service with Templates (contrail-services.yaml)

Customize Contrail services for your network by modifying the **contrail-services.yaml** file. We provide a sample at **~/tripleo-heat-templates/environments/contrail/contrail-services.yaml**. Look through that file for explanations of the parameters.

Here is the **contrail-services.yaml** for our example.

```
parameter_defaults:
  ServiceNetMap:
      <leave unchanged>
  NovaLiveMigrationPermitPostCopy: false
  NeutronMetadataProxySharedSecret: secret
  ContrailRegistry: REGISTRY_FOR_CONTRAIL_CONTAINERS:REGISTRY_PORT
  ContrailImageTag: CONTRAIL_TAG
  ContrailDefaults:
    APPLY_DEFAULTS: "True"
  ContrailSettings:
    VROUTER_ENCRYPTION: false
    VROUTER_GATEWAY: 192.168.33.1
    BGP_ASN: 64512
    BGP_AUTO_MESH: true
```

NOTE: APPLY_DEFAULTS

When Contrail is deployed for the first time, the default value of APPLY_DEFAULTS parameter in the ContrailDefaults section needs to be set to 'True'. This enables provisioning parameters present inside the template to use day0 configuration whenever a config provisioning container is restarted. Thus, the provisioning parameters are template driven and any changes to Contrail settings should be done through TripleO templates.

Contrail Networking allows you to configure some global configuration parameters like VXLAN network id mode, linklocal configuration, IBGP auto mesh configuration, enabling 4byte_AS, and changing BGP Global ASN through its web user interface. If you want to manage your cluster through web user interface, then you need to set

APPLY_DEFAULTS=False in ContrailDefaults section and deploy your cluster again by running openstack overcloud deploy. This additional step is required because when you have changed Contrail global configuration parameters through web user interface, then there is a possibility for these global configuration parameters to be overwritten if any config provisioner container is restarted. In order to avoid these values to be overwritten, set APPLY_DEFAULTS as 'False' and deploy Contrail again by running openstack overcloud deploy command. As a result, the global configuration parameters remain unchanged as provisioning is not executed again.

For example, if you set APPLY_DEFAULTS=False through TripleO template, deploy your Contrail cluster, set VxLAN Identifier Mode to 'User Configured' from web user interface, and restart config provisioner container, then VxLAN Identifier Mode remains 'User Configured' after the restart of config provisioner container. On the contrary, if APPLY_DEFAULTS is set to True, then after the restart of config provisioner container, VxLAN Identifier Mode will change to its default value, which is Automatic.

For example, if you set APPLY_DEFAULTS=False through TripleO template, deploy your Contrail cluster, set VxLAN Identifier Mode to 'User Configured' from web user interface, and restart config provisioner container, then VxLAN Identifier Mode remains 'User Configured' after the restart of config provisioner container. On the contrary, if APPLY_DEFAULTS is set to True, then after the restart of config provisioner container, VxLAN Identifier Mode changes to its default value, which is Automatic.

```
APPLY_DEFAULTS=True/False (default: True)
```

Contrail Plugins (contrail-plugins.yaml)

The Contrail plugins file contains various settings and refers to many other files used by Contrail. It is located at **~/tripleo-heat-templates/environments/contrail/contrail-plugins.yaml**.

Don't change this file or change the location of any of the files referenced.

Create the Overcloud

1. Locate the environment files you created in the previous procedures.
2. Deploy the overcloud.

   Double check to make sure the referenced files below are at the locations specified.

```
openstack overcloud deploy --templates tripleo-heat-templates/ \
  --stack overcloud --libvirt-type kvm \
  -n /home/stack/tripleo-heat-templates/environments/contrail/network_data.yaml \
```

```
  -r /home/stack/tripleo-heat-templates/environments/contrail/roles_data.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/rhsm.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/overcloud-baremetal-
deployed.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/overcloud-networks-
deployed.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/overcloud-vip-deployed.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-containers.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
  -e /home/stack/containers-prepare-parameter.yaml
```

You've now installed Contrail in RHOSP 17.1 in our example deployment.

Advanced Configuration

IN THIS SECTION

Contrail Networking provides a rich set of capabilities that go beyond our basic example. The following sections contain additional example YAML configuration that might apply to your deployment. This additional configuration is unrelated to our example.

Before using, convert these YAML examples to Jinja2 format. See Red Hat documentation for information on how to do this.

Advanced vRouter Kernel Mode Configuration

In addition to the standard NIC configuration, the vRouter kernel mode supports VLAN, Bond, and Bond + VLAN modes. The configuration snippets below only show the relevant section of the NIC template configuration for each mode.

Table 18: Advanced Interface Types (Kernel Mode)

Interface Types	Example Configuration

VLAN	<pre>- name: enp2s0
	  type: interface
	  use_dhcp: false
	- type: vlan
	  device: enp2s0
	  vlan_id: {{ tenant_vlan_id }}
	  use_dhcp: false
	- name: vhost0
	  type: contrail_vrouter
	  addresses:
	  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}
	  members:
	  - name: vlan{{ tenant_vlan_id }}
	  type: interface
	  use_dhcp: false
	  mtu: 1500
	  use_dhcp: false</pre>

Bond	<pre>- name: bond0
	  type: linux_bond
	  bonding_options: mode=4 xmit_hash_policy=layer2+3
	  use_dhcp: false
	  members:
	  - type: interface
	  name: enp2s0
	  - type: interface
	  name: enp3s0
	- name: vhost0
	  type: contrail_vrouter
	  mtu: 1500
	  use_dhcp: false
	  addresses:
	  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}
	  members:
	  - name: bond0
	  type: interface
	  use_dhcp: false</pre>

Table 18: Advanced Interface Types (Kernel Mode) ***(Continued)***

Interface Types	Example Configuration

Bond + VLAN	- name: bond0
	  type: linux_bond
	  bonding_options: mode=4 xmit_hash_policy=layer2+3
	  use_dhcp: false
	  members:
	  - type: interface
	  name: enp2s0
	  - type: interface
	  name: enp3s0
	- device: bond0
	  type: vlan
	  vlan_id: {{ tenant_vlan_id }}
	  use_dhcp: false
	- name: vhost0
	  type: contrail_vrouter
	  mtu: 1500
	  use_dhcp: false
	  addresses:
	  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}
	  members:
	  - name: vlan{{ tenant_vlan_id }}
	  type: interface
	  use_dhcp: false

Advanced vRouter DPDK Mode Configuration

In addition to the standard NIC configuration, the vRouter DPDK mode supports Standard, VLAN, Bond, and Bond + VLAN modes.

Network Environment Configuration:

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages:

```
 # For Intel CPU
  ContrailDpdkParameters:
```

```
    KernelArgs: "intel_iommu=on iommu=pt default_hugepagesz=1GB hugepagesz=1G hugepages=4
hugepagesz=2M hugepages=1024"
    ExtraSysctlSettings:
      # must be equal to value from kernel args: hugepages=4
      vm.nr_hugepages:
        value: 4
      vm.max_map_count:
        value: 128960
```

See the following NIC template configurations for vRouter DPDK mode. The configuration snippets below only show the relevant section of the NIC configuration for each mode.

Table 19: Advanced Interface Types (DPDK)

Interface Types	Example Configuration

Standard	`- name: vhost0`
	`type: contrail_vrouter_dpdk`
	`driver: uio_pci_generic`
	`cpu_list: 0x01`
	`mtu: 1500`
	`use_dhcp: false`
	`addresses:`
	`- ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}`
	`members:`
	`- name: enp2s0`
	`type: interface`
	`use_dhcp: false`

Table 19: Advanced Interface Types (DPDK) *(Continued)*

Interface Types	Example Configuration

VLAN	`- name: vhost0`
	`  type: contrail_vrouter_dpdk`
	`  driver: uio_pci_generic`
	`  cpu_list: 0x01`
	`  mtu: 1500`
	`  use_dhcp: false`
	`  vlan_id: {{ tenant_vlan_id }}`
	`  addresses:`
	`  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}`
	`  members:`
	`  - name: enp2s0`
	`  type: interface`
	`  use_dhcp: false`

Bond	`- name: vhost0`
	`  type: contrail_vrouter_dpdk`
	`  driver: uio_pci_generic`
	`  cpu_list: 0x01`
	`  mtu: 1500`
	`  use_dhcp: false`
	`  bond_mode: 4`
	`  bond_policy: layer2+3`
	`  addresses:`
	`  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}`
	`  members:`
	`  - name: enp2s0`
	`  type: interface`
	`  use_dhcp: false`
	`  - name: enp3s0`
	`  type: interface`
	`  use_dhcp: false`

Table 19: Advanced Interface Types (DPDK) *(Continued)*

Interface Types	Example Configuration

Bond + VLAN	<pre>- name: vhost0
	  type: contrail_vrouter_dpdk
	  driver: uio_pci_generic
	  cpu_list: 0x01
	  mtu: 1500
	  use_dhcp: false
	  bond_mode: 4
	  bond_policy: layer2+3
	  vlan_id: {{ tenant_vlan_id }}
	  addresses:
	  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}
	  members:
	  - name: enp2s0
	  type: interface
	  use_dhcp: false
	  - name: enp3s0
	  type: interface
	  use_dhcp: false</pre>

Advanced vRouter SRIOV + Kernel Mode Configuration

vRouter SRIOV + Kernel mode can be used in the following combinations:

- Standard
- VLAN
- Bond
- Bond + VLAN

Network environment configuration:

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages:

```
ContrailSriovParameters:
    KernelArgs: "intel_iommu=on iommu=pt default_hugepagesz=1GB hugepagesz=1G hugepages=4
hugepagesz=2M hugepages=1024"
    ExtraSysctlSettings:
      # must be equal to value from 1G kernel args: hugepages=4
      vm.nr_hugepages:
        value: 4
```

SRIOV PF/VF settings:

```
NovaPCIPassthrough:
- devname: "ens2f1"
  physical_network: "sriov1"
ContrailSriovNumVFs: ["ens2f1:7"]
```

The SRIOV NICs are not configured in the NIC templates. However, vRouter NICs must still be configured. See the following NIC template configurations for vRouter kernel mode. The configuration snippets below only show the relevant section of the NIC configuration for each mode.

Table 20: Interface Types (SR-IOV + Kernel Mode)

Interface Types	Example Configuration

Table 20: Interface Types (SR-IOV + Kernel Mode) *(Continued)*

Interface Types	Example Configuration

VLAN	- name: ens2f1
	  type: interface
	  use_dhcp: false
	- type: vlan
	  device: ens2f1
	  vlan_id: {{ tenant_vlan_id }}
	  use_dhcp: false
	- name: vhost0
	  type: contrail_vrouter
	  addresses:
	  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}
	  members:
	  - name: vlan{{ tenant_vlan_id }}
	  type: interface
	  use_dhcp: false
	  mtu: 1500
	  use_dhcp: false

Bond	- name: bond0
	  type: linux_bond
	  bonding_options: mode=4 xmit_hash_policy=layer2+3
	  use_dhcp: false
	  members:
	  - type: interface
	  name: ens2f1
	  - type: interface
	  name: ens3f1
	- name: vhost0
	  type: contrail_vrouter
	  mtu: 1500
	  use_dhcp: false
	  addresses:
	  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}
	  members:
	  - name: bond0
	  type: interface
	  use_dhcp: false

Table 20: Interface Types (SR-IOV + Kernel Mode) ***(Continued)***

Interface Types	Example Configuration

Bond + VLAN	<pre>- name: bond0
	  type: linux_bond
	  bonding_options: mode=4 xmit_hash_policy=layer2+3
	  use_dhcp: false
	  members:
	  - type: interface
	  name: ens2f1
	  - type: interface
	  name: ens3f1
	- device: bond0
	  type: vlan
	  vlan_id: {{ tenant_vlan_id }}
	  use_dhcp: false
	- name: vhost0
	  type: contrail_vrouter
	  mtu: 1500
	  use_dhcp: false
	  addresses:
	  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}
	  members:
	  - name: vlan{{ tenant_vlan_id }}
	  type: interface
	  use_dhcp: false</pre>

Advanced vRouter SRIOV + DPDK Mode Configuration

Use vRouter SRIOV + DPDK in the following combinations:

- Standard
- VLAN
- Bond
- Bond + VLAN

Network environment configuration:

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages

```
ContrailSriovParameters:
    KernelArgs: "intel_iommu=on iommu=pt default_hugepagesz=1GB hugepagesz=1G hugepages=4
hugepagesz=2M hugepages=1024"
    ExtraSysctlSettings:
      # must be equal to value from 1G kernel args: hugepages=4
      vm.nr_hugepages:
        value: 4
```

SRIOV PF/VF settings

```
NovaPCIPassthrough:
         - devname: "ens2f1"
physical_network: "sriov1"
ContrailSriovNumVFs: ["ens2f1:7"]
```

The SRIOV NICs are not configured in the NIC templates. However, vRouter NICs must still be configured. See the following NIC template configurations for vRouter DPDK mode. The configuration snippets below only show the relevant section of the NIC configuration for each mode.

Table 21: Interface Types (SR-IOV + DPDK)

Interface Types	Example Configuration

Standard	`- name: vhost0`
	`  type: contrail_vrouter_dpdk`
	`  driver: uio_pci_generic`
	`  cpu_list: 0x01`
	`  mtu: 1500`
	`  use_dhcp: false`
	`  addresses:`
	`  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}`
	`  members:`
	`  - name: ens2f1`
	`  type: interface`
	`  use_dhcp: false`

Table 21: Interface Types (SR-IOV + DPDK) *(Continued)*

Interface Types	Example Configuration

VLAN	`- name: vhost0`
	`  type: contrail_vrouter_dpdk`
	`  driver: uio_pci_generic`
	`  cpu_list: 0x01`
	`  mtu: 1500`
	`  use_dhcp: false`
	`  vlan_id: {{ tenant_vlan_id }}`
	`  addresses:`
	`  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}`
	`  members:`
	`  - name: ens2f1`
	`  type: interface`
	`  use_dhcp: false`

Bond	`- name: vhost0`
	`  type: contrail_vrouter_dpdk`
	`  driver: uio_pci_generic`
	`  cpu_list: 0x01`
	`  mtu: 1500`
	`  use_dhcp: false`
	`  bond_mode: 4`
	`  bond_policy: layer2+3`
	`  addresses:`
	`  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}`
	`  members:`
	`  - name: ens2f1`
	`  type: interface`
	`  use_dhcp: false`
	`  - name: ens2f1`
	`  type: interface`
	`  use_dhcp: false`

Table 21: Interface Types (SR-IOV + DPDK) *(Continued)*

Interface Types	Example Configuration

Bond + VLAN	<pre>- name: vhost0
	  type: contrail_vrouter_dpdk
	  driver: uio_pci_generic
	  cpu_list: 0x01
	  mtu: 1500
	  use_dhcp: false
	  bond_mode: 4
	  bond_policy: layer2+3
	  vlan_id: {{ tenant_vlan_id }}
	  addresses:
	  - ip_netmask: {{ tenant_ip }}/{{ tenant_cidr }}
	  members:
	  - name: ens2f1
	  type: interface
	  use_dhcp: false
	  - name: ens3f1
	  type: interface
	  use_dhcp: false</pre>

CHAPTER 7

Setting Up Contrail with Red Hat OpenStack 16.1

IN THIS CHAPTER

Understanding Red Hat OpenStack Platform Director

IN THIS SECTION

Red Hat OpenStack Platform Director

Starting with Contrail Networking Release 2008, Contrail Networking supports using Contrail with Red Hat OpenStack Platform Director 16.1.

This chapter explains how to integrate a Contrail Networking Release 2008 (or higher) installation with Red Hat OpenStack Platform Director 16.1.

Red Hat OpenStack Platform provides an installer called the Red Hat OpenStack Platform director (RHOSPd or OSPd), which is a toolset based on the OpenStack project TripleO (OOO, OpenStack on OpenStack). TripleO is an open source project that uses features of OpenStack to deploy a fully functional, tenant-facing OpenStack environment.

TripleO can be used to deploy an RDO-based OpenStack environment integrated with Tungsten Fabric. Red Hat OpenStack Platform director can be used to deploy an RHOSP-based OpenStack environment integrated with Contrail Networking.

OSPd uses the concepts of undercloud and overcloud. OSPd sets up an undercloud, a single server running an operator-facing deployment that contains the OpenStack components needed to deploy and manage an overcloud, a tenant-facing deployment that hosts user workloads.

The overcloud is the deployed solution that can represent a cloud for any purpose, such as production, staging, test, and so on. The operator can select to deploy to their environment any of the available overcloud roles, such as controller, compute, and the like.

OSPd leverages existing core components of OpenStack including Nova, Ironic, Neutron, Heat, Glance, and Ceilometer to deploy OpenStack on bare metal hardware.

- Nova and Ironic are used in the undercloud to manage the bare metal instances that comprise the infrastructure for the overcloud.
- Neutron is used to provide a networking environment in which to deploy the overcloud.
- Glance stores machine images.
- Ceilometer collects metrics about the overcloud.

For more information about OSPd architecture, see OSPd documentation.

Contrail Networking Roles

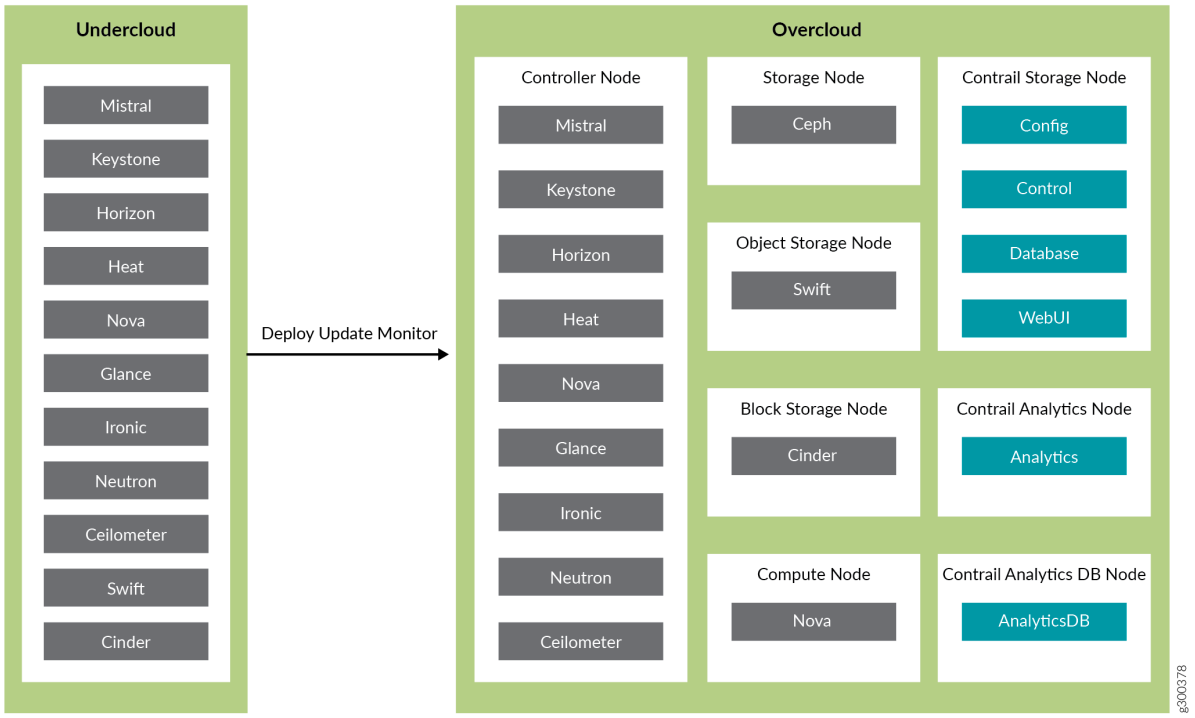
OSPd supports composable roles, which are groups of services that you define through Heat templates. Composable roles allow you to integrate Contrail Networking into the overcloud environment.

The following are the Contrail Networking roles used for integrating into the overcloud:

- Contrail Controller
- Contrail Analytics
- Contrail Analytics Database
- Contrail-TSN
- Contrail-DPDK

Figure 38 on page 320 shows the relationship and components of an undercloud and overcloud architecture for Contrail Networking.

Figure 38: Undercloud and Overcloud with Roles



RVM and KVM Requirements

Starting in Contrail Networking Release 21.4, Contrail Networking was enhanced to operate with hosts using Red Hat Virtualization (RHV). You must use RHV-based hosts in place of KVM-based hosts in RHOSP 16.1 environments starting in Contrail Networking Release 21.4 and in all future Contrail Networking releases.

In Release 21.3 and earlier Contrail Networking releases, this procedure is performed with hosts using Kernel-based Virtual Machine (KVM).

Undercloud Requirements

The undercloud is a single server or VM that hosts the OpenStack Platform director, which is an OpenStack installation used to provision OpenStack on the overcloud.

See Undercloud Requirements for the compute requirements of the undercloud.

Overcloud Requirements

The overcloud roles can be deployed to bare metal servers or to virtual machines (VMs), but the compute nodes must be deployed to bare metal systems. Every overcloud node must support IPMI for booting up from the undercloud using PXE.

Ensure the following requirements are met for the Contrail Networking nodes per role.

- Non-high availability: A minimum of 4 overcloud nodes are needed for control plane roles for a non-high availability deployment:
  - 1x contrail-config (includes Contrail control)
  - 1x contrail-analytics
  - 1x contrail-analytics-database
  - 1x OpenStack controller
- High availability: A minimum of 12 overcloud nodes are needed for control plane roles for a high availability deployment:
  - 3x contrail-config (includes Contrail control)
  - 3x contrail-analytics
  - 3x contrail-analytics-database
  - 3x OpenStack controller

  If the control plane roles are deployed to VMs, use 3 separate physical servers and deploy one role of each kind to each physical server.

See Overcloud Requirements for the compute requirements of the overcloud.

Networking Requirements

As a minimum, the installation requires two networks:

- provisioning network - This is the private network that the undercloud uses to provision the overcloud.
- external network - This is the externally-routable network you use to access the undercloud and overcloud nodes.

Ensure the following requirements are met for the provisioning network:

- One NIC from every machine must be in the same broadcast domain of the provisioning network, and it should be the same NIC on each of the overcloud machines. For example, if you use the

second NIC on the first overcloud machine, you should use the second NIC on each additional overcloud machine.

During installation, these NICs will be referenced by a single name across all overcloud machines.

- The provisioning network NIC should not be the same NIC that you are using for remote connectivity to the undercloud machine. During the undercloud installation, an Open vSwitch bridge will be created for Neutron, and the provisioning NIC will be bridged to the Open vSwitch bridge. Consequently, connectivity would be lost if the provisioning NIC was also used for remote connectivity to the undercloud machine.
- The provisioning NIC on the overcloud nodes must be untagged.
- You must have the MAC address of the NIC that will PXE boot the IPMI information for the machine on the provisioning network. The IPMI information will include such things as the IP address of the IPMI NIC and the IPMI username and password.
- All of the networks must be available to all of the Contrail Networking roles and computes.

While the provisioning and external networks are sufficient for basic applications, you should create additional networks in most overcloud environments to provide isolation for the different traffic types by assigning network traffic to specific network interfaces or bonds.

When isolated networks are configured, the OpenStack services are configured to use the isolated networks. If no isolated networks are configured, all services run on the provisioning network. If only some isolated networks are configured, traffic belonging to a network not configured runs on the provisioning network.

The following networks are typically deployed when using network isolation topology:

- Provisioning - used by the undercloud to provision the overcloud
- Internal API - used by OpenStack services to communicate with each other
- Tenant - used for tenant overlay data plane traffic (one network per tenant)
- Storage - used for storage data traffic
- Storage Management - used for storage control and management traffic
- External - provides external access to the undercloud and overcloud, including external access to the web UIs and public APIs
- Floating IP - provides floating IP access to the tenant network (can either be merged with external or can be a separate network)
- Management - provides access for system administration

Compatibility Matrix

The following combinations of Operating System/OpenStack/Deployer/Contrail Networking are supported:

Table 22: Compatibility Matrix

Operating System	OpenStack	Deployer	Contrail Networking
RHEL 8.2	OSP16	OSPd16	Contrail Networking 2008 or higher

Installation Summary

The general installation procedure is as follows:

- Set up the infrastructure, which is the set of servers or VMs that host the undercloud and overcloud, including the provisioning network that connects them together.
- Set up the undercloud, which is the OSPd application.
- Set up the overcloud, which is the set of services in the tenant-facing network. Contrail Networking is part of the overcloud.

For more information on installing and using the RHOSPd, see Red Hat documentation.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
2008	Starting with Contrail Networking Release 2008, Contrail Networking supports using Contrail with Red Hat OpenStack Platform Director 16.1.

Setting Up the Infrastructure (Contrail Networking Release 21.3 or Earlier)

SUMMARY

Follow this topic to set up the infrastructure for a Contrail Networking deployment in a RHOSP 16.1 environment when you are using Contrail Networking Release 21.3 or earlier.

IN THIS SECTION

When to Use This Procedure

You should use this topic to set up the infrastructure for a Contrail Networking deployment in a RHOSP 16.1 environment when you are using Contrail Networking Release 21.3 or earlier.

This procedure shows you how to set up the infrastructure for the installation when the hosts are using Kernel-based Virtual Machine (KVM).

Starting in Contrail Networking Release 21.4, Contrail Networking was enhanced to operate with hosts using Red Hat Virtualization (RHV). You must use RHV-based hosts in place of KVM-based hosts in RHOSP 16.1 environments starting in Contrail Networking Release 21.4 and in all future Contrail Networking releases. See Setting Up the Infrastructure (Contrail Networking Release 21.4 or Later) .

Target Configuration (Example)

Undercloud and overcloud KVM hosts require virtual switches and virtual machine definitions to be configured. You can deploy any KVM host operating system version that supports KVM and OVS. The following example shows a RHEL/CentOS based system. If you are using RHEL, you must subscribe the system.

The following example illustrates all control plane functions as Virtual Machines hosted on KVM hosts.

There are different ways to create the infrastructure providing the control plane elements. To illustrate the installation procedure, we will use four host machines for the infrastructure, each running KVM. KVM1 contains a VM running the undercloud while KVM2 through KVM4 each contains a VM running an OpenStack controller and a Contrail controller (Table 23 on page 325).

Table 23: Control Plane Infrastructure

KVM Host	Virtual Machines
KVM1	undercloud
KVM2	OpenStack Controller 1, Contrail Contoller 1
KVM3	OpenStack Controller 2, Contrail Contoller 2
KVM4	OpenStack Controller 3, Contrail Contoller 3

Figure 39 on page 325 shows the physical connectivity where each KVM host and each compute node has two interfaces that connect to an external switch. These interfaces attach to separate virtual bridges within the VM, allowing for two physically separate networks (external and provisioning networks).

Figure 39: Physical View

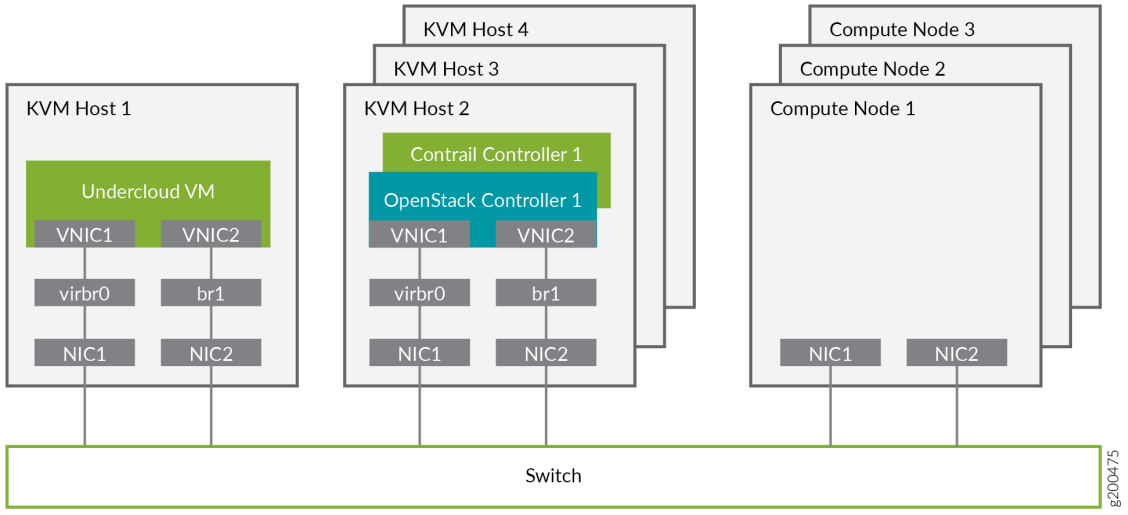
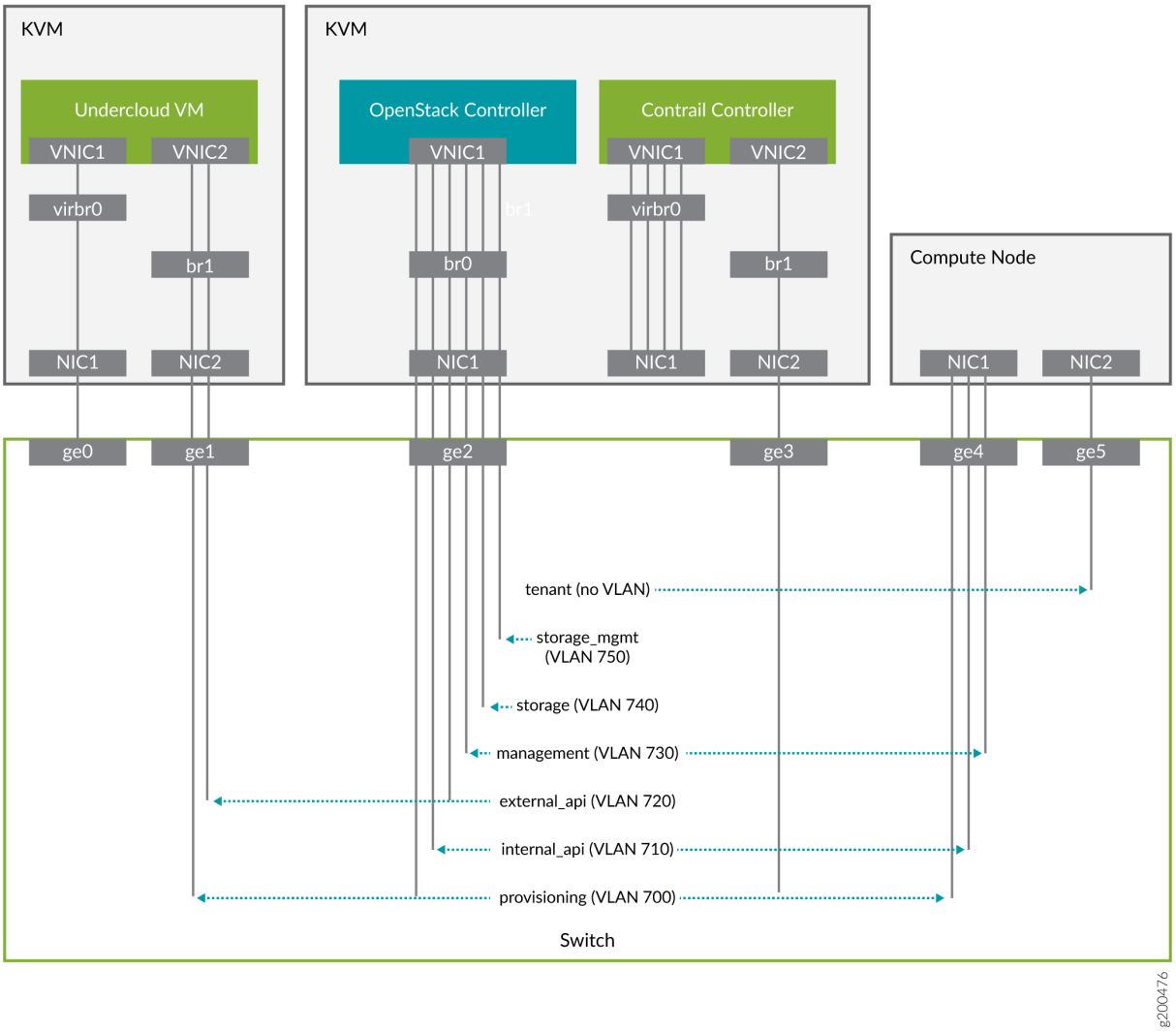


Figure 40 on page 326 shows the logical view of the connectivity where VLANs are used to provide further network separation for the different OpenStack network types.

Figure 40: Logical View



The following sections describe how to configure the infrastructure, the undercloud, and finally the overcloud.

Configure the External Physical Switch

Configure the ports and VLANs on the external physical switch according to the following table:

Table 24: External Physical Switch Port and VLAN Configuration

Port	Trunked VLAN	Native VLAN
ge0	-	-

Table 24: External Physical Switch Port and VLAN Configuration *(Continued)*

Port	Trunked VLAN	Native VLAN
ge1	700, 720	-
ge2	700, 710, 720, 730, 740, 750	-
ge3	-	-
ge4	710, 730	700
ge5	-	-

Configure KVM Hosts

Use this example procedure to install the required packages and start KVM and Open vSwitch on each undercloud and overcloud KVM host.

1. Log in to a KVM host.
2. Install the required packages.

```
yum install -y libguestfs \
   libguestfs-tools \
   openvswitch \
   virt-install \
   kvm libvirt \
   libvirt-python \
   python-virtualbmc \
   python-virtinst
```

3. Start KVM and Open vSwitch.

```
systemctl start libvirtd
systemctl start openvswitch
```

4. Additionally, on the overcloud nodes only, create and start the virtual switches br0 and br1.

Table 25: vSwitch Configuration

Bridge	Trunked VLAN	Native VLAN
br0	710, 720, 730 740, 750	700
br1	-	-

```
# Create the virtual switches and bind them to the respective interfaces.
ovs-vsctl add-br br0
ovs-vsctl add-br br1
ovs-vsctl add-port br0 NIC1
ovs-vsctl add-port br1 NIC2


# Create the configuration file for br0.
cat << EOF > br0.xml
<network>
   <name>br0</name>
   <forward mode='bridge'/>
   <bridge name='br0'/>
   <virtualport type='openvswitch'/>
   <portgroup name='overcloud'/>
      <vlan trunk='yes'>
         <tag id='700' nativeMode='untagged'/>
         <tag id='710'/>
         <tag id='720'/>
         <tag id='730'/>
         <tag id='740'/>
         <tag id='750'/>
      </vlan>
   </portgroup>
</network>
EOF


# Create the configuration file for br1.
cat << EOF > br1.xml
<network>
   <name>br1</name>
```

```
   <forward mode='bridge'/>
   <bridge name='br1'/>
   <virtualport type='openvswitch'/>
</network>
EOF


# Create the br0 network based on the configuration file.
virsh net-define br0.xml
virsh net-start br0
virsh net-autostart br0


# Create the br1 network based on the configuration file.
virsh net-define br1.xml
virsh net-start br1
virsh net-autostart br1
```

5. Repeat step 1 through step 4 for each KVM host.

Create the Overcloud VM Definitions on the Overcloud KVM Hosts

Use this example procedure on each overcloud KVM host (KVM2 to KVM4) to do the following:

- create the VM definitions for that overcloud KVM host
- create and start a virtual baseboard management controller for that overcloud KVM host so that the VM can be managed using IPMI
- create an **ironic_list** file to be used by the undercloud

This example procedure creates a VM definition consisting of 2 compute nodes, 1 Contrail controller node, and 1 OpenStack controller node on each overcloud KVM host.

1. Log in to an overcloud KVM host.
2. Specify the roles you want to create.

```
ROLES=compute:2,contrail-controller:1,control:1
```

3. Create the VM definitions.

```
# Initialize and specify the IPMI user and password you want to use.
num=0
ipmi_user=<user>
```

```
ipmi_password=<password>
libvirt_path=/var/lib/libvirt/images
port_group=overcloud
prov_switch=br0
/bin/rm ironic_list


# For each role and instance specified in the ROLES variable:
#    - create the VM definition
#    - create and start a virtual baseboard management controller (vbmc)
#    - store the VM information into an ironic_list file (for later use in the undercloud)
IFS=',' read -ra role_list <<< "${ROLES}"
for role in ${role_list[@]}; do
   role_name=`echo $role|cut -d ":" -f 1`
   role_count=`echo $role|cut -d ":" -f 2`
   for count in `seq 1 ${role_count}`; do
      echo $role_name $count
      qemu-img create -f qcow2 ${libvirt_path}/${role_name}_${count}.qcow2 99G
      virsh define /dev/stdin <<EOF
      $(virt-install --name ${role_name}_${count} \
         --disk ${libvirt_path}/${role_name}_${count}.qcow2 \
         --vcpus=4 \
         --ram=16348 \
         --network network=br0,model=virtio,portgroup=${port_group} \
         --network network=br1,model=virtio \
         --virt-type kvm \
         --cpu host \
         --import \
         --os-variant rhel8.2 \
         --serial pty \
         --console pty,target_type=virtio \
         --graphics vnc \
         --print-xml)
EOF
      vbmc add ${role_name}_${count} --port 1623${num} --username ${ipmi_user} --password $
{ipmi_password}
      vbmc start ${role_name}_${count}
      prov_mac=`virsh domiflist ${role_name}_${count}|grep ${prov_switch}|awk '{print $5}'`
      vm_name=${role_name}-${count}-`hostname -s`
      kvm_ip=`ip route get 1  |grep src |awk '{print $7}'`
      echo ${prov_mac} ${vm_name} ${kvm_ip} ${role_name} 1623${num}>> ironic_list
      num=$(expr $num + 1)
```

```
    done
done
```

4. Repeat step 1 through step 3 on each overcloud KVM host.

CAUTION: This procedure creates one **ironic_list** file per overcloud KVM host. Combine the contents of each file into a single **ironic_list** file on the undercloud.

The following shows the resulting **ironic_list** file after you combine the contents from each separate file:

```
52:54:00:e7:ca:9a compute-1-5b3s31 10.87.64.32 compute 16230 52:54:00:30:6c:3f compute-2-5b3s31
10.87.64.32 compute 16231 52:54:00:9a:0c:d5 contrail-controller-1-5b3s31 10.87.64.32 contrail-
controller 16232 52:54:00:cc:93:d4 control-1-5b3s31 10.87.64.32 control 16233 52:54:00:28:10:d4
compute-1-5b3s30 10.87.64.31 compute 16230 52:54:00:7f:36:e7 compute-2-5b3s30 10.87.64.31 compute
16231 52:54:00:32:e5:3e contrail-controller-1-5b3s30 10.87.64.31 contrail-controller 16232
52:54:00:d4:31:aa control-1-5b3s30 10.87.64.31 control 16233 52:54:00:d1:d2:ab compute-1-5b3s32
10.87.64.33 compute 16230 52:54:00:ad:a7:cc compute-2-5b3s32 10.87.64.33 compute 16231
52:54:00:55:56:50 contrail-controller-1-5b3s32 10.87.64.33 contrail-controller 16232
52:54:00:91:51:35 control-1-5b3s32 10.87.64.33 control 16233
```

Create the Undercloud VM Definition on the Undercloud KVM Host

Use this example procedure on the undercloud KVM host (KVM1) to create the undercloud VM definition and to start the undercloud VM.

1. Create the images directory.

```
mkdir ~/images
cd images
```

2. Retrieve the image.

   RHEL

   Download rhel-server-8.2-update-1-x86_64-kvm.qcow2 from RedHat portal to ~/images.

```
cloud_image=~/images/rhel-server-8.2-update-1-x86_64-kvm.qcow2
```

3. Customize the undercloud image.

```
undercloud_name=queensa
undercloud_suffix=local
```

```
root_password=<password>
stack_password=<password>
export LIBGUESTFS_BACKEND=direct
qemu-img create -f qcow2 /var/lib/libvirt/images/${undercloud_name}.qcow2 100G
virt-resize --expand /dev/sda1 ${cloud_image} /var/lib/libvirt/images/$
{undercloud_name}.qcow2
virt-customize  -a /var/lib/libvirt/images/${undercloud_name}.qcow2 \
--run-command 'xfs_growfs /' \
--root-password password:${root_password} \
--hostname ${undercloud_name}.${undercloud_suffix} \
--run-command 'useradd stack' \
--password stack:password:${stack_password} \
--run-command 'echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack' \
--chmod 0440:/etc/sudoers.d/stack \
--run-command 'sed -i "s/PasswordAuthentication no/PasswordAuthentication yes/g" /etc/ssh/
sshd_config' \
--run-command 'systemctl enable sshd' \
--run-command 'yum remove -y cloud-init' \
--selinux-relabel
```

NOTE: As part of the undercloud definition, a user called **stack** is created. This user will be used later to install the undercloud.

4. Define the undercloud virsh template.

```
vcpus=8
vram=32000
virt-install --name ${undercloud_name} \
--disk /var/lib/libvirt/images/${undercloud_name}.qcow2 \
--vcpus=${vcpus} \
--ram=${vram} \
--network network=default,model=virtio \
--network network=br0,model=virtio,portgroup=overcloud \
--virt-type kvm \
--import \
--os-variant rhel8.2 \
--graphics vnc \
--serial pty \
--noautoconsole \
--console pty,target_type=virtio
```

5. Start the undercloud VM.

```
virsh start ${undercloud_name}
```

6. Retrieve the undercloud IP address. It might take several seconds before the IP address is available.

```
undercloud_ip=`virsh domifaddr ${undercloud_name} |grep ipv4 |awk '{print $4}' |awk -F"/"
'{print $1}'` ssh-copy-id ${undercloud_ip}
```

Setting Up the Undercloud

SUMMARY

Follow this topic to setup the undercloud for a Contrail Networking deployment with RHOSP 16.

IN THIS SECTION

Follow this topic to setup the undercloud for a Contrail Networking deployment with RHOSP 16.

Contrail Networking was enhanced to operate with hosts using Red Hat Virtualization (RHV) in Contrail Networking Release 21.4. Prior to this enhancement, Contrail Networking was supported in environments with hosts using Kernel-based Virtual Machine (KVM) only.

These instructions apply to both environments.

Install the Undercloud

Use this example procedure to install the undercloud.

1. Log in to the undercloud VM from the undercloud KVM host.

```
ssh ${undercloud_ip}
```

2. Configure the hostname.

```
undercloud_name=`hostname -s`
undercloud_suffix=`hostname -d`
```

```
hostnamectl set-hostname ${undercloud_name}.${undercloud_suffix}
hostnamectl set-hostname --transient ${undercloud_name}.${undercloud_suffix}
```

3. Add the hostname to the **/etc/hosts** file. The following example assumes the management interface is eth0.

```
undercloud_ip=`ip addr sh dev eth0 | grep "inet " | awk '{print $2}' | awk -F"/" '{print $1}'`
echo ${undercloud_ip} ${undercloud_name}.${undercloud_suffix} ${undercloud_name} >> /etc/hosts
```

4. Set up the repositories.

   RHEL

```
#Register with Satellite (can be done with CDN as well)
satellite_fqdn=device.example.net
act_key=xxx
org=example
yum localinstall -y http://${satellite_fqdn}/pub/katello-ca-consumer-latest.noarch.rpm
subscription-manager register --activationkey=${act_key} --org=${org}
```

5. Install the Tripleo client.

```
yum install -y python-tripleoclient tmux
```

6. Copy the undercloud configuration file sample and modify the configuration as required. See Red Hat documentation for information on how to modify that file.

```
su - stack
cp /usr/share/python-tripleoclient/undercloud.conf.sample ~/undercloud.conf
vi ~/undercloud.conf
```

7. Install the undercloud.

```
openstack undercloud install
source stackrc
```

8. If you are using a satellite for deployment, manually update the hostname and satellite IP addresses in your /etc/hosts/ file.

To perform this procedure using the VI editor:

```
(undercloud) [stack@osp16-5c5s36 ~]$ sudo vi /etc/hosts
```

and manually enter your hostname and satellite IP address in the file while using the editor.

This step ensures that the overcloud deployment is successful later in the procedure.

You should also perform this step if the overcloud deployment fails later in the procedure and a failed lookup URL message appears on the console as the reason.

A sample failed lookup URL error message when you experience this issue:.

```
=======================
TASK [redhat-subscription : SATELLITE | Run Satellite 6 tasks] *****************
Tuesday 30 March 2021  12:11:25 -0400 (0:00:00.490)       0:13:39.737 *********
included: /usr/share/ansible/roles/redhat-subscription/tasks/satellite-6.yml for overcloud-
controller-0, overcloud-controller-1, overcloud-controller-2
TASK [redhat-subscription : SATELLITE 6 | Set Satellite server CA as a fact] ***Tuesday 30
March 2021  12:11:26 -0400 (0:00:00.730)       0:13:40.467 *********
fatal: [overcloud-controller-0]: FAILED! => {"msg": "An unhandled exception occurred while
running the lookup plugin 'url'. Error was a <class 'ansible.errors.AnsibleError'>, original
message: Failed lookup url for  : <urlopen error [Errno -2] Name or service not
known>"}fatal: [overcloud-controller-1]: FAILED! => {"msg": "An unhandled exception occurred
while running the lookup plugin 'url'. Error was a <class 'ansible.errors.AnsibleError'>,
original message: Failed lookup url for  : <urlopen error [Errno -2] Name or service not
known>"}

fatal: [overcloud-controller-2]: FAILED! => {"msg": "An unhandled exception occurred while
running the lookup plugin 'url'. Error was a <class 'ansible.errors.AnsibleError'>, original
message: Failed lookup url for  : <urlopen error [Errno -2] Name or service not known>"}
```

Perform Post-Install Configuration

1. Configure a forwarding path between the provisioning network and the external network:

```
sudo iptables -A FORWARD -i br-ctlplane -o eth0 -j ACCEPT
sudo iptables -A FORWARD -i eth0 -o br-ctlplane -m state --state RELATED,ESTABLISHED -j
ACCEPT
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

2. Add the external API interface:

```
sudo ip link add name vlan720 link br-ctlplane type vlan id 720
sudo ip addr add 10.2.0.254/24 dev vlan720
sudo ip link set dev vlan720 up
```

3. Add the `stack` user to the docker group:

```
newgrp docker
exit
su - stack
source stackrc
```

4. Manually add the satellite IP address and hostname into the **/etc/hosts/** file.

Setting Up the Overcloud

SUMMARY

Follow this topic to setup the overcloud for a Contrail Networking deployment with RHOSP 16.

IN THIS SECTION

Follow this topic to setup the overcloud for a Contrail Networking deployment with RHOSP 16.

Contrail Networking was enhanced to operate with hosts using Red Hat Virtualization (RHV) in Contrail Networking Release 21.4. Prior to this enhancement, Contrail Networking was supported in environments with hosts using Kernel-based Virtual Machine (KVM) only.

These instructions apply to both environments unless otherwise noted. In cases where the running virtualization engine impacts this procedure, the steps to perform in environments using RHV or KVM are noted.

Configuring the Overcloud

Use this example procedure on the undercloud to set up the configuration for the overcloud.

1. Specify the name server to be used:

```
undercloud_nameserver=8.8.8.8
openstack subnet set `openstack subnet show ctlplane-subnet -c id -f value` --dns-nameserver $
{undercloud_nameserver}
```

2. Retrieve and upload the overcloud images.

   a. Create the image directory:

```
mkdir images
cd images
```

   b. Retrieve the overcloud images from either the RDO project or from Red Hat.

      OSP16

```
sudo yum install -y rhosp-director-images rhosp-director-images-ipa
for i in /usr/share/rhosp-director-images/overcloud-full-latest-16.0.tar /usr/share/rhosp-
director-images/ironic-python-agent-latest-16.0.tar ; do tar -xvf $i; done
```

   c. Upload the overcloud images:

```
cd
openstack overcloud image upload --image-path /home/stack/images/
```

3. Prepare OpenStack's bare metal provisioning (Ironic).

   The Ironic driver installation depends on the virtualization engine running for Red Hat Openstack:

   - Red Hat Virtualization (RHV, Contrail Networking Release 21.4 and later): Use staging-ovirt to download the Ironic driver.

     See the Creating virtualized control planes document from Red Hat to enable the control plane with the staging-ovirt driver.

   - Kernel-based Virtual Machine (KVM, Contrail Networking Release 21.3 and earlier releases that support RHOSP16): Use the IPMI driver to download the Ironic driver.

     The IPMI driver download procedure is provided in these steps.

NOTE: Make sure to combine the **ironic_list** files from the three overcloud KVM hosts.

a. Add the overcloud VMs to Ironic:

```
ipmi_password=<password>
ipmi_user=<user>
while IFS= read -r line; do
  mac=`echo $line|awk '{print $1}'`
  name=`echo $line|awk '{print $2}'`
  kvm_ip=`echo $line|awk '{print $3}'`
  profile=`echo $line|awk '{print $4}'`
  ipmi_port=`echo $line|awk '{print $5}'`
  uuid=`openstack baremetal node create --driver ipmi \
                                        --property cpus=4 \
                                        --property memory_mb=16348 \
                                        --property local_gb=100 \
                                        --property cpu_arch=x86_64 \
                                        --driver-info ipmi_username=${ipmi_user}  \
                                        --driver-info ipmi_address=${kvm_ip} \
                                        --driver-info ipmi_password=${ipmi_password} \
                                        --driver-info ipmi_port=${ipmi_port} \
                                        --name=${name} \
                                        --property capabilities=profile:$
{profile},boot_option:local \
                                        -c uuid -f value`
  openstack baremetal port create --node ${uuid} ${mac}
done < <(cat ironic_list)

DEPLOY_KERNEL=$(openstack image show bm-deploy-kernel -f value -c id)
DEPLOY_RAMDISK=$(openstack image show bm-deploy-ramdisk -f value -c id)

for i in `openstack baremetal node list -c UUID -f value`; do
  openstack baremetal node set $i --driver-info deploy_kernel=$DEPLOY_KERNEL --driver-info
deploy_ramdisk=$DEPLOY_RAMDISK
done

for i in `openstack baremetal node list -c UUID -f value`; do
  openstack baremetal node show $i -c properties -f value
done
```

b. Introspect the overcloud node:

```
for node in $(openstack baremetal node list -c UUID -f value) ; do
  openstack baremetal node manage $node
done
openstack overcloud node introspect --all-manageable --provide
```

4. Create Flavor:

```
for i in compute-dpdk \
compute-sriov \
contrail-controller \
contrail-analytics \
contrail-database \
contrail-analytics-database; do
  openstack flavor create $i --ram 4096 --vcpus 1 --disk 40
  openstack flavor set --property "capabilities:boot_option"="local" \
                       --property "capabilities:profile"="${i}" ${i}
  openstack flavor set --property resources:CUSTOM_BAREMETAL=1 --property
resources:DISK_GB='0'
                       --property resources:MEMORY_MB='0'
                       --property resources:VCPU='0' ${i}
done
```

5. Copy the TripleO heat templates.

```
cp -r /usr/share/openstack-tripleo-heat-templates/ tripleo-heat-templates
```

6. Download and copy the Contrail heat templates from https://support.juniper.net/support/downloads.

```
tar -xzvf contrail-tripleo-heat-templates-<version>.tgz
cp -r contrail-tripleo-heat-templates/* tripleo-heat-templates/
```

7. Create **rhsm.yaml** file with your RedHat credentials

```
parameter_defaults:
  RhsmVars:
    rhsm_repos:
      - fast-datapath-for-rhel-8-x86_64-rpms
      - openstack-16.1-for-rhel-8-x86_64-rpms
```

```
    - satellite-tools-6.5-for-rhel-8-x86_64-rpms
    - ansible-2-for-rhel-8-x86_64-rpms
    - rhel-8-for-x86_64-highavailability-rpms
    - rhel-8-for-x86_64-appstream-rpms
    - rhel-8-for-x86_64-baseos-rpms
  rhsm_username: "YOUR_REDHAT_LOGIN"
  rhsm_password: "YOUR_REDHAT_PASSWORD"
  rhsm_org_id: "YOUR_REDHAT_ID"
  rhsm_pool_ids: "YOUR_REDHAT_POOL_ID"
```

8. Create and upload the OpenStack containers.

   a. Create the OpenStack container file.

   NOTE: The container must be created based on the OpenStack program.

   OSP16

```
sudo openstack tripleo container image prepare \
  -e ~/containers-prepare-parameter.yaml
  -e ~/rhsm.yaml > ~/overcloud_containers.yaml

sudo openstack overcloud container image upload --config-file ~/overcloud_containers.yaml
```

   b. Upload the OpenStack containers:

```
openstack overcloud container image upload --config-file ~/local_registry_images.yaml
```

9. Create and upload the Contrail containers.

   a. Create the Contrail container file.

   NOTE: This step is optional. The Contrail containers can be downloaded from external registries later.

```
cd ~/tf-heat-templates/tools/contrail
./import_contrail_container.sh -f container_outputfile -r registry -t tag [-i insecure] [-
u username] [-p password] [-c certificate path]
```

   Here are few examples of importing Contrail containers from different sources:

- Import from password protected public registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r hub.juniper.net/contrail -
u USERNAME -p PASSWORD -t 1234
```

- Import from Dockerhub:

```
./import_contrail_container.sh -f /tmp/contrail_container -r docker.io/
opencontrailnightly -t 1234
```

- Import from private secure registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r device.example.net:5443 -
c http://device.example.net/pub/device.example.net.crt -t 1234
```

- Import from private insecure registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r 10.0.0.1:5443 -i 1 -t 1234
```

b. Upload Contrail containers to the undercloud registry:

```
openstack overcloud container image upload --config-file /tmp/contrail_container
```

Customizing the Contrail Service with Templates (contrail-services.yaml)

This section contains information to customize Contrail services for your network by modifying the **contrail-services.yaml** file.

- APPLY_DEFAULTS Settings customization - When Contrail is deployed for the first time, the default value of APPLY_DEFAULTS parameter in the ContrailSettings/global ContrailSettings section needs to be set to 'True'.

```
APPLY_DEFAULTS=True/False (default: True)
```

There are multiple ContrailSettings sections referring to global configuration and to specific roles like e.g. DPDK nodes. This enables provisioning parameters present inside the template to use day0 configuration whenever a config provisioning container is restarted. Thus, the provisioning

parameters are template driven and any changes to Contrail settings should be done through TripleO templates.

Contrail Networking allows you to configure some global configuration parameters like VXLAN network id mode, linklocal configuration, IBGP auto mesh configuration, enabling 4byte_AS, and changing BGP Global ASN through its web user interface. If you want to manage your cluster through web user interface, then you need to set APPLY_DEFAULTS=False in ContrailSettings section and deploy your cluster again by running openstack overcloud deploy. This additional step is required because when you have changed Contrail global configuration parameters through web user interface, then there is a possibility for these global configuration parameters to be overwritten if any config provisioner container is restarted. In order to avoid these values to be overwritten, set APPLY_DEFAULTS as ’False’ and deploy Contrail again by running openstack overcloud deploy command. As a result, the global configuration parameters remain unchanged as provisioning is not executed again.

For example, if you set APPLY_DEFAULTS=False through TripleO template, deploy your Contrail cluster, set VxLAN Identifier Mode to ’User Configured’ from web user interface, and restart config provisioner container, then VxLAN Identifier Mode will remain ’User Configured’ after the restart of config provisioner container. On the contrary, if APPLY_DEFAULTS is set to True, then after the restart of config provisioner container, VxLAN Identifier Mode will change to its default value, which is Automatic.

For example, if you set APPLY_DEFAULTS=False through TripleO template, deploy your Contrail cluster, set VxLAN Identifier Mode to ’User Configured’ from web user interface, and restart config provisioner container, then VxLAN Identifier Mode will remain ’User Configured’ after the restart of config provisioner container. On the contrary, if APPLY_DEFAULTS is set to True, then after the restart of config provisioner container, VxLAN Identifier Mode will change to its default value, which is Automatic.

Example:

```
parameter_defaults:
  ContrailSettings:
    APPLY_DEFAULTS: true
    VROUTER_GATEWAY: 10.0.0.1
    # KEY1: value1
    # KEY2: value2

    VXLAN_VN_ID_MODE: "configured"
    ENCAP_PRIORITY: "VXLAN,MPLSoUDP,MPLSoGRE"

  ContrailControllerParameters:
    AAAMode: rbac
```

- Contrail Services customization

```
vi ~/tripleo-heat-templates/environments/contrail-services.yaml
parameter_defaults:
  ContrailSettings:
    VROUTER_GATEWAY: 10.0.0.1
    # KEY1: value1
    # KEY2: value2

    VXLAN_VN_ID_MODE: "configured"
    ENCAP_PRIORITY: "VXLAN,MPLSoUDP,MPLSoGRE"

  ContrailControllerParameters:
    AAAMode: rbac
```

- Contrail registry settings

```
vi ~/tripleo-heat-templates/environments/contrail-services.yaml
```

Here are few examples of default values for various registries:

- Public Juniper registry

```
parameter_defaults:
  ContrailRegistry: hub.juniper.net/contrail
  ContrailRegistryUser: <USER>
  ContrailRegistryPassword: <PASSWORD>
```

- Insecure registry

```
parameter_defaults:
  ContrailRegistryInsecure: true
  DockerInsecureRegistryAddress: 10.87.64.32:5000,192.168.24.1:8787
  ContrailRegistry: 10.87.64.32:5000
```

- Private secure registry

```
parameter_defaults:
  ContrailRegistryCertUrl: http://device.example.net/pub/device.example.net.crt
  ContrailRegistry: device.example.net:5443
```

- Contrail Container image settings

```
parameter_defaults:
  ContrailImageTag: queens-5.0-104-rhel-queens
```

Customizing the Contrail Network with Templates

IN THIS SECTION

Overview

In order to customize the network, define different networks and configure the overcloud nodes NIC layout. TripleO supports a flexible way of customizing the network.

The following networking customization example uses network as:

Table 26: Network Customization

Network	VLAN	overcloud Nodes
provisioning	-	All
internal_api	710	All
external_api	720	OpenStack CTRL
storage	740	OpenStack CTRL, Computes
storage_mgmt	750	OpenStack CTRL
tenant	-	Contrail CTRL, Computes

Roles Configuration (roles_data_contrail_aio.yaml)

IN THIS SECTION

The networks must be activated per role in the roles_data file:

```
vi ~/tripleo-heat-templates/roles_data_contrail_aio.yaml
```

OpenStack Controller

```
###############################################################################
# Role: Controller                                                            #
###############################################################################
- name: Controller
  description: |
    Controller role that has all the controler services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  tags:
    - primary
    - controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
```

Compute Node

```
###############################################################################
# Role: Compute                                                               #
###############################################################################
- name: Compute
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi
    - Tenant
    - Storage
```

Contrail Controller

```
###############################################################################
# Role: ContrailController                                                    #
###############################################################################
- name: ContrailController
```

```
  description: |
    ContrailController role that has all the Contrail controler services loaded
    and handles config, control and webui functions
  CountDefault: 1
  tags:
    - primary
    - contrailcontroller
  networks:
    - InternalApi
    - Tenant
```

Compute DPDK

```
################################################################################
# Role: ContrailDpdk                                                           #
################################################################################
- name: ContrailDpdk
  description: |
    Contrail Dpdk Node role
  CountDefault: 0
  tags:
    - contraildpdk
  networks:
    - InternalApi
    - Tenant
    - Storage
```

Compute SRIOV

```
################################################################################
# Role: ContrailSriov
################################################################################
- name: ContrailSriov
  description: |
    Contrail Sriov Node role
  CountDefault: 0
  tags:
    - contrailsriov
  networks:
    - InternalApi
```

```
    - Tenant
    - Storage
```

Compute CSN

```
###############################################################################
# Role: ContrailTsn
###############################################################################
- name: ContrailTsn
  description: |
    Contrail Tsn Node role
  CountDefault: 0
  tags:
    - contrailtsn
  networks:
    - InternalApi
    - Tenant
    - Storage
```

Network Parameter Configuration (contrail-net.yaml)

```
cat ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml
resource_registry:
  OS::TripleO::Controller::Net::SoftwareConfig: ../../network/config/contrail/controller-nic-
config.yaml
  OS::TripleO::ContrailController::Net::SoftwareConfig: ../../network/config/contrail/contrail-
controller-nic-config.yaml
  OS::TripleO::ContrailControlOnly::Net::SoftwareConfig: ../../network/config/contrail/contrail-
controller-nic-config.yaml
  OS::TripleO::Compute::Net::SoftwareConfig: ../../network/config/contrail/compute-nic-
config.yaml
  OS::TripleO::ContrailDpdk::Net::SoftwareConfig: ../../network/config/contrail/contrail-dpdk-
nic-config.yaml
  OS::TripleO::ContrailSriov::Net::SoftwareConfig: ../../network/config/contrail/contrail-sriov-
nic-config.yaml
  OS::TripleO::ContrailTsn::Net::SoftwareConfig: ../../network/config/contrail/contrail-tsn-nic-
config.yaml

parameter_defaults:
  # Customize all these values to match the local environment
```

```
TenantNetCidr: 10.0.0.0/24
InternalApiNetCidr: 10.1.0.0/24
ExternalNetCidr: 10.2.0.0/24
StorageNetCidr: 10.3.0.0/24
StorageMgmtNetCidr: 10.4.0.0/24
# CIDR subnet mask length for provisioning network
ControlPlaneSubnetCidr: '24'
# Allocation pools
TenantAllocationPools: [{'start': '10.0.0.10', 'end': '10.0.0.200'}]
InternalApiAllocationPools: [{'start': '10.1.0.10', 'end': '10.1.0.200'}]
ExternalAllocationPools: [{'start': '10.2.0.10', 'end': '10.2.0.200'}]
StorageAllocationPools: [{'start': '10.3.0.10', 'end': '10.3.0.200'}]
StorageMgmtAllocationPools: [{'start': '10.4.0.10', 'end': '10.4.0.200'}]
# Routes
ControlPlaneDefaultRoute: 192.168.24.1
InternalApiDefaultRoute: 10.1.0.1
ExternalInterfaceDefaultRoute: 10.2.0.1
# Vlans
InternalApiNetworkVlanID: 710
ExternalNetworkVlanID: 720
StorageNetworkVlanID: 730
StorageMgmtNetworkVlanID: 740
TenantNetworkVlanID: 3211
# Services
EC2MetadataIp: 192.168.24.1  # Generally the IP of the undercloud
DnsServers: ["172.x.x.x"]
NtpServer: 10.0.0.1
```

Network Interface Configuration (*-NIC-*.yaml)

IN THIS SECTION

NIC configuration files exist per role in the following directory:

```
cd ~/tripleo-heat-templates/network/config/contrail
```

OpenStack Controller

```
heat_template_version: rocky

description: >
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role. This is an example for a Nova compute node using
  Contrail vrouter and the vhost0 interface.
parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal_api network
    type: string
  InternalApiDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the internal api network.
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage_mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
```

```
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 50
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  ExternalInterfaceDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the external network.
    type: string
  ManagementInterfaceDefaultRoute: # Commented out by default in this template
    default: unset
    description: The default route of the management network.
    type: string
```

```
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
              - type: interface
                name: nic1
                use_dhcp: false
                dns_servers:
                  get_param: DnsServers
                addresses:
                - ip_netmask:
                    list_join:
                      - '/'
                      - - get_param: ControlPlaneIp
                        - get_param: ControlPlaneSubnetCidr
                routes:
                - ip_netmask: 169.x.x.x/32
                  next_hop:
                    get_param: EC2MetadataIp
                - default: true
                  next_hop:
                    get_param: ControlPlaneDefaultRoute
              - type: vlan
                vlan_id:
                  get_param: InternalApiNetworkVlanID
                device: nic1
```

```
                addresses:
                - ip_netmask:
                    get_param: InternalApiIpSubnet
              - type: vlan
                vlan_id:
                  get_param: ExternalNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: ExternalIpSubnet
              - type: vlan
                vlan_id:
                  get_param: StorageNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: StorageIpSubnet
              - type: vlan
                vlan_id:
                  get_param: StorageMgmtNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: StorageMgmtIpSubnet
outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl
```

Contrail Controller

```
heat_template_version: rocky
description: >
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role. This is an example for a Nova compute node using
  Contrail vrouter and the vhost0 interface.

parameters:
  ControlPlaneIp:
    default: ''
```

```
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal_api network
    type: string
  InternalApiDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the internal api network.
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage_mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
```

```
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 50
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  ExternalInterfaceDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the external network.
    type: string
  ManagementInterfaceDefaultRoute: # Commented out by default in this template
    default: unset
    description: The default route of the management network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
```

```
          params:
            $network_config:
              network_config:
              - type: interface
                name: nic1
                use_dhcp: false
                dns_servers:
                  get_param: DnsServers
                addresses:
                - ip_netmask:
                    list_join:
                      - '/'
                      - - get_param: ControlPlaneIp
                        - get_param: ControlPlaneSubnetCidr
                routes:
                - ip_netmask: 169.x.x.x/32
                  next_hop:
                    get_param: EC2MetadataIp
                - default: true
                  next_hop:
                    get_param: ControlPlaneDefaultRoute
              - type: vlan
                vlan_id:
                  get_param: InternalApiNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: InternalApiIpSubnet
              - type: interface
                name: nic2
                use_dhcp: false
                addresses:
                - ip_netmask:
                    get_param: TenantIpSubnet
outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl
```

Compute Node

```
heat_template_version: rocky
description: >
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role. This is an example for a Nova compute node using
  Contrail vrouter and the vhost0 interface.
parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal_api network
    type: string
  InternalApiDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the internal api network.
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage_mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  ExternalNetworkVlanID:
    default: 10
```

```
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 50
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  ExternalInterfaceDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the external network.
    type: string
  ManagementInterfaceDefaultRoute: # Commented out by default in this template
    default: unset
    description: The default route of the management network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
```

```
    description: The IP address of the EC2 metadata server.
    type: string
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
              - type: interface
                name: nic1
                use_dhcp: false
                dns_servers:
                  get_param: DnsServers
                addresses:
                - ip_netmask:
                    list_join:
                      - '/'
                      - - get_param: ControlPlaneIp
                        - get_param: ControlPlaneSubnetCidr
                routes:
                - ip_netmask: 169.x.x.x/32
                  next_hop:
                    get_param: EC2MetadataIp
                - default: true
                  next_hop:
                    get_param: ControlPlaneDefaultRoute
              - type: vlan
                vlan_id:
                  get_param: InternalApiNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: InternalApiIpSubnet
              - type: vlan
                vlan_id:
                  get_param: StorageNetworkVlanID
                device: nic1
```

```
                addresses:
                - ip_netmask:
                    get_param: StorageIpSubnet
              - type: contrail_vrouter
                name: vhost0
                use_dhcp: false
                members:
                  -
                    type: interface
                    name: nic2
                    use_dhcp: false
                addresses:
                - ip_netmask:
                    get_param: TenantIpSubnet

outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl
```

Advanced vRouter Kernel Mode Configuration

IN THIS SECTION

In addition to the standard NIC configuration, the vRouter kernel mode supports VLAN, Bond, and Bond + VLAN modes. The configuration snippets below only show the relevant section of the NIC template configuration for each mode.

VLAN

```
- type: vlan
  vlan_id:
```

```
    get_param: TenantNetworkVlanID
  device: nic2
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
        str_replace:
          template: vlanVLANID
          params:
            VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Bond

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
   -
     type: interface
     name: nic2
   -
     type: interface
     name: nic3
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name: bond0
      use_dhcp: false
  addresses:
```

```
  - ip_netmask:
      get_param: TenantIpSubnet
```

Bond + VLAN

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
   -
     type: interface
     name: nic2
   -
     type: interface
     name: nic3
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: bond0
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
        str_replace:
          template: vlanVLANID
          params:
            VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Advanced vRouter DPDK Mode Configuration

IN THIS SECTION

In addition to the standard NIC configuration, the vRouter DPDK mode supports Standard, VLAN, Bond, and Bond + VLAN modes.

Network Environment Configuration:

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages:

```
 # For Intel CPU
  ContrailDpdkParameters:
    KernelArgs: "intel_iommu=on iommu=pt default_hugepagesz=1GB hugepagesz=1G hugepages=4
hugepagesz=2M hugepages=1024"
    ExtraSysctlSettings:
      # must be equal to value from kernel args: hugepages=4
      vm.nr_hugepages:
        value: 4
      vm.max_map_count:
        value: 128960
```

See the following NIC template configurations for vRouter DPDK mode. The configuration snippets below only show the relevant section of the NIC configuration for each mode.

Standard

```
- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
```

```
  driver: uio_pci_generic
  cpu_list: 0x01
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

VLAN

```
 - type: contrail_vrouter_dpdk
              name: vhost0
              use_dhcp: false
              driver: uio_pci_generic
              cpu_list: 0x01
              vlan_id:
                get_param: TenantNetworkVlanID
              members:
                -
                  type: interface
                  name: nic2
                  use_dhcp: false
              addresses:
              - ip_netmask:
                  get_param: TenantIpSubnet
```

Bond

```
- type: contrail_vrouter_dpdk
              name: vhost0
              use_dhcp: false
              driver: uio_pci_generic
              cpu_list: 0x01
              bond_mode: 4
              bond_policy: layer2+3
              members:
                -
```

```
                type: interface
                name: nic2
                use_dhcp: false
              -
                type: interface
                name: nic3
                use_dhcp: false
            addresses:
            - ip_netmask:
                get_param: TenantIpSubnet
```

Bond + VLAN

```
- type: contrail_vrouter_dpdk
            name: vhost0
            use_dhcp: false
            driver: uio_pci_generic
            cpu_list: 0x01
            vlan_id:
              get_param: TenantNetworkVlanID
            bond_mode: 4
            bond_policy: layer2+3
            members:
              -
                type: interface
                name: nic2
                use_dhcp: false
              -
                type: interface
                name: nic3
                use_dhcp: false
            addresses:
            - ip_netmask:
                get_param: TenantIpSubnet
```

Advanced vRouter SRIOV + Kernel Mode Configuration

IN THIS SECTION

vRouter SRIOV + Kernel mode can be used in the following combinations:

- Standard
- VLAN
- Bond
- Bond + VLAN

Network environment configuration:

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages:

```
ContrailSriovParameters:
    KernelArgs: "intel_iommu=on iommu=pt default_hugepagesz=1GB hugepagesz=1G hugepages=4
hugepagesz=2M hugepages=1024"
    ExtraSysctlSettings:
      # must be equal to value from 1G kernel args: hugepages=4
      vm.nr_hugepages:
        value: 4
```

SRIOV PF/VF settings:

```
NovaPCIPassthrough:
- devname: "ens2f1"
  physical_network: "sriov1"
ContrailSriovNumVFs: ["ens2f1:7"]
```

The SRIOV NICs are not configured in the NIC templates. However, vRouter NICs must still be configured. See the following NIC template configurations for vRouter kernel mode. The configuration snippets below only show the relevant section of the NIC configuration for each mode.

VLAN

```
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: nic2
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
        str_replace:
          template: vlanVLANID
          params:
            VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Bond

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
   -
     type: interface
     name: nic2
   -
     type: interface
     name: nic3
- type: contrail_vrouter
```

```
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name: bond0
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Bond + VLAN

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
   -
     type: interface
     name: nic2
   -
     type: interface
     name: nic3
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: bond0
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
        str_replace:
          template: vlanVLANID
          params:
            VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
```

```
  - ip_netmask:
      get_param: TenantIpSubnet
```

Advanced vRouter SRIOV + DPDK Mode Configuration

IN THIS SECTION

vRouter SRIOV + DPDK can be used in the following combinations:

- Standard
- VLAN
- Bond
- Bond + VLAN

Network environment configuration:

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages

```
ContrailSriovParameters:
    KernelArgs: "intel_iommu=on iommu=pt default_hugepagesz=1GB hugepagesz=1G hugepages=4
hugepagesz=2M hugepages=1024"
    ExtraSysctlSettings:
      # must be equal to value from 1G kernel args: hugepages=4
      vm.nr_hugepages:
        value: 4
```

SRIOV PF/VF settings

```
NovaPCIPassthrough:
- devname: "ens2f1"
  physical_network: "sriov1"
ContrailSriovNumVFs: ["ens2f1:7"]
```

The SRIOV NICs are not configured in the NIC templates. However, vRouter NICs must still be configured. See the following NIC template configurations for vRouter DPDK mode. The configuration snippets below only show the relevant section of the NIC configuration for each mode.

Standard

```
- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

VLAN

```
 - type: contrail_vrouter_dpdk
             name: vhost0
             use_dhcp: false
             driver: uio_pci_generic
             cpu_list: 0x01
             vlan_id:
               get_param: TenantNetworkVlanID
             members:
               -
                 type: interface
                 name: nic2
```

```
                use_dhcp: false
            addresses:
            - ip_netmask:
                get_param: TenantIpSubnet
```

Bond

```
- type: contrail_vrouter_dpdk
            name: vhost0
            use_dhcp: false
            driver: uio_pci_generic
            cpu_list: 0x01
            bond_mode: 4
            bond_policy: layer2+3
            members:
              -
                type: interface
                name: nic2
                use_dhcp: false
              -
                type: interface
                name: nic3
                use_dhcp: false
            addresses:
            - ip_netmask:
                get_param: TenantIpSubnet
```

Bond + VLAN

```
 - type: contrail_vrouter_dpdk
            name: vhost0
            use_dhcp: false
            driver: uio_pci_generic
            cpu_list: 0x01
            vlan_id:
              get_param: TenantNetworkVlanID
            bond_mode: 4
            bond_policy: layer2+3
            members:
              -
```

```
                type: interface
                name: nic2
                use_dhcp: false
              -
                type: interface
                name: nic3
                use_dhcp: false
            addresses:
            - ip_netmask:
                get_param: TenantIpSubnet
```

Installing Overcloud

1. Deployment:

```
openstack overcloud deploy --templates tripleo-heat-templates/ \
  --stack overcloud --libvirt-type kvm \
  --roles-file $role_file \
  -e tripleo-heat-templates/environments/rhsm.yaml \
  -e tripleo-heat-templates/environments/network-isolation.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-services.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-net.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
  -e containers-prepare-parameter.yaml \
  -e rhsm.yaml
```

2. Validation Test:

```
source overcloudrc
curl -O http://download.cirros-cloud.net/0.3.5/cirros-0.3.5-x86_64-disk.img
openstack image create --container-format bare --disk-format qcow2 --file cirros-0.3.5-x86_64-
disk.img cirros
openstack flavor create --public cirros --id auto --ram 64 --disk 0 --vcpus 1
openstack network create net1
openstack subnet create --subnet-range 1.0.0.0/24 --network net1 sn1
nova boot --image cirros --flavor cirros --nic net-id=`openstack network show net1 -c id -f
value` --availability-zone nova:overcloud-novacompute-0.localdomain c1
nova list
```

CHAPTER 8

Setting Up Contrail with Red Hat OpenStack 16.2

IN THIS CHAPTER

Understanding Red Hat OpenStack Platform Director 16.2

IN THIS SECTION

Red Hat OpenStack Platform Director

Starting with Contrail Networking Release 21.3, Contrail Networking supports using Contrail with Red Hat OpenStack Platform Director 16.2.

This chapter explains how to integrate a Contrail Networking Release 21.3 (or higher) installation with Red Hat OpenStack Platform Director 16.2.

Red Hat OpenStack Platform provides an installer called the Red Hat OpenStack Platform director (RHOSPd or OSPd), which is a toolset based on the OpenStack project TripleO (OOO, OpenStack on OpenStack). TripleO is an open source project that uses features of OpenStack to deploy a fully functional, tenant-facing OpenStack environment.

TripleO can be used to deploy an RDO-based OpenStack environment integrated with Tungsten Fabric. Red Hat OpenStack Platform director can be used to deploy an RHOSP-based OpenStack environment integrated with Contrail Networking.

OSPd uses the concepts of undercloud and overcloud. OSPd sets up an undercloud, a single server running an operator-facing deployment that contains the OpenStack components needed to deploy and manage an overcloud, a tenant-facing deployment that hosts user workloads.

The overcloud is the deployed solution that can represent a cloud for any purpose, such as production, staging, test, and so on. The operator can select to deploy to their environment any of the available overcloud roles, such as controller, compute, and the like.

OSPd leverages existing core components of OpenStack including Nova, Ironic, Neutron, Heat, Glance, and Ceilometer to deploy OpenStack on bare metal hardware.

- Nova and Ironic are used in the undercloud to manage the bare metal instances that comprise the infrastructure for the overcloud.
- Neutron is used to provide a networking environment in which to deploy the overcloud.
- Glance stores machine images.
- Ceilometer collects metrics about the overcloud.

For more information about OSPd architecture, see OSPd documentation.

Contrail Networking Roles

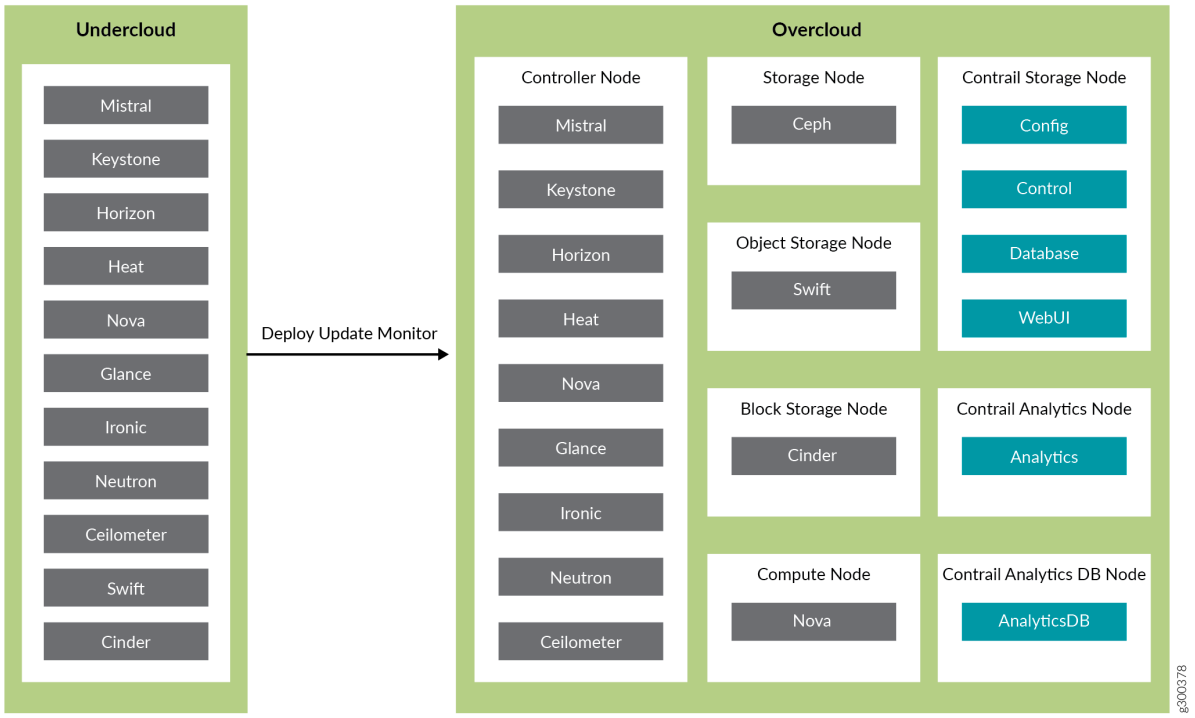
OSPd supports composable roles, which are groups of services that you define through Heat templates. Composable roles allow you to integrate Contrail Networking into the overcloud environment.

The following are the Contrail Networking roles used for integrating into the overcloud:

- Contrail Controller
- Contrail Analytics
- Contrail Analytics Database
- Contrail-TSN
- Contrail-DPDK

Figure 41 on page 375 shows the relationship and components of an undercloud and overcloud architecture for Contrail Networking.

Figure 41: Undercloud and Overcloud with Roles



RVM and KVM Requirements

Starting in Contrail Networking Release 21.4, Contrail Networking was enhanced to operate with hosts using Red Hat Virtualization (RHV). You must use RHV-based hosts in place of KVM-based hosts in RHOSP 16.2 environments starting in Contrail Networking Release 21.4 and in all future Contrail Networking releases.

In Contrail Networking releases prior to Release 21.3, this procedure is performed with hosts using Kernel-based Virtual Machine (KVM).

Undercloud Requirements

The undercloud is a single server or VM that hosts the OpenStack Platform director, which is an OpenStack installation used to provision OpenStack on the overcloud.

See Undercloud Requirements for the compute requirements of the undercloud.

Overcloud Requirements

The overcloud roles can be deployed to bare metal servers or to virtual machines (VMs), but the compute nodes must be deployed to bare metal systems. Every overcloud node must support IPMI for booting up from the undercloud using PXE.

Ensure the following requirements are met for the Contrail Networking nodes per role.

- Non-high availability: A minimum of 4 overcloud nodes are needed for control plane roles for a non-high availability deployment:
  - 1x contrail-config (includes Contrail control)
  - 1x contrail-analytics
  - 1x contrail-analytics-database
  - 1x OpenStack controller
- High availability: A minimum of 12 overcloud nodes are needed for control plane roles for a high availability deployment:
  - 3x contrail-config (includes Contrail control)
  - 3x contrail-analytics
  - 3x contrail-analytics-database
  - 3x OpenStack controller

  If the control plane roles are deployed to VMs, use 3 separate physical servers and deploy one role of each kind to each physical server.

See Overcloud Requirements for the compute requirements of the overcloud.

Networking Requirements

As a minimum, the installation requires two networks:

- provisioning network - This is the private network that the undercloud uses to provision the overcloud.
- external network - This is the externally-routable network you use to access the undercloud and overcloud nodes.

Ensure the following requirements are met for the provisioning network:

- One NIC from every machine must be in the same broadcast domain of the provisioning network, and it should be the same NIC on each of the overcloud machines. For example, if you use the

second NIC on the first overcloud machine, you should use the second NIC on each additional overcloud machine.

During installation, these NICs will be referenced by a single name across all overcloud machines.

- The provisioning network NIC should not be the same NIC that you are using for remote connectivity to the undercloud machine. During the undercloud installation, an Open vSwitch bridge will be created for Neutron, and the provisioning NIC will be bridged to the Open vSwitch bridge. Consequently, connectivity would be lost if the provisioning NIC was also used for remote connectivity to the undercloud machine.
- The provisioning NIC on the overcloud nodes must be untagged.
- You must have the MAC address of the NIC that will PXE boot the IPMI information for the machine on the provisioning network. The IPMI information will include such things as the IP address of the IPMI NIC and the IPMI username and password.
- All of the networks must be available to all of the Contrail Networking roles and computes.

While the provisioning and external networks are sufficient for basic applications, you should create additional networks in most overcloud environments to provide isolation for the different traffic types by assigning network traffic to specific network interfaces or bonds.

When isolated networks are configured, the OpenStack services are configured to use the isolated networks. If no isolated networks are configured, all services run on the provisioning network. If only some isolated networks are configured, traffic belonging to a network not configured runs on the provisioning network.

The following networks are typically deployed when using network isolation topology:

- Provisioning - used by the undercloud to provision the overcloud
- Internal API - used by OpenStack services to communicate with each other
- Tenant - used for tenant overlay data plane traffic (one network per tenant)
- Storage - used for storage data traffic
- Storage Management - used for storage control and management traffic
- External - provides external access to the undercloud and overcloud, including external access to the web UIs and public APIs
- Floating IP - provides floating IP access to the tenant network (can either be merged with external or can be a separate network)
- Management - provides access for system administration

Compatibility Matrix

The following combinations of Operating System/OpenStack/Deployer/Contrail Networking are supported:

Table 27: Compatibility Matrix

Operating System	OpenStack	Deployer	Contrail Networking
RHEL 8.2 or 8.4	OSP16.2	RHOSP16 Director	Contrail Networking 21.3
RHEL 8.4	OSP16.2	RHOSP16 Director	Contrail Networking 21.4
RHEL 8.4	OSP16.2.3	RHOSP16 Director	Contrail Networking 21.4.L1
RHEL 8.4	OSP16.2.4	RHOSP16 Director	Contrail Networking 21.4.L2

Installation Summary

The general installation procedure is as follows:

- Set up the infrastructure, which is the set of servers or VMs that host the undercloud and overcloud, including the provisioning network that connects them together.
- Set up the undercloud, which is the OSPd application.
- Set up the overcloud, which is the set of services in the tenant-facing network. Contrail Networking is part of the overcloud.

For more information on installing and using the RHOSPd, see Red Hat documentation.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
2008	Starting with Contrail Networking Release 21.3, Contrail Networking supports using Contrail with Red Hat OpenStack Platform Director 16.2.

Setting Up the Infrastructure (Contrail Networking Release 21.4 or Later)

SUMMARY

Follow this topic to set up the infrastructure for a Contrail Networking deployment in a RHOSP 16 environment when you are using Contrail Networking Release 21.4 or later.

IN THIS SECTION

When to Use This Procedure

You should use this topic to set up the infrastructure for a Contrail Networking deployment in a RHOSP 16 environment when you are using Contrail Networking Release 21.4 or later.

This procedure shows you how to set up the infrastructure for the installation when the hosts are using Red Hat Virtualization (RHV). Contrail Networking was enhanced to operate with hosts using Red Hat Virtualization (RHV) in Release 21.4.

In Release 21.3 and earlier, this procedure is performed with hosts using Kernel-based Virtual Machine (KVM). See "Setting Up the Infrastructure (Contrail Networking Release 21.3 or Earlier) " on page 324.

Understanding Red Hat Virtualization

This procedure shows an example of how to set up the infrastructure for a Contrail Networking deployment in a RHOSP 16 environment when the hosts are using Red Hat Virtualization (RHV).

RHV is an enterprise virtualization platform built on Red Hat Enterprise Linux and KVM. RHV is developed and fully supported by Red Hat.

The purpose of this topic is to illustrate one method of deploying Contrail Networking in a RHOSP 16 environment using RHOSP 16. The documentation of related RHV components is beyond the scope of this topic.

For additional information on RHV, see Product Documentation for Red Hat Virtualization from Red Hat.

For additional information on RHV installation, see the Installing Red Hat Virtualization as a self-hosted engine using the command line document from Red Hat.

Prepare the Red Hat Virtualization Manager Hosts

Prepare the Red Hat Virtualization Manager hosts using the instructions provided by Red Hat. See the Installing Red Hat Virtualization Hosts section of the Installing Hosts for Red Hat Virtualization chapter of the Installing Red Hat Virtualization as a self-hosted engine using the command line guide from Red Hat.

Deploy Hosts with Red Hat Enterprise Linux

IN THIS SECTION

Red Hat Enterprise Linux (RHEL) must run to enable RHV.

This section provides an example of how to deploy RHEL8.

Install and enable required software

This example shows how to obtain, install, and enable the software required to operate Red Hat Enterprise Linux 8.

```
# Register node with RedHat subscription
# (for satellite check RedHat instruction)
sudo subscription-manager register \
  --username {username} \
  --password {password}

# Attach pools that allow to enable all required repos
# e.g.:
```

```
sudo subscription-manager attach \
  --pool {RHOSP16.2 pool ID} \
  --pool {Red Hat Virtualization Manager pool ID}

# Enable repos
sudo subscription-manager repos \
    --disable='*' \
    --enable=rhel-8-for-x86_64-baseos-rpms \
    --enable=rhel-8-for-x86_64-appstream-rpms \
    --enable=rhv-4-mgmt-agent-for-rhel-8-x86_64-rpms \
    --enable=fast-datapath-for-rhel-8-x86_64-rpms \
    --enable=advanced-virt-for-rhel-8-x86_64-rpms \
    --enable=openstack-16.2-cinderlib-for-rhel-8-x86_64-rpms \
    --enable=rhceph-4-tools-for-rhel-8-x86_64-rpms

# Remove cloud-init (in case if it virt test setup and cloud image used for deploy)
sudo dnf remove -y cloud-init || true

# Enable dnf modules and update system
# For Red Hat Virtualization Manager 4.4 use virt:av
# (for previous versions check RedHat documentation)
sudo dnf module reset -y virt
sudo dnf module enable -y virt:av
sudo dnf distro-sync -y --nobest
sudo dnf upgrade -y --nobest

# Enable firewall
sudo dnf install -y firewalld
sudo systemctl enable --now firewalld

# Check current active zone
sudo firewall-cmd --get-active-zones
# exmaple of zones:
#     public
#       interfaces:  eth0

# Add virbr0 interface into the active zone for ovirtmgmt, e.g.
sudo firewall-cmd --zone=public --change-zone=virbr0 --permanent
sudo firewall-cmd --zone=public --add-forward --permanent
# Ensure used interfaces in one zone
sudo firewall-cmd --get-active-zones
# exmaple of zones:
#     [stack@node-10-0-10-147 ~]$ sudo firewall-cmd --get-active-zones
```

```
#     public
#       interfaces:  eth0 virbr0

# Enable https and cockpit for RHVM web access and monitoring
sudo firewall-cmd --permanent \
  --add-service=https \
  --add-service=cockpit \
  --add-service nfs

sudo firewall-cmd --permanent \
  --add-port 2223/tcp \
  --add-port 5900-6923/tcp \
  --add-port 2223/tcp \
  --add-port 5900-6923/tcp \
  --add-port 111/tcp --add-port 111/udp \
  --add-port 2049/tcp --add-port 2049/udp \
  --add-port 4045/tcp --add-port 4045/udp \
  --add-port 1110/tcp --add-port 1110/udp

# Prepare NFS Storage
# adjust sysctl settings
cat {{ EOF | sudo tee /etc/sysctl.d/99-nfs-tf-rhv.conf
net.ipv4.tcp_mem=4096 65536 4194304
net.ipv4.tcp_rmem=4096 65536 4194304
net.ipv4.tcp_wmem=4096 65536 4194304
net.core.rmem_max=8388608
net.core.wmem_max=8388608
EOF
sudo sysctl --system
# install and enable NFS services
sudo dnf install -y nfs-utils
sudo systemctl enable --now nfs-server
sudo systemctl enable --now rpcbind
# prepare special user required by Red Hat Virtualization
getent group kvm || sudo groupadd kvm -g 36
sudo useradd vdsm -u 36 -g kvm
exports="/storage *(rw,all_squash,anonuid=36,anongid=36)\n"
for s in vmengine undercloud ipa overcloud ; do
  sudo mkdir -p /storage/$s
  exports+="/storage/$s *(rw,all_squash,anonuid=36,anongid=36)\n"
done
sudo chown -R 36:36 /storage
sudo chmod -R 0755 /storage
```

```
# add storage directory to exports
echo -e "$exports" | sudo tee /etc/exports
# restart NFS services
sudo systemctl restart rpcbind
sudo systemctl restart nfs-server
# check exports
sudo exportfs

# Rebbot system In case if newer kernel availalbe in /lib/modules
latest_kv=$(ls -1 /lib/modules | sort -V | tail -n 1)
active_kv=$(uname -r)
if [[ "$latest_kv" != "$active_kv" ]] ; then
  echo "INFO: newer kernel version $latest_kv is available, active one is $active_kv"
  echo "Perform reboot..."
  sudo reboot
fi
```

Confirm the Domain Names

Before proceeding, ensure that the fully qualified domain names (FQDNs) can be resolved by DNS or by the /etc/hosts on all nodes.

```
[stack@node-10-0-10-147 ~]$ cat /etc/hosts
# Red Hat Virtualization Manager VM
10.0.10.200  vmengine.dev.clouddomain          vmengine.dev          vmengine
# Red Hat Virtualization Hosts
10.0.10.147  node-10-0-10-147.dev.clouddomain  node-10-0-10-147.dev  node-10-0-10-147
10.0.10.148  node-10-0-10-148.dev.clouddomain  node-10-0-10-148.dev  node-10-0-10-148
10.0.10.149  node-10-0-10-149.dev.clouddomain  node-10-0-10-149.dev  node-10-0-10-149
10.0.10.150  node-10-0-10-150.dev.clouddomain  node-10-0-10-150.dev  node-10-0-10-150
```

Deploy Red Hat Virtualization Manager on the First Node

IN THIS SECTION

This section shows how to deploy Red Hat Virtual Manager (RHVM).

Enable the Red Hat Virtualization Manager Appliance

To enable the Red Hat Virtualization Manager Appliance:

```
sudo dnf install -y \
  tmux \
  rhvm-appliance \
  ovirt-hosted-engine-setup
```

Deploy the Self-Hosted Engine

To deploy the self-hosted engine:

```
# !!! During deploy you need answer questions
sudo hosted-engine --deploy

# example of adding ansible vars into deploy command
#   sudo hosted-engine --deploy --ansible-extra-vars=he_ipv4_subnet_prefix=10.0.10
# example of an answer:
#   ...
#   Please specify the storage you would like to use (glusterfs, iscsi, fc, nfs)[nfs]:
#   Please specify the nfs version you would like to use (auto, v3, v4, v4_0, v4_1, v4_2)[auto]:
#   Please specify the full shared storage connection path to use (example: host:/path):
10.0.10.147:/storage/vmengine
#   ...
```

NOTE: Ensure all required interfaces are in one zone for IP Forwarding before proceeding with the NFS task during deployment.

```
sudo firewall-cmd --get-active-zones
# exmaple of zones:
```

```
#     [stack@node-10-0-10-147 ~]$ sudo firewall-cmd --get-active-zones
#     public
#       interfaces: ovirtmgmt eth0 virbr0
```

Enable virh CLI to Use oVirt Authentication

To enable virh cli to use oVirt authentication:

```
sudo ln -s /etc/ovirt-hosted-engine/virsh_auth.conf  /etc/libvirt/auth.conf
```

Enabling the Red Hat Virtualization Manager Repositories

To enable the RHVM repositories:

1. Login into RHVM

   ```
   ssh root@vmengine
   ```

2. Associate the Red Hat Virtualization Manager subscription and enable repositories:

   ```
   sudo subscription-manager register --username {username} --password {password}
   # Attach pools that allow to enable all required repos
   # e.g.:
   sudo subscription-manager attach \
     --pool {RHOSP16.2 pool ID} \
     --pool {Red Hat Virtualization Manager pool ID}
   # Enable repos
   sudo subscription-manager repos \
       --disable='*' \
       --enable=rhel-8-for-x86_64-baseos-rpms \
       --enable=rhel-8-for-x86_64-appstream-rpms \
       --enable=rhv-4.4-manager-for-rhel-8-x86_64-rpms \
       --enable=fast-datapath-for-rhel-8-x86_64-rpms \
       --enable=advanced-virt-for-rhel-8-x86_64-rpms \
       --enable=openstack-16.2-cinderlib-for-rhel-8-x86_64-rpms \
       --enable=rhceph-4-tools-for-rhel-8-x86_64-rpms
   # Enable modules and sync
   sudo dnf module -y enable pki-deps
   ```

```
sudo dnf module -y enable postgresql:12
sudo dnf distro-sync -y --nobest
```

Deploy Nodes and Enable Networking

IN THIS SECTION

Follow the tasks in this section to deploy nodes and enable networking:

Prepare the Ansible env Files

To prepare the Ansible env files:

```
# Common variables
# !!! Adjust to your setup - especially undercloud_mgmt_ip and
#     ipa_mgmt_ip to allow SSH to this machines (e.g. choose IPs from ovirtmgmt network)
cat << EOF > common-env.yaml
---
ovirt_hostname: vmengine.dev.clouddomain
ovirt_user: "admin@internal"
ovirt_password: "qwe123QWE"

datacenter_name: Default

# to access hypervisors
ssh_public_key: false
ssh_root_password: "qwe123QWE"

# gateway for VMs (undercloud and ipa)
mgmt_gateway: "10.0.10.1"
# dns to be set in ipa and initial dns for UC
# k8s nodes uses ipa as dns
dns_server:  "10.0.10.1"
```

```
undercloud_name: "undercloud"
undercloud_mgmt_ip: "10.0.10.201"
undercloud_ctlplane_ip: "192.168.24.1"

ipa_name: "ipa"
ipa_mgmt_ip: "10.0.10.205"
ipa_ctlplane_ip: "192.168.24.5"

overcloud_domain: "dev.clouddomain"
EOF

# Hypervisor nodes
# !! Adjust to your setup
# Important: ensure you use correct node name for already registered first hypervisor
# (it is registed at the RHVM deploy command hosted-engine --deploy)
cat << EOF > nodes.yaml
---
nodes:
  # !!! Adjust networks and power management options for your needs
  - name: node-10-0-10-147.dev.clouddomain
    ip: 10.0.10.147
    cluster: Default
    comment: 10.0.10.147
    networks:
      - name: ctlplane
        phy_dev: eth1
      - name: tenant
        phy_dev: eth2
    # provide power management if needed (for all nodes)
    # pm:
    #   address: 192.168.122.1
    #   port: 6230
    #   user: ipmi
    #   password: qwe123QWE
    #   type: ipmilan
    #   options:
    #     ipmilanplus: true
  - name: node-10-0-10-148.dev.clouddomain
    ip: 10.0.10.148
    cluster: node-10-0-10-148
    comment: 10.0.10.148
    networks:
      - name: ctlplane
```

```
        phy_dev: eth1
      - name: tenant
        phy_dev: eth2
  - name: node-10-0-10-149.dev.clouddomain
    ip: 10.0.10.149
    cluster: node-10-0-10-149
    comment: 10.0.10.149
    networks:
      - name: ctlplane
        phy_dev: eth1
      - name: tenant
        phy_dev: eth2
  - name: node-10-0-10-150.dev.clouddomain
    ip: 10.0.10.150
    cluster: node-10-0-10-150
    comment: 10.0.10.150
    networks:
      - name: ctlplane
        phy_dev: eth1
      - name: tenant
        phy_dev: eth2
# !!! Adjust storages according to your setup architecture
storage:
  - name: undercloud
    mountpoint: "/storage/undercloud"
    host: node-10-0-10-147.dev.clouddomain
    address: node-10-0-10-147.dev.clouddomain
  - name: ipa
    mountpoint: "/storage/ipa"
    host: node-10-0-10-147.dev.clouddomain
    address: node-10-0-10-147.dev.clouddomain
  - name: node-10-0-10-148-overcloud
    mountpoint: "/storage/overcloud"
    host: node-10-0-10-148.dev.clouddomain
    address: node-10-0-10-148.dev.clouddomain
  - name: node-10-0-10-149-overcloud
    mountpoint: "/storage/overcloud"
    host: node-10-0-10-149.dev.clouddomain
    address: node-10-0-10-149.dev.clouddomain
  - name: node-10-0-10-150-overcloud
    mountpoint: "/storage/overcloud"
    host: node-10-0-10-150.dev.clouddomain
    address: node-10-0-10-150.dev.clouddomain
```

```
EOF

# Playbook to register hypervisor nodes in RHVM, create storage pools and networks
# Adjust values to your setup!!!
cat << EOF > infra.yaml
- hosts: localhost
  tasks:
  - name: Get RHVM token
    ovirt_auth:
      url: "https://{{ ovirt_hostname }}/ovirt-engine/api"
      username: "{{ ovirt_user }}"
      password: "{{ ovirt_password }}"
      insecure: true
  - name: Create datacenter
    ovirt_datacenter:
      state: present
      auth: "{{ ovirt_auth }}"
      name: "{{ datacenter_name }}"
      local: false
  - name: Create clusters {{ item.name }}
    ovirt_cluster:
      state: present
      auth: "{{ ovirt_auth }}"
      name: "{{ item.cluster }}"
      data_center: "{{ datacenter_name }}"
      ksm: true
      ballooning: true
      memory_policy: server
    with_items:
       - "{{ nodes }}"
  - name: List host in datacenter
    ovirt_host_info:
      auth: "{{ ovirt_auth }}"
      pattern: "datacenter={{ datacenter_name }}"
    register: host_list
  - set_fact:
      hostnames: []
  - name: List hostname
    set_fact:
      hostnames: "{{ hostnames + [ item.name ] }}"
    with_items:
      - "{{ host_list['ovirt_hosts'] }}"
  - name: Register in RHVM
```

```
    ovirt_host:
      state: present
      auth: "{{ ovirt_auth }}"
      name: "{{ item.name }}"
      cluster: "{{ item.cluster }}"
      address: "{{ item.ip }}"
      comment: "{{ item.comment | default(item.ip) }}"
      power_management_enabled: "{{ item.power_management_enabled | default(false) }}"
      # unsupported in rhel yet - to avoid reboot create node via web
      # reboot_after_installation: "{{ item.reboot_after_installation | default(false) }}"
      reboot_after_upgrade: "{{ item.reboot_after_upgrade | default(false) }}"
      public_key: "{{ ssh_public_key }}"
      password: "{{ ssh_root_password }}"
    register: task_result
    until: not task_result.failed
    retries: 5
    delay: 10
    when: item.name not in hostnames
    with_items:
       - "{{ nodes }}"
  - name: Register Power Management for host
    ovirt_host_pm:
      state: present
      auth: "{{ ovirt_auth }}"
      name: "{{ item.name }}"
      address: "{{ item.pm.address }}"
      username: "{{ item.pm.user }}"
      password: "{{ item.pm.password }}"
      type: "{{ item.pm.type }}"
      options: "{{ item.pm.pm_options | default(omit) }}"
    when: item.pm is defined
    with_items:
       - "{{ nodes }}"
  - name: Create storage domains
    ovirt_storage_domain:
      state: present
      auth: "{{ ovirt_auth }}"
      data_center: "{{ datacenter_name }}"
      name: "{{ item.name }}"
      domain_function: "data"
      host: "{{ item.host }}"
      nfs:
        address: "{{ item.address | default(item.host) }}"
```

```
        path: "{{ item.mountpoint }}"
        version: "auto"
    register: task_result
    until: not task_result.failed
    retries: 5
    delay: 10
    with_items:
       - "{{ storage }}"
  - name: Create logical networks
    ovirt_network:
      state: present
      auth: "{{ ovirt_auth }}"
      data_center: "{{ datacenter_name }}"
      name: "{{ datacenter_name }}-{{ item.1.name }}"
      clusters:
      - name: "{{ item.0.cluster }}"
      vlan_tag: "{{ item.1.vlan | default(omit)}}"
      vm_network: true
    with_subelements:
      - "{{ nodes }}"
      - networks
  - name: Create host networks
    ovirt_host_network:
      state: present
      auth: "{{ ovirt_auth }}"
      networks:
      - name: "{{ datacenter_name }}-{{ item.1.name }}"
        boot_protocol: none
      name: "{{ item.0.name }}"
      interface: "{{ item.1.phy_dev }}"
    with_subelements:
      - "{{ nodes }}"
      - networks
  - name: Remove vNICs network_filter
    ovirt.ovirt.ovirt_vnic_profile:
      state: present
      auth: "{{ ovirt_auth }}"
      name: "{{ datacenter_name }}-{{ item.1.name }}"
      network: "{{ datacenter_name }}-{{ item.1.name }}"
      data_center: "{{ datacenter_name }}"
      network_filter: ""
    with_subelements:
      - "{{ nodes }}"
```

```
      - networks
  - name: Revoke SSO Token
    ovirt_auth:
      state: absent
      ovirt_auth: "{{ ovirt_auth }}"
EOF
```

Deploy Nodes and Networking

To deploy the nodes and enable networking:

```
ansible-playbook \
  --extra-vars="@common-env.yaml" \
  --extra-vars="@nodes.yaml" \
  infra.yaml
```

Check Hosts

If a host is in Reboot status, go to the extended menu and select 'Confirm Host has been rebooted"'

Prepare images

To prepare the images:

1. Make a folder for the images:

   ```
   mkdir ~/images
   ```

2. Download the RHEL8.4 base image from RedHat downloads (Red Hat account required). Move the files into the ~/images directory that you created in the previous step.

Create Overcloud VMs

IN THIS SECTION

Follow the instructions in this section to create the overcloud VMs:

Prepare Images for the Kubernetes Cluster

If you are deploying the Contrail Control plane in a Kubernetes cluster, follow this example to prepare the images for the Contrail Controllers:

```
cd
cloud_image=images/rhel-8.4-x86_64-kvm.qcow2
root_password=contrail123
stack_password=contrail123
export LIBGUESTFS_BACKEND=direct
qemu-img create -f qcow2 images/overcloud.qcow2 100G
virt-resize --expand /dev/sda3 ${cloud_image} images/overcloud.qcow2
virt-customize  -a images/overcloud.qcow2 \
  --run-command 'xfs_growfs /' \
  --root-password password:${root_password} \
  --run-command 'useradd stack' \
  --password stack:password:${stack_password} \
  --run-command 'echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack' \
  --chmod 0440:/etc/sudoers.d/stack \
  --run-command 'sed -i "s/PasswordAuthentication no/PasswordAuthentication yes/g" /etc/ssh/
sshd_config' \
  --run-command 'systemctl enable sshd' \
  --selinux-relabel
```

Note that Kubernetes has to be deployed separately on the nodes. This can be done a variety of ways. For information on performing this task using Kubespray, see this Kubespray page on Github.

Contrail Controllers can be deployed using the TF operator on top of Kubernetes. See the TF Operator Github page.

Prepare Overcloud VM Definitions

To prepare the overcloud VM definitions:

```
# Overcloud VMs definitions
# Adjust values to your setup!!!
# For deploying Contrail Control plane in a Kuberentes cluster
# remove contrail controller nodes as they are not managed by RHOSP. They to be created at next
steps.
cat << EOF > vms.yaml
```

```
---
vms:
  - name: controller-0
    disk_size_gb: 100
    memory_gb: 16
    cpu_cores: 4
    nics:
      - name: eth0
        interface: virtio
        profile_name: "{{ datacenter_name }}-ctlplane"
        mac_address: "52:54:00:16:54:d8"
      - name: eth1
        interface: virtio
        profile_name: "{{ datacenter_name }}-tenant"
    cluster: node-10-0-10-148
    storage: node-10-0-10-148-overcloud
  - name: contrail-controller-0
    disk_size_gb: 100
    memory_gb: 16
    cpu_cores: 4
    nics:
      - name: eth0
        interface: virtio
        profile_name: "{{ datacenter_name }}-ctlplane"
        mac_address: "52:54:00:d6:2b:03"
      - name: eth1
        interface: virtio
        profile_name: "{{ datacenter_name }}-tenant"
    cluster: node-10-0-10-148
    storage: node-10-0-10-148-overcloud
  - name: contrail-controller-1
    disk_size_gb: 100
    memory_gb: 16
    cpu_cores: 4
    nics:
      - name: eth0
        interface: virtio
        profile_name: "{{ datacenter_name }}-ctlplane"
        mac_address: "52:54:00:d6:2b:13"
      - name: eth1
        interface: virtio
        profile_name: "{{ datacenter_name }}-tenant"
    cluster: node-10-0-10-149
```

```
    storage: node-10-0-10-149-overcloud
  - name: contrail-controller-2
    disk_size_gb: 100
    memory_gb: 16
    cpu_cores: 4
    nics:
      - name: eth0
        interface: virtio
        profile_name: "{{ datacenter_name }}-ctlplane"
        mac_address: "52:54:00:d6:2b:23"
      - name: eth1
        interface: virtio
        profile_name: "{{ datacenter_name }}-tenant"
    cluster: node-10-0-10-150
    storage: node-10-0-10-150-overcloud
EOF

# Playbook for overcloud VMs
# !!! Adjustto your setup
cat << EOF > overcloud.yaml
- hosts: localhost
  tasks:
  - name: Get RHVM token
    ovirt_auth:
      url: "https://{{ ovirt_hostname }}/ovirt-engine/api"
      username: "{{ ovirt_user }}"
      password: "{{ ovirt_password }}"
      insecure: true
  - name: Create disks
    ovirt_disk:
      auth: "{{ ovirt_auth }}"
      name: "{{ item.name }}"
      interface: virtio
      size: "{{ item.disk_size_gb }}GiB"
      format: cow
      image_path: "{{ item.image | default(omit) }}"
      storage_domain: "{{ item.storage }}"
    register: task_result
    ignore_errors: yes
    until: not task_result.failed
    retries: 5
    delay: 10
    with_items:
```

```
      - "{{ vms }}"
  - name: Deploy VMs
    ovirt.ovirt.ovirt_vm:
      auth: "{{ ovirt_auth }}"
      state: "{{ item.state | default('present') }}"
      cluster: "{{ item.cluster }}"
      name: "{{ item.name }}"
      memory: "{{ item.memory_gb }}GiB"
      cpu_cores: "{{ item.cpu_cores }}"
      type: server
      high_availability: yes
      placement_policy: pinned
      operating_system: rhel_8x64
      disk_format: cow
      graphical_console:
        protocol:
          - spice
          - vnc
      serial_console: yes
      nics: "{{ item.nics | default(omit) }}"
      disks:
        - name: "{{ item.name }}"
          bootable: True
      storage_domain: "{{ item.storage }}"
      cloud_init: "{{ item.cloud_init | default(omit) }}"
      cloud_init_nics: "{{ item.cloud_init_nics | default(omit) }}"
    retries: 5
    delay: 2
    with_items:
      - "{{ vms }}"
  - name: Revoke SSO Token
    ovirt_auth:
      state: absent
      ovirt_auth: "{{ ovirt_auth }}"
EOF

ansible-playbook \
  --extra-vars="@common-env.yaml" \
  --extra-vars="@vms.yaml" \
  overcloud.yaml
```

Create Contrail Control Plane VMs for Kubernetes-based Deployments

IN THIS SECTION

Follow the instructions in this section in side-by-side deployments where the Contrail Control plane is deployed as a separate Kubernetes-based cluster.

Customize VM image for Kubernetes VMs

To customize the VM image for Kubernetes VMs:

```
cd
cloud_image=images/rhel-8.4-x86_64-kvm.qcow2
root_password=contrail123
stack_password=contrail123
export LIBGUESTFS_BACKEND=direct
qemu-img create -f qcow2 images/k8s.qcow2 100G
virt-resize --expand /dev/sda3 ${cloud_image} images/k8s.qcow2
virt-customize  -a images/k8s.qcow2 \
  --run-command 'xfs_growfs /' \
  --root-password password:${root_password} \
  --password stack:password:${stack_password} \
  --run-command 'echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack' \
  --chmod 0440:/etc/sudoers.d/stack \
  --run-command 'sed -i "s/PasswordAuthentication no/PasswordAuthentication yes/g" /etc/ssh/
sshd_config' \
  --run-command 'systemctl enable sshd' \
  --selinux-relabel
```

Define the Kubernetes VMs

To define the Kubernetes VMs:

```
# !!! Adjust to your setup (addresses in ctlplane, tenant and mgmt networks)
cat << EOF > k8s-vms.yaml
---
vms:
  - name: contrail-controller-0
    state: running
    disk_size_gb: 100
    memory_gb: 16
    cpu_cores: 4
    nics:
      - name: eth0
        interface: virtio
        profile_name: "{{ datacenter_name }}-ctlplane"
        mac_address: "52:54:00:16:54:d8"
      - name: eth1
        interface: virtio
        profile_name: "{{ datacenter_name }}-tenant"
      - name: eth2
        interface: virtio
        profile_name: "ovirtmgmt"
    cluster: node-10-0-10-148
    storage: node-10-0-10-148-overcloud
    image: "images/k8s.qcow2"
    cloud_init:
      # ctlplane network
      host_name: "contrail-controller-0.{{ overcloud_domain }}"
      dns_search: "{{ overcloud_domain }}"
      dns_servers: "{{ ipa_ctlplane_ip }}"
      nic_name: "eth0"
      nic_boot_protocol_v6: none
      nic_boot_protocol: static
      nic_ip_address: "192.168.24.7"
      nic_gateway: "{{ undercloud_ctlplane_ip }}"
      nic_netmask: "255.255.255.0"
    cloud_init_nics:
      # tenant network
      - nic_name: "eth1"
        nic_boot_protocol_v6: none
```

```
        nic_boot_protocol: static
        nic_ip_address: "10.0.0.201"
        nic_netmask: "255.255.255.0"
      # mgmt network
      - nic_name: "eth2"
        nic_boot_protocol_v6: none
        nic_boot_protocol: static
        nic_ip_address: "10.0.10.210"
        nic_netmask: "255.255.255.0"
  - name: contrail-controller-1
    state: running
    disk_size_gb: 100
    memory_gb: 16
    cpu_cores: 4
    nics:
      - name: eth0
        interface: virtio
        profile_name: "{{ datacenter_name }}-ctlplane"
        mac_address: "52:54:00:d6:2b:03"
      - name: eth1
        interface: virtio
        profile_name: "{{ datacenter_name }}-tenant"
      - name: eth2
        interface: virtio
        profile_name: "ovirtmgmt"
    cluster: node-10-0-10-149
    storage: node-10-0-10-149-overcloud
    image: "images/k8s.qcow2"
    cloud_init:
      host_name: "contrail-controller-1.{{ overcloud_domain }}"
      dns_search: "{{ overcloud_domain }}"
      dns_servers: "{{ ipa_ctlplane_ip }}"
      nic_name: "eth0"
      nic_boot_protocol_v6: none
      nic_boot_protocol: static
      nic_ip_address: "192.168.24.8"
      nic_gateway: "{{ undercloud_ctlplane_ip }}"
      nic_netmask: "255.255.255.0"
    cloud_init_nics:
      - nic_name: "eth1"
        nic_boot_protocol_v6: none
        nic_boot_protocol: static
        nic_ip_address: "10.0.0.202"
```

```
        nic_netmask: "255.255.255.0"
      # mgmt network
      - nic_name: "eth2"
        nic_boot_protocol_v6: none
        nic_boot_protocol: static
        nic_ip_address: "10.0.10.211"
        nic_netmask: "255.255.255.0"
  - name: contrail-controller-2
    state: running
    disk_size_gb: 100
    memory_gb: 16
    cpu_cores: 4
    nics:
      - name: eth0
        interface: virtio
        profile_name: "{{ datacenter_name }}-ctlplane"
        mac_address: "52:54:00:d6:2b:23"
      - name: eth1
        interface: virtio
        profile_name: "{{ datacenter_name }}-tenant"
      - name: eth2
        interface: virtio
        profile_name: "ovirtmgmt"
    cluster: node-10-0-10-150
    storage: node-10-0-10-150-overcloud
    image: "images/k8s.qcow2"
    cloud_init:
      host_name: "contrail-controller-1.{{ overcloud_domain }}"
      dns_search: "{{ overcloud_domain }}"
      dns_servers: "{{ ipa_ctlplane_ip }}"
      nic_name: "eth0"
      nic_boot_protocol_v6: none
      nic_boot_protocol: static
      nic_ip_address: "192.168.24.9"
      nic_gateway: "{{ undercloud_ctlplane_ip }}"
      nic_netmask: "255.255.255.0"
    cloud_init_nics:
      - nic_name: "eth1"
        nic_boot_protocol_v6: none
        nic_boot_protocol: static
        nic_ip_address: "10.0.0.203"
        nic_netmask: "255.255.255.0"EOF
      # mgmt network
```

```
      - nic_name: "eth2"
        nic_boot_protocol_v6: none
        nic_boot_protocol: static
        nic_ip_address: "10.0.10.212"
        nic_netmask: "255.255.255.0"
EOF

ansible-playbook \
  --extra-vars="@common-env.yaml" \
  --extra-vars="@k8s-vms.yaml" \
  overcloud.yaml
```

Configure VLANs for RHOSP Internal API networks

To SSH to Kubernetes nodes and configure VLANS for RHOSP Internal API Networks:

```
# Example

# ssh to a node
ssh stack@192.168.24.7

# !!!Adjust to your setup and repeate for all Contrail Controller nodes
cat {{EOF | sudo tee /etc/sysconfig/network-scripts/ifcfg-vlan710
ONBOOT=yes
BOOTPROTO=static
HOTPLUG=no
NM_CONTROLLED=no
PEERDNS=no
USERCTL=yes
VLAN=yes
DEVICE=vlan710
PHYSDEV=eth0
IPADDR=10.1.0.7
NETMASK=255.255.255.0
EOF
sudo ifup vlan710

# Do same for external vlan if needed
```

Create Undercloud VM

IN THIS SECTION

Follow the instructions in this section to the create the undercloud VM:

Customize the image for Undercloud VM

To customer the image for the undercloud VM:

```
cd
cloud_image=images/rhel-8.4-x86_64-kvm.qcow2
undercloud_name=undercloud
domain_name=dev.clouddomain
root_password=contrail123
stack_password=contrail123
export LIBGUESTFS_BACKEND=direct
qemu-img create -f qcow2 images/${undercloud_name}.qcow2 100G
virt-resize --expand /dev/sda3 ${cloud_image} images/${undercloud_name}.qcow2
virt-customize  -a images/${undercloud_name}.qcow2 \
  --run-command 'xfs_growfs /' \
  --root-password password:${root_password} \
  --hostname ${undercloud_name}.${domain_name} \
  --run-command 'useradd stack' \
  --password stack:password:${stack_password} \
  --run-command 'echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack' \
  --chmod 0440:/etc/sudoers.d/stack \
  --run-command 'sed -i "s/PasswordAuthentication no/PasswordAuthentication yes/g" /etc/ssh/
sshd_config' \
  --run-command 'systemctl enable sshd' \
  --selinux-relabel
```

Define Undercloud VM

To define the undercloud VM:

```
cat << EOF > undercloud.yaml
- hosts: localhost
  tasks:
  - set_fact:
      cluster: "Default"
      storage: "undercloud"
  - name: get RHVM token
    ovirt_auth:
      url: "https://{{ ovirt_hostname }}/ovirt-engine/api"
      username: "{{ ovirt_user }}"
      password: "{{ ovirt_password }}"
      insecure: true
  - name: create disks
    ovirt_disk:
      auth: "{{ ovirt_auth }}"
      name: "{{ undercloud_name }}"
      interface: virtio
      format: cow
      size: 100GiB
      image_path: "images/{{ undercloud_name }}.qcow2"
      storage_domain: "{{ storage }}"
    register: task_result
    ignore_errors: yes
    until: not task_result.failed
    retries: 5
    delay: 10
  - name: deploy vms
    ovirt.ovirt.ovirt_vm:
      auth: "{{ ovirt_auth }}"
      state: running
      cluster: "{{ cluster }}"
      name: "{{ undercloud_name }}"
      memory: 32GiB
      cpu_cores: 8
      type: server
      high_availability: yes
      placement_policy: pinned
      operating_system: rhel_8x64
```

```
      cloud_init:
        host_name: "{{ undercloud_name }}.{{ overcloud_domain }}"
        dns_search: "{{ overcloud_domain }}"
        dns_servers: "{{ dns_server | default(mgmt_gateway) }}"
        nic_name: "eth0"
        nic_boot_protocol_v6: none
        nic_boot_protocol: static
        nic_ip_address: "{{ undercloud_mgmt_ip }}"
        nic_gateway: "{{ mgmt_gateway }}"
        nic_netmask: "255.255.255.0"
      cloud_init_nics:
        - nic_name: "eth1"
          nic_boot_protocol_v6: none
          nic_boot_protocol: static
          nic_ip_address: "{{ undercloud_ctlplane_ip }}"
          nic_netmask: "255.255.255.0"
      disk_format: cow
      graphical_console:
        protocol:
          - spice
          - vnc
      serial_console: yes
      nics:
       - name: eth0
         interface: virtio
         profile_name: "ovirtmgmt"
       - name: eth1
         interface: virtio
         profile_name: "{{ datacenter_name }}-ctlplane"
      disks:
        - name: "{{ undercloud_name }}"
          bootable: true
      storage_domain: "{{ storage }}"
  - name: revoke SSO token
    ovirt_auth:
      state: absent
      ovirt_auth: "{{ ovirt_auth }}"
EOF

ansible-playbook --extra-vars="@common-env.yaml" undercloud.yaml
```

Create FreeIPA VM

IN THIS SECTION

To create the FreeIPA VM:

Customize VM image for RedHat IDM (FreeIPA) VM

Follow this example to customer the VM image for the RedHat IDM image.

This example is setup for a TLS everywhere deployment.

```
cd
cloud_image=images/rhel-8.4-x86_64-kvm.qcow2
ipa_name=ipa
domain_name=dev.clouddomain
qemu-img create -f qcow2 images/${ipa_name}.qcow2 100G
virt-resize --expand /dev/sda3 ${cloud_image} images/${ipa_name}.qcow2
virt-customize  -a images/${ipa_name}.qcow2 \
  --run-command 'xfs_growfs /' \
  --root-password password:${root_password} \
  --hostname ${ipa_name}.${domain_name} \
  --run-command 'sed -i "s/PasswordAuthentication no/PasswordAuthentication yes/g" /etc/ssh/
sshd_config' \
  --run-command 'systemctl enable sshd' \
  --selinux-relabel
```

Enable the RedHat IDM (FreeIPA) VM

To enable the RedHat IDM VM:

```
cat << EOF > ipa.yaml
- hosts: localhost
  tasks:
  - set_fact:
      cluster: "Default"
      storage: "ipa"
  - name: get RHVM token
    ovirt_auth:
      url: "https://{{ ovirt_hostname }}/ovirt-engine/api"
      username: "{{ ovirt_user }}"
      password: "{{ ovirt_password }}"
      insecure: true
  - name: create disks
    ovirt_disk:
      auth: "{{ ovirt_auth }}"
      name: "{{ ipa_name }}"
      interface: virtio
      format: cow
      size: 100GiB
      image_path: "images/{{ ipa_name }}.qcow2"
      storage_domain: "{{ storage }}"
    register: task_result
    ignore_errors: yes
    until: not task_result.failed
    retries: 5
    delay: 10
  - name: deploy vms
    ovirt.ovirt.ovirt_vm:
      auth: "{{ ovirt_auth }}"
      state: running
      cluster: "{{ cluster }}"
      name: "{{ ipa_name }}"
      memory: 4GiB
      cpu_cores: 2
      type: server
      high_availability: yes
      placement_policy: pinned
      operating_system: rhel_8x64
```

```
      cloud_init:
        host_name: "{{ ipa_name }}.{{ overcloud_domain }}"
        dns_search: "{{ overcloud_domain }}"
        dns_servers: "{{ dns_server | default(mgmt_gateway) }}"
        nic_name: "eth0"
        nic_boot_protocol_v6: none
        nic_boot_protocol: static
        nic_ip_address: "{{ ipa_mgmt_ip }}"
        nic_gateway: "{{ mgmt_gateway }}"
        nic_netmask: "255.255.255.0"
      cloud_init_nics:
        - nic_name: "eth1"
          nic_boot_protocol_v6: none
          nic_boot_protocol: static
          nic_ip_address: "{{ ipa_ctlplane_ip }}"
          nic_netmask: "255.255.255.0"
      disk_format: cow
      graphical_console:
        protocol:
          - spice
          - vnc
      serial_console: yes
      nics:
       - name: eth0
         interface: virtio
         profile_name: "ovirtmgmt"
       - name: eth1
         interface: virtio
         profile_name: "{{ datacenter_name }}-ctlplane"
      disks:
        - name: "{{ ipa_name }}"
          bootable: true
      storage_domain: "{{ storage }}"
  - name: revoke SSO token
    ovirt_auth:
      state: absent
      ovirt_auth: "{{ ovirt_auth }}"
EOF

ansible-playbook --extra-vars="@common-env.yaml" ipa.yaml
```

Access to RHVM via a web browser

RHVM can be accessed only using the engine FQDN or one of the engine alternate FQDNs. For example, https://vmengine.dev.clouddomain. Please ensure that the FQDN can be resolved.

Access to VMs via serial console

To access the VMs via serial console, see the RedHat documentation or the oVirt documentation.

Setting Up the Undercloud for RHOSP 16.2

SUMMARY

Follow this topic to setup the undercloud for a Contrail Networking deployment with RHOSP 16.2.

IN THIS SECTION

Follow this topic to setup the undercloud for a Contrail Networking deployment with RHOSP 16.2.

Contrail Networking was enhanced to operate with hosts using Red Hat Virtualization (RHV) in Contrail Networking Release 21.4.L2 or later. Prior to this enhancement, Contrail Networking was supported in environments with hosts using Kernel-based Virtual Machine (KVM) only.

These instructions apply to both environments.

Install the Undercloud

Use this example procedure to install the undercloud.

1. Log in to the undercloud VM from the undercloud KVM host.

```
ssh ${undercloud_ip}
```

2. Configure the hostname.

```
undercloud_name=`hostname -s`
undercloud_suffix=`hostname -d`
```

```
hostnamectl set-hostname ${undercloud_name}.${undercloud_suffix}
hostnamectl set-hostname --transient ${undercloud_name}.${undercloud_suffix}
```

3. Add the hostname to the **/etc/hosts** file. The following example assumes the management interface is eth0.

```
undercloud_ip=`ip addr sh dev eth0 | grep "inet " | awk '{print $2}' | awk -F"/" '{print $1}'`
echo ${undercloud_ip} ${undercloud_name}.${undercloud_suffix} ${undercloud_name} >> /etc/hosts
```

4. Set up the repositories.

   RHEL

```
#Register with Satellite (can be done with CDN as well)
satellite_fqdn=device.example.net
act_key=xxx
org=example
yum localinstall -y http://${satellite_fqdn}/pub/katello-ca-consumer-latest.noarch.rpm
subscription-manager register --activationkey=${act_key} --org=${org}
```

5. Install the Tripleo client.

```
yum install -y python-tripleoclient tmux
```

6. Copy the undercloud configuration file sample and modify the configuration as required. See Red Hat documentation for information on how to modify that file.

```
su - stack
cp /usr/share/python-tripleoclient/undercloud.conf.sample ~/undercloud.conf
vi ~/undercloud.conf
```

7. Install the undercloud.

```
openstack undercloud install
source stackrc
```

8. If you are using a satellite for deployment, manually update the hostname and satellite IP addresses in your /etc/hosts/ file.

To perform this procedure using the VI editor:

```
(undercloud) [stack@osp16-5c5s36 ~]$ sudo vi /etc/hosts
```

and manually enter your hostname and satellite IP address in the file while using the editor.

This step ensures that the overcloud deployment is successful later in the procedure.

You should also perform this step if the overcloud deployment fails later in the procedure and a failed lookup URL message appears on the console as the reason.

A sample failed lookup URL error message when you experience this issue:.

```
=======================
TASK [redhat-subscription : SATELLITE | Run Satellite 6 tasks] *****************
Tuesday 30 March 2021  12:11:25 -0400 (0:00:00.490)       0:13:39.737 *********
included: /usr/share/ansible/roles/redhat-subscription/tasks/satellite-6.yml for overcloud-
controller-0, overcloud-controller-1, overcloud-controller-2
TASK [redhat-subscription : SATELLITE 6 | Set Satellite server CA as a fact] ***Tuesday 30
March 2021  12:11:26 -0400 (0:00:00.730)       0:13:40.467 *********
fatal: [overcloud-controller-0]: FAILED! => {"msg": "An unhandled exception occurred while
running the lookup plugin 'url'. Error was a <class 'ansible.errors.AnsibleError'>, original
message: Failed lookup url for  : <urlopen error [Errno -2] Name or service not
known>"}fatal: [overcloud-controller-1]: FAILED! => {"msg": "An unhandled exception occurred
while running the lookup plugin 'url'. Error was a <class 'ansible.errors.AnsibleError'>,
original message: Failed lookup url for  : <urlopen error [Errno -2] Name or service not
known>"}

fatal: [overcloud-controller-2]: FAILED! => {"msg": "An unhandled exception occurred while
running the lookup plugin 'url'. Error was a <class 'ansible.errors.AnsibleError'>, original
message: Failed lookup url for  : <urlopen error [Errno -2] Name or service not known>"}
```

Perform Post-Install Configuration

After installing the undercloud,

1. Configure a forwarding path between the provisioning network and the external network:

```
sudo iptables -A FORWARD -i br-ctlplane -o eth0 -j ACCEPT
sudo iptables -A FORWARD -i eth0 -o br-ctlplane -m state --state RELATED,ESTABLISHED -j
```

```
ACCEPT
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

2. Add the external API interface:

```
sudo ip link add name vlan720 link br-ctlplane type vlan id 720
sudo ip addr add 10.2.0.254/24 dev vlan720
sudo ip link set dev vlan720 up
```

3. Add the `stack` user to the docker group:

```
newgrp docker
exit
su - stack
source stackrc
```

4. Manually add the satellite IP address and hostname into the **/etc/hosts/** file.

Setting Up the Overcloud for RHOSP 16.2

SUMMARY

Follow this topic to setup the overcloud for a Contrail Networking deployment with RHOSP 16.2.

IN THIS SECTION

Follow this topic to setup the overcloud for a Contrail Networking deployment with RHOSP 16.2.

Contrail Networking was enhanced to operate with hosts using Red Hat Virtualization (RHV) in Contrail Networking Release 21.4.L2 or later. Prior to this enhancement, Contrail Networking was supported in environments with hosts using Kernel-based Virtual Machine (KVM) only.

These instructions apply to both environments unless otherwise noted. In cases where the running virtualization engine impacts this procedure, the steps to perform in environments using RHV or KVM are noted.

Configuring the Overcloud

Use this example procedure on the undercloud to set up the configuration for the overcloud.

1. Specify the name server to be used:

```
undercloud_nameserver=8.8.8.8
openstack subnet set `openstack subnet show ctlplane-subnet -c id -f value` --dns-nameserver $
{undercloud_nameserver}
```

2. Retrieve and upload the overcloud images.

   a. Create the image directory:

```
mkdir images
cd images
```

   b. Retrieve the overcloud images from either the RDO project or from RHOSO 16.2.

```
sudo yum install -y rhosp-director-images rhosp-director-images-ipa
for i in /usr/share/rhosp-director-images/overcloud-full-latest-16.0.tar /usr/share/rhosp-
director-images/ironic-python-agent-latest-16.0.tar ; do tar -xvf $i; done
```

   c. Upload the overcloud images:

```
cd
openstack overcloud image upload --image-path /home/stack/images/
```

3. Prepare OpenStack's bare metal provisioning (Ironic).

   The Ironic driver installation depends on the virtualization engine running for Red Hat Openstack:

   - Red Hat Virtualization (RHV, Contrail Networking Release 21.4.L2 and later): Use staging-ovirt to download the Ironic driver.

     See the Creating virtualized control planes document from Red Hat to enable the control plane with the staging-ovirt driver.

- Kernel-based Virtual Machine (KVM, Contrail Networking Release 21.3 and earlier releases that support RHOSP16): Use the IPMI driver to download the Ironic driver.

  The IPMI driver download procedure is provided in these steps.

NOTE: Make sure to combine the **ironic_list** files from the three overcloud KVM hosts.

a. Add the overcloud VMs to Ironic:

```
ipmi_password=<password>
ipmi_user=<user>
while IFS= read -r line; do
  mac=`echo $line|awk '{print $1}'`
  name=`echo $line|awk '{print $2}'`
  kvm_ip=`echo $line|awk '{print $3}'`
  profile=`echo $line|awk '{print $4}'`
  ipmi_port=`echo $line|awk '{print $5}'`
  uuid=`openstack baremetal node create --driver ipmi \
                                        --property cpus=4 \
                                        --property memory_mb=16348 \
                                        --property local_gb=100 \
                                        --property cpu_arch=x86_64 \
                                        --driver-info ipmi_username=${ipmi_user}  \
                                        --driver-info ipmi_address=${kvm_ip} \
                                        --driver-info ipmi_password=${ipmi_password} \
                                        --driver-info ipmi_port=${ipmi_port} \
                                        --name=${name} \
                                        --property capabilities=profile:$
{profile},boot_option:local \
                                        -c uuid -f value`
  openstack baremetal port create --node ${uuid} ${mac}
done < <(cat ironic_list)

DEPLOY_KERNEL=$(openstack image show bm-deploy-kernel -f value -c id)
DEPLOY_RAMDISK=$(openstack image show bm-deploy-ramdisk -f value -c id)

for i in `openstack baremetal node list -c UUID -f value`; do
  openstack baremetal node set $i --driver-info deploy_kernel=$DEPLOY_KERNEL --driver-info
deploy_ramdisk=$DEPLOY_RAMDISK
done
```

```
for i in `openstack baremetal node list -c UUID -f value`; do
  openstack baremetal node show $i -c properties -f value
done
```

b. Introspect the overcloud node:

```
for node in $(openstack baremetal node list -c UUID -f value) ; do
  openstack baremetal node manage $node
done
openstack overcloud node introspect --all-manageable --provide
```

4. Create Flavor:

```
for i in compute-dpdk \
compute-sriov \
contrail-controller \
contrail-analytics \
contrail-database \
contrail-analytics-database; do
  openstack flavor create $i --ram 4096 --vcpus 1 --disk 40
  openstack flavor set --property "capabilities:boot_option"="local" \
                       --property "capabilities:profile"="${i}" ${i}
  openstack flavor set --property resources:CUSTOM_BAREMETAL=1 --property
resources:DISK_GB='0'
                       --property resources:MEMORY_MB='0'
                       --property resources:VCPU='0' ${i}
done
```

5. Copy the TripleO heat templates.

```
cp -r /usr/share/openstack-tripleo-heat-templates/ tripleo-heat-templates
```

6. Download and copy the Contrail heat templates from https://support.juniper.net/support/downloads.

```
tar -xzvf contrail-tripleo-heat-templates-<version>.tgz
cp -r contrail-tripleo-heat-templates/* tripleo-heat-templates/
```

7. Create **rhsm.yaml** file with your RedHat credentials

```
parameter_defaults:
  RhsmVars:
    rhsm_repos:
      - fast-datapath-for-rhel-8-x86_64-rpms
      - openstack-16.1-for-rhel-8-x86_64-rpms
      - satellite-tools-6.5-for-rhel-8-x86_64-rpms
      - ansible-2-for-rhel-8-x86_64-rpms
      - rhel-8-for-x86_64-highavailability-rpms
      - rhel-8-for-x86_64-appstream-rpms
      - rhel-8-for-x86_64-baseos-rpms
    rhsm_username: "YOUR_REDHAT_LOGIN"
    rhsm_password: "YOUR_REDHAT_PASSWORD"
    rhsm_org_id: "YOUR_REDHAT_ID"
    rhsm_pool_ids: "YOUR_REDHAT_POOL_ID"
```

8. Create and upload the OpenStack containers.

   a. Create the OpenStack container file.

   NOTE: The container must be created based on the OpenStack program.

   OSP16.2

```
sudo openstack tripleo container image prepare \
  -e ~/containers-prepare-parameter.yaml
  -e ~/rhsm.yaml > ~/overcloud_containers.yaml

sudo openstack overcloud container image upload --config-file ~/overcloud_containers.yaml
```

   b. Upload the OpenStack containers:

```
openstack overcloud container image upload --config-file ~/local_registry_images.yaml
```

9. Create and upload the Contrail containers.

   a. Create the Contrail container file.

NOTE: This step is optional. The Contrail containers can be downloaded from external registries later.

```
cd ~/tf-heat-templates/tools/contrail
./import_contrail_container.sh -f container_outputfile -r registry -t tag [-i insecure] [-
u username] [-p password] [-c certificate path]
```

Here are few examples of importing Contrail containers from different sources:

- Import from password protected public registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r hub.juniper.net/contrail -
u USERNAME -p PASSWORD -t 1234
```

- Import from Dockerhub:

```
./import_contrail_container.sh -f /tmp/contrail_container -r docker.io/
opencontrailnightly -t 1234
```

- Import from private secure registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r device.example.net:5443 -
c http://device.example.net/pub/device.example.net.crt -t 1234
```

- Import from private insecure registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r 10.0.0.1:5443 -i 1 -t 1234
```

b. Upload Contrail containers to the undercloud registry:

```
openstack overcloud container image upload --config-file /tmp/contrail_container
```

Customizing the Contrail Service with Templates (contrail-services.yaml)

This section contains information to customize Contrail services for your network by modifying the **contrail-services.yaml** file.

- APPLY_DEFAULTS Settings customization - When Contrail is deployed for the first time, the default value of APPLY_DEFAULTS parameter in the ContrailDefaults section needs to be set to ’True’. This enables provisioning parameters present inside the template to use day0 configuration whenever a config provisioning container is restarted. Thus, the provisioning parameters are template driven and any changes to Contrail settings should be done through TripleO templates.

  Contrail Networking allows you to configure some global configuration parameters like VXLAN network id mode, linklocal configuration, IBGP auto mesh configuration, enabling 4byte_AS, and changing BGP Global ASN through its web user interface. If you want to manage your cluster through web user interface, then you need to set APPLY_DEFAULTS=False in ContrailDefaults section and deploy your cluster again by running openstack overcloud deploy. This additional step is required because when you have changed Contrail global configuration parameters through web user interface, then there is a possibility for these global configuration parameters to be overwritten if any config provisioner container is restarted. In order to avoid these values to be overwritten, set APPLY_DEFAULTS as ’False’ and deploy Contrail again by running openstack overcloud deploy command. As a result, the global configuration parameters remain unchanged as provisioning is not executed again.

  For example, if you set APPLY_DEFAULTS=False through TripleO template, deploy your Contrail cluster, set VxLAN Identifier Mode to ’User Configured’ from web user interface, and restart config provisioner container, then VxLAN Identifier Mode will remain ’User Configured’ after the restart of config provisioner container. On the contrary, if APPLY_DEFAULTS is set to True, then after the restart of config provisioner container, VxLAN Identifier Mode will change to its default value, which is Automatic.

  For example, if you set APPLY_DEFAULTS=False through TripleO template, deploy your Contrail cluster, set VxLAN Identifier Mode to ’User Configured’ from web user interface, and restart config provisioner container, then VxLAN Identifier Mode will remain ’User Configured’ after the restart of config provisioner container. On the contrary, if APPLY_DEFAULTS is set to True, then after the restart of config provisioner container, VxLAN Identifier Mode will change to its default value, which is Automatic.

  ```
  APPLY_DEFAULTS=True/False (default: True)
  ```

- Contrail Services customization

  ```
  vi ~/tripleo-heat-templates/environments/contrail-services.yaml
  parameter_defaults:
    ContrailSettings:
      VROUTER_GATEWAY: 10.0.0.1
      # KEY1: value1
  ```

```
    # KEY2: value2

    VXLAN_VN_ID_MODE: "configured"
    ENCAP_PRIORITY: "VXLAN,MPLSoUDP,MPLSoGRE"

  ContrailControllerParameters:
    AAAMode: rbac
```

- Contrail registry settings

```
vi ~/tripleo-heat-templates/environments/contrail-services.yaml
```

Here are few examples of default values for various registries:

- Public Juniper registry

```
parameter_defaults:
  ContrailRegistry: hub.juniper.net/contrail
  ContrailRegistryUser: <USER>
  ContrailRegistryPassword: <PASSWORD>
```

- Insecure registry

```
parameter_defaults:
  ContrailRegistryInsecure: true
  DockerInsecureRegistryAddress: 10.87.64.32:5000,192.168.24.1:8787
  ContrailRegistry: 10.87.64.32:5000
```

- Private secure registry

```
parameter_defaults:
  ContrailRegistryCertUrl: http://device.example.net/pub/device.example.net.crt
  ContrailRegistry: device.example.net:5443
```

- Contrail Container image settings

```
parameter_defaults:
  ContrailImageTag: queens-5.0-104-rhel-queens
```

Customizing the Contrail Network with Templates

IN THIS SECTION

Overview

In order to customize the network, define different networks and configure the overcloud nodes NIC layout. TripleO supports a flexible way of customizing the network.

The following networking customization example uses network as:

Table 28: Network Customization

Network	VLAN	overcloud Nodes
provisioning	-	All
internal_api	710	All
external_api	720	OpenStack CTRL
storage	740	OpenStack CTRL, Computes
storage_mgmt	750	OpenStack CTRL
tenant	-	Contrail CTRL, Computes

Roles Configuration (roles_data_contrail_aio.yaml)

IN THIS SECTION

The networks must be activated per role in the roles_data file:

```
vi ~/tripleo-heat-templates/roles_data_contrail_aio.yaml
```

OpenStack Controller

```
###############################################################################
# Role: Controller                                                            #
###############################################################################
- name: Controller
  description: |
    Controller role that has all the controler services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  tags:
    - primary
    - controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
```

Compute Node

```
###############################################################################
# Role: Compute                                                               #
###############################################################################
- name: Compute
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi
    - Tenant
    - Storage
```

Contrail Controller

```
###############################################################################
# Role: ContrailController                                                    #
###############################################################################
- name: ContrailController
  description: |
    ContrailController role that has all the Contrail controler services loaded
    and handles config, control and webui functions
  CountDefault: 1
  tags:
    - primary
    - contrailcontroller
  networks:
    - InternalApi
    - Tenant
```

Compute DPDK

```
###############################################################################
# Role: ContrailDpdk                                                          #
###############################################################################
- name: ContrailDpdk
  description: |
    Contrail Dpdk Node role
```

```
  CountDefault: 0
  tags:
    - contraildpdk
  networks:
    - InternalApi
    - Tenant
    - Storage
```

Compute SRIOV

```
###############################################################################
# Role: ContrailSriov
###############################################################################
- name: ContrailSriov
  description: |
    Contrail Sriov Node role
  CountDefault: 0
  tags:
    - contrailsriov
  networks:
    - InternalApi
    - Tenant
    - Storage
```

Compute CSN

```
###############################################################################
# Role: ContrailTsn
###############################################################################
- name: ContrailTsn
  description: |
    Contrail Tsn Node role
  CountDefault: 0
  tags:
    - contrailtsn
  networks:
    - InternalApi
    - Tenant
    - Storage
```

Network Parameter Configuration (contrail-net.yaml)

```
cat ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml
resource_registry:
  OS::TripleO::Controller::Net::SoftwareConfig: ../../network/config/contrail/controller-nic-
config.yaml
  OS::TripleO::ContrailController::Net::SoftwareConfig: ../../network/config/contrail/contrail-
controller-nic-config.yaml
  OS::TripleO::ContrailControlOnly::Net::SoftwareConfig: ../../network/config/contrail/contrail-
controller-nic-config.yaml
  OS::TripleO::Compute::Net::SoftwareConfig: ../../network/config/contrail/compute-nic-
config.yaml
  OS::TripleO::ContrailDpdk::Net::SoftwareConfig: ../../network/config/contrail/contrail-dpdk-
nic-config.yaml
  OS::TripleO::ContrailSriov::Net::SoftwareConfig: ../../network/config/contrail/contrail-sriov-
nic-config.yaml
  OS::TripleO::ContrailTsn::Net::SoftwareConfig: ../../network/config/contrail/contrail-tsn-nic-
config.yaml

parameter_defaults:
  # Customize all these values to match the local environment
  TenantNetCidr: 10.0.0.0/24
  InternalApiNetCidr: 10.1.0.0/24
  ExternalNetCidr: 10.2.0.0/24
  StorageNetCidr: 10.3.0.0/24
  StorageMgmtNetCidr: 10.4.0.0/24
  # CIDR subnet mask length for provisioning network
  ControlPlaneSubnetCidr: '24'
  # Allocation pools
  TenantAllocationPools: [{'start': '10.0.0.10', 'end': '10.0.0.200'}]
  InternalApiAllocationPools: [{'start': '10.1.0.10', 'end': '10.1.0.200'}]
  ExternalAllocationPools: [{'start': '10.2.0.10', 'end': '10.2.0.200'}]
  StorageAllocationPools: [{'start': '10.3.0.10', 'end': '10.3.0.200'}]
  StorageMgmtAllocationPools: [{'start': '10.4.0.10', 'end': '10.4.0.200'}]
  # Routes
  ControlPlaneDefaultRoute: 192.168.24.1
  InternalApiDefaultRoute: 10.1.0.1
  ExternalInterfaceDefaultRoute: 10.2.0.1
  # Vlans
  InternalApiNetworkVlanID: 710
  ExternalNetworkVlanID: 720
  StorageNetworkVlanID: 730
```

```
  StorageMgmtNetworkVlanID: 740
  TenantNetworkVlanID: 3211
  # Services
  EC2MetadataIp: 192.168.24.1  # Generally the IP of the undercloud
  DnsServers: ["172.x.x.x"]
  NtpServer: 10.0.0.1
```

Network Interface Configuration (*-NIC-*.yaml)

IN THIS SECTION

NIC configuration files exist per role in the following directory:

```
cd ~/tripleo-heat-templates/network/config/contrail
```

OpenStack Controller

```
heat_template_version: rocky

description: >
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role. This is an example for a Nova compute node using
  Contrail vrouter and the vhost0 interface.
parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
```

```
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal_api network
    type: string
  InternalApiDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the internal api network.
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage_mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 50
    description: Vlan ID for the tenant network traffic.
```

```
    type: number
  ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  ExternalInterfaceDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the external network.
    type: string
  ManagementInterfaceDefaultRoute: # Commented out by default in this template
    default: unset
    description: The default route of the management network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
              - type: interface
                name: nic1
```

```
    use_dhcp: false
    dns_servers:
      get_param: DnsServers
    addresses:
    - ip_netmask:
        list_join:
          - '/'
          - - get_param: ControlPlaneIp
            - get_param: ControlPlaneSubnetCidr
    routes:
    - ip_netmask: 169.x.x.x/32
      next_hop:
        get_param: EC2MetadataIp
    - default: true
      next_hop:
        get_param: ControlPlaneDefaultRoute
  - type: vlan
    vlan_id:
      get_param: InternalApiNetworkVlanID
    device: nic1
    addresses:
    - ip_netmask:
        get_param: InternalApiIpSubnet
  - type: vlan
    vlan_id:
      get_param: ExternalNetworkVlanID
    device: nic1
    addresses:
    - ip_netmask:
        get_param: ExternalIpSubnet
  - type: vlan
    vlan_id:
      get_param: StorageNetworkVlanID
    device: nic1
    addresses:
    - ip_netmask:
        get_param: StorageIpSubnet
  - type: vlan
    vlan_id:
      get_param: StorageMgmtNetworkVlanID
    device: nic1
    addresses:
    - ip_netmask:
```

```
                    get_param: StorageMgmtIpSubnet
outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl
```

Contrail Controller

```
heat_template_version: rocky
description: >
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role. This is an example for a Nova compute node using
  Contrail vrouter and the vhost0 interface.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal_api network
    type: string
  InternalApiDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the internal api network.
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage_mgmt network
    type: string
  TenantIpSubnet:
```

```
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 50
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  ExternalInterfaceDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the external network.
    type: string
  ManagementInterfaceDefaultRoute: # Commented out by default in this template
```

```
    default: unset
    description: The default route of the management network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
              - type: interface
                name: nic1
                use_dhcp: false
                dns_servers:
                  get_param: DnsServers
                addresses:
                - ip_netmask:
                    list_join:
                      - '/'
                      - - get_param: ControlPlaneIp
                        - get_param: ControlPlaneSubnetCidr
                routes:
                - ip_netmask: 169.x.x.x/32
                  next_hop:
                    get_param: EC2MetadataIp
                - default: true
                  next_hop:
                    get_param: ControlPlaneDefaultRoute
              - type: vlan
                vlan_id:
```

```
                  get_param: InternalApiNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: InternalApiIpSubnet
              - type: interface
                name: nic2
                use_dhcp: false
                addresses:
                - ip_netmask:
                    get_param: TenantIpSubnet
outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl
```

Compute Node

```
heat_template_version: rocky
description: >
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role. This is an example for a Nova compute node using
  Contrail vrouter and the vhost0 interface.
parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal_api network
    type: string
  InternalApiDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the internal api network.
    type: string
```

```
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage_mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 50
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
```

```
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  ExternalInterfaceDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the external network.
    type: string
  ManagementInterfaceDefaultRoute: # Commented out by default in this template
    default: unset
    description: The default route of the management network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
              - type: interface
                name: nic1
                use_dhcp: false
                dns_servers:
                  get_param: DnsServers
                addresses:
                - ip_netmask:
                    list_join:
                      - '/'
                      - - get_param: ControlPlaneIp
                        - get_param: ControlPlaneSubnetCidr
```

```
                routes:
                - ip_netmask: 169.x.x.x/32
                  next_hop:
                    get_param: EC2MetadataIp
                - default: true
                  next_hop:
                    get_param: ControlPlaneDefaultRoute
              - type: vlan
                vlan_id:
                  get_param: InternalApiNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: InternalApiIpSubnet
              - type: vlan
                vlan_id:
                  get_param: StorageNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: StorageIpSubnet
              - type: contrail_vrouter
                name: vhost0
                use_dhcp: false
                members:
                  -
                    type: interface
                    name: nic2
                    use_dhcp: false
                addresses:
                - ip_netmask:
                    get_param: TenantIpSubnet

outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl
```

Advanced vRouter Kernel Mode Configuration

IN THIS SECTION

In addition to the standard NIC configuration, the vRouter kernel mode supports VLAN, Bond, and Bond + VLAN modes. The configuration snippets below only show the relevant section of the NIC template configuration for each mode.

VLAN

```
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: nic2
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
        str_replace:
          template: vlanVLANID
          params:
            VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Bond

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
   -
     type: interface
     name: nic2
   -
     type: interface
     name: nic3
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name: bond0
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Bond + VLAN

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
   -
     type: interface
     name: nic2
   -
     type: interface
     name: nic3
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
```

```
    device: bond0
  - type: contrail_vrouter
    name: vhost0
    use_dhcp: false
    members:
      -
        type: interface
        name:
          str_replace:
            template: vlanVLANID
            params:
              VLANID: {get_param: TenantNetworkVlanID}
        use_dhcp: false
    addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
```

Advanced vRouter DPDK Mode Configuration

IN THIS SECTION

In addition to the standard NIC configuration, the vRouter DPDK mode supports Standard, VLAN, Bond, and Bond + VLAN modes.

Network Environment Configuration:

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages:

```
 # For Intel CPU
  ContrailDpdkParameters:
```

```
    KernelArgs: "intel_iommu=on iommu=pt default_hugepagesz=1GB hugepagesz=1G hugepages=4
hugepagesz=2M hugepages=1024"
    ExtraSysctlSettings:
      # must be equal to value from kernel args: hugepages=4
      vm.nr_hugepages:
        value: 4
      vm.max_map_count:
        value: 128960
```

See the following NIC template configurations for vRouter DPDK mode. The configuration snippets below only show the relevant section of the NIC configuration for each mode.

Standard

```
- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

VLAN

```
 - type: contrail_vrouter_dpdk
             name: vhost0
             use_dhcp: false
             driver: uio_pci_generic
             cpu_list: 0x01
             vlan_id:
               get_param: TenantNetworkVlanID
             members:
               -
                 type: interface
```

```
                name: nic2
                use_dhcp: false
            addresses:
            - ip_netmask:
                get_param: TenantIpSubnet
```

Bond

```
- type: contrail_vrouter_dpdk
            name: vhost0
            use_dhcp: false
            driver: uio_pci_generic
            cpu_list: 0x01
            bond_mode: 4
            bond_policy: layer2+3
            members:
              -
                type: interface
                name: nic2
                use_dhcp: false
              -
                type: interface
                name: nic3
                use_dhcp: false
            addresses:
            - ip_netmask:
                get_param: TenantIpSubnet
```

Bond + VLAN

```
 - type: contrail_vrouter_dpdk
            name: vhost0
            use_dhcp: false
            driver: uio_pci_generic
            cpu_list: 0x01
            vlan_id:
              get_param: TenantNetworkVlanID
            bond_mode: 4
            bond_policy: layer2+3
            members:
```

```
        -
          type: interface
          name: nic2
          use_dhcp: false
        -
          type: interface
          name: nic3
          use_dhcp: false
      addresses:
      - ip_netmask:
          get_param: TenantIpSubnet
```

Advanced vRouter SRIOV + Kernel Mode Configuration

IN THIS SECTION

vRouter SRIOV + Kernel mode can be used in the following combinations:

- Standard
- VLAN
- Bond
- Bond + VLAN

Network environment configuration:

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages:

```
ContrailSriovParameters:
    KernelArgs: "intel_iommu=on iommu=pt default_hugepagesz=1GB hugepagesz=1G hugepages=4
```

```
hugepagesz=2M hugepages=1024"
    ExtraSysctlSettings:
      # must be equal to value from 1G kernel args: hugepages=4
      vm.nr_hugepages:
        value: 4
```

SRIOV PF/VF settings:

```
NovaPCIPassthrough:
- devname: "ens2f1"
  physical_network: "sriov1"
ContrailSriovNumVFs: ["ens2f1:7"]
```

The SRIOV NICs are not configured in the NIC templates. However, vRouter NICs must still be configured. See the following NIC template configurations for vRouter kernel mode. The configuration snippets below only show the relevant section of the NIC configuration for each mode.

VLAN

```
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: nic2
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
        str_replace:
          template: vlanVLANID
          params:
            VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Bond

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
   -
     type: interface
     name: nic2
   -
     type: interface
     name: nic3
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name: bond0
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Bond + VLAN

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
   -
     type: interface
     name: nic2
   -
     type: interface
     name: nic3
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
```

```
    device: bond0
  - type: contrail_vrouter
    name: vhost0
    use_dhcp: false
    members:
      -
        type: interface
        name:
          str_replace:
            template: vlanVLANID
            params:
              VLANID: {get_param: TenantNetworkVlanID}
        use_dhcp: false
    addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
```

Advanced vRouter SRIOV + DPDK Mode Configuration

IN THIS SECTION

vRouter SRIOV + DPDK can be used in the following combinations:

- Standard
- VLAN
- Bond
- Bond + VLAN

Network environment configuration:

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages

```
ContrailSriovParameters:
    KernelArgs: "intel_iommu=on iommu=pt default_hugepagesz=1GB hugepagesz=1G hugepages=4
hugepagesz=2M hugepages=1024"
    ExtraSysctlSettings:
      # must be equal to value from 1G kernel args: hugepages=4
      vm.nr_hugepages:
        value: 4
```

SRIOV PF/VF settings

```
NovaPCIPassthrough:
- devname: "ens2f1"
  physical_network: "sriov1"
ContrailSriovNumVFs: ["ens2f1:7"]
```

The SRIOV NICs are not configured in the NIC templates. However, vRouter NICs must still be configured. See the following NIC template configurations for vRouter DPDK mode. The configuration snippets below only show the relevant section of the NIC configuration for each mode.

Standard

```
- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
  addresses:
```

```
  - ip_netmask:
      get_param: TenantIpSubnet
```

VLAN

```
 - type: contrail_vrouter_dpdk
             name: vhost0
             use_dhcp: false
             driver: uio_pci_generic
             cpu_list: 0x01
             vlan_id:
               get_param: TenantNetworkVlanID
             members:
               -
                 type: interface
                 name: nic2
                 use_dhcp: false
             addresses:
             - ip_netmask:
                 get_param: TenantIpSubnet
```

Bond

```
- type: contrail_vrouter_dpdk
             name: vhost0
             use_dhcp: false
             driver: uio_pci_generic
             cpu_list: 0x01
             bond_mode: 4
             bond_policy: layer2+3
             members:
               -
                 type: interface
                 name: nic2
                 use_dhcp: false
               -
                 type: interface
                 name: nic3
                 use_dhcp: false
             addresses:
```

```
            - ip_netmask:
                get_param: TenantIpSubnet
```

Bond + VLAN

```
- type: contrail_vrouter_dpdk
            name: vhost0
            use_dhcp: false
            driver: uio_pci_generic
            cpu_list: 0x01
            vlan_id:
              get_param: TenantNetworkVlanID
            bond_mode: 4
            bond_policy: layer2+3
            members:
              -
                type: interface
                name: nic2
                use_dhcp: false
              -
                type: interface
                name: nic3
                use_dhcp: false
            addresses:
            - ip_netmask:
                get_param: TenantIpSubnet
```

Installing Overcloud

Perform the following procedure to install Overcloud.

1. Deployment:

```
openstack overcloud deploy --templates tripleo-heat-templates/ \
  --stack overcloud --libvirt-type kvm \
  --roles-file $role_file \
  -e tripleo-heat-templates/environments/rhsm.yaml \
  -e tripleo-heat-templates/environments/network-isolation.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-services.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-net.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
```

```
  -e containers-prepare-parameter.yaml \
  -e rhsm.yaml
```

2. Validation Test:

```
source overcloudrc
curl -O http://download.cirros-cloud.net/0.3.5/cirros-0.3.5-x86_64-disk.img
openstack image create --container-format bare --disk-format qcow2 --file cirros-0.3.5-x86_64-
disk.img cirros
openstack flavor create --public cirros --id auto --ram 64 --disk 0 --vcpus 1
openstack network create net1
openstack subnet create --subnet-range 1.0.0.0/24 --network net1 sn1
nova boot --image cirros --flavor cirros --nic net-id=`openstack network show net1 -c id -f
value` --availability-zone nova:overcloud-novacompute-0.localdomain c1
nova list
```

CHAPTER 9

Setting Up Contrail with Red Hat OpenStack 13

IN THIS CHAPTER

Understanding Red Hat OpenStack Platform Director

IN THIS SECTION

Red Hat OpenStack Platform Director

This chapter explains how to integrate a Contrail 5.1.*x* installation (or higher) with Red Hat OpenStack Platform Director 13.

Red Hat OpenStack Platform provides an installer called the Red Hat OpenStack Platform director (RHOSPd or OSPd), which is a toolset based on the OpenStack project TripleO (OOO, OpenStack on OpenStack). TripleO is an open source project that uses features of OpenStack to deploy a fully functional, tenant-facing OpenStack environment.

TripleO can be used to deploy an RDO-based OpenStack environment integrated with Tungsten Fabric. Red Hat OpenStack Platform director can be used to deploy an RHOSP-based OpenStack environment integrated with Contrail.

OSPd uses the concepts of undercloud and overcloud. OSPd sets up an undercloud, a single server running an operator-facing deployment that contains the OpenStack components needed to deploy and manage an overcloud, a tenant-facing deployment that hosts user workloads.

The overcloud is the deployed solution that can represent a cloud for any purpose, such as production, staging, test, and so on. The operator can select to deploy to their environment any of the available overcloud roles, such as controller, compute, and the like.

OSPd leverages existing core components of OpenStack including Nova, Ironic, Neutron, Heat, Glance, and Ceilometer to deploy OpenStack on bare metal hardware.

- Nova and Ironic are used in the undercloud to manage the bare metal instances that comprise the infrastructure for the overcloud.
- Neutron is used to provide a networking environment in which to deploy the overcloud.
- Glance stores machine images.
- Ceilometer collects metrics about the overcloud.

For more information about OSPd architecture, see OSPd documentation.

Contrail Roles

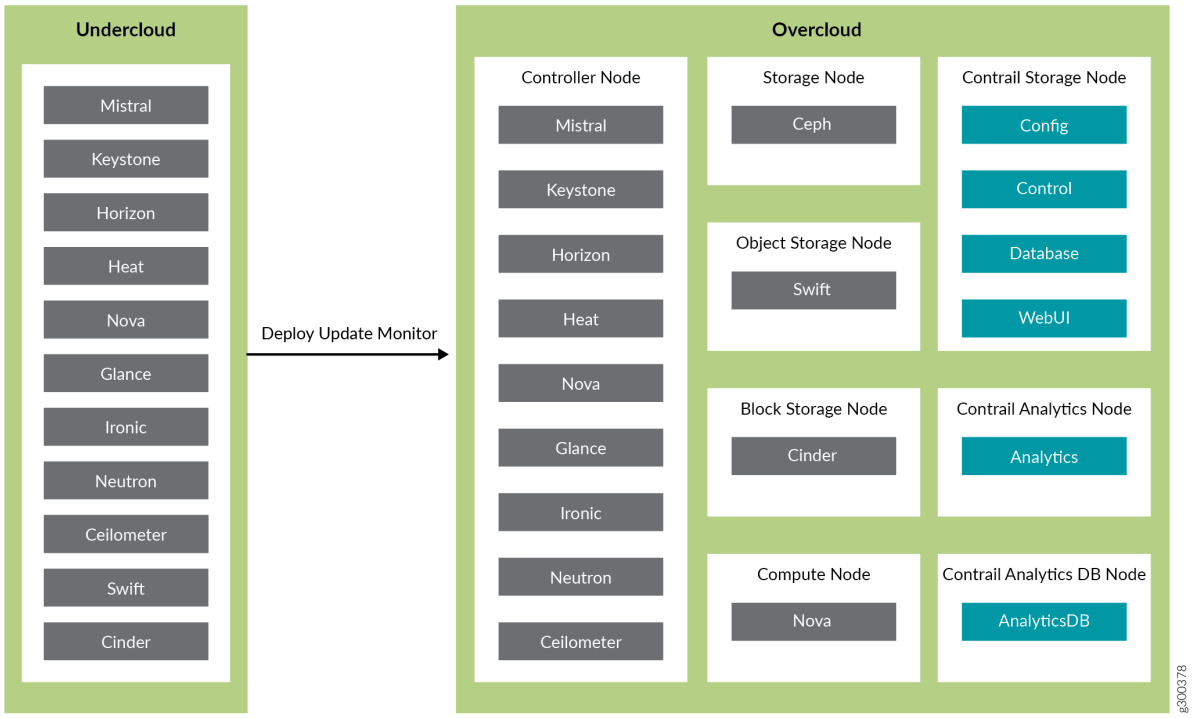
OSPd supports composable roles, which are groups of services that you define through Heat templates. Composable roles allow you to integrate Contrail into the overcloud environment.

The following are the Contrail roles used for integrating into the overcloud:

- Contrail Controller
- Contrail Analytics
- Contrail Analytics Database
- Contrail-TSN
- Contrail-DPDK

Figure 42 on page 450 shows the relationship and components of an undercloud and overcloud architecture for Contrail.

Figure 42: Undercloud and Overcloud with Roles



Undercloud Requirements

The undercloud is a single server or VM that hosts the OpenStack Platform director, which is an OpenStack installation used to provision OpenStack on the overcloud.

See Undercloud Requirements for the compute requirements of the undercloud.

Overcloud Requirements

The overcloud roles can be deployed to bare metal servers or to virtual machines (VMs), but the compute nodes must be deployed to bare metal systems. Every overcloud node must support IPMI for booting up from the undercloud using PXE.

Ensure the following requirements are met for the Contrail nodes per role.

- Non-high availability: A minimum of 4 overcloud nodes are needed for control plane roles for a non-high availability deployment:
  - 1x contrail-config (includes Contrail control)

    - 1x contrail-analytics
    - 1x contrail-analytics-database
    - 1x OpenStack controller
- High availability: A minimum of 12 overcloud nodes are needed for control plane roles for a high availability deployment:
    - 3x contrail-config (includes Contrail control)
    - 3x contrail-analytics
    - 3x contrail-analytics-database
    - 3x OpenStack controller

    If the control plane roles are deployed to VMs, use 3 separate physical servers and deploy one role of each kind to each physical server.

See Overcloud Requirements for the compute requirements of the overcloud.

Networking Requirements

As a minimum, the installation requires two networks:

- provisioning network - This is the private network that the undercloud uses to provision the overcloud.
- external network - This is the externally-routable network you use to access the undercloud and overcloud nodes.

Ensure the following requirements are met for the provisioning network:

- One NIC from every machine must be in the same broadcast domain of the provisioning network, and it should be the same NIC on each of the overcloud machines. For example, if you use the second NIC on the first overcloud machine, you should use the second NIC on each additional overcloud machine.

  During installation, these NICs will be referenced by a single name across all overcloud machines.
- The provisioning network NIC should not be the same NIC that you are using for remote connectivity to the undercloud machine. During the undercloud installation, an Open vSwitch bridge will be created for Neutron, and the provisioning NIC will be bridged to the Open vSwitch bridge. Consequently, connectivity would be lost if the provisioning NIC was also used for remote connectivity to the undercloud machine.
- The provisioning NIC on the overcloud nodes must be untagged.

- You must have the MAC address of the NIC that will PXE boot the IPMI information for the machine on the provisioning network. The IPMI information will include such things as the IP address of the IPMI NIC and the IPMI username and password.
- All of the networks must be available to all of the Contrail roles and computes.

While the provisioning and external networks are sufficient for basic applications, you should create additional networks in most overcloud environments to provide isolation for the different traffic types by assigning network traffic to specific network interfaces or bonds.

When isolated networks are configured, the OpenStack services are configured to use the isolated networks. If no isolated networks are configured, all services run on the provisioning network. If only some isolated networks are configured, traffic belonging to a network not configured runs on the provisioning network.

The following networks are typically deployed when using network isolation topology:

- Provisioning - used by the undercloud to provision the overcloud
- Internal API - used by OpenStack services to communicate with each other
- Tenant - used for tenant overlay data plane traffic (one network per tenant)
- Storage - used for storage data traffic
- Storage Management - used for storage control and management traffic
- External - provides external access to the undercloud and overcloud, including external access to the web UIs and public APIs
- Floating IP - provides floating IP access to the tenant network (can either be merged with external or can be a separate network)
- Management - provides access for system administration

For more information on the different network types, see Planning Networks.

For more information on networking requirements, see Networking Requirements.

Compatibility Matrix

The following combinations of Operating System/OpenStack/Deployer/Contrail are supported:

Table 29: Compatibility Matrix

Operating System	OpenStack	Deployer	Contrail
RHEL 7.5	OSP13	OSPd13	Contrail 5.1.*x* or higher
CentOS 7.5	RDO queens/stable	tripleo queens/stable	Tungsten Fabric (latest)

Installation Summary

The general installation procedure is as follows:

- Set up the infrastructure, which is the set of servers or VMs that host the undercloud and overcloud, including the provisioning network that connects them together.
- Set up the undercloud, which is the OSPd application.
- Set up the overcloud, which is the set of services in the tenant-facing network. Contrail is part of the overcloud.

For more information on installing and using the RHOSPd, see Red Hat documentation.

Setting Up the Infrastructure

IN THIS SECTION

Target Configuration (Example)

Undercloud and overcloud KVM hosts require virtual switches and virtual machine definitions to be configured. You can deploy any KVM host operating system version that supports KVM and OVS. The

following example shows a RHEL/CentOS based system. If you are using RHEL, you must subscribe the system.

The following example illustrates all control plane functions as Virtual Machines hosted on KVM hosts.

There are different ways to create the infrastructure providing the control plane elements. To illustrate the installation procedure, we will use four host machines for the infrastructure, each running KVM. KVM1 contains a VM running the undercloud while KVM2 through KVM4 each contains a VM running an OpenStack controller and a Contrail controller (Table 30 on page 454).

Table 30: Control Plane Infrastructure

KVM Host	Virtual Machines
KVM1	undercloud
KVM2	OpenStack Controller 1, Contrail Contoller 1
KVM3	OpenStack Controller 2, Contrail Contoller 2
KVM4	OpenStack Controller 3, Contrail Contoller 3

Figure 43 on page 455 shows the physical connectivity where each KVM host and each compute node has two interfaces that connect to an external switch. These interfaces attach to separate virtual bridges within the VM, allowing for two physically separate networks (external and provisioning networks).

Figure 43: Physical View

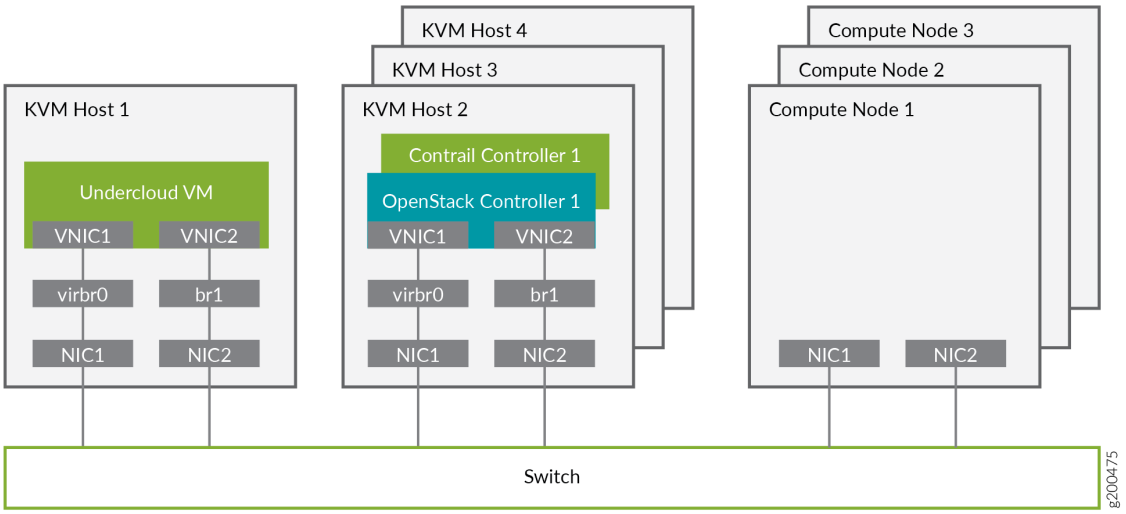
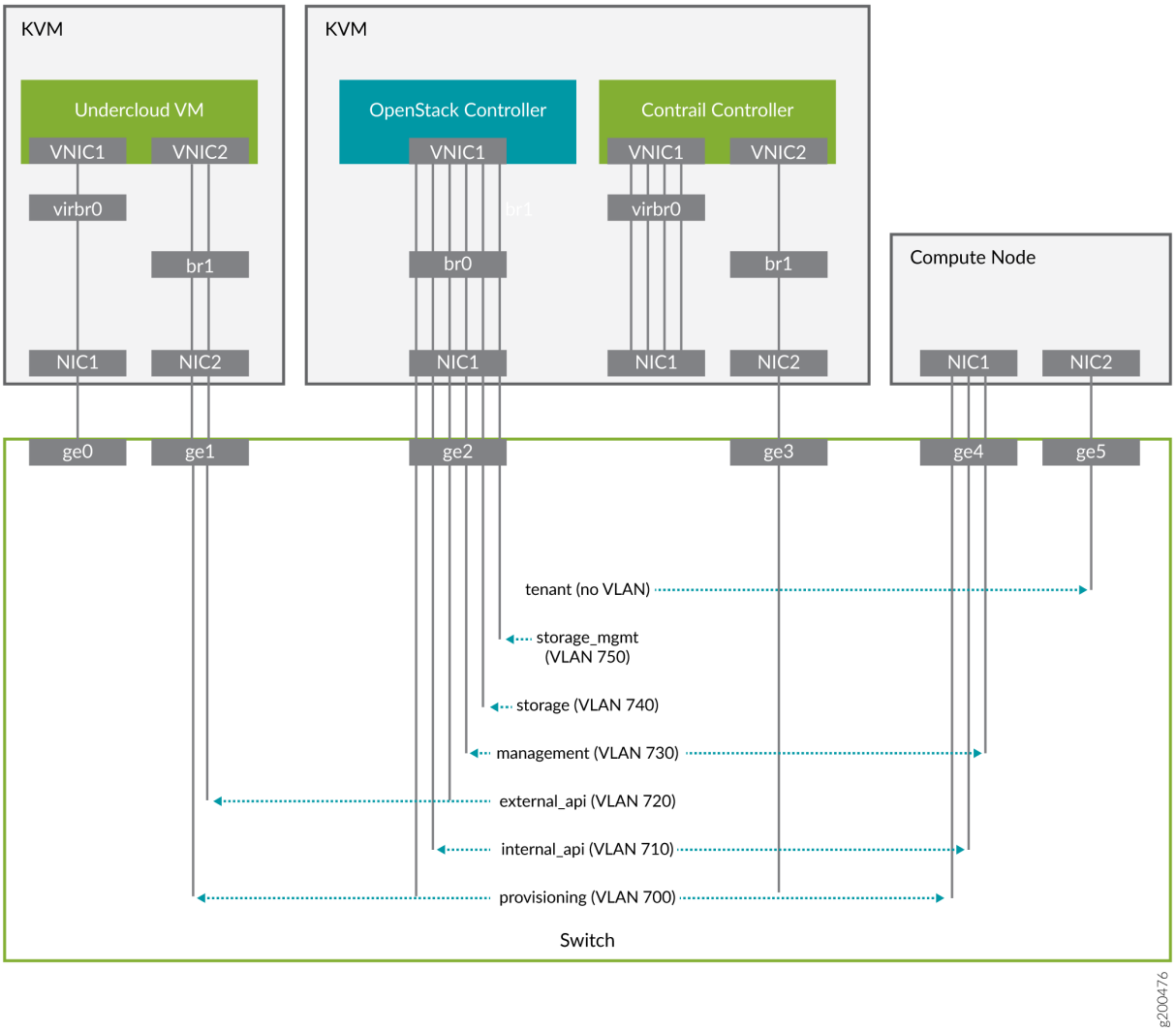


Figure 44 on page 456 shows the logical view of the connectivity where VLANs are used to provide further network separation for the different OpenStack network types.

Figure 44: Logical View



The following sections describe how to configure the infrastructure, the undercloud, and finally the overcloud.

Configure the External Physical Switch

Configure the ports and VLANs on the external physical switch according to the following table:

Table 31: External Physical Switch Port and VLAN Configuration

Port	Trunked VLAN	Native VLAN
ge0	-	-

Table 31: External Physical Switch Port and VLAN Configuration *(Continued)*

Port	Trunked VLAN	Native VLAN
ge1	700, 720	-
ge2	700, 710, 720, 730, 740, 750	-
ge3	-	-
ge4	710, 730	700
ge5	-	-

Configure KVM Hosts

Use this example procedure to install the required packages and start KVM and Open vSwitch on each undercloud and overcloud KVM host.

1. Log in to a KVM host.
2. Install the required packages.

```
yum install -y libguestfs \
   libguestfs-tools \
   openvswitch \
   virt-install \
   kvm libvirt \
   libvirt-python \
   python-virtualbmc \
   python-virtinst
```

3. Start KVM and Open vSwitch.

```
systemctl start libvirtd
systemctl start openvswitch
```

4. Additionally, on the overcloud nodes only, create and start the virtual switches br0 and br1.

Table 32: vSwitch Configuration

Bridge	Trunked VLAN	Native VLAN
br0	710, 720, 730 740, 750	700
br1	-	-

```
# Create the virtual switches and bind them to the respective interfaces.
ovs-vsctl add-br br0
ovs-vsctl add-br br1
ovs-vsctl add-port br0 NIC1
ovs-vsctl add-port br1 NIC2


# Create the configuration file for br0.
cat << EOF > br0.xml
<network>
   <name>br0</name>
   <forward mode='bridge'/>
   <bridge name='br0'/>
   <virtualport type='openvswitch'/>
   <portgroup name='overcloud'/>
      <vlan trunk='yes'>
         <tag id='700' nativeMode='untagged'/>
         <tag id='710'/>
         <tag id='720'/>
         <tag id='730'/>
         <tag id='740'/>
         <tag id='750'/>
      </vlan>
   </portgroup>
</network>
EOF


# Create the configuration file for br1.
cat << EOF > br1.xml
<network>
   <name>br1</name>
```

```
   <forward mode='bridge'/>
   <bridge name='br1'/>
   <virtualport type='openvswitch'/>
</network>
EOF


# Create the br0 network based on the configuration file.
virsh net-define br0.xml
virsh net-start br0
virsh net-autostart br0


# Create the br1 network based on the configuration file.
virsh net-define br1.xml
virsh net-start br1
virsh net-autostart br1
```

5. Repeat step 1 through step 4 for each KVM host.

Create the Overcloud VM Definitions on the Overcloud KVM Hosts

Use this example procedure on each overcloud KVM host (KVM2 to KVM4) to do the following:

- create the VM definitions for that overcloud KVM host
- create and start a virtual baseboard management controller for that overcloud KVM host so that the VM can be managed using IPMI
- create an **ironic_list** file to be used by the undercloud

This example procedure creates a VM definition consisting of 2 compute nodes, 1 Contrail controller node, and 1 OpenStack controller node on each overcloud KVM host.

1. Log in to an overcloud KVM host.
2. Specify the roles you want to create.

```
ROLES=compute:2,contrail-controller:1,control:1
```

3. Create the VM definitions.

```
# Initialize and specify the IPMI user and password you want to use.
num=0
ipmi_user=<user>
```

```
ipmi_password=<password>
libvirt_path=/var/lib/libvirt/images
port_group=overcloud
prov_switch=br0
/bin/rm ironic_list


# For each role and instance specified in the ROLES variable:
#    - create the VM definition
#    - create and start a virtual baseboard management controller (vbmc)
#    - store the VM information into an ironic_list file (for later use in the undercloud)
IFS=',' read -ra role_list <<< "${ROLES}"
for role in ${role_list[@]}; do
   role_name=`echo $role|cut -d ":" -f 1`
   role_count=`echo $role|cut -d ":" -f 2`
   for count in `seq 1 ${role_count}`; do
      echo $role_name $count
      qemu-img create -f qcow2 ${libvirt_path}/${role_name}_${count}.qcow2 99G
      virsh define /dev/stdin <<EOF
      $(virt-install --name ${role_name}_${count} \
         --disk ${libvirt_path}/${role_name}_${count}.qcow2 \
         --vcpus=4 \
         --ram=16348 \
         --network network=br0,model=virtio,portgroup=${port_group} \
         --network network=br1,model=virtio \
         --virt-type kvm \
         --cpu host \
         --import \
         --os-variant rhel7 \
         --serial pty \
         --console pty,target_type=virtio \
         --graphics vnc \
         --print-xml)
EOF
      vbmc add ${role_name}_${count} --port 1623${num} --username ${ipmi_user} --password $
{ipmi_password}
      vbmc start ${role_name}_${count}
      prov_mac=`virsh domiflist ${role_name}_${count}|grep ${prov_switch}|awk '{print $5}'`
      vm_name=${role_name}-${count}-`hostname -s`
      kvm_ip=`ip route get 1  |grep src |awk '{print $7}'`
      echo ${prov_mac} ${vm_name} ${kvm_ip} ${role_name} 1623${num}>> ironic_list
      num=$(expr $num + 1)
```

```
    done
done
```

4. Repeat step 1 through step 3 on each overcloud KVM host.

CAUTION: This procedure creates one **ironic_list** file per overcloud KVM host. Combine the contents of each file into a single **ironic_list** file on the undercloud.

The following shows the resulting **ironic_list** file after you combine the contents from each separate file:

```
52:54:00:e7:ca:9a compute-1-5b3s31 10.87.64.32 compute 16230 52:54:00:30:6c:3f compute-2-5b3s31
10.87.64.32 compute 16231 52:54:00:9a:0c:d5 contrail-controller-1-5b3s31 10.87.64.32 contrail-
controller 16232 52:54:00:cc:93:d4 control-1-5b3s31 10.87.64.32 control 16233 52:54:00:28:10:d4
compute-1-5b3s30 10.87.64.31 compute 16230 52:54:00:7f:36:e7 compute-2-5b3s30 10.87.64.31 compute
16231 52:54:00:32:e5:3e contrail-controller-1-5b3s30 10.87.64.31 contrail-controller 16232
52:54:00:d4:31:aa control-1-5b3s30 10.87.64.31 control 16233 52:54:00:d1:d2:ab compute-1-5b3s32
10.87.64.33 compute 16230 52:54:00:ad:a7:cc compute-2-5b3s32 10.87.64.33 compute 16231
52:54:00:55:56:50 contrail-controller-1-5b3s32 10.87.64.33 contrail-controller 16232
52:54:00:91:51:35 control-1-5b3s32 10.87.64.33 control 16233
```

Create the Undercloud VM Definition on the Undercloud KVM Host

Use this example procedure on the undercloud KVM host (KVM1) to create the undercloud VM definition and to start the undercloud VM.

1. Create the images directory.

```
mkdir ~/images
cd images
```

2. Retrieve the image.

- CentOS

```
curl https://cloud.centos.org/centos/7/images/CentOS-7-x86_64-GenericCloud-1802.qcow2.xz -
o CentOS-7-x86_64-GenericCloud-1802.qcow2.xz
unxz -d images/CentOS-7-x86_64-GenericCloud-1802.qcow2.xz
cloud_image=~/images/CentOS-7-x86_64-GenericCloud-1802.qcow2
```

- RHEL

```
Download rhel-server-7.5-update-1-x86_64-kvm.qcow2 from the Red Hat portal to ~/images.
cloud_image=~/images/rhel-server-7.5-update-1-x86_64-kvm.qcow2
```

3. Customize the undercloud image.

```
undercloud_name=queensa
undercloud_suffix=local
root_password=<password>
stack_password=<password>
export LIBGUESTFS_BACKEND=direct
qemu-img create -f qcow2 /var/lib/libvirt/images/${undercloud_name}.qcow2 100G
virt-resize --expand /dev/sda1 ${cloud_image} /var/lib/libvirt/images/$
{undercloud_name}.qcow2
virt-customize  -a /var/lib/libvirt/images/${undercloud_name}.qcow2 \
--run-command 'xfs_growfs /' \
--root-password password:${root_password} \
--hostname ${undercloud_name}.${undercloud_suffix} \
--run-command 'useradd stack' \
--password stack:password:${stack_password} \
--run-command 'echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack' \
--chmod 0440:/etc/sudoers.d/stack \
--run-command 'sed -i "s/PasswordAuthentication no/PasswordAuthentication yes/g" /etc/ssh/
sshd_config' \
--run-command 'systemctl enable sshd' \
--run-command 'yum remove -y cloud-init' \
--selinux-relabel
```

NOTE: As part of the undercloud definition, a user called **stack** is created. This user will be used later to install the undercloud.

4. Define the undercloud virsh template.

```
vcpus=8
vram=32000
virt-install --name ${undercloud_name} \
--disk /var/lib/libvirt/images/${undercloud_name}.qcow2 \
--vcpus=${vcpus} \
--ram=${vram} \
```

```
--network network=default,model=virtio \
--network network=br0,model=virtio,portgroup=overcloud \
--virt-type kvm \
--import \
--os-variant rhel7 \
--graphics vnc \
--serial pty \
--noautoconsole \
--console pty,target_type=virtio
```

5. Start the undercloud VM.

```
virsh start ${undercloud_name}
```

6. Retrieve the undercloud IP address. It might take several seconds before the IP address is available.

```
undercloud_ip=`virsh domifaddr ${undercloud_name} |grep ipv4 |awk '{print $4}' |awk -F"/"
'{print $1}'` ssh-copy-id ${undercloud_ip}
```

Setting Up the Undercloud

IN THIS SECTION

Install the Undercloud

Use this example procedure to install the undercloud.

1. Log in to the undercloud VM from the undercloud KVM host.

```
ssh ${undercloud_ip}
```

2. Configure the hostname.

```
undercloud_name=`hostname -s`
undercloud_suffix=`hostname -d`
hostnamectl set-hostname ${undercloud_name}.${undercloud_suffix}
hostnamectl set-hostname --transient ${undercloud_name}.${undercloud_suffix}
```

3. Add the hostname to the **/etc/hosts** file. The following example assumes the management interface is eth0.

```
undercloud_ip=`ip addr sh dev eth0 | grep "inet " | awk '{print $2}' | awk -F"/" '{print $1}'`
echo ${undercloud_ip} ${undercloud_name}.${undercloud_suffix} ${undercloud_name} >> /etc/hosts
```

4. Set up the repositories.

   - CentOS

```
tripleo_repos=`python -c 'import requests;r = requests.get("https://trunk.rdoproject.org/
centos7-queens/current"); print r.text ' | grep python2-tripleo-repos|awk -F"href=\""
'{print $2}' | awk -F"\"" '{print $1}'`
yum install -y https://trunk.rdoproject.org/centos7-queens/current/${tripleo_repos}
tripleo-repos -b queens current
```

   - RHEL

```
#Register with Satellite (can be done with CDN as well)
satellite_fqdn=device.example.net
act_key=xxx
org=example
yum localinstall -y http://${satellite_fqdn}/pub/katello-ca-consumer-latest.noarch.rpm
subscription-manager register --activationkey=${act_key} --org=${org}
```

5. Install the Tripleo client.

```
yum install -y python-tripleoclient tmux
```

6. Copy the undercloud configuration file sample and modify the configuration as required. See Red Hat documentation for information on how to modify that file.

```
su - stack
cp /usr/share/instack-undercloud/undercloud.conf.sample ~/undercloud.conf
vi ~/undercloud.conf
```

7. Install the undercloud.

```
openstack undercloud install
source stackrc
```

Perform Post-Install Configuration

1. Configure a forwarding path between the provisioning network and the external network:

```
sudo iptables -A FORWARD -i br-ctlplane -o eth0 -j ACCEPT
sudo iptables -A FORWARD -i eth0 -o br-ctlplane -m state --state RELATED,ESTABLISHED -j
ACCEPT
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

2. Add the external API interface:

```
sudo ip link add name vlan720 link br-ctlplane type vlan id 720
sudo ip addr add 10.2.0.254/24 dev vlan720
sudo ip link set dev vlan720 up
```

3. Add the `stack` user to the docker group:

```
newgrp docker
exit
su - stack
source stackrc
```

Setting Up the Overcloud

IN THIS SECTION

Configuring the Overcloud

Use this example procedure on the undercloud to set up the configuration for the overcloud.

1. Specify the name server to be used:

```
undercloud_nameserver=8.8.8.8
openstack subnet set `openstack subnet show ctlplane-subnet -c id -f value` --dns-nameserver $
{undercloud_nameserver}
```

2. Retrieve and upload the overcloud images.

   a. Create the image directory:

```
mkdir images
cd images
```

   b. Retrieve the overcloud images from either the RDO project or from Red Hat.

   - TripleO

```
curl -O https://images.rdoproject.org/queens/rdo_trunk/current-tripleo-rdo/ironic-
python-agent.tar
curl -O https://images.rdoproject.org/queens/rdo_trunk/current-tripleo-rdo/overcloud-
full.tar
tar xvf ironic-python-agent.tar
tar xvf overcloud-full.tar
```

- OSP13

```
sudo yum install -y rhosp-director-images rhosp-director-images-ipa
for i in /usr/share/rhosp-director-images/overcloud-full-latest-13.0.tar /usr/share/
rhosp-director-images/ironic-python-agent-latest-13.0.tar ; do tar -xvf $i; done
```

c. Upload the overcloud images:

```
cd
openstack overcloud image upload --image-path /home/stack/images/
```

3. Prepare OpenStack's bare metal provisioning (Ironic).

Ironic is an integrated OpenStack program that provisions bare metal machines instead of virtual machines. It is best thought of as a bare metal hypervisor API and a set of plugins that interact with the bare metal hypervisors.

NOTE: Make sure to combine the **ironic_list** files from the three overcloud KVM hosts.

a. Add the overcloud VMs to Ironic:

```
ipmi_password=<password>
ipmi_user=<user>
while IFS= read -r line; do
  mac=`echo $line|awk '{print $1}'`
  name=`echo $line|awk '{print $2}'`
  kvm_ip=`echo $line|awk '{print $3}'`
  profile=`echo $line|awk '{print $4}'`
  ipmi_port=`echo $line|awk '{print $5}'`
  uuid=`openstack baremetal node create --driver ipmi \
                                        --property cpus=4 \
                                        --property memory_mb=16348 \
                                        --property local_gb=100 \
                                        --property cpu_arch=x86_64 \
                                        --driver-info ipmi_username=${ipmi_user}  \
                                        --driver-info ipmi_address=${kvm_ip} \
                                        --driver-info ipmi_password=${ipmi_password} \
                                        --driver-info ipmi_port=${ipmi_port} \
                                        --name=${name} \
                                        --property capabilities=profile:$
```

```
{profile},boot_option:local \
                                          -c uuid -f value`
  openstack baremetal port create --node ${uuid} ${mac}
done < <(cat ironic_list)

DEPLOY_KERNEL=$(openstack image show bm-deploy-kernel -f value -c id)
DEPLOY_RAMDISK=$(openstack image show bm-deploy-ramdisk -f value -c id)

for i in `openstack baremetal node list -c UUID -f value`; do
  openstack baremetal node set $i --driver-info deploy_kernel=$DEPLOY_KERNEL --driver-info
deploy_ramdisk=$DEPLOY_RAMDISK
done

for i in `openstack baremetal node list -c UUID -f value`; do
  openstack baremetal node show $i -c properties -f value
done
```

b. Introspect the overcloud node:

```
for node in $(openstack baremetal node list -c UUID -f value) ; do
  openstack baremetal node manage $node
done
openstack overcloud node introspect --all-manageable --provide
```

c. Add Baremetal Server (BMS) to Ironic.

- Create rules for automated profiling.

  Evaluate the attributes of the physical server. The server will automatically be profiled based on the rules.

  The following example shows how to create a rule for system manufacturer as “Supermicro” and memory greater or equal to 128 GB.

```
cat << EOF > ~/rule_compute.json
[
 {
     "description": "set physical compute",
     "conditions": [
         {"op": "eq", "field": "data://auto_discovered", "value": true},
         {"op": "eq", "field": "data://inventory.system_vendor.manufacturer",
          "value": "Supermicro"},
```

```
          {"op": "ge", "field": "memory_mb", "value": 128000}
      ],
      "actions": [
          {"action": "set-attribute", "path": "driver_info/ipmi_username",
           "value": "<user>"},
          {"action": "set-attribute", "path": "driver_info/ipmi_password",
           "value": "<password>"},
          {"action": "set-capability", "name": "profile", "value": "compute"},
          {"action": "set-attribute", "path": "driver_info/ipmi_address","value":
"{data[inventory][bmc_address]}"}
      ]
 }
]
EOF
```

You can import the rule by:

```
openstack baremetal introspection rule import ~/rule_compute.json
```

- Scan the BMC IP range and automatically add new servers matching the above rule by:

```
ipmi_range=10.87.122.25/32
ipmi_password=<password>
ipmi_user=<user>
openstack overcloud node discover --range ${ipmi_range} \
  --credentials ${ipmi_user}:${ipmi_password} \
  --introspect --provide
```

4. Create Flavor:

```
for i in compute-dpdk \
compute-sriov \
contrail-controller \
contrail-analytics \
contrail-database \
contrail-analytics-database; do
  openstack flavor create $i --ram 4096 --vcpus 1 --disk 40
  openstack flavor set --property "capabilities:boot_option"="local" \
                       --property "capabilities:profile"="${i}" ${i}
done
```

5. Copy the TripleO heat templates.

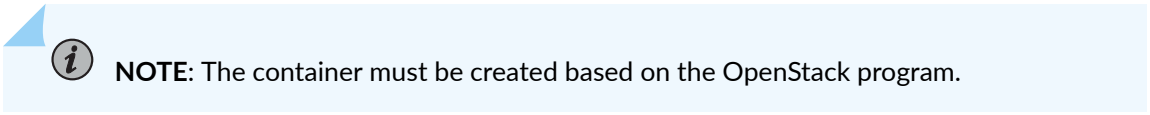
```
cp -r /usr/share/openstack-tripleo-heat-templates/ tripleo-heat-templates
```

6. Download and copy the Contrail heat templates from https://support.juniper.net/support/downloads.

```
tar -xzvf contrail-tripleo-heat-templates-<version>.tgz
cp -r contrail-tripleo-heat-templates/* tripleo-heat-templates/
```

7. Create and upload the OpenStack containers.

   a. Create the OpenStack container file.



NOTE: The container must be created based on the OpenStack program.

- TripleO

```
openstack overcloud container image prepare \
  --namespace docker.io/tripleoqueens \
  --tag current-tripleo \
  --tag-from-label rdo_version \
  --output-env-file=~/overcloud_images.yaml

tag=`grep "docker.io/tripleoqueens" docker_registry.yaml |tail -1 |awk -F":" '{print
$3}'`

openstack overcloud container image prepare \
  --namespace docker.io/tripleoqueens \
  --tag ${tag} \
  --push-destination 192.168.24.1:8787 \
  --output-env-file=~/overcloud_images.yaml \
  --output-images-file=~/local_registry_images.yaml
```

- OSP13

```
openstack overcloud container image prepare \
 --push-destination=192.168.24.1:8787  \
 --tag-from-label {version}-{release} \
 --output-images-file ~/local_registry_images.yaml  \
 --namespace=registry.access.Red Hat.com/rhosp13  \
```

```
  --prefix=openstack-  \
  --tag-from-label {version}-{release}  \
  --output-env-file ~/overcloud_images.yaml
```

b. Upload the OpenStack containers:

```
openstack overcloud container image upload --config-file ~/local_registry_images.yaml
```

8. Create and upload the Contrail containers.

a. Create the Contrail container file.

NOTE: This step is optional. The Contrail containers can be downloaded from external registries later.

```
cd ~/tripleo-heat-templates/tools/contrail
./import_contrail_container.sh -f container_outputfile -r registry -t tag [-i insecure] [-
u username] [-p password] [-c certificate pat
```

Here are few examples of importing Contrail containers from different sources:

- Import from password protected public registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r hub.juniper.net/contrail -
u USERNAME -p PASSWORD -t 1234
```

- Import from Dockerhub:

```
./import_contrail_container.sh -f /tmp/contrail_container -r docker.io/
opencontrailnightly -t 1234
```

- Import from private secure registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r device.example.net:5443 -
c http://device.example.net/pub/device.example.net.crt -t 1234
```

- Import from private insecure registry:

```
./import_contrail_container.sh -f /tmp/contrail_container -r 10.0.0.1:5443 -i 1 -t 1234
```

b. Upload Contrail containers to the undercloud registry:

```
openstack overcloud container image upload --config-file /tmp/contrail_container
```

Customizing the Contrail Service with Templates (contrail-services.yaml)

This section contains information to customize Contrail services for your network by modifying the **contrail-services.yaml** file.

- Contrail Services customization

```
vi ~/tripleo-heat-templates/environments/contrail-services.yaml
parameter_defaults:
  ContrailSettings:
    VROUTER_GATEWAY: 10.0.0.1
    # KEY1: value1
    # KEY2: value2

    VXLAN_VN_ID_MODE: "configured"
    ENCAP_PRIORITY: "VXLAN,MPLSoUDP,MPLSoGRE"

  ContrailControllerParameters:
    AAAMode: rbac
```

- Contrail registry settings

```
vi ~/tripleo-heat-templates/environments/contrail-services.yaml
```

Here are few examples of default values for various registries:

- Public Juniper registry

```
parameter_defaults:
  ContrailRegistry: hub.juniper.net/contrail
```

```
  ContrailRegistryUser: <USER>
  ContrailRegistryPassword: <PASSWORD>
```

- Insecure registry

```
parameter_defaults:
  ContrailRegistryInsecure: true
  DockerInsecureRegistryAddress: 10.87.64.32:5000,192.168.24.1:8787
  ContrailRegistry: 10.87.64.32:5000
```

- Private secure registry

```
parameter_defaults:
  ContrailRegistryCertUrl: http://device.example.net/pub/device.example.net.crt
  ContrailRegistry: device.example.net:5443
```

- Contrail Container image settings

```
parameter_defaults:
  ContrailImageTag: queens-5.0-104-rhel-queens
```

Customizing the Contrail Network with Templates

IN THIS SECTION

Overview

In order to customize the network, define different networks and configure the overcloud nodes NIC layout. TripleO supports a flexible way of customizing the network.

The following networking customization example uses network as:

Table 33: Network Customization

Network	VLAN	overcloud Nodes
provisioning	-	All
internal_api	710	All
external_api	720	OpenStack CTRL
storage	740	OpenStack CTRL, Computes
storage_mgmt	750	OpenStack CTRL
tenant	-	Contrail CTRL, Computes

Roles Configuration (roles_data_contrail_aio.yaml)

IN THIS SECTION

The networks must be activated per role in the roles_data file:

```
vi ~/tripleo-heat-templates/roles_data_contrail_aio.yaml
```

OpenStack Controller

```
###############################################################################
# Role: Controller                                                            #
###############################################################################
- name: Controller
  description: |
    Controller role that has all the controler services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  tags:
    - primary
    - controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
```

Compute Node

```
###############################################################################
# Role: Compute                                                               #
###############################################################################
- name: Compute
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi
    - Tenant
    - Storage
```

Contrail Controller

```
###############################################################################
# Role: ContrailController                                                    #
###############################################################################
- name: ContrailController
  description: |
    ContrailController role that has all the Contrail controler services loaded
    and handles config, control and webui functions
  CountDefault: 1
  tags:
    - primary
    - contrailcontroller
  networks:
    - InternalApi
    - Tenant
```

Compute DPDK

```
###############################################################################
# Role: ContrailDpdk                                                          #
###############################################################################
- name: ContrailDpdk
  description: |
    Contrail Dpdk Node role
  CountDefault: 0
  tags:
    - contraildpdk
  networks:
    - InternalApi
    - Tenant
    - Storage
```

Compute SRIOV

```
###############################################################################
# Role: ContrailSriov
###############################################################################
- name: ContrailSriov
```

```
  description: |
    Contrail Sriov Node role
  CountDefault: 0
  tags:
    - contrailsriov
  networks:
    - InternalApi
    - Tenant
    - Storage
```

Compute CSN

```
###############################################################################
# Role: ContrailTsn
###############################################################################
- name: ContrailTsn
  description: |
    Contrail Tsn Node role
  CountDefault: 0
  tags:
    - contrailtsn
  networks:
    - InternalApi
    - Tenant
    - Storage
```

Network Parameter Configuration (contrail-net.yaml)

```
cat ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml
resource_registry:
  OS::TripleO::Controller::Net::SoftwareConfig: ../../network/config/contrail/controller-nic-
config.yaml
  OS::TripleO::ContrailController::Net::SoftwareConfig: ../../network/config/contrail/contrail-
controller-nic-config.yaml
  OS::TripleO::ContrailControlOnly::Net::SoftwareConfig: ../../network/config/contrail/contrail-
controller-nic-config.yaml
  OS::TripleO::Compute::Net::SoftwareConfig: ../../network/config/contrail/compute-nic-
config.yaml
  OS::TripleO::ContrailDpdk::Net::SoftwareConfig: ../../network/config/contrail/contrail-dpdk-
nic-config.yaml
```

```
  OS::TripleO::ContrailSriov::Net::SoftwareConfig: ../../network/config/contrail/contrail-sriov-
nic-config.yaml
  OS::TripleO::ContrailTsn::Net::SoftwareConfig: ../../network/config/contrail/contrail-tsn-nic-
config.yaml

parameter_defaults:
  # Customize all these values to match the local environment
  TenantNetCidr: 10.0.0.0/24
  InternalApiNetCidr: 10.1.0.0/24
  ExternalNetCidr: 10.2.0.0/24
  StorageNetCidr: 10.3.0.0/24
  StorageMgmtNetCidr: 10.4.0.0/24
  # CIDR subnet mask length for provisioning network
  ControlPlaneSubnetCidr: '24'
  # Allocation pools
  TenantAllocationPools: [{'start': '10.0.0.10', 'end': '10.0.0.200'}]
  InternalApiAllocationPools: [{'start': '10.1.0.10', 'end': '10.1.0.200'}]
  ExternalAllocationPools: [{'start': '10.2.0.10', 'end': '10.2.0.200'}]
  StorageAllocationPools: [{'start': '10.3.0.10', 'end': '10.3.0.200'}]
  StorageMgmtAllocationPools: [{'start': '10.4.0.10', 'end': '10.4.0.200'}]
  # Routes
  ControlPlaneDefaultRoute: 192.168.24.1
  InternalApiDefaultRoute: 10.1.0.1
  ExternalInterfaceDefaultRoute: 10.2.0.1
  # Vlans
  InternalApiNetworkVlanID: 710
  ExternalNetworkVlanID: 720
  StorageNetworkVlanID: 730
  StorageMgmtNetworkVlanID: 740
  TenantNetworkVlanID: 3211
  # Services
  EC2MetadataIp: 192.168.24.1  # Generally the IP of the undercloud
  DnsServers: ["172.x.x.x"]
  NtpServer: 10.0.0.1
```

Network Interface Configuration (*-NIC-*.yaml)

IN THIS SECTION

NIC configuration files exist per role in the following directory:

```
cd ~/tripleo-heat-templates/network/config/contrail
```

OpenStack Controller

```
heat_template_version: queens

description: >
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role. This is an example for a Nova compute node using
  Contrail vrouter and the vhost0 interface.
parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal_api network
    type: string
  InternalApiDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the internal api network.
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
```

```
    default: ''
    description: IP address/subnet on the storage_mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 50
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  ExternalInterfaceDefaultRoute: # Not used by default in this template
```

```
    default: '10.0.0.1'
    description: The default route of the external network.
    type: string
  ManagementInterfaceDefaultRoute: # Commented out by default in this template
    default: unset
    description: The default route of the management network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
              - type: interface
                name: nic1
                use_dhcp: false
                dns_servers:
                  get_param: DnsServers
                addresses:
                - ip_netmask:
                    list_join:
                      - '/'
                      - - get_param: ControlPlaneIp
                        - get_param: ControlPlaneSubnetCidr
                routes:
                - ip_netmask: 169.x.x.x/32
                  next_hop:
                    get_param: EC2MetadataIp
```

```
                - default: true
                  next_hop:
                    get_param: ControlPlaneDefaultRoute
              - type: vlan
                vlan_id:
                  get_param: InternalApiNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: InternalApiIpSubnet
              - type: vlan
                vlan_id:
                  get_param: ExternalNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: ExternalIpSubnet
              - type: vlan
                vlan_id:
                  get_param: StorageNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: StorageIpSubnet
              - type: vlan
                vlan_id:
                  get_param: StorageMgmtNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: StorageMgmtIpSubnet
outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl
```

Contrail Controller

```
heat_template_version: queens
description: >
```

```
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role. This is an example for a Nova compute node using
  Contrail vrouter and the vhost0 interface.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal_api network
    type: string
  InternalApiDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the internal api network.
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage_mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
```

```
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 50
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  ExternalInterfaceDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the external network.
    type: string
  ManagementInterfaceDefaultRoute: # Commented out by default in this template
    default: unset
    description: The default route of the management network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
resources:
  OsNetConfigImpl:
```

```
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
              - type: interface
                name: nic1
                use_dhcp: false
                dns_servers:
                  get_param: DnsServers
                addresses:
                - ip_netmask:
                    list_join:
                      - '/'
                      - - get_param: ControlPlaneIp
                        - get_param: ControlPlaneSubnetCidr
                routes:
                - ip_netmask: 169.x.x.x/32
                  next_hop:
                    get_param: EC2MetadataIp
                - default: true
                  next_hop:
                    get_param: ControlPlaneDefaultRoute
              - type: vlan
                vlan_id:
                  get_param: InternalApiNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: InternalApiIpSubnet
              - type: interface
                name: nic2
                use_dhcp: false
                addresses:
                - ip_netmask:
                    get_param: TenantIpSubnet
outputs:
  OS::stack_id:
```

```
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl
```

Compute Node

```
heat_template_version: queens
description: >
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role. This is an example for a Nova compute node using
  Contrail vrouter and the vhost0 interface.
parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal_api network
    type: string
  InternalApiDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the internal api network.
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage_mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
```

```
    default: ''
    description: IP address/subnet on the management network
    type: string
  ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 50
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  ExternalInterfaceDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the external network.
    type: string
  ManagementInterfaceDefaultRoute: # Commented out by default in this template
    default: unset
    description: The default route of the management network.
    type: string
  DnsServers: # Override this via parameter_defaults
```

```
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
              - type: interface
                name: nic1
                use_dhcp: false
                dns_servers:
                  get_param: DnsServers
                addresses:
                - ip_netmask:
                    list_join:
                      - '/'
                      - - get_param: ControlPlaneIp
                        - get_param: ControlPlaneSubnetCidr
                routes:
                - ip_netmask: 169.x.x.x/32
                  next_hop:
                    get_param: EC2MetadataIp
                - default: true
                  next_hop:
                    get_param: ControlPlaneDefaultRoute
              - type: vlan
                vlan_id:
                  get_param: InternalApiNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
```

```
                    get_param: InternalApiIpSubnet
              - type: vlan
                vlan_id:
                  get_param: StorageNetworkVlanID
                device: nic1
                addresses:
                - ip_netmask:
                    get_param: StorageIpSubnet
              - type: contrail_vrouter
                name: vhost0
                use_dhcp: false
                members:
                  -
                    type: interface
                    name: nic2
                    use_dhcp: false
                addresses:
                - ip_netmask:
                    get_param: TenantIpSubnet

outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl
```

Advanced vRouter Kernel Mode Configuration

IN THIS SECTION

In addition to the standard NIC configuration, the vRouter kernel mode supports VLAN, Bond, and Bond + VLAN modes. The configuration snippets below only show the relevant section of the NIC template configuration for each mode.

VLAN

```
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: nic2
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
        str_replace:
          template: vlanVLANID
          params:
            VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Bond

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
   -
     type: interface
     name: nic2
   -
     type: interface
     name: nic3
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
```

```
      name: bond0
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Bond + VLAN

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
   -
     type: interface
     name: nic2
   -
     type: interface
     name: nic3
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: bond0
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
        str_replace:
          template: vlanVLANID
          params:
            VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Advanced vRouter DPDK Mode Configuration

IN THIS SECTION

In addition to the standard NIC configuration, the vRouter DPDK mode supports Standard, VLAN, Bond, and Bond + VLAN modes.

Network Environment Configuration:

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages:

```
parameter_defaults:
  ContrailDpdkHugepages1GB: 10
```

See the following NIC template configurations for vRouter DPDK mode. The configuration snippets below only show the relevant section of the NIC configuration for each mode.

Standard

```
- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  members:
    -
      type: interface
      name: nic2
      use_dhcp: false
  addresses:
```

```
  - ip_netmask:
      get_param: TenantIpSubnet
```

VLAN

```
 - type: contrail_vrouter_dpdk
             name: vhost0
             use_dhcp: false
             driver: uio_pci_generic
             cpu_list: 0x01
             vlan_id:
               get_param: TenantNetworkVlanID
             members:
               -
                 type: interface
                 name: nic2
                 use_dhcp: false
             addresses:
             - ip_netmask:
                 get_param: TenantIpSubnet
```

Bond

```
- type: contrail_vrouter_dpdk
             name: vhost0
             use_dhcp: false
             driver: uio_pci_generic
             cpu_list: 0x01
             bond_mode: 4
             bond_policy: layer2+3
             members:
               -
                 type: interface
                 name: nic2
                 use_dhcp: false
               -
                 type: interface
                 name: nic3
                 use_dhcp: false
             addresses:
```

```
            - ip_netmask:
                get_param: TenantIpSubnet
```

Bond + VLAN

```
- type: contrail_vrouter_dpdk
            name: vhost0
            use_dhcp: false
            driver: uio_pci_generic
            cpu_list: 0x01
            vlan_id:
              get_param: TenantNetworkVlanID
            bond_mode: 4
            bond_policy: layer2+3
            members:
              -
                type: interface
                name: nic2
                use_dhcp: false
              -
                type: interface
                name: nic3
                use_dhcp: false
            addresses:
            - ip_netmask:
                get_param: TenantIpSubnet
```

Advanced vRouter SRIOV + Kernel Mode Configuration

IN THIS SECTION

vRouter SRIOV + Kernel mode can be used in the following combinations:

- Standard
- VLAN
- Bond
- Bond + VLAN

Network environment configuration:

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages:

```
parameter_defaults:
  ContrailSriovHugepages1GB: 10
```

SRIOV PF/VF settings:

```
NovaPCIPassthrough:
- devname: "ens2f1"
  physical_network: "sriov1"
ContrailSriovNumVFs: ["ens2f1:7"]
```

The SRIOV NICs are not configured in the NIC templates. However, vRouter NICs must still be configured. See the following NIC template configurations for vRouter kernel mode. The configuration snippets below only show the relevant section of the NIC configuration for each mode.

VLAN

```
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: nic2
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
```

```
        str_replace:
          template: vlanVLANID
          params:
            VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Bond

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
  members:
   -
     type: interface
     name: nic2
   -
     type: interface
     name: nic3
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name: bond0
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Bond + VLAN

```
- type: linux_bond
  name: bond0
  bonding_options: "mode=4 xmit_hash_policy=layer2+3"
  use_dhcp: false
```

```
  members:
   -
     type: interface
     name: nic2
   -
     type: interface
     name: nic3
- type: vlan
  vlan_id:
    get_param: TenantNetworkVlanID
  device: bond0
- type: contrail_vrouter
  name: vhost0
  use_dhcp: false
  members:
    -
      type: interface
      name:
        str_replace:
          template: vlanVLANID
          params:
            VLANID: {get_param: TenantNetworkVlanID}
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

Advanced vRouter SRIOV + DPDK Mode Configuration

IN THIS SECTION

vRouter SRIOV + DPDK can be used in the following combinations:

- Standard
- VLAN
- Bond
- Bond + VLAN

Network environment configuration:

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

Enable the number of hugepages

```
parameter_defaults:
  ContrailSriovMode: dpdk
  ContrailDpdkHugepages1GB: 10
  ContrailSriovHugepages1GB: 10
```

SRIOV PF/VF settings

```
NovaPCIPassthrough:
- devname: "ens2f1"
  physical_network: "sriov1"
ContrailSriovNumVFs: ["ens2f1:7"]
```

The SRIOV NICs are not configured in the NIC templates. However, vRouter NICs must still be configured. See the following NIC template configurations for vRouter DPDK mode. The configuration snippets below only show the relevant section of the NIC configuration for each mode.

Standard

```
- type: contrail_vrouter_dpdk
  name: vhost0
  use_dhcp: false
  driver: uio_pci_generic
  cpu_list: 0x01
  members:
    -
      type: interface
      name: nic2
```

```
      use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: TenantIpSubnet
```

VLAN

```
 - type: contrail_vrouter_dpdk
              name: vhost0
              use_dhcp: false
              driver: uio_pci_generic
              cpu_list: 0x01
              vlan_id:
                get_param: TenantNetworkVlanID
              members:
                -
                  type: interface
                  name: nic2
                  use_dhcp: false
              addresses:
              - ip_netmask:
                  get_param: TenantIpSubnet
```

Bond

```
- type: contrail_vrouter_dpdk
              name: vhost0
              use_dhcp: false
              driver: uio_pci_generic
              cpu_list: 0x01
              bond_mode: 4
              bond_policy: layer2+3
              members:
                -
                  type: interface
                  name: nic2
                  use_dhcp: false
                -
                  type: interface
                  name: nic3
```

```
                use_dhcp: false
            addresses:
            - ip_netmask:
                get_param: TenantIpSubnet
```

Bond + VLAN

```
- type: contrail_vrouter_dpdk
            name: vhost0
            use_dhcp: false
            driver: uio_pci_generic
            cpu_list: 0x01
            vlan_id:
              get_param: TenantNetworkVlanID
            bond_mode: 4
            bond_policy: layer2+3
            members:
              -
                type: interface
                name: nic2
                use_dhcp: false
              -
                type: interface
                name: nic3
                use_dhcp: false
            addresses:
            - ip_netmask:
                get_param: TenantIpSubnet
```

Advanced Scenarios

Remote Compute

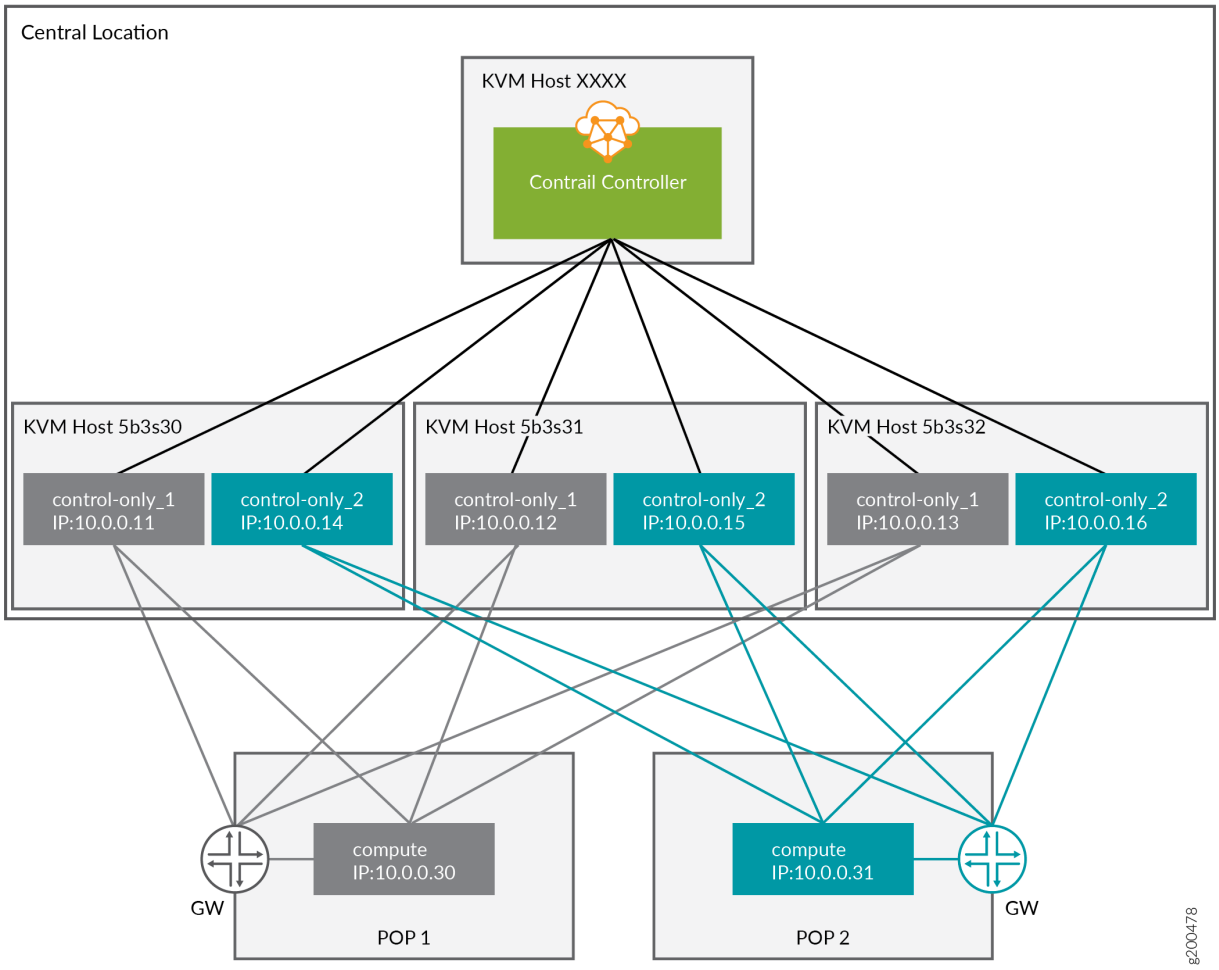
Remote Compute extends the data plane to remote locations (POP) whilest keeping the control plane central. Each POP will have its own set of Contrail control services, which are running in the central location. The difficulty is to ensure that the compute nodes of a given POP connect to the Control nodes assigned to that POC. The Control nodes must have predictable IP addresses and the compute nodes have to know these IP addresses. In order to achieve that the following methods are used:

- Custom Roles
- Static IP assignment

- Precise Node placement
- Per Node hieradata

Each overcloud node has a unique DMI UUID. This UUID is known on the undercloud node as well as on the overcloud node. Hence, this UUID can be used for mapping node specific information. For each POP, a Control role and a Compute role has to be created.

Overview



Mapping Table

Table 34: Mapping Table

Nova Name	Ironic Name	UUID	KVM	IP Address	POP
overcloud -contrailcontrolonly -0	control-only-1-5b3s30	Ironic UUID: 7d758dce-2784-45fd-be09-5a41eb53e764 DMI UUID: 73F8D030-E896-4A95-A9F5-E1A4FEBE322D	5b3s30	10.0.0.11	POP1
overcloud -contrailcontrolonly -1	control-only-2-5b3s30	Ironic UUID: d26abdeb-d514-4a37-a7fb-2cd2511c351f DMI UUID: 14639A66-D62C-4408-82EE-FDDC4E509687	5b3s30	10.0.0.14	POP2
overcloud -contrailcontrolonly -2	control-only-1-5b3s31	Ironic UUID: 91dd9fa9-e8eb-4b51-8b5e-bbaffb6640e4 DMI UUID: 28AB0B57-D612-431E-B177-1C578AE0FEA4	5b3s31	10.0.0.12	POP1
overcloud -contrailcontrolonly -3	control-only-2-5b3s31	Ironic UUID: 09fa57b8-580f-42ec-bf10-a19573521ed4 DMI UUID: 09BEC8CB-77E9-42A6-AFF4-6D4880FD87D0	5b3s31	10.0.0.15	POP2
overcloud -contrailcontrolonly -4	control-only-1-5b3s32	Ironic UUID: 4766799-24c8-4e3b-af54-353f2b796ca4 DMI UUID: 3993957A-ECBF-4520-9F49-0AF6EE1667A7	5b3s32	10.0.0.13	POP1

Table 34: Mapping Table *(Continued)*

Nova Name	Ironic Name	UUID	KVM	IP Address	POP
overcloud -contrailcontrolonly -5	control-only-2-5b3s32	Ironic UUID: 58a803ae-a785-470e-9789-139abbfa74fb DMI UUID: AF92F485-C30C-4D0A-BDC4-C6AE97D06A66	5b3s32	10.0.0.16	POP2

ControlOnly preparation

Add ControlOnly overcloud VMs to overcloud KVM host

NOTE: This has to be done on the overcloud KVM hosts

Two ControlOnly overcloud VM definitions will be created on each of the overcloud KVM hosts.

```
ROLES=control-only:2
num=4
ipmi_user=<user>
ipmi_password=<password>
libvirt_path=/var/lib/libvirt/images
port_group=overcloud
prov_switch=br0

/bin/rm ironic_list
IFS=',' read -ra role_list <<< "${ROLES}"
for role in ${role_list[@]}; do
  role_name=`echo $role|cut -d ":" -f 1`
  role_count=`echo $role|cut -d ":" -f 2`
  for count in `seq 1 ${role_count}`; do
    echo $role_name $count
    qemu-img create -f qcow2 ${libvirt_path}/${role_name}_${count}.qcow2 99G
    virsh define /dev/stdin <<EOF
 $(virt-install --name ${role_name}_${count} \
--disk ${libvirt_path}/${role_name}_${count}.qcow2 \
--vcpus=4 \
```

```
--ram=16348 \
--network network=br0,model=virtio,portgroup=${port_group} \
--network network=br1,model=virtio \
--virt-type kvm \
--cpu host \
--import \
--os-variant rhel7 \
--serial pty \
--console pty,target_type=virtio \
--graphics vnc \
--print-xml)
EOF
    vbmc add ${role_name}_${count} --port 1623${num} --username ${ipmi_user} --password $
{ipmi_password}
    vbmc start ${role_name}_${count}
    prov_mac=`virsh domiflist ${role_name}_${count}|grep ${prov_switch}|awk '{print $5}'`
    vm_name=${role_name}-${count}-`hostname -s`
    kvm_ip=`ip route get 1  |grep src |awk '{print $7}'`
    echo ${prov_mac} ${vm_name} ${kvm_ip} ${role_name} 1623${num}>> ironic_list
    num=$(expr $num + 1)
  done
done
```

NOTE: The generated *ironic_list* will be needed on the undercloud to import the nodes to Ironic.

Get the ironic_lists from the overcloud KVM hosts and combine them.

```
cat ironic_list_control_only
52:54:00:3a:2f:ca control-only-1-5b3s30 10.87.64.31 control-only 16234
52:54:00:31:4f:63 control-only-2-5b3s30 10.87.64.31 control-only 16235
52:54:00:0c:11:74 control-only-1-5b3s31 10.87.64.32 control-only 16234
52:54:00:56:ab:55 control-only-2-5b3s31 10.87.64.32 control-only 16235
52:54:00:c1:f0:9a control-only-1-5b3s32 10.87.64.33 control-only 16234
52:54:00:f3:ce:13 control-only-2-5b3s32 10.87.64.33 control-only 16235
```

Import:

```
ipmi_password=<password>
ipmi_user=<user>
```

```
DEPLOY_KERNEL=$(openstack image show bm-deploy-kernel -f value -c id)
DEPLOY_RAMDISK=$(openstack image show bm-deploy-ramdisk -f value -c id)

num=0
while IFS= read -r line; do
  mac=`echo $line|awk '{print $1}'`
  name=`echo $line|awk '{print $2}'`
  kvm_ip=`echo $line|awk '{print $3}'`
  profile=`echo $line|awk '{print $4}'`
  ipmi_port=`echo $line|awk '{print $5}'`
  uuid=`openstack baremetal node create --driver ipmi \
                                        --property cpus=4 \
                                        --property memory_mb=16348 \
                                        --property local_gb=100 \
                                        --property cpu_arch=x86_64 \
                                        --driver-info ipmi_username=${ipmi_user}  \
                                        --driver-info ipmi_address=${kvm_ip} \
                                        --driver-info ipmi_password=${ipmi_password} \
                                        --driver-info ipmi_port=${ipmi_port} \
                                        --name=${name} \
                                        --property capabilities=boot_option:local \
                                        -c uuid -f value`
  openstack baremetal node set ${uuid} --driver-info deploy_kernel=$DEPLOY_KERNEL --driver-info
deploy_ramdisk=$DEPLOY_RAMDISK
  openstack baremetal port create --node ${uuid} ${mac}
  openstack baremetal node manage ${uuid}
  num=$(expr $num + 1)
done < <(cat ironic_list_control_only)
```

ControlOnly node introspection

```
openstack overcloud node introspect --all-manageable --provide
```

Get the ironic UUID of the ControlOnly nodes

```
openstack baremetal node list |grep control-only
| 7d758dce-2784-45fd-be09-5a41eb53e764 | control-only-1-5b3s30  | None | power off | available |
False |
| d26abdeb-d514-4a37-a7fb-2cd2511c351f | control-only-2-5b3s30  | None | power off | available |
False |
```

```
| 91dd9fa9-e8eb-4b51-8b5e-bbaffb6640e4 | control-only-1-5b3s31  | None | power off | available |
False |
| 09fa57b8-580f-42ec-bf10-a19573521ed4 | control-only-2-5b3s31  | None | power off | available |
False |
| f4766799-24c8-4e3b-af54-353f2b796ca4 | control-only-1-5b3s32  | None | power off | available |
False |
| 58a803ae-a785-470e-9789-139abbfa74fb | control-only-2-5b3s32  | None | power off | available |
False |
```

The first ControlOnly node on each of the overcloud KVM hosts will be used for POP1, the second for POP2, and so and so forth.

Get the ironic UUID of the POP compute nodes:

```
openstack baremetal node list |grep compute
| 91d6026c-b9db-49cb-a685-99a63da5d81e | compute-3-5b3s30 | None | power off | available | False
|
| 8028eb8c-e1e6-4357-8fcf-0796778bd2f7 | compute-4-5b3s30 | None | power off | available | False
|
| b795b3b9-c4e3-4a76-90af-258d9336d9fb | compute-3-5b3s31 | None | power off | available | False
|
| 2d4be83e-6fcc-4761-86f2-c2615dd15074 | compute-4-5b3s31 | None | power off | available | False
|
```

The first two compute nodes belong to POP1 the second two compute nodes belong to POP2.

Create an input YAML using the ironic UUIDs:

```
 ~/subcluster_input.yaml
---
- subcluster: subcluster1
  asn: "65413"
  control_nodes:
    - uuid: 7d758dce-2784-45fd-be09-5a41eb53e764
      ipaddress: 10.0.0.11
    - uuid: 91dd9fa9-e8eb-4b51-8b5e-bbaffb6640e4
      ipaddress: 10.0.0.12
    - uuid: f4766799-24c8-4e3b-af54-353f2b796ca4
      ipaddress: 10.0.0.13
  compute_nodes:
    - uuid: 91d6026c-b9db-49cb-a685-99a63da5d81e
      vrouter_gateway: 10.0.0.1
```

```
    - uuid: 8028eb8c-e1e6-4357-8fcf-0796778bd2f7
      vrouter_gateway: 10.0.0.1
- subcluster: subcluster2
  asn: "65414"
  control_nodes:
    - uuid: d26abdeb-d514-4a37-a7fb-2cd2511c351f
      ipaddress: 10.0.0.14
    - uuid: 09fa57b8-580f-42ec-bf10-a19573521ed4
      ipaddress: 10.0.0.15
    - uuid: 58a803ae-a785-470e-9789-139abbfa74fb
      ipaddress: 10.0.0.16
  compute_nodes:
    - uuid: b795b3b9-c4e3-4a76-90af-258d9336d9fb
      vrouter_gateway: 10.0.0.1
    - uuid: 2d4be83e-6fcc-4761-86f2-c2615dd15074
      vrouter_gateway: 10.0.0.1
```

NOTE: Only control_nodes, compute_nodes, dpdk_nodes and sriov_nodes are supported.

Generate subcluster environment:

```
~/tripleo-heat-templates/tools/contrail/create_subcluster_environment.py -i ~/
subcluster_input.yaml \
               -o ~/tripleo-heat-templates/environments/contrail/contrail-subcluster.yaml
```

Check subcluster environment file:

```
cat ~/tripleo-heat-templates/environments/contrail/contrail-subcluster.yaml
parameter_defaults:
  NodeDataLookup:
    041D7B75-6581-41B3-886E-C06847B9C87E:
      contrail_settings:
        CONTROL_NODES: 10.0.0.14,10.0.0.15,10.0.0.16
        SUBCLUSTER: subcluster2
        VROUTER_GATEWAY: 10.0.0.1
    09BEC8CB-77E9-42A6-AFF4-6D4880FD87D0:
      contrail_settings:
        BGP_ASN: '65414'
        SUBCLUSTER: subcluster2
```

```
    14639A66-D62C-4408-82EE-FDDC4E509687:
      contrail_settings:
        BGP_ASN: '65414'
        SUBCLUSTER: subcluster2
    28AB0B57-D612-431E-B177-1C578AE0FEA4:
      contrail_settings:
        BGP_ASN: '65413'
        SUBCLUSTER: subcluster1
    3993957A-ECBF-4520-9F49-0AF6EE1667A7:
      contrail_settings:
        BGP_ASN: '65413'
        SUBCLUSTER: subcluster1
    73F8D030-E896-4A95-A9F5-E1A4FEBE322D:
      contrail_settings:
        BGP_ASN: '65413'
        SUBCLUSTER: subcluster1
    7933C2D8-E61E-4752-854E-B7B18A424971:
      contrail_settings:
        CONTROL_NODES: 10.0.0.14,10.0.0.15,10.0.0.16
        SUBCLUSTER: subcluster2
        VROUTER_GATEWAY: 10.0.0.1
    AF92F485-C30C-4D0A-BDC4-C6AE97D06A66:
      contrail_settings:
        BGP_ASN: '65414'
        SUBCLUSTER: subcluster2
    BB9E9D00-57D1-410B-8B19-17A0DA581044:
      contrail_settings:
        CONTROL_NODES: 10.0.0.11,10.0.0.12,10.0.0.13
        SUBCLUSTER: subcluster1
        VROUTER_GATEWAY: 10.0.0.1
    E1A809DE-FDB2-4EB2-A91F-1B3F75B99510:
      contrail_settings:
        CONTROL_NODES: 10.0.0.11,10.0.0.12,10.0.0.13
        SUBCLUSTER: subcluster1
        VROUTER_GATEWAY: 10.0.0.1
```

Add contrail-subcluster.yaml, contrail-ips-from-pool-all.yaml and contrail-scheduler-hints.yaml to the OpenStack deploy command:

```
openstack overcloud deploy --templates ~/tripleo-heat-templates \
 -e ~/overcloud_images.yaml \
 -e ~/tripleo-heat-templates/environments/network-isolation.yaml \
 -e ~/tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
 -e ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
 -e ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
 -e ~/tripleo-heat-templates/environments/contrail/contrail-subcluster.yaml \
 -e ~/tripleo-heat-templates/environments/contrail/contrail-ips-from-pool-all.yaml \
 -e ~/tripleo-heat-templates/environments/contrail/contrail-scheduler-hints.yaml \
 --roles-file ~/tripleo-heat-templates/roles_data_contrail_aio.yaml
```

Installing Overcloud

1. Deployment:

```
openstack overcloud deploy --templates ~/tripleo-heat-templates \
-e ~/overcloud_images.yaml \
-e ~/tripleo-heat-templates/environments/network-isolation.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
--roles-file ~/tripleo-heat-templates/roles_data_contrail_aio.yaml
```

2. Validation Test:

```
source overcloudrc
curl -O http://download.cirros-cloud.net/0.3.5/cirros-0.3.5-x86_64-disk.img
openstack image create --container-format bare --disk-format qcow2 --file cirros-0.3.5-x86_64-
disk.img cirros
openstack flavor create --public cirros --id auto --ram 64 --disk 0 --vcpus 1
openstack network create net1
openstack subnet create --subnet-range 1.0.0.0/24 --network net1 sn1
nova boot --image cirros --flavor cirros --nic net-id=`openstack network show net1 -c id -f
value` --availability-zone nova:overcloud-novacompute-0.localdomain c1
nova list
```

RELATED DOCUMENTATION

Installing a Nested Red Hat OpenShift Container Platform 3.11 Cluster Using Contrail Ansible Deployer

Using Netronome SmartNIC vRouter with Contrail Networking

NOTE: The Netronome SmartNIC vRouter technology covered in this document is available for evaluation purposes only. It is not intended for deployment in production networks.

Contrail supports Netronome Agilio CX SmartNICs for Contrail Networking deployment with Red Hat OpenStack Platform Director (RHOSPd) 13 environment.

This feature will enable service providers to improve the forwarding performance which includes packets per second (PPS) of vRouter. This will optimize server CPU usage and you can deploy more Virtual network functions (VNFs) per server.

Benefits:

- Increased PPS capacity of Contrail vRouter datapath allowing applications to reach their full processing capacity.
- Reclaimed CPU cores from Contrail vRouter off-loading allowing more VMs and VNFs to be deployed per server.

The goal of this topic is to provide a procedure for deploying accelerated vRouter compute nodes.

Before you begin:

- Equip compute nodes with Netronome Agilio CX SmartNIC.

  For details, refer to Agilio CX SmartNICs.

- Retrieve *Agilio heat-template plugin*.

  Register on Netronome support site at https://help.netronome.com and provide Docker Hub credentials.

  Netronome will provide the TripleO templates for SmartNIC vRouter deployment on compute nodes. Also, Netronome will authorize Docker Hub registry access.

  For details, refer to Netronome Agilio vRouter 19*xx* deployment guide.

- Note the following version tags:

```
AGILIO_TAG="2.38-rhel-queens FORWARDER_TAG="2.38-rhel-queens
```

Procedure:

NOTE: If you have multiple undercloud nodes deployed, you must perform the following procedure on the same node.

1. Configure *Agilio* plugin.

   For details, refer to Netronome agilio-ovs-openstack-plugin GitHub Repository.

   a. Extract the *Agilio* plugin archive and copy the **agilio-plugin** folder into the **tripleo-heat-templates** directory.

   ```
   [stack@queensa ~]$ tar -xzvf rhosp-contrail-agilio-heat-plugin-5-34.tgz agilio-plugin/ agilio-plugin/
   agilio-vrouter.yaml agilio-plugin/agilio_upgrade.sh agilio-plugin/deploy_rhosp.sh agilio-plugin/nfp_udev.sh
   agilio-plugin/agilio-env.yaml agilio-plugin/version agilio-plugin/README.md [stack@queensa ~]$ cp -r
   agilio-plugin/ tripleo-heat-templates/
   ```

   b. Navigate to the **agilio-plugin** directory on the undercloud node.

   ```
   [tripleo-heat-templates]$ cd agilio-plugin/
   ```

   c. Modify **agilio-env.yaml** file as per your environment.

   NOTE: Reserve at least 1375*2 MB hugepages for *virtio-forwarder*.

   Sample **agilio-env.yaml** file:

```
resource_registry:
  OS::TripleO::NodeExtraConfigPost: agilio-vrouter.yaml

parameter_defaults:
  # Hugepages
  ContrailVrouterHugepages2MB: "8192"
  # IOMMU
  ComputeParameters:
    KernelArgs: "intel_iommu=on iommu=pt isolcpus=1,2 "

  ComputeCount: 3

  # Aditional config
  ControlPlaneDefaultRoute: 10.0.x.1
  EC2MetadataIp: 10.0.x.1  # Generally the IP of the Undercloud
```

```
  DnsServers: ["8.8.8.8","192.168.3.3"]
  NtpServer: ntp.is.co.za
  ContrailRegistryInsecure: true
  DockerInsecureRegistryAddress: 172.x.x.150:6666,10.0.x.1:8787
  ContrailRegistry: 172.x.x.150:6666
  ContrailImageTag: <container_tag>-rhel-queens

# Fix DB Diskspace too low issue
  ContrailAnalyticsDBMinDiskGB: 40
```

d. Add Docker Hub credentials to **tripleo-heat-templates/agilio-plugin/agililo_upgrade.sh** file to retrieve containers from **AGILIO_REPO="docker.io/netronomesystems/"** repository.

```
#GENERAL DOCKER CONFIG DOCKER_USR=****** #ENTER_DOCKER_USERNAME_HERE DOCKER_PASS=******
#ENTER_DOCKER_PASSWORD_HERE
```

```
[root@overcloud-novacompute-2 heat-admin]# docker ps -a | grep virtio_for 7d5af8a2591d docker.io/
netronomesystems/virtio-forwarder:2.38-rhel-queens "./entrypoint.sh" 30 seconds ago Up 15 seconds
virtio_forwarder
```

```
[root@overcloud-novacompute-2 heat-admin]# docker ps -a | grep agilio c7c611b5168b docker.io/
netronomesystems/agilio-vrouter:2.38-rhel-queens "./entrypoint.sh" 46 seconds ago Up 38 seconds
agilio_vrouter
```

2. Prepare the Contrail Networking cluster for deployment.

   Refer to the following topics for deployment:

   - "Understanding Red Hat OpenStack Platform Director" on page 448
   - "Setting Up the Infrastructure" on page 453
   - "Setting Up the Undercloud" on page 463
   - "Setting Up the Overcloud" on page 466

   NOTE: Do not perform steps for "Installing Overcloud" on page 509.

3. Deploy the cluster by one of the following ways:

   - Add `agilio-env.yaml` to installing overcloud step as mentioned in "Installing Overcloud" on page 509 topic.

     ```
     openstack overcloud deploy --templates ~/tripleo-heat-templates -e ~/overcloud_images.yaml -e ~/tripleo-
     heat-templates/environments/network-isolation.yaml -e ~/tripleo-heat-templates/environments/contrail/
     contrail-plugins.yaml -e ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml -e ~/
     ```

```
tripleo-heat-templates/environments/contrail/contrail-net.yaml -e ~/tripleo-heat-templates/agilio-plugin/
agilio-env.yaml --roles-file ~/tripleo-heat-templates/roles_data_contrail_aio.yaml
```

Or

- Run the following command:

```
deploy_rhosp.sh

-e ~/tripleo-heat-templates/agilio-plugin/agilio-env.yaml
```

On completing above steps successfully, refer to Netronome agilio-ovs-openstack-plugin GitHub Repository on how to spin up the accelerated VMs.

RELATED DOCUMENTATION

Understanding Red Hat OpenStack Platform Director | 448

Setting Up the Infrastructure | 453

Setting Up the Undercloud | 463

Setting Up the Overcloud | 466

Installing OpenStack Octavia LBaaS with RHOSP in Contrail Networking

Contrail Networking Release 2005 supports Octavia as LBaaS. The deployment supports RHOSP and Juju platforms.

With Octavia as LBaaS, Contrail Networking is only maintaining network connectivity and is not involved in any load balancing functions.

For each OpenStack load balancer creation, Octavia launches a VM known as *amphora VM*. The VM starts the HAPROXY when listener is created for the load balancer in OpenStack. Whenever the load balancer gets updated in OpenStack, *amphora VM* updates the running HAPROXY configuration. The *amphora VM* is deleted on deleting the load balancer.

Contrail Networking provides connectivity to *amphora VM* interfaces. *Amphora VM* has two interfaces; one for management and the other for data. The management interface is used by the Octavia services for the management communication. Since, Octavia services are running in the underlay network and *amphora VM* is running in the overlay network, SDN gateway is needed to reach the overlay network. The data interface is used for load balancing.

Follow the procedure to install OpenStack Octavia LBaaS with Contrail Networking:

1. Deploy RHOSP13 with Contrail Networking without Octavia.

```
openstack overcloud deploy --templates tripleo-heat-templates/ \
--roles-file tripleo-heat-templates/roles_data_contrail_aio.yaml \
-e environment-rhel-registration.yaml \
-e tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/rhel-registration-resource-registry.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-net-single.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
-e misc_opts.yaml \
-e contrail-parameters.yaml \
-e docker_registry.yaml
```

2. Make a copy of **tripleo-heat-templates/docker/services/octavia/octavia-deployment-config.yaml** file.

```
cp tripleo-heat-templates/docker/services/octavia/octavia-deployment-config.yaml tripleo-heat-templates/docker/services/octavia/octavia-deployment-config.bak
```

3. Make the following changes in *generate_certs* section of the **tripleo-heat-templates/docker/services/octavia/octavia-deployment-config.yaml** file.

```
conditions:

  generate_certs:
      and:
      - get_param: OctaviaGenerateCerts
      - or:
        - equals:
          - get_param: StackAction
          - CREATE
        - equals:
          - get_param: StackAction
          - UPDATE
```

4. Deploy RHOSP13 with Octavia services.

```
openstack overcloud deploy --templates tripleo-heat-templates/ \ --roles-file tripleo-heat-templates/roles_data_contrail_aio.yaml \
```

```
-e environment-rhel-registration.yaml \
-e tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/rhel-registration-resource-
registry.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-net-single.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
-e tripleo-heat-templates/environments/services/octavia.yaml \
-e misc_opts.yaml \
-e contrail-parameters.yaml \
-e docker_registry.yaml
```

5. Rollback changes in **tripleo-heat-templates/docker/services/octavia/octavia-deployment-config.yaml** file.

```
cp tripleo-heat-templates/docker/services/octavia/octavia-deployment-config.bak tripleo-heat-
templates/docker/services/octavia/octavia-deployment-config.yaml
```

Here is an example for creating and testing load balancer:

Prerequisites:

- You must have connectivity between Octavia controller and amphora instances,
- You must have OpenStack services into LXD containers.
- You must have separate interfaces for control plane and data plane.

1. Create private network.

```
openstack network create private
openstack subnet create private --network private --subnet-range 10.10.10.0/24 --allocation-
pool
start=10.10.10.50,end=10.10.10.70 --gateway none
```

2. Create security group.

```
openstack security group create allow_all
openstack security group rule create --ingress --protocol any --prefix '0.0.0.0/0' allow_all
```

3. Check available flavors and images. You can create them, if needed.

```
openstack flavor list
openstack image list
```

4. Create two servers for load balancer.

```
openstack server create --flavor test_flavor --image cirros --security-group allow_all --
network private cirros1
openstack server create --flavor test_flavor --image cirros --security-group allow_all --
network private cirros2
```

5. Create additional server to test load balancer.

```
openstack server create --flavor test_flavor --image cirros --security-group allow_all --
network private cirros-test
```

6. Check status and IP addresses.

```
openstack server list --long
```

7. Create simple HTTP server on every cirros. Login on both the cirros instances and run following commands:

```
MYIP=$(ifconfig eth0|grep 'inet addr'|awk -F: '{print $2}'| awk '{print $1}') while true;
do echo -e "HTTP/1.0 200 OK\r\n\r\nWelcome to $MYIP" | sudo nc -l -p 80 ; done&
```

8. Create load balancer

```
openstack loadbalancer create --name lb1 --vip-subnet-id private
```

Make sure *provisioning_status* is *Active*.

```
openstack loadbalancer show lb1
```

9. Setup load balancer

```
openstack loadbalancer listener create --protocol HTTP --protocol-port 80 --name listener1
lb1
openstack loadbalancer show lb1  # Wait for the provisioning_status to be ACTIVE.
openstack loadbalancer pool create --lb-algorithm ROUND_ROBIN --listener listener1 --
protocol HTTP --name pool1
openstack loadbalancer healthmonitor create --delay 5 --timeout 2 --max-retries 1 --type
HTTP pool1
openstack loadbalancer member create --subnet-id private --address 10.10.10.50 --protocol-
port 80 pool1
openstack loadbalancer member create --subnet-id private --address 10.10.10.51 --protocol-
port 80 pool1
```

IP addresses 10.10.10.50 and 10.10.10.51 belong to VMs created with test http server in step "7" on page 516.

10. Check the status of load balancer.

```
openstack loadbalancer show lb1  # Wait for the provisioning_status to be ACTIVE.
openstack loadbalancer pool list
openstack loadbalancer pool show pool1
openstack loadbalancer member list pool1
openstack loadbalancer listener list
```

11. Login to load balancer client and verify if round robin works.

```
cirros@169.x.0.9's password:
$ curl 10.10.10.50
Welcome to 10.10.10.52
$ curl 10.10.10.50
Welcome to 10.10.10.53
$ curl 10.10.10.50
Welcome to 10.10.10.52
$ curl 10.10.10.50
Welcome to 10.10.10.53
$ curl 10.10.10.50
Welcome to 10.10.10.52
$ curl 10.10.10.50
Welcome to 10.10.10.53
```

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
2005	Contrail Networking Release 2005 supports Octavia as LBaaS.

RELATED DOCUMENTATION

- Support for OpenStack LBaaS | **564**
- Using Load Balancers in Contrail | **550**
- Installing OpenStack Octavia LBaaS with Juju Charms in Contrail Networking | **670**

CHAPTER 10

Configuring Virtual Networks

IN THIS CHAPTER

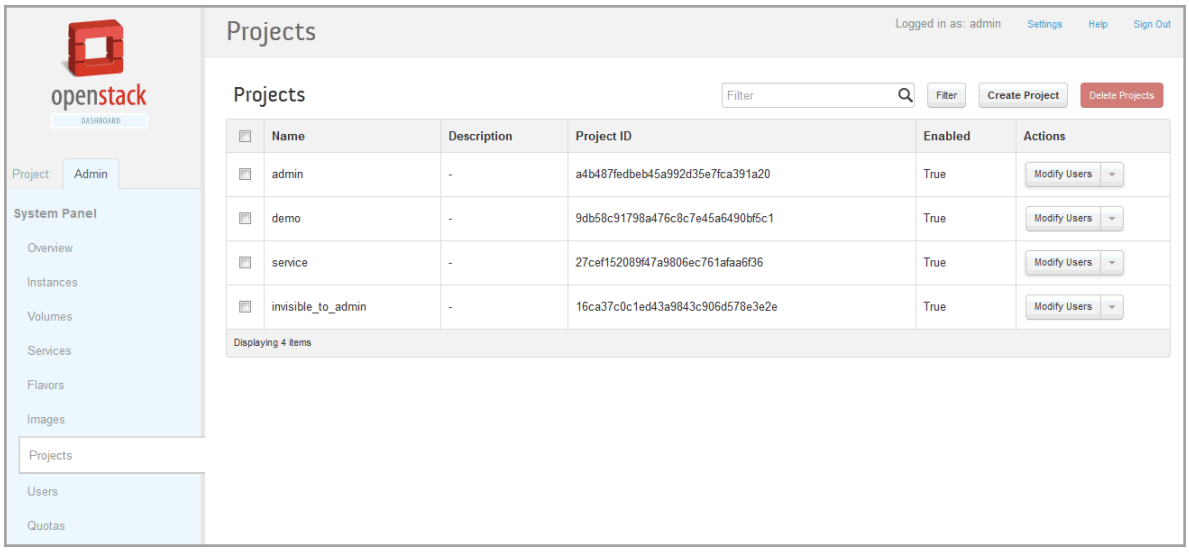
Creating Projects in OpenStack for Configuring Tenants in Contrail

In Contrail, a tenant configuration is called a project. A project is created for each set of virtual machines (VMs) and virtual networks (VNs) that are configured as a discrete entity for the tenant.

Projects are created, managed, and edited at the OpenStack **Projects** page.

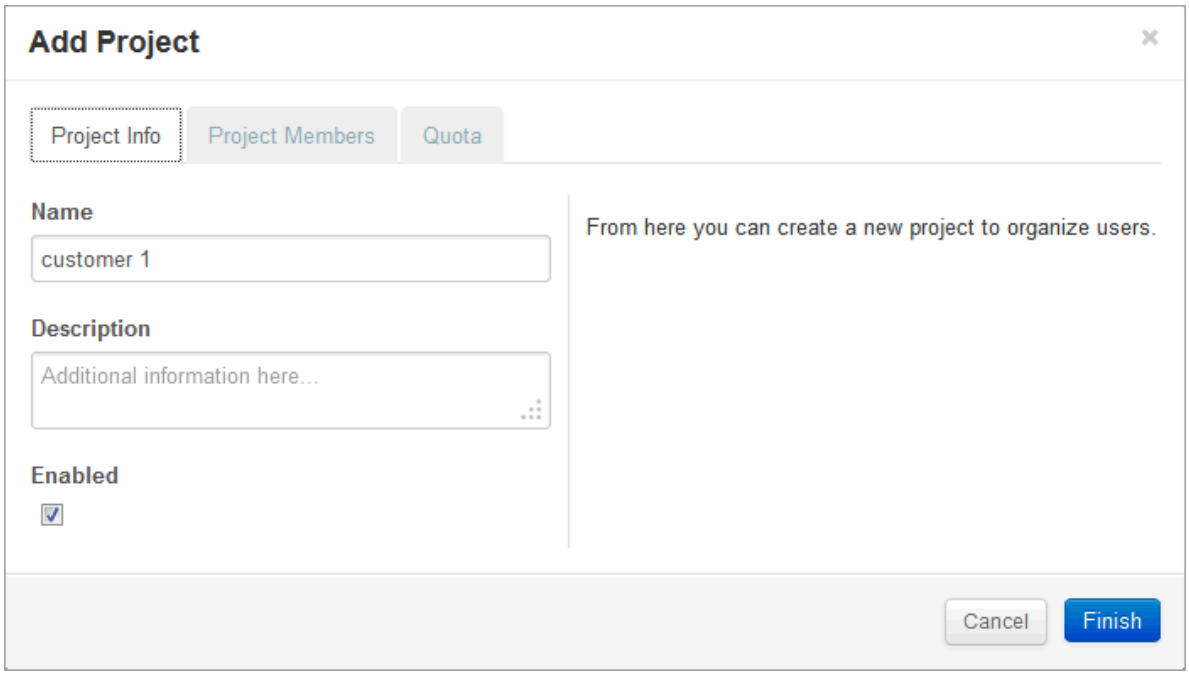
1. Click the **Admin** tab on the OpenStack dashboard, then click the **Projects** link to access the **Projects** page; see Figure 45 on page 520.

Figure 45: OpenStack Projects



2. In the upper right, click the **Create Project** button to access the **Add Project** window; see Figure 46 on page 520.

Figure 46: Add Project



3. In the **Add Project** window, on the **Project Info** tab, enter a **Name** and a **Description** for the new project, and select the **Enabled** check box to activate this project.

4. In the **Add Project** window, select the **Project Members** tab, and assign users to this project. Designate each user as **admin** or as **Member**.

   As a general rule, one person should be a super user in the **admin** role for all projects and a user with a **Member** role should be used for general configuration purposes.

5. Click **Finish** to create the project.

Refer to OpenStack documentation for more information about creating and managing projects.

RELATED DOCUMENTATION

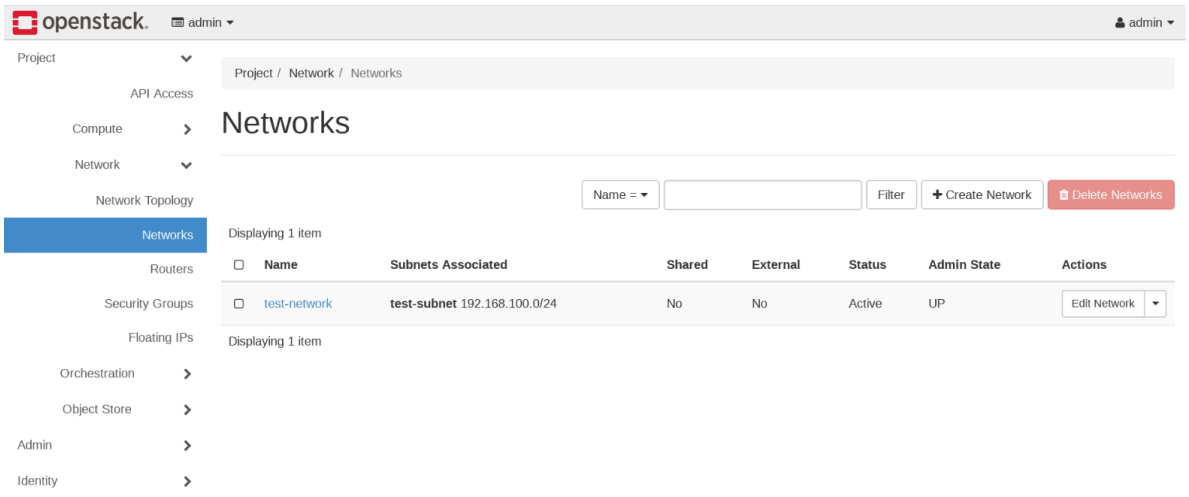
> *Creating a Virtual Network with Juniper Networks Contrail*
>
> Creating a Virtual Network with OpenStack Contrail | **521**
>
> OpenStack documentation

Creating a Virtual Network with OpenStack Contrail

You can create virtual networks in Contrail Networking from the OpenStack. The following procedure shows how to create a virtual network when using OpenStack.

1. To create a virtual network when using OpenStack Contrail, select **Project > Network > Networks**. The **Networks** page is displayed. See Figure 47 on page 522.

Figure 47: Networks Page



2. Click **Create Network**. The **Create Network** window is displayed. See Figure 48 on page 522 and Figure 49 on page 523.

Figure 48: Create Networks

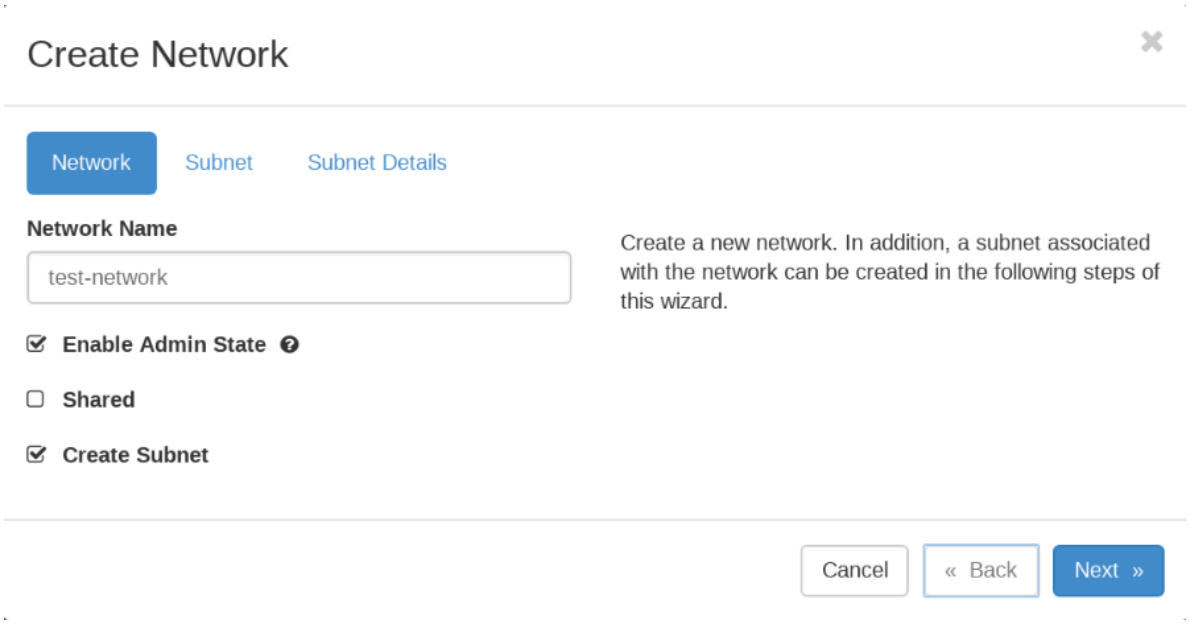
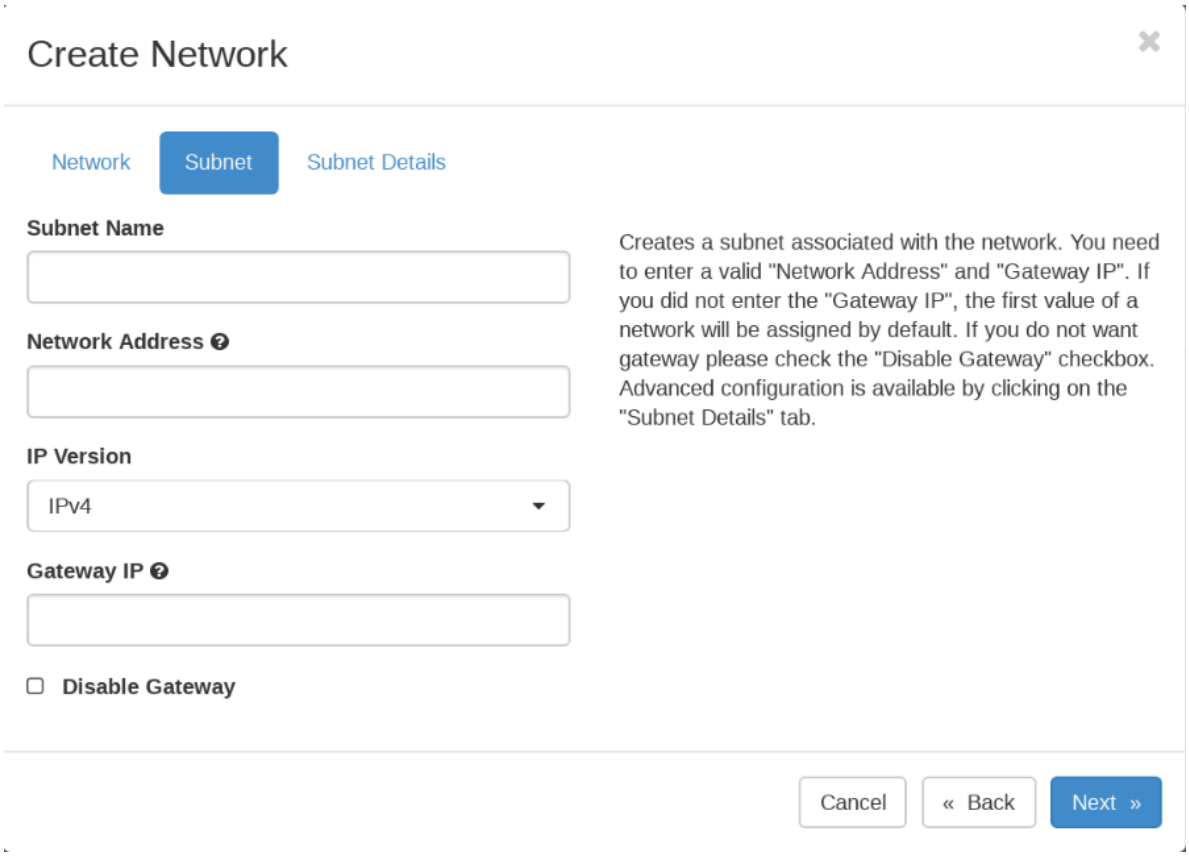


Figure 49: Subnet and Gateway Details



3. Click the **Network** and **Subnet** tabs to complete the fields in the **Create Network** window. See field descriptions in Table 35 on page 523.

Table 35: Create Network Fields

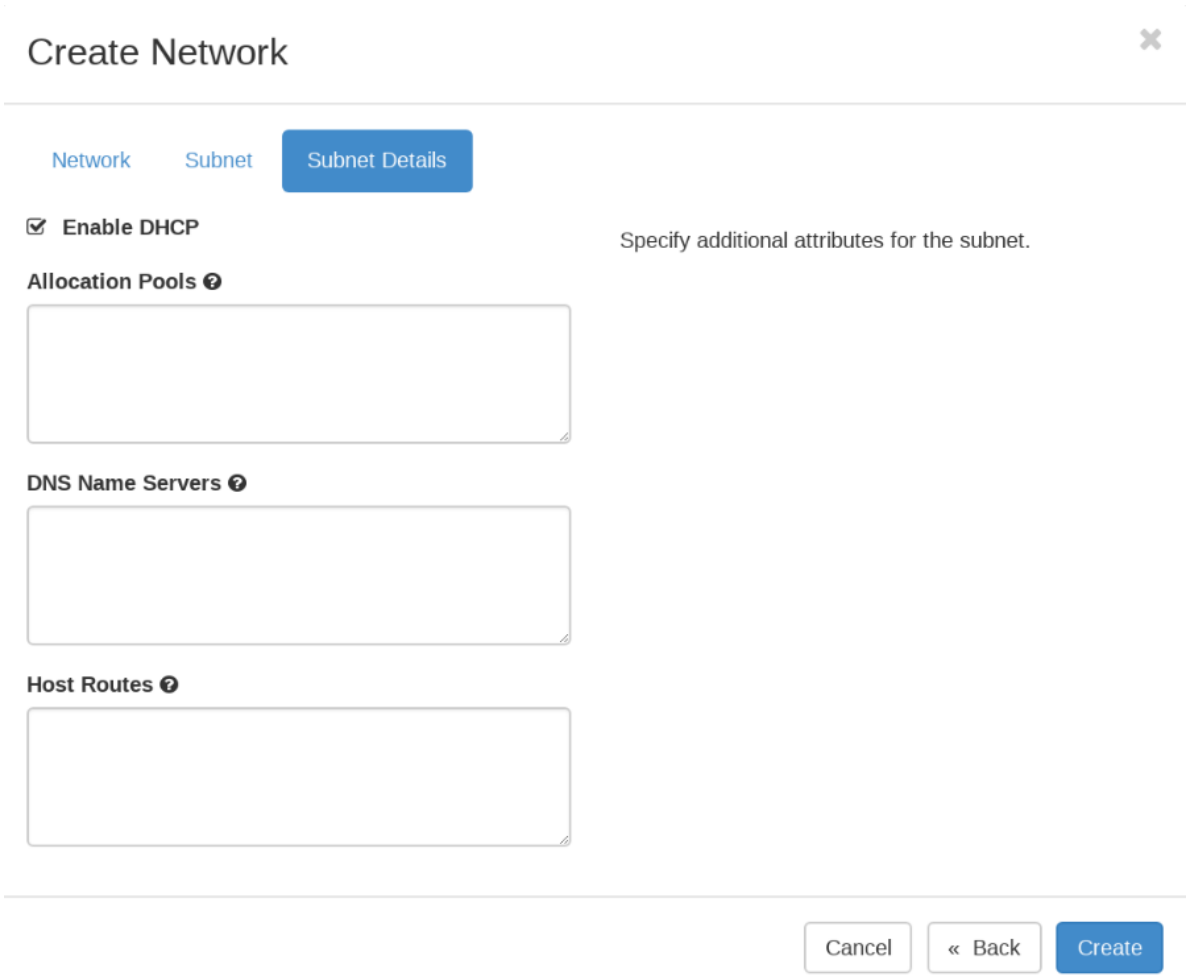
Field	Description
Network Name	Enter a name for the network.
Subnet Name	Enter a name for the subnetwork.
Network Address	Enter the network address in CIDR format.
IP Version*	Select IPv4 or IPv6.

Table 35: Create Network Fields *(Continued)*

Field	Description
Gateway IP	Optionally, enter an explicit gateway IP address for the IP address block. Check the Disable Gateway box if no gateway is to be used.

4. Click the **Subnet Details** tab to specify the Allocation Pool, DNS Name Servers, and Host Routes.

Figure 50: Additional Subnet Attributes



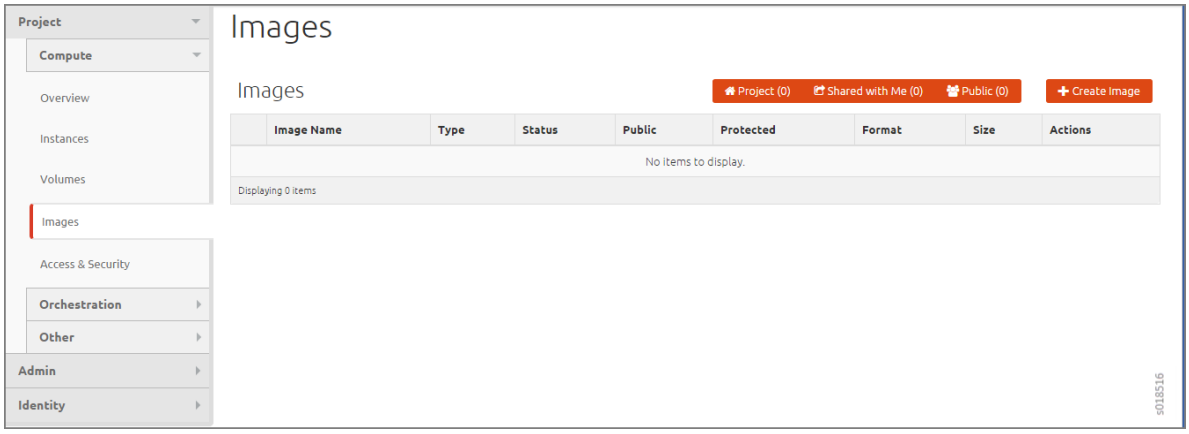
5. To save your network, click **Create** , or click **Cancel** to discard your work and start over.

Creating an Image for a Project in OpenStack Contrail

To specify an image to upload to the Image Service for a project in your system by using the OpenStack dashboard:

1. In OpenStack, select **Project > Compute > Images**. The Images window is displayed. See Figure 51 on page 525.

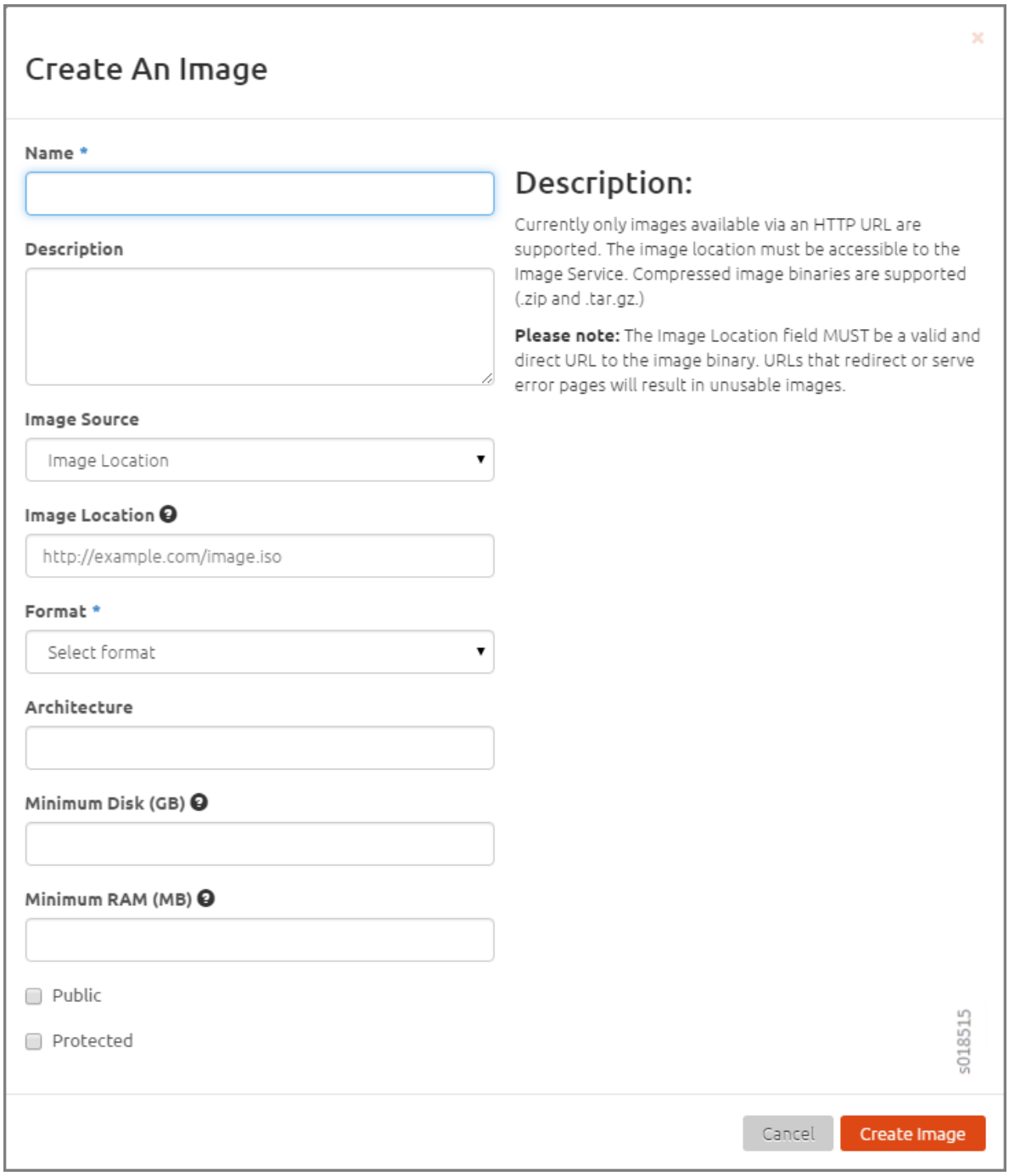
Figure 51: OpenStack Images Window



2. Make sure you have selected the correct project to which you are associating an image.
3. Click **Create Image**.

   The **Create An Image** window is displayed. See Figure 52 on page 526.

Figure 52: OpenStack Create An Image Window



4. Complete the fields to specify your image. Table 36 on page 527 describes each of the fields on the window.

> **NOTE**: Only images available through an HTTP URL are supported, and the image location must be accessible to the Image Service. Compressed image binaries are supported (***.zip** and ***.tar.gz**).

Table 36: Create an Image Fields

Field	Description
Name	Enter a name for this image.
Description	Enter a description for the image.
Image Source	Select **Image File** or **Image Location**. If you select **Image File**, you are prompted to browse to the local location of the file.
Image Location	Enter an external HTTP URL from which to load the image. The URL must be a valid and direct URL to the image binary. URLs that redirect or serve error pages result in unusable images.
Format	Required field. Select the format of the image from a list: AKI– Amazon Kernel Image AMI– Amazon Machine Image ARI– Amazon Ramdisk Image ISO– Optical Disk Image QCOW2– QEMU Emulator Raw– An unstructured image format VDI– Virtual Disk Imade VHD– Virtual Hard Disk VMDK– Virtual Machine Disk
Architecture	Enter the architecture.
Minimum Disk (GB)	Enter the minimum disk size required to boot the image. If you do not specify a size, the default is 0 (no minimum).

Table 36: Create an Image Fields *(Continued)*

Field	Description
Minimum Ram (MB)	Enter the minimum RAM required to boot the image. If you do not specify a size, the default is 0 (no minimum).
Public	Select this check box if this is a public image. Leave unselected for a private image.
Protected	Select this check box for a protected image.

5. When you are finished, click **Create Image**.

Using Security Groups with Virtual Machines (Instances)

IN THIS SECTION

Security Groups Overview

A **security group** is a container for security group rules. Security groups and security group rules allow administrators to specify the type of traffic that is allowed to pass through a port. When a virtual machine (VM) is created in a virtual network (VN), a security group can be associated with the VM when it is launched. If a security group is not specified, a port is associated with a default security group. The default security group allows both ingress and egress traffic. Security rules can be added to the default security group to change the traffic behavior.

Creating Security Groups and Adding Rules

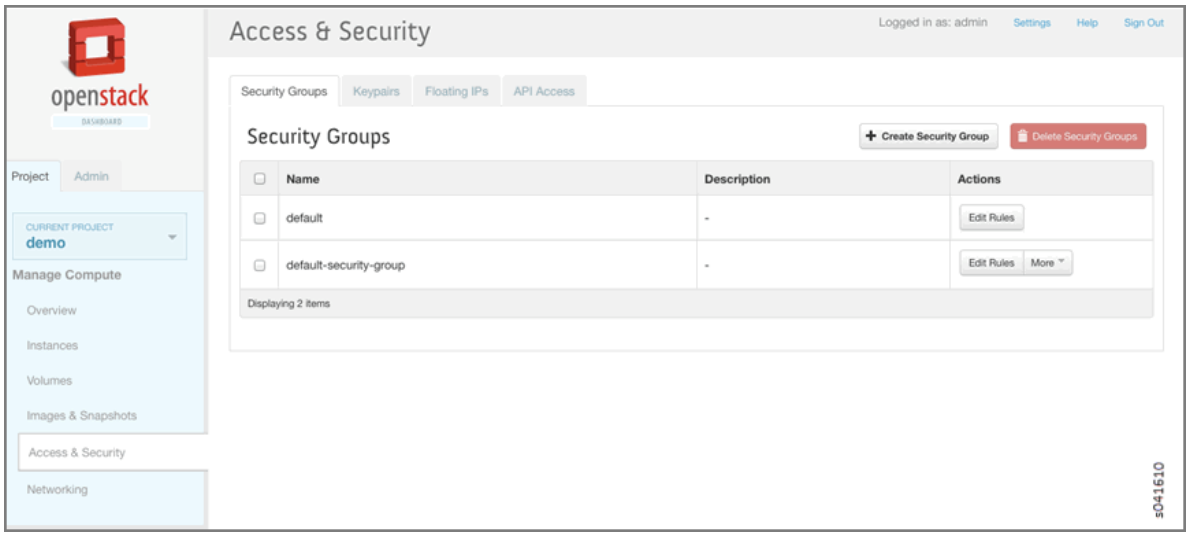
A default security group is created for each project. You can add security rules to the default security group and you can create additional security groups and add rules to them. The security groups are then associated with a VM, when the VM is launched or at a later date.

To add rules to a security group:

1. From the OpenStack interface, click the **Project** tab, select **Access & Security**, and click the **Security Groups** tab.

   Any existing security groups are listed under the **Security Groups** tab, including the default security group; see Figure 53 on page 529.

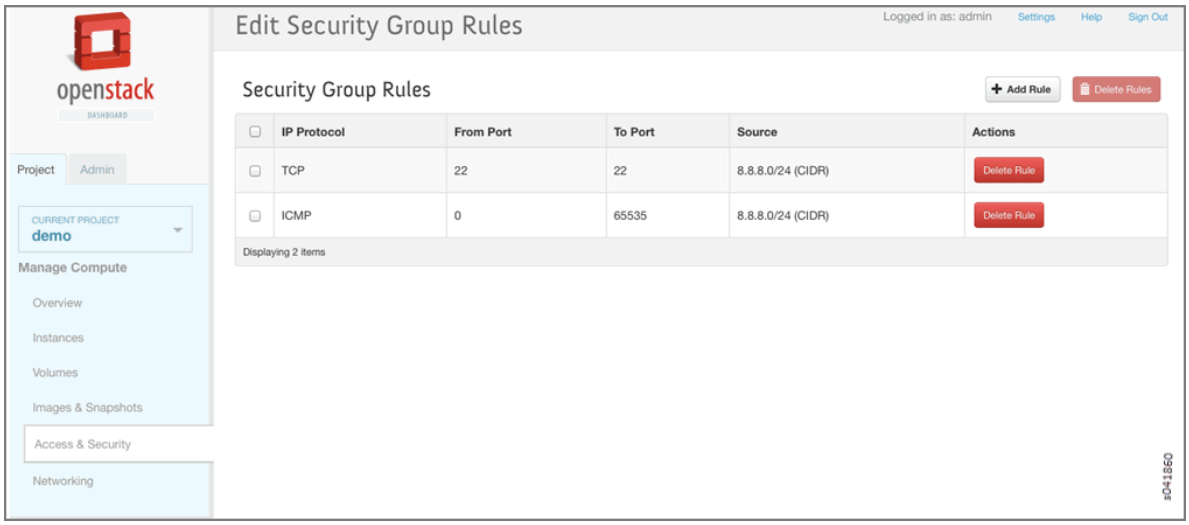
Figure 53: Security Groups



2. Select the **default-security-group** and click **Edit Rules** in the **Actions** column.

   The **Edit Security Group Rules** window is displayed; see Figure 54 on page 530. Any rules already associated with the security group are listed.

Figure 54: Edit Security Group Rules



3. Click **Add Rule** to add a new rule; see Figure 55 on page 531.

Figure 55: Add Rule

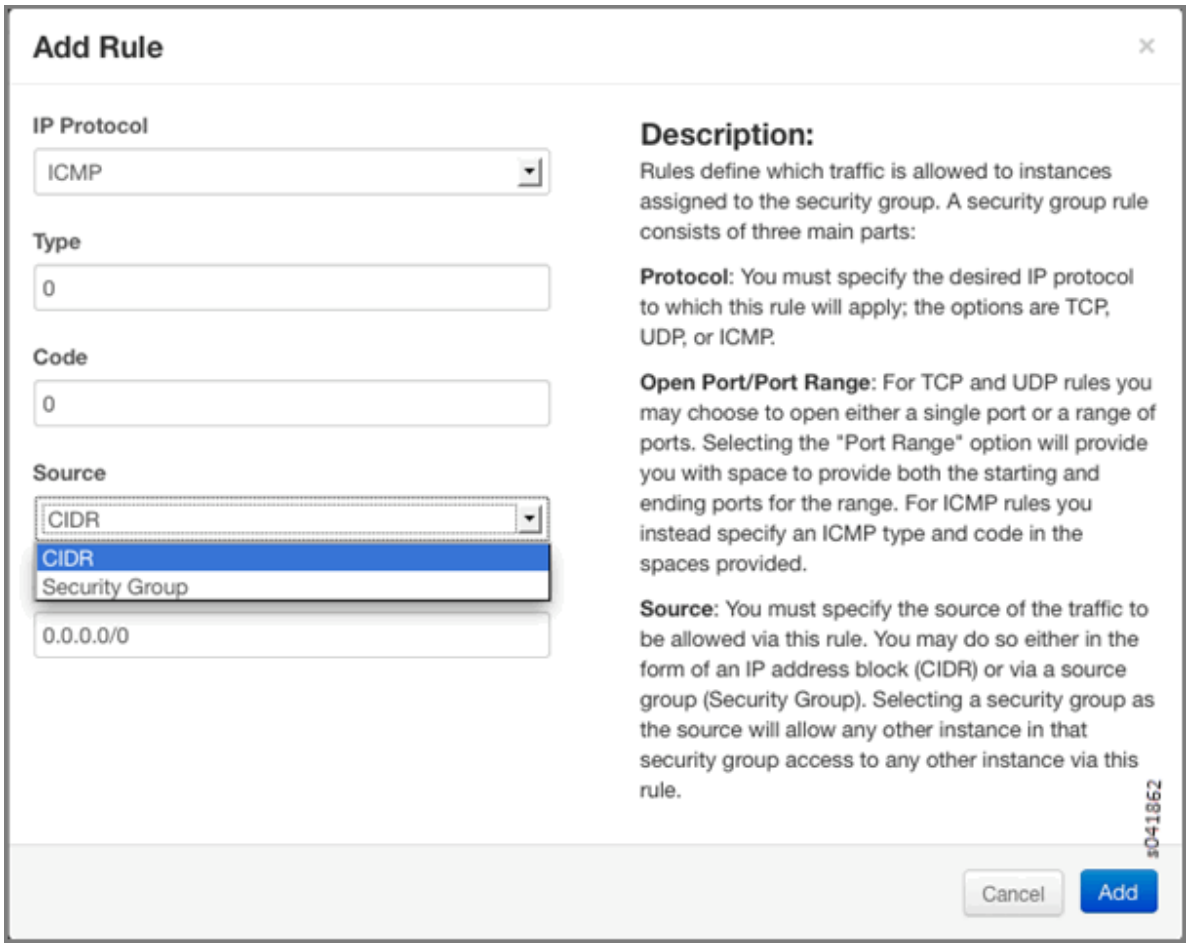


Table 37: Add Rule Fields

Column	Description
IP Protocol	Select the IP protocol to apply for this rule: TCP, UDP, ICMP.
From Port	Select the port from which traffic originates to apply this rule. For TCP and UDP, enter a single port or a range of ports. For ICMP rules, enter an ICMP type code.
To Port	The port to which traffic is destined that applies to this rule, using the same options as in the **From Port** field.

Table 37: Add Rule Fields *(Continued)*

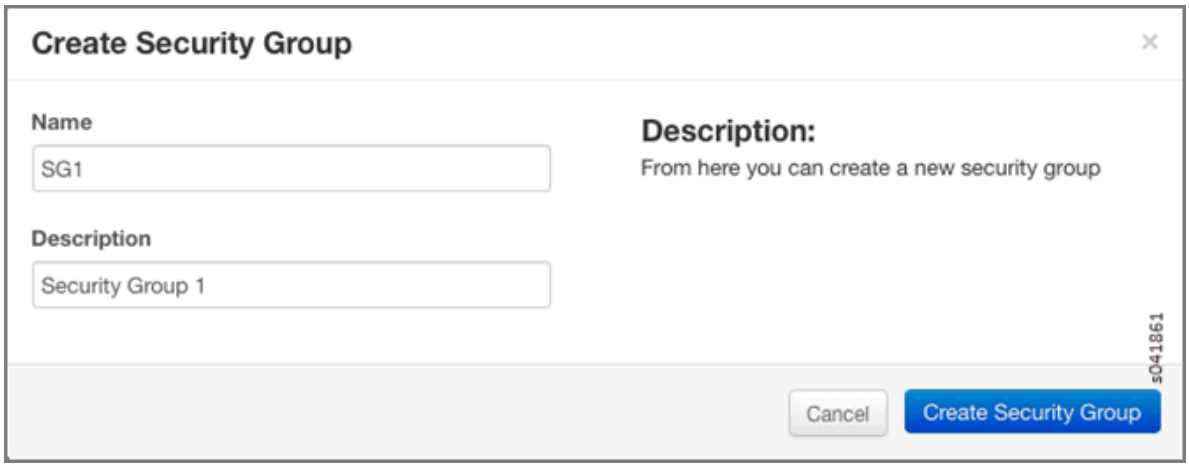
Column	Description
Source	Select the source of traffic to be allowed by this rule. Specify subnet—the CIDR IP address or address block of the inter-domain source of the traffic that applies to this rule, or you can choose security group as source. Selecting security group as source allows any other instance in that security group access to any other instance via this rule.

4. Click **Create Security Group** to create additional security groups.

   The **Create Security Group** window is displayed; see Figure 56 on page 532.

   Each new security group has a unique 32-bit security group ID and an ACL is associated with the configured rules.

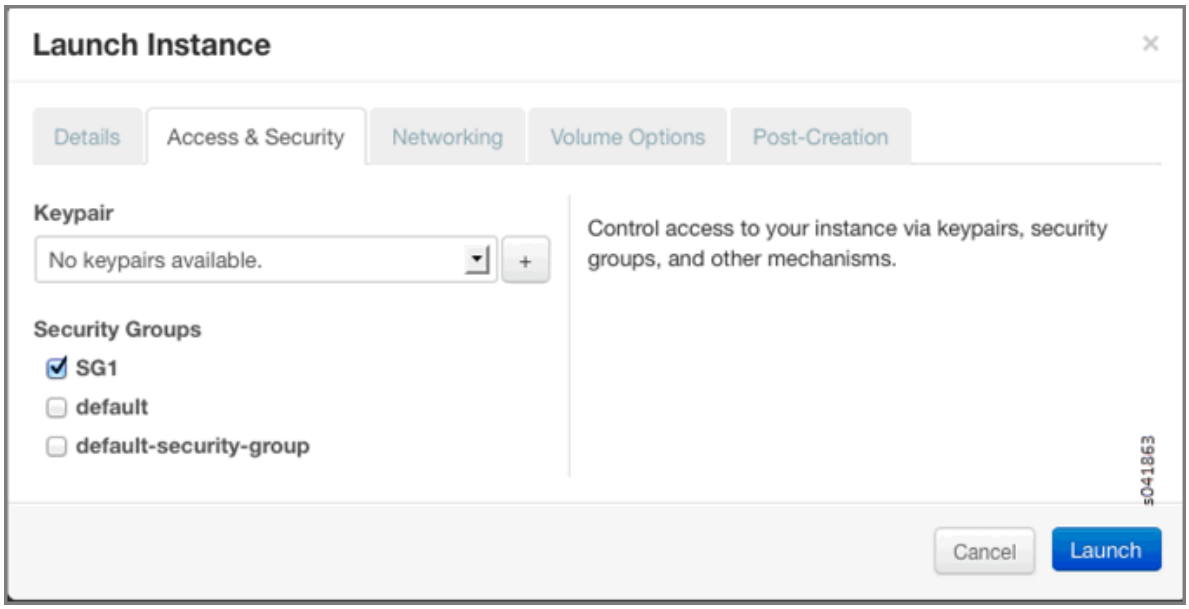
Figure 56: Create Security Group



5. When an instance is launched, there is an opportunity to associate a security group; see Figure 57 on page 533.

   In the **Security Groups** list, select the security group name to associate with the instance.

Figure 57: Associate Security Group at Launch Instance



6. You can verify that security groups are attached by viewing the `SgListReq` and `IntfReq` associated with the `agent.xml`.

CHAPTER 11

Using Contrail Resources in Heat Templates

IN THIS CHAPTER

Using the Contrail Heat Template

IN THIS SECTION

Heat is the orchestration engine of the OpenStack program. Heat enables launching multiple cloud applications based on templates that are comprised of text files.

Introduction to Heat

A Heat template describes the infrastructure for a cloud application, such as networks, servers, floating IP addresses, and the like, and can be used to manage the entire life cycle of that application.

When the application infrastructure changes, the Heat templates can be modified to automatically reflect those changes. Heat can also delete all application resources if the system is finished with an application.

Heat templates can record the relationships between resources, for example, which networks are connected by means of policy enforcements, and consequently call OpenStack REST APIs that create

the necessary infrastructure, in the correct order, needed to launch the application managed by the Heat template.

Heat Architecture

Heat is implemented by means of Python applications, including the following:

- `heat-client`—The CLI tool that communicates with the `heat-api` application to run Heat APIs.
- `heat-api`—Provides an OpenStack native REST API that processes API requests by sending them to the Heat engine over remote procedure calls (RPCs).
- `heat-engine`—Responsible for orchestrating the launch of templates and providing events back to the API consumer.

Support for Heat Version 2 Resources

Starting with Contrail Release 3.0.2, Contrail Heat resources and templates are autogenerated from the Contrail schema, using Heat Version 2 resources. Contrail Release 3.0.2 is the minimum required version for using Heat with Contrail in 3.x releases. The Contrail Heat Version 2 resources are of the following hierarchy: `OS::ContrailV2::<ResourceName>`.

The generated resources and templates are part of the Contrail Python package, and are located in the following directory in the target installation:

/usr/lib/python2.7/dist-packages/vnc_api/gen/heat/

The **heat/** directory has the following subdirectories:

- **resources/**—Contains all the resources for the contrail-heat plugin, which runs in the context of the Heat engine service.
- **templates/**—Contains sample templates for each resource. Each sample template presents every possible parameter in the schema. Use the sample templates as a reference when you build up more complex templates for your network design.
- **env/**—Contains the environment for input to each template.

The following contains a list of all the generated plug-in resources that are supported by contrail-heat :

https://github.com/tungstenfabric/tf-heat-plugin/tree/master/contrail_heat/new_templates

Deprecation of Heat Version 1 Resources

Heat Version 1 resources within the hierarchy `OS::Contrail::<ResourceName>` are being deprecated, and you should not create new service templates using the Heat Version 1 templates.

Heat Version 2 with Service Templates and Port Tuple Sample Workflow

With Contrail service templates Version 2, the user can create ports and bind them to a virtual machine (VM)-based service instance, by means of a port-tuple object. All objects created with the Version 2 service template are directly visible to the Contrail Heat engine, and are directly managed by Heat.

The following shows the basic workflow steps for creating a port tuple and service instance that will be managed by Heat:

1. Create a service template. Select 2 in the Version field.
2. Create a service instance for the service template just created.
3. Create a port-tuple object.
4. Create ports, using Nova VM launch or without a VM launch.
5. Label each port as left, right, mgmt, and so on, and add the ports to the port-tuple object.

   Use a unique label for each of the ports in a single port tuple. The labels named left and right are used for forwarding.
6. Link the port tuple to a service instance.
7. Launch the service instance.

Example: Creating a Service Template Using Heat

The following is an example of how to create a service template using Heat.

1. Define a template to create the service template.

```
service_template.yaml
heat_template_version:  2013--05--23
description: >
   HOT template to create a service template
parameters:
   name:
      type: string
      description: Name of service template
   mode:
      type: string
      description: service mode
   type:
      type: string
```

```
      description: service type
   image:
      type: string
      description: Name of the image
   flavor:
      type: string
      description: Flavor
   service_interface_type_list:
      type: string
      description: List of interface types
   shared_ip_list:
      type: string
      description: List of shared ip enabled--disabled
   static_routes_list:
      type: string
      description: List of static routes enabled--disabled

resources:
   service_template:
      type: OS::ContrailV2::ServiceTemplate
      properties:
         name: { get_param: name }
         service_mode: { get_param: mode }
         service_type: { get_param: type }
         image_name: { get_param: image }
         flavor: { get_param: flavor }
         service_interface_type_list: { "Fn::Split" : [ ",", Ref:
service_interface_type_list ] }
         shared_ip_list: { "Fn::Split" : [ ",", Ref: shared_ip_list ] }
         static_routes_list: { "Fn::Split" : [ ",", Ref: static_routes_list ] }
   outputs:
      service_template_fq_name:
         description: FQ name of the service template
         value: { get_attr: [ service_template, fq_name] }

}
```

2. Create an environment file to define the values to put in the variables in the template file.

```
service_template.env

parameters:
```

```
name: contrail_svc_temp

mode: transparent

type: firewall

image: cirros

flavor: m1.tiny

service_interface_type_list: management,left,right,other

shared_ip_list: True,True,False,False

static_routes_list: False,True,False,False
```

3. Create the Heat stack by launching the template and the environment file, using the following command:

```
heat stack create stack1 -f service_template.yaml -e service_template.env
```

OR use this command for recent versions of OpenStack

```
openstack stack create -e <env-file-name> -t <template-file-name> <stack-name>
```

RELATED DOCUMENTATION

Service Chain Version 2 with Port Tuple

CHAPTER 12

QoS Support in Contrail Networking

IN THIS CHAPTER

Quality of Service in Contrail

IN THIS SECTION

Overview: Quality of Service

Quality of service (QoS) in networking provides the ability to control reliability, bandwidth, latency, and other traffic management features. Network traffic can be marked with QoS bits (DSCP, 802.1p, and MPLS EXP) that intermediate network switches and routers can use to provide service guarantees.

Contrail QoS Model

The QoS model in Contrail Networking has the following features:

- All packet forwarding devices, such as vRouter and the gateway, combine to form a system.
- Interfaces to the system are the ports from which the system sends and receives packets, such as tap interfaces and physical ports.
- Fabric interfaces are where the overlay traffic is tunneled.
- QoS is applied at the ingress to the system, for example, upon traffic from the interfaces to the fabric.
- At egress, packets are stripped of their tunnel headers and sent to interface queues, based on the forwarding class. No marking from the outer packet to the inner packet is considered at this time.

Features of Fabric Interfaces

Fabric interfaces, unlike other interfaces, are always shared. Therefore, fabric interfaces are common property. Consequently, traffic classes and QoS marking on the fabric must be controlled by the system administrator. The administrator might choose to provision different classes of service on the fabric.

In Contrail, classes of service are determined by both of the following:

- Queueing on the fabric interface, including queues, scheduling of queues, and drop policies, and
- forwarding class, a method of marking that controls how packets are sent to the fabric, including marking and identifying which queue to use.

Tenants can define which forwarding class their traffic can use, deciding which packets use which forwarding class. The Contrail QoS configuration object has a mapping table, mapping the incoming DSCP or 802.1p value to the forwarding class mapping.

The QoS configuration can also be applied to a virtual network, an interface, or a network policy.

QoS Configuration Parameters for Provisioning

Configuring QoS in Contrail Networking Release 5.0 and Later

This section describes how to provision QoS in Contrail Networking release 5.0 and later.

1. Define the hardware queues and priority group in the **instances.yaml** file under the vrouter role as shown below.

```
nodeh5:
    ip: 10.xxx.xxx.109
    provider: bms
    roles:
      vrouter:
        VROUTER_GATEWAY: 192.168.1.45
        PRIORITY_ID: 0,1,2,3,4,5,6,7
        PRIORITY_BANDWIDTH: 0,10,0,20,0,30,0,40
        PRIORITY_SCHEDULING: strict,rr,strict,rr,strict,rr,strict,rr
        QOS_QUEUE_ID: 3,11,18,28,36,43,61,53
        QOS_LOGICAL_QUEUES: "[ 1, 6-10, 12-15];[40-46];[70-74, 75, 80-95];[115];[140-143,
145];[175];[245];[215]"
        QOS_DEF_HW_QUEUE: True
      openstack_compute:
```

2. In the already provisioned setup, define the QoS configuration in the **/etc/contrail/common_vrouter.env** file as shown in the following sample.

```
PRIORITY_ID=0,1,2,3,4,5,6,7
PRIORITY_BANDWIDTH=0,10,0,20,0,30,0,40
PRIORITY_SCHEDULING=strict,rr,strict,rr,strict,rr,strict,rr
QOS_QUEUE_ID=3,11,18,28,36,43,61,53
QOS_LOGICAL_QUEUES="[ 1, 6-10, 12-15];[40-46];[70-74, 75, 80-95];[115];[140-143, 145];[175];
[245];[215]"
QOS_DEF_HW_QUEUE=True
```

3. Execute the `execute docker-compose up -d under /etc/contrail/vrouter/` command.

Queuing Implementation

The vRouter provides the infrastructure to use queues supplied by the network interface, a method that is also called hardware queueing. Network interface cards (NICs) that implement hardware queueing have their own set of scheduling algorithms associated with the queues. The Contrail implementation is designed to work with most NICs, however, the method is tested only on an Intel-based 10G NIC, also called Niantic.

Contrail QoS Configuration Objects

Contrail QoS configuration objects include the:

- forwarding class
- QoS configuration object (`qos-config`)

The forwarding class object specifies parameters for marking and queuing, including:

- The DSCP, 802.1p, and MPLS EXP values to be written on packets.
- The queue index to be used for the packet.

The QoS configuration object specifies a mapping from DSCP, 802.1p, and MPLS EXP values to the corresponding forwarding class.

The QoS configuration has an option to specify the default forwarding class ID to use to select the forwarding class for all unspecified DSCP, 802.1p, and MPLS EXP values.

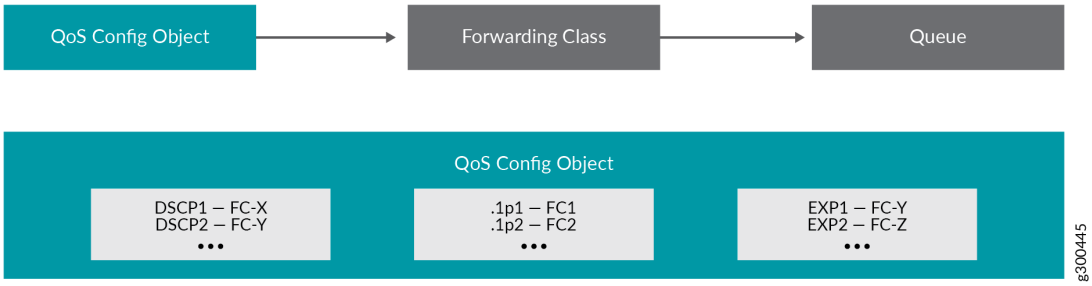
If the default forwarding class ID is not specified by the user, it defaults to the forwarding class with ID 0.

Processing of QoS marked packets to look up the corresponding forwarding class to be applied works as follows:

- For an IP packet, the DSCP map is used .
- For a Layer 2 packet, the 802.1p map is used.
- For an MPLS-tunneled packet with MPLS EXP values specified, the EXP bit value is used with the MPLS EXP map.
- If the QoS configuration is untrusted, only the default forwarding class is specified, and all incoming values of the DSCP, 802.1p, and EXP bits in the packet are mapped to the same default forwarding class.

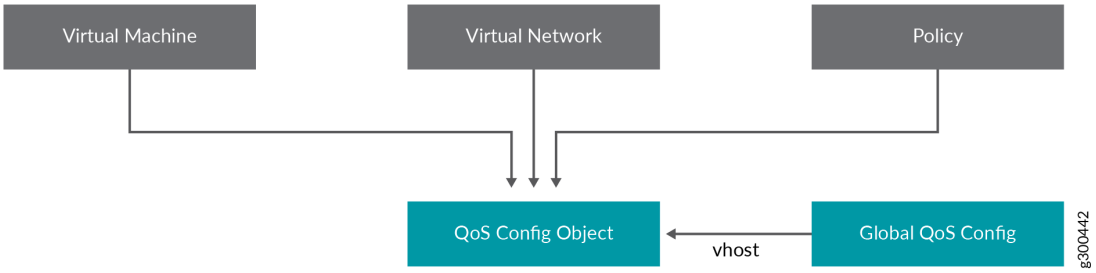
Figure 58 on page 543 shows the processing of QoS packets.

Figure 58: Processing of QoS Packets



A virtual machine interface, virtual network, and network policy can refer to the QoS configuration object. The QoS configuration object can be specified on the vhost so that underlay traffic can also be subjected to marking and queuing. See Figure 59 on page 543.

Figure 59: Referring to the QoS Object



Example: Mapping Traffic to Forwarding Classes

This example shows how traffic forwarding classes are defined and how the QoS configuration object is defined to map the QoS bits to forwarding classes.

Table 38 on page 544 shows two forwarding class objects defined. FC1 marks the traffic with high priority values and queues it to Queue 0. FC2 marks the traffic as best effort and queues the traffic to Queue 1.

Table 38: Forwarding Class Mapping

Name	ID	DSCP	802.1p	MPLS EXP	Queue
FC1	1	10	7	7	0
FC2	2	38	0	0	1

In Table 39 on page 544, the QoS configuration object DSCP values of 10, 18, and 26 are mapped to a forwarding class with ID 1, which is forwarding class FC1. All other IP packets are mapped to the forwarding class with ID 2, which is FC2. All traffic with an 802.1p value of 6 or 7 are mapped to forwarding class FC1, and the remaining traffic is mapped to FC2.

Table 39: QoS Configuration Object Mapping

DSCP	Forwarding Class ID	802.1p	Forwarding Class ID	MPLS EXP	Forwarding Class ID
10	1	6	1	5	1
18	1	7	1	7	1
26	1	*	2	*	1
*	2				

QoS Configuration Object Marking on the Packet

The following sections describes how QoS configuration object marking is handled in various circumstances.

Traffic Originated by a Virtual Machine Interface

- If a VM interface sends an IP packet to another VM in a remote compute node, the DSCP value in the IP header is used to look into the qos-config table, and the tunnel header is marked with DSCP, 802.1p, and MPLS EXP bits as specified by the forwarding class.

- If a VM sends a Layer 2 non-IP packet with an 802.1p value, the 802.1p value is used to look into the qos-config table, and the corresponding forwarding class DSCP, 802.1p, and MPLS EXP value is written to the tunnel header.
- If a VM sends an IP packet to a VM in same compute node, the packet headers are not changed while forwarding. The original packet remains unchanged.

Traffic Destined to a Virtual Machine Interface

For traffic destined to a VMI, if a tunneled packet is received, the tunnel headers are stripped off and the packet is sent to the interface. No marking is done from the outer packet to inner packet.

Traffic from a vhost Interface

The QoS configuration can be applied on IP traffic coming from a vhost interface. The DSCP value in the packet is used to look into the qos-config object specified on the vhost, and the corresponding forwarding class DSCP and 802.1p values are overwritten on the packet.

Traffic from fabric interface

The QoS configuration can be applied while receiving the packet on an Ethernet interface of a compute node, and the corresponding forwarding class DSCP and 802.1p values are overwritten on the packet.

QoS Configuration Priority by Level

The QoS configuration can be specified at different levels.

The levels that can be configured with QoS and their order of priority:

1. in policy
2. on `virtual-network`
3. on `virtual-machine-interface`

Queuing

Contrail Networking supports QoS. These sections provide an overview of the queuing features available in Contrail Networking.

The queue to which a packet is sent is specified by the forwarding class.

Queue Selection in Datapath

In vRouter, in the data path, the forwarding class number specifies the actual physical hardware queue to which the packet needs to be sent, not to a logical selection as in other parts of Contrail. There is a mapping table in the vRouter configuration file, to translate the physical queue number from the logical queue number.

Hardware Queueing in Linux kernel based vRouter

If Xmit-Packet-Steering (XPS) is enabled, the kernel chooses the queue, from those available in a list of queues. If the kernel selects the queue, packets will not be sent to the vRouter-specified queue.

To disable this mapping:

- have a kernel without CONFIG_XPS option
- write zeros to the mapping file in /sys/class/net//queues/tx-X/xps_cpus .

When this mapping is disabled, the kernel will send packets to the specific hardware queue.

To verify:

See individual queue statistics in the output of 'ethtool -S ' command.

Parameters for QoS Scheduling Configuration

The following shows sample scheduling configuration for hardware queues on the compute node.

The priority group ID and the corresponding scheduling algorithm and bandwidth to be used by the priority group can be configured.

Possible values for the scheduling algorithm include:

- strict
- rr (round-robin)

When round-robin scheduling is used, the percentage of total hardware queue bandwidth that can be used by the priority group is specified in the bandwidth parameter.

The following configuration and provisioning is applicable only for compute nodes running Niantic NICs and running kernel based vrouter.

```
qos_niantic =  {
       ‘compute1': [
                          { 'priority_id': '1', 'scheduling': 'strict', 'bandwidth': '0'},
```

```
                        { 'priority_id': '2', 'scheduling': 'rr', 'bandwidth': '20'},
                        { 'priority_id': '3', 'scheduling': 'rr', 'bandwidth': '10’}
        ],
        ‘compute2' :[
                        { 'priority_id': '1', 'scheduling': 'strict', 'bandwidth': '0'},
                        { 'priority_id': '1', 'scheduling': 'rr', 'bandwidth': '30’}
         ]
}
```

RELATED DOCUMENTATION

Configuring Network QoS Parameters | 547

https://github.com/Juniper/contrail-controller/wiki/QoS

Configuring Network QoS Parameters

IN THIS SECTION

Overview

You can use the OpenStack Nova command-line interface (CLI) to specify a quality of service (QoS) setting for a virtual machine’s network interface, by setting the `quota` of a Nova flavor. Any virtual machine created with that Nova flavor will inherit all of the specified QoS settings. Additionally, if the virtual machine that was created with the QoS settings has multiple interfaces in different virtual networks, the same QoS settings will be applied to all of the network interfaces associated with the virtual machine. The QoS settings can be specified in unidirectional or bidirectional mode.

The `quota` driver in Neutron converts QoS parameters into `libvirt` network settings of the virtual machine.

The QoS parameters available in the quota driver only cover rate limiting the network interface. There are no specifications available for policy-based QoS at this time.

QoS Configuration Examples

Although the QoS setting can be specified in quota by using either Horizon or CLI, quota creation using CLI is more robust and stable, therefore, creating by CLI is the recommended method.

Example

CLI for Nova flavor has the following format:

```
nova flavor-key <flavor_name> set quota:vif_<direction> _<param_name> = value
```

where:

<flavor_name> is the name of an existing Nova flavor.

vif_*<direction>*_*<param_name>* is the inbound or outbound QoS data name.

QoS `vif` types include the following:

- `vif_inbound_average` lets you specify the average rate of inbound (receive) traffic, in kilobytes/sec.
- `vif_outbound_average` lets you specify the average rate of outbound (transmit) traffic, in kilobytes/sec.
- Optional: `vif_inbound_peak` and `vif_outbound_peak` specify the maximum rate of inbound and outbound traffic, respectively, in kilobytes/sec.
- Optional: `vif_inbound_burst` and `vif_outbound_peak` specify the amount of kilobytes that can be received or transmitted, respectively, in a single burst at the peak rate.

Details for various QoS parameters for `libvirt` can be found at http://libvirt.org/formatnetwork.html.

The following example shows an inbound average of 800 kilobytes/sec, a peak of 1000 kilobytes/sec, and a burst amount of 30 kilobytes.

```
nova flavor-key m1.small set quota:vif_inbound_average=800
nova flavor-key m1.small set quota:vif_inbound_peak=1000
nova flavor-key m1.small set quota:vif_inbound_burst=30
```

The following is an example of specified outbound parameters:

```
nova flavor-key m1.small set quota:vif_outbound_average=800
nova flavor-key m1.small set quota:vif_outbound_peak=1000
nova flavor-key m1.small set quota:vif_outbound_burst=30
```

After the Nova flavor is configured for QoS, a virtual machine instance can be created, using either Horizon or CLI. The instance will have network settings corresponding to the nova flavor-key, as in the following:

```
<interface type="ethernet">
      <mac address="02:a3:a0:87:7f:61"/>
      <model type="virtio"/>
      <script path=""/>
      <target dev="tapa3a0877f-61"/>
      <bandwidth>
        <inbound average="800" peak="1000" burst="30"/>
        <outbound average="800" peak="1000" burst="30"/>
      </bandwidth>
    </interface>
```

Limitations

- The stock `libvirt` does not support rate limiting of `ethernet` interface types. Consequently, settings like those in the example for the guest interface will not result in any `tc qdisc` settings for the corresponding tap device in the host.
- The `nova flavor-key rxtx_factor` takes a float as an input and acts as a scaling factor for receive (inbound) and transmit (outbound) throughputs. This key is only available to Neutron extensions (private extensions). The Contrail Neutron plugin doesn't implement this private extension. Consequently, setting the `nova flavor-key rxtx_factor` will not have any effect on the QoS setting of the network interface(s) of any virtual machine created with that nova flavor.
- The outbound rate limits of a virtual machine interface are not strictly achieved. The outbound throughput of a virtual machine network interface is always less than the average outbound limit specified in the virtual machine's libvirt configuration file. The same behavior is also seen when using a Linux bridge.

CHAPTER 13

Load Balancers

IN THIS CHAPTER

Using Load Balancers in Contrail

IN THIS SECTION

As of Contrail Release 3.0, load balancer LBaaS features are available. This topic includes:

Invoking LBaaS Drivers

The provider field specified in the pool configuration determines which load balancer drivers are selected. The load balancer driver selected is responsible for configuring the external hardware or virtual machine load balancer.

Supported load balancer drivers include:

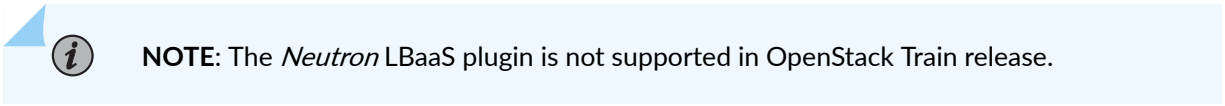
- HAProxy
- A10 Networks
- F5 Networks
- Avi Networks

Benefits of Creating Configuration Objects

Starting with Contrail 3.0, the Neutron LBaaS plugin creates required configuration objects (such as pool, VIP, members, and monitor) in the Contrail API server, instead of within the Neutron plugin context, as in previous releases.

This method of configuration has the following benefits:

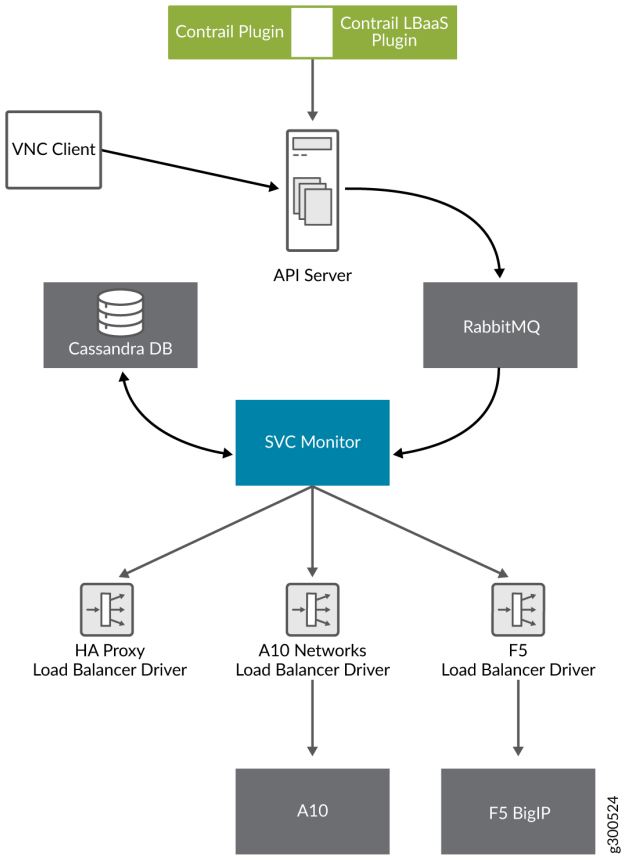
- Configuration objects can be created in multiple ways: from Neutron, from virtual controller APIs, or from the Contrail UI.
- The load balancer driver can make inline calls, such as REST or SUDS, to configure the external load balancer device.
- The load balancer driver can use Contrail service monitor infrastructure, such as database, logging, and API server.



NOTE: The *Neutron* LBaaS plugin is not supported in OpenStack Train release.

Figure 60 on page 552 provides an overview of the Contrail LBaaS components.

Figure 60: Contrail LBaaS components with neutron-lbaas



Using a Service Appliance Set as the LBaaS Provider

In OpenStack Neutron, the load balancer provider is statically configured in `neutron.conf`, which requires restart of the Neutron server when configuring a new provider. The following is an example of the service provider configuration in `neutron.conf`.

```
[service_providers]
service_provider = LOADBALANCER:Opencontrail:neutron_plugin_contrail.plugins.opencontrail.
loadbalancer.driver.OpencontrailLoadbalancerDriver:default
```

In Contrail Release 3.0 and greater, the Neutron LBaaS provider is configured by using the object `service-appliance-set`. All of the configuration parameters of the LBaaS driver are populated to the `service-appliance-set` object and passed to the driver.

During initialization, the service monitor creates a default service appliance set with a default LBaaS provider, which uses an HAProxy-based load balancer. The service appliance set consists of individual

service appliances for load balancing the traffic. The service appliances can be physical devices or virtual machines.

Sample Configuration: Service Appliance Set

The following is a sample configuration of the service appliance set for the LBaaS provider:

```
{
  "service-appliance-set": {
    "fq_name": [
      "default-global-system-config",
      "f5"
    ],
    "service_appliance_driver":
"svc_monitor.services.loadbalancer.drivers.f5.f5_driver.OpencontrailF5LoadbalancerDriver",
    "parent_type": "global-system-config",
    "service_appliance_set_properties": {
      "key_value_pair": [
        {
          "key": "sync_mode",
          "value": "replication"
        },
       {
          "key": "global_routed_mode",
          "value": "True"
        }
      ]
    },
    "name": "f5"
  }
}
```

Sample Configuration: Single Service Appliance

The following is a sample configuration of a single service appliance:

```
{
  "service-appliance": {
    "fq_name": [
      "default-global-system-config",
      "f5",
      "bigip"
    ],
```

```
    "parent_type": "service-appliance-set",
    "service_appliance_ip_address": "<ip address>",
    "service_appliance_user_credentials": {
      "username": "admin",
      "password": "<password>"
    },
    "name": "bigip"
  }
}
```

Understanding the Load Balancer Agent

The load balancer agent is a module in the service monitor. The service monitor listens on the RabbitMQ configuration messaging queue (`vnc_config.object-update`) to get configuration objects. The dependency tracker triggers changes to all related objects, based on configuration updates.

The dependency tracker is informed to notify the pool object whenever the VIP, member, or health monitor object is modified.

Whenever there is an update to the pool object, either directly due to a pool update or due to a dependency update, the load balancer agent in the service monitor is notified.

The load balancer agent module handles the following:

- Loading and unloading LBaaS driver-based service appliance set configuration.
- Providing the abstract driver class for the load balancer driver.
- Invoking the LBaaS driver.
- Load balancer-related configuration.

F5 Networks Load Balancer Integration in Contrail

This section details use of the F5 load balancer driver with Contrail.

Contrail Release 3.0 implements an LBaaS driver that supports a physical or virtual F5 Networks load balancer, using the abstract load balancer driver class, `ContrailLoadBalancerAbstractDriver`.

This driver is invoked from the load balancer agent of the `contrail-svc-monitor`. The driver makes a BIG-IP interface call to configure the F5 Networks device. All of the configuration parameters used to tune the driver are configured in the `service-appliance-set` object and passed to the driver by the load balancer agent while loading the driver.

The F5 load balancer driver uses the BIG-IP interface version V1.0.6, which is a Python package extracted from the load balancer plugin provided by F5 Networks. The driver uses either a SOAP API or a REST API.

F5 Load Balancer Global Routed Mode

The F5 load balancer driver is programmed in `global routed` mode using a property of the `service-appliance-set`.

This section describes the features and requirements of the F5 load balancer driver configured in global routed mode.

The following are features of the global routed mode.

- All virtual IP addresses (VIPs) are assumed to be routable from clients and all members are routable from the F5 device.
- All access to and from the F5 device is assumed to be globally routed, with no segregation between tenant services on the F5 device. Consequently, do NOT configure overlapping addresses across tenants and networks.
- The F5 device can be attached to the corporate network or to the IP fabric.

The following are requirements to support global routed mode of an F5 device used with LBaaS:

- The entire configuration of the F5 device for Layer 2 and Layer 3 is preprovisioned.
- All tenant networks and all IP fabrics are in the same namespace as the corporate network.
- All VIPs are in the same namespace as the tenant and corporate networks.

Traffic Flow in Global Routed Mode

This section describes and illustrates the behavior of traffic flow in global routed mode.

The information in this section is based on a model that includes the following network topology:

Corporate Network --- DC Gateway (MX device) --- IP Fabric --- Compute nodes

The Corporate Network, the IP Fabric and all tenant networks use IP addresses from a single namespace, there is no overlap of the addresses in the networks. The F5 devices can be attached to the Corporate Network or to the IP Fabric, and are configured to use the global routed mode.

The role of the MX Series device is to route post-proxy traffic, coming from the F5 device in the underlay, to the pool members in the overlay. In the reverse direction, the MX device takes traffic coming from the pool members in the overlay and routes it back to the F5 device in the underlay.

The MX device is preprovisioned with the following:

- VRF connected to pool network 2
- ability to route traffic from inet.0 to the pool network

The MX routes the traffic from inet.0 to public VRF and sends traffic to the compute node where the pool member is instantiated.

The F5 device is preprovisioned with the following:

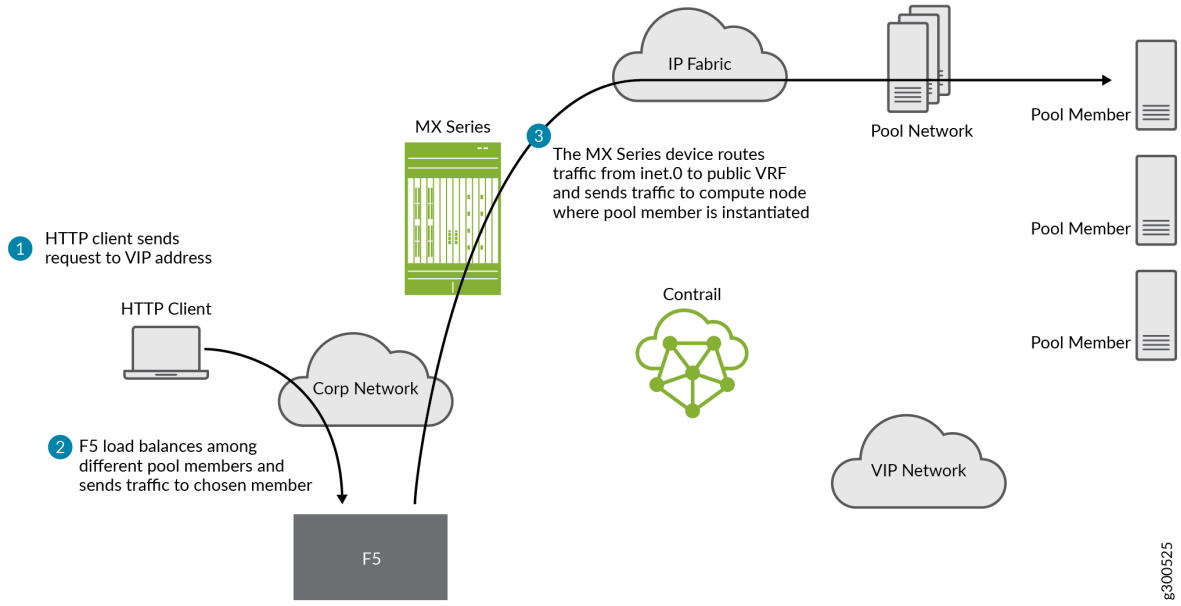
- publish route to attract VIP traffic
- pool network subnet route that points to the MX device

The F5 device is responsible for attracting traffic destined to all the VIPs, by advertising a subnet route that covers all VIPs using IGP.

The F5 device load balances among different pool members and sends traffic to the chosen member.

The following figure shows the traffic flow in global routed mode.

Figure 61: Global Routed Traffic Flow



A similar result can also be achieved on the switch to which the F5 is attached, by publishing the VIP subnet in IGP and using a static route to point the VIP traffic to the F5 device.

The MX should attract the reverse traffic from the pool members going back to the F5.

Routing Traffic to Pool Members

For post load balancing traffic going from the F5 device to the pool members, the MX Series device needs to attract traffic for all the tenant networks.

Routing Reverse Traffic from Pool Members to the F5 Device

The MX should attract the reverse traffic from the pool members going back to the F5.

Initial Configuration on an F5 Device

- The operator is responsible for ensuring that the F5 device attracts traffic to all VIP subnets by injecting the route for the VIP subnet into IGP. Alternately, the switch to which F5 is connected can advertise the VIP subnet route and use the static route to send VIP traffic to the F5 device.
- In the global routed mode, the F5 uses AutoMap SNAT for all VIP traffic.

Initial Configuration on an MX Series Device Used as DC Gateway

- The operator must identify a super-net that contains all tenant network subnets (pool members across multiple pools) and advertise its route into corporate and fabric networks, using IGP (preferred) or static routes.
- The operator must add a static route for the super-net into inet.0 with a next-hop of public.inet.0.
- The operator must create a public VRF and get its default route imported into the VRF. This is to attract the return traffic from pool members to the F5 device (VIP destination).

Configuration on MX Device for Each Pool Member

- For each member virtual network, the operator adds a policy to connect the member pool virtual network to the public virtual network.
- As new member virtual networks are connected to the public virtual network by policy, corresponding targets are imported by the public VRF on MX. The Contrail Device Manager generates the configuration of import, export targets for public VRF on the MX device.
- The operator must ensure that security group rules for the member virtual network ports allow traffic coming from the F5 device.

Example: Creating a Load Balancer

Use the following steps to create a load balancer in Contrail Release 3.0 and greater.

1. To configure a service appliance set, use the script in **/opt/contrail/utils** to create a load balancer provider. With the script, you specify the driver and name of the selected provider. Additional configuration can be performed using the key-value pair property configuration.

   ```
   /opt/contrail/utils/service_appliance_set.py --api_server_ip <ip address>--api_server_port 8082 --oper add --admin_user admin --admin_password <password> --admin_tenant_name admin --name f5 --driver "svc_monitor.services.loadbalancer.drivers.f5.f5_driver.OpencontrailF5LoadbalancerDriver" --properties '{"use_snat": "True", "num_snat": "1", "global_routed_mode":"True", "sync_mode": "replication", "vip_vlan": "trial2"}'
   ```

2. Add the actual device information of the load balancer.

   ```
   /opt/contrail/utils/service_appliance.py --api_server_ip <ip address>--api_server_port 8082 --oper add --admin_user admin --admin_password <password> --admin_tenant_name admin --name bigip --service_appliance_set f5 --device_ip 10.204.216.113 --user_credential '{"user": "admin", "password": "<password>"}'
   ```

3. Refer to the load balancer provider while configuring the pool.

   ```
   neutron lb-pool-create --lb-method ROUND_ROBIN --name web_service --protocol HTTP --provider "f5" --subnet-id <subnet id>
   ```

4. Add members to the load balancer pool. Both bare metal webserver and overlay webserver are allowed as pool members. The F5 device can load balance the traffic among all pool members.

   ```
   neutron lb-member-create --address <ip address>--protocol-port 8080 --weight 3 web_service

   neutron lb-member-create --address <ip address> --protocol-port 8080 --weight 2 web_service
   ```

5. Create a VIP for the load balancer pool.

   ```
   neutron lb-vip-create --name httpserver --protocol-port 80 --protocol HTTP web_service --subnet-id <subnet id>
   ```

6. Create the health monitor and associate it with the load balancer pool.

   ```
   neutron lb-healthmonitor-create --delay 3 --type HTTP --max-retries 3 --timeout 3

   neutron lb-healthmonitor-associate <nnnnn-nnnnn-nnnn-> web_service
   ```

Using the Avi Networks Load Balancer for Contrail

If you are using the Avi LBaaS driver in an OpenStack Contrail environment, there are two possible modes that are mutually-exclusive. The Avi Vantage cloud configuration is exactly the same in both modes:

- Neutron-based Avi LBaaS driver - In this mode, the Avi LBaaS driver derives from Neutron and resides in the Neutron server process. This mode enables coexistence of multiple Neutron LBaaS providers.

- Contrail-based Avi LBaaS driver - In this mode, the Avi LBaaS driver derives from Contrail and resides in the service-monitor process. This mode enables coexistence of multiple Contrail LBaaS providers.

> **NOTE**: In a Contrail environment, you cannot have a mix of Contrail LBaaS and Neutron LBaaS. You must select a mode that is compatible with the current environment.

Installing the Avi LBaaS Neutron Driver

Use the following procedure to install the Avi Networks LBaaS load balancer driver for the Neutron server for Contrail.

The following steps are performed on the Neutron server host.

1. Determine the installed version of the Contrail Neutron plugin.

```
$ contrail-version neutron-plugin-contrail
Package Version
------------------------ ------------
neutron-plugin-contrail 3.0.2.0-51
```

2. Adjust the `neutron.conf` database connection URL.

```
$ vi /etc/neutron/neutron.conf
# if using mysql
connection = mysql+pymysql://neutron:c0ntrail123@127.0.0.1/neutron
```

3. Populate and upgrade the Neutron database schema.

```
# to upgrade to head
$ neutron-db-manage upgrade head
# to upgrade to a specific version
$ neutron-db-manage --config-file /etc/neutron/neutron.conf upgrade liberty
```

4. Drop foreign key constraints.

```
# obtain current mysql token
$ cat /etc/contrail/mysql.token
fabe17d9dd5ae798f7ea
```

```
$ mysql -u root -p
Enter password: fabe17d9dd5ae798f7ea

mysql> use neutron;

mysql> show create table vips;
# CONSTRAINT `vips_ibfk_1` FOREIGN KEY (`port_id`) REFERENCES `ports` (`id`) - ports table is
not used by Contrail
mysql> alter table vips drop FOREIGN KEY vips_ibfk_1;

mysql> show create table lbaas_loadbalancers;
# CONSTRAINT `fk_lbaas_loadbalancers_ports_id` FOREIGN KEY (`vip_port_id`) REFERENCES `ports`
(`id`)
mysql> alter table lbaas_loadbalancers drop FOREIGN KEY fk_lbaas_loadbalancers_ports_id;
```

5. To install the Avi LBaaS plugin, continue with steps from the readme file that downloads with the Avi LBaaS software. You can perform either a local installation or a manual installation. The following are sample installation steps.

   - For a local installation:

```
# LBaaS v1 driver
$ ./install.sh --aname avi_adc --aip

  <controller_ip|controller_vip>
    --auser

     --apass

# LBaaS v2 driver
$ ./install.sh --aname avi_adc_v2 --aip
     <controller_ip|controller_vip>
       --auser

        --apass

         --v2
```

- For a manual installation:

```
# LBaaS v1 driver
$ vi /etc/neutron/neutron.conf
#service_plugins =
neutron_plugin_contrail.plugins.opencontrail.loadbalancer.plugin.LoadBalancerPlugin
service_plugins = neutron_lbaas.services.loadbalancer.plugin.LoadBalancerPlugin
[service_providers]
service_provider =
LOADBALANCER:Avi_ADC:neutron_lbaas.services.loadbalancer.drivers.avi.avi_driver.AviLbaaSDri
ver

[avi_adc]
address=10.1.11.4
user=admin
password=avi123
cloud=jcos

# LBaaS v2 driver
$ vi /etc/neutron/neutron.conf
#service_plugins =
neutron_plugin_contrail.plugins.opencontrail.loadbalancer.plugin.LoadBalancerPlugin
service_plugins = neutron_lbaas.services.loadbalancer.plugin.LoadBalancerPluginv2
[service_providers]
service_provider = LOADBALANCERV2:avi_adc_v2:neutron_lbaas.drivers.avi.driver.AviDriver

[avi_adc_v2]
controller_ip=10.1.11.3
username=admin
password=avi123

$ service neutron-server restart
$ neutron service-provider-list
```

Installing the Avi LBaaS Contrail Driver

Use the following procedure to install the Avi Networks LBaaS load balancer driver for Contrail.

The following steps are performed on the Contrail `api-server` host.

1. Determine the installed version of the Contrail Neutron plugin.

```
$ contrail-version neutron-plugin-contrail
Package Version
------------------------ ------------
neutron-plugin-contrail 3.0.2.0-51
```

2. Install the Avi driver.

```
# LBaaS v2 driver
$ ./install.sh --aname ocavi_adc_v2 --aip

  <controller_ip|controller_vip>
    --auser

     --apass

      --v2 --no-restart --no-confmodify
```

3. Set up the service appliance set.

NOTE: If `neutron_lbaas` doesn't exist on the `api-server` node, adjust the driver path to the correct path location for `neutron_lbaas`.

```
$ /opt/contrail/utils/service_appliance_set.py --api_server_ip 10.xx.xx.100 --api_server_port 8082 --oper add
--admin_user admin --admin_password <password> --admin_tenant_name admin --name ocavi_adc_v2 --driver
"neutron_lbaas.drivers.avi.avi_ocdriver.OpencontrailAviLoadbalancerDriver" --properties '{"address":
"10.1.xx.3", "user": "admin", "password": "avi123", "cloud": "Default-Cloud"}'
```

4. To delete the service appliance set.

```
$ /opt/contrail/utils/service_appliance_set.py --api_server_ip 10.xx.xx.100 --api_server_port 8082 --oper del
--admin_user admin --admin_password <password> --admin_tenant_name admin --name ocavi_adc_v2
```

Configuring the Avi Controller

1. If OpenStack endpoints are private IPs and Contrail provides a public front-end IP to those endpoints, use iptables to DNAT. On the AviController only, perform iptable NAT to reach the private IPs.

```
$ iptables -t nat -I OUTPUT --dest 17x.xx.xx.50 -j DNAT --to-dest 10.xx.xx.100
```

2. To configure the Avi controller during cloud configuration, select the “Integration with Contrail” checkbox and provide the endpoint URL of the Contrail VNC api-server. Use the Keystone credentials from the OpenStack configuration to authenticate with the api-server service.

Example Configuration Settings

```
: > show cloud jcos
    +---------------------------+------------------------------------------+
    | Field                     | Value                                    |
    +---------------------------+------------------------------------------+
    | uuid                      | cloud-104bb7e6-a9d2-4b34-a4c5-d94be659bb91 |
    | name                      | jcos                                     |
    | vtype                     | CLOUD_OPENSTACK                          |
    | openstack_configuration   |                                          |
    |   username                | admin                                    |
    |   admin_tenant            | demo                                     |
    |   keystone_host           | 17x.xx.xx.50                             |
    |   mgmt_network_name       | mgmtnw                                   |
    |   privilege               | WRITE_ACCESS                             |
    |   use_keystone_auth       | True                                     |
    |   region                  | RegionOne                                |
    |   hypervisor              | KVM                                      |
    |   tenant_se               | True                                     |
    |   import_keystone_tenants | True                                     |
    |   anti_affinity           | True                                     |
    |   port_security           | False                                    |
    |   security_groups         | True                                     |
    |   allowed_address_pairs   | True                                     |
    |   free_floatingips        | True                                     |
    |   img_format              | OS_IMG_FMT_AUTO                          |
    |   use_admin_url           | True                                     |
    |   use_internal_endpoints  | False                                    |
    |   config_drive            | True                                     |
    |   insecure                | True                                     |
    |   intf_sec_ips            | False                                    |
    |   external_networks       | False                                    |
    |   neutron_rbac            | True                                     |
    |   nuage_port              | 8443                                     |
    |   contrail_endpoint       | http://10.10.10.100:8082                 |
    | apic_mode                 | False                                    |
    | dhcp_enabled              | True                                     |
    | mtu                       | 1500 bytes                               |
```

```
| prefer_static_routes      | False                                      |
| enable_vip_static_routes  | False                                      |
| license_type              | LIC_CORES                                  |
| tenant_ref                | admin                                      |
+---------------------------+--------------------------------------------+
```

RELATED DOCUMENTATION

Configuring Load Balancing as a Service in Contrail | 568

Support for OpenStack LBaaS | 564

Installing OpenStack Octavia LBaaS with RHOSP in Contrail Networking | 513

Installing OpenStack Octavia LBaaS with Juju Charms in Contrail Networking | 670

Support for OpenStack LBaaS

IN THIS SECTION

OpenStack Neutron LBaaS Version 2.0

Starting with Contrail Networking Release 3.1, Contrail provides support for the OpenStack Load Balancer as a Service (LBaaS) Version 2.0 APIs in the Liberty release of OpenStack.

Platform Support

Table 40 on page 565 shows which Contrail with OpenStack release combinations support which version of OpenStack LBaaS APIs.

Table 40: Contrail OpenStack Platform Support for LBaaS Versions

Contrail OpenStack Platform	LBaaS Support
Contrail-3.1-Liberty (and subsequent OS releases)	Only LBaaS v2 is supported.
Contrail-3.0-Liberty (and subsequent OS releases)	LBaaS v1 is default. LBaaS v2 is Beta.
<Contrail-any-release>-Kilo (and previous OS releases)	Only LBaaS v1 is supported.

Using OpenStack LBaaS Version 2.0

The OpenStack LBaaS Version 2.0 extension enables tenants to manage load balancers for VMs, for example, load-balancing client traffic from a network to application services, such as VMs, on the same network. The LBaaS Version 2.0 extension is used to create and manage load balancers, listeners, pools, members of a pool, and health monitors, and to view the status of a resource.

For LBaaS v2.0, the Contrail controller aggregates the configuration by provider. For example, if `haproxy` is the provider, the controller generates the configuration for `haproxy` and eliminates the need to send all of the load-balancer resources to the `vrouter-agent`; only the generated configuration is sent, as part of the service instance.

For more information about OpenStack v2.0 APIs, refer to the section *LBaaS 2.0 (STABLE) (lbaas, loadbalancers, listeners, health_monitors, pools, members)*, at http://developer.openstack.org/api-ref-networking-v2-ext.html.

LBaaS v2.0 also allows users to listen to multiple ports for the same virtual IP, by decoupling the virtual IP address from the port.

The object model has the following resources:

- Load balancer—Holds the virtual IP address
- Listeners—One or many listeners with different ports, protocols, and so on
- Pools
- Members
- Health monitors

Support for Multiple Certificates per Listener

Multiple certificates per listener are supported, with OpenStack Barbican as the storage for certificates. OpenStack Barbican is a REST API designed for the secure storage, provisioning, and management of secrets such as passwords, encryption keys, and X.509 certificates.

The following is an example CLI to store certificates in Barbican:

```
- barbican --os-identity-api-version 2.0 secret store --payload-content-type='text/plain' --name='certificate' --payload="$(cat server.crt)"
```

For more information about OpenStack Barbican, see: https://wiki.openstack.org/wiki/Barbican.

Neutron Load-Balancer Creation

NOTE: This procedure is written using the *Neutron* LBaaS plugin v1.0. Starting with the OpenStack Train release, *neutron-lbaas* is replaced by *Octavia* Some commands are different due to the plugin change. See the Red Hat *Octavia* documentation for the equivalent procedure: https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/15/html/networking_guide/sec-octavia

The following is an example of Neutron load-balancer creation:

```
- neutron net-create private-net

- neutron subnet-create --name private-subnet private-net 10.30.30.0/24

- neutron lbaas-loadbalancer-create $(neutron subnet-list | awk '/ private-subnet / {print $2}') --name lb1

- neutron lbaas-listener-create --loadbalancer lb1 --protocol-port 443 --protocol TERMINATED_HTTPS --name listener1 --default-tls-container=$(barbican --os-identity-api-version 2.0 container list | awk '/ tls_container / {print $2}')

- neutron lbaas-pool-create --name pool1 --protocol HTTP --listener listener1 --lb-algorithm ROUND_ROBIN

- neutron lbaas-member-create --subnet private-subnet --address 30.30.30.10 --protocol-port 80 mypool
```

```
- neutron lbaas-member-create --subnet private-subnet --address 30.30.30.11 --protocol-port 80
mypool
```

OpenStack Octavia LBaaS

Using Octavia Load-Balancer

Contrail Networking Release 2005 supports Octavia as LBaaS. The deployment supports RHOSP and Juju platforms.

With Octavia as LBaaS, Contrail Networking is only maintaining network connectivity and is not involved in any load balancing functions.

For each OpenStack load balancer creation, Octavia launches a VM known as *amphora VM*. The VM starts the HAPROXY when listener is created for the load balancer in OpenStack. Whenever the load balancer gets updated in OpenStack, *amphora VM* updates the running HAPROXY configuration. The *amphora VM* is deleted on deleting the load balancer.

Contrail Networking provides connectivity to *amphora VM* interfaces. *Amphora VM* has two interfaces; one for management and the other for data. The management interface is used by the Octavia services for the management communication. Since, Octavia services are running in the underlay network and *amphora VM* is running in the overlay network, SDN gateway is needed to reach the overlay network. The data interface is used for load balancing the traffic.

If the load balancer service is exposed to public, you must create the load balancer VIP in the public subnet. The load balancer members can be in the public or private subnet.

You must create network policy between public network and private network if the load balancer members are in the private network.

Octavia Load-Balancer Creation

The following is an example of Octavia load-balancer creation:

```
openstack loadbalancer listener create --protocol HTTP --protocol-port 80 --name listener1 lb1
openstack loadbalancer show lb1 # Wait for the provisioning_status to be ACTIVE.
openstack loadbalancer pool create --lb-algorithm ROUND_ROBIN --listener listener1 --protocol
HTTP --name pool1
openstack loadbalancer healthmonitor create --delay 5 --timeout 2 --max-retries 1 --type HTTP
pool1
openstack loadbalancer member create --subnet-id private --address 10.10.10.50 --protocol-port
80 pool1
```

```
openstack loadbalancer member create --subnet-id private --address 10.10.10.51 --protocol-port
80 pool1
```

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
2005	Contrail Networking Release 2005 supports Octavia as LBaaS.

RELATED DOCUMENTATION

- https://wiki.openstack.org/wiki/Barbican
- http://developer.openstack.org/api-ref-networking-v2-ext.html
- https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/15/html/networking_guide/sec-octavia
- https://docs.openstack.org/octavia/queens/user/guides/basic-cookbook.html
- Installing OpenStack Octavia LBaaS with RHOSP in Contrail Networking | **513**
- Installing OpenStack Octavia LBaaS with Juju Charms in Contrail Networking | **670**
- Using Load Balancers in Contrail | **550**
- Configuring Load Balancing as a Service in Contrail | **568**

Configuring Load Balancing as a Service in Contrail

IN THIS SECTION

Overview: Load Balancing as a Service

Load Balancing as a Service (LBaaS) is a feature available through OpenStack Neutron. Contrail Release 1.20 and greater allows the use of the Neutron API for LBaaS to apply open source load balancing technologies to provision a load balancer in the Contrail system.

The LBaaS load balancer enables the creation of a pool of virtual machines serving applications, all front-ended by a virtual-ip. The LBaaS implementation has the following features:

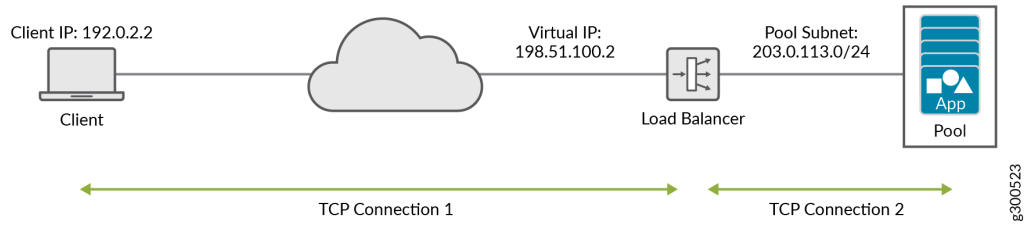
- Load balancing of traffic from clients to a pool of backend servers. The load balancer proxies all connections to its virtual IP.
- Provides load balancing for HTTP, TCP, and HTTPS.
- Provides health monitoring capabilities for applications, including HTTP, TCP, and ping.
- Enables floating IP association to `virtual-ip` for public access to the backend pool.

In Figure 62 on page 569, the load balancer is launched with the virtual IP address 198.51.100.2. The backend pool of virtual machine applications (App Pool) is on the subnet 203.0.113.0/24. Each of the application virtual machines gets an IP address (virtual-ip) from the pool subnet. When a client connects to the `virtual-ip` for accessing the application, the load balancer proxies the TCP connection on its `virtual-ip`, then creates a new TCP connection to one of the virtual machines in the pool.

The pool member is selected using one of following methods:

- weighted round robin (WRR), based on the weight assignment
- least connection, selects the member with the fewest connections
- source IP selects based on the `source-ip` of the packet

Figure 62: Load Balancing as a Service in Contrail



Additionally, the load balancer monitors the health of each pool member using the following methods:

- Monitors TCP by creating a TCP connection at intervals.

- Monitors HTTP by creating a TCP connection and issuing an HTTP request at intervals.
- Monitors ping by checking if a member can be reached by pinging.

Contrail LBaaS Implementation

Contrail supports the OpenStack LBaaS Neutron APIs and creates relevant objects for LBaaS, including `virtual-ip`, `loadbalancer-pool`, `loadbalancer-member`, and `loadbalancer-healthmonitor`. Contrail creates a service instance when a `loadbalancer-pool` is associated with a `virtual-ip` object. The service scheduler then launches a namespace on a randomly selected virtual router and spawns HAProxy into that namespace. The configuration for HAProxy is picked up from the load balancer objects. Contrail supports high availability of namespaces and HAProxy by spawning active and standby on two different vrouters.

A Note on Installation

To use the LBaaS feature, HAProxy, version 1.5 or greater and `iproute2`, version 3.10.0 or greater must both be installed on the Contrail compute nodes.

If you are using fabic commands for installation, the haproxy and iproute2 packages will be installed automatically with LBaaS if you set the following:

```
env.enable_lbaas=True
```

Use the following to check the version of the `iproute2` package on your system and verify the installation:

```
root@nodeh5:/var/log# ip -V
ip utility, iproute2-ss130716
root@nodeh5:/var/log#
```

You can also view the server yml file to verify the `env.enable_lbaas=True`.

Limitations

LBaaS currently has these limitations:

- A pool should not be deleted before deleting the VIP.
- Multiple VIPs cannot be associated with the same pool. If pool needs to be reused, create another pool with the same members and bind it to the second VIP.
- Members cannot be moved from one pool to another. If needed, first delete the members from one pool, then add to a different pool.
- In case of active-standby failover, namespaces might not get cleaned up when the agent restarts.

- The floating-ip association needs to select the VIP port and not the service ports.

Configuring LBaaS Using CLI

The LBaaS feature is enabled on Contrail through Neutron API calls. The following procedure shows how to create a pool network and a VIP network using CLI. The VIP network is created in the public network and members are added in the pool network.

NOTE: The following procedures are written using the *Neutron* LBaaS plugin v1.0. Starting with the OpenStack Train release, *neutron-lbaas* is replaced by *Octavia*. Some commands are different due to the plugin change. See the Red Hat *Octavia* documentation for the equivalent procedure: https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/15/html/networking_guide/sec-octavia

Creating a Load Balancer

Use the following steps to create a load balancer in Contrail.

1. Create a VIP network.

```
neutron net-create vipnet

neutron subnet-create --name vipsubnet vipnet 198.51.100.2
```

2. Create a pool network.

```
neutron net-create poolnet

neutron subnet-create --name poolsubnet poolnet 203.0.113.0/24
```

3. Create a pool for HTTP.

```
neutron lb-pool-create --lb-method ROUND_ROBIN --name mypool --protocol HTTP --subnet-id poolsubnet
```

4. Add members to the pool.

```
neutron lb-member-create --address 203.0.113.3 --protocol-port 80 mypool

neutron lb-member-create --address 203.0.113.4 --protocol-port 80 mypool
```

5. Create a VIP for HTTP and associate it to the pool.

```
neutron lb-vip-create --name myvip --protocol-port 80 --protocol HTTP--subnet-id vipsubnet mypool
```

Deleting a Load Balancer

Use the following steps to delete a load balancer in Contrail.

1. Delete the VIP.

   ```
   neutron lb-vip-delete <vip-uuid>
   ```

2. Delete members from the pool.

   ```
   neutron lb-member-delete <member-uuid>
   ```

3. Delete the pool.

   ```
   neutron lb-pool-delete <pool-uuid>
   ```

Managing Healthmonitor for Load Balancer

Use the following commands to create a healthmonitor, associate a healthmonitor to a pool, disassociate a healthmonitor, and delete a healthmonitor.

1. Create a healthmonitor.

   ```
   neutron lb-healthmonitor-create --delay 20 --timeout 10 --max-retries 3 --type HTTP
   ```

2. Associate a healthmonitor to a pool.

   ```
   neutron lb-healthmonitor-associate <healthmonitor-uuid> mypool
   ```

3. Disassociate a healthmonitor from a pool.

   ```
   neutron lb-healthmonitor-disassociate <healthmonitor-uuid> mypool
   ```

Configuring an SSL VIP with an HTTP Backend Pool

Use the following steps to configure an SSL VIP with an HTTP backend pool.

1. Copy an SSL certificate to all compute nodes.

   ```
   scp ssl_certificate.pem <compute-node-ip> <certificate-path>
   ```

2. Update the information in **/etc/contrail/contrail-vrouter-agent.conf**.

```
# SSL certificate path haproxy

haproxy_ssl_cert_path=<certificate-path>
```

3. Restart `contrail-vrouter-agent`.

```
service contrail-vrouter-agent restart
```

4. Create a VIP for port 443 (SSL).

```
neutron lb-vip-create --name myvip --protocol-port 443 --protocol HTTP --subnet-id vipsubnet mypool
```

Configuring LBaaS using the Contrail Command UI

Create, edit, or delete load balancers using the Contrail Command UI. Use the following guidelines when creating load balancers:

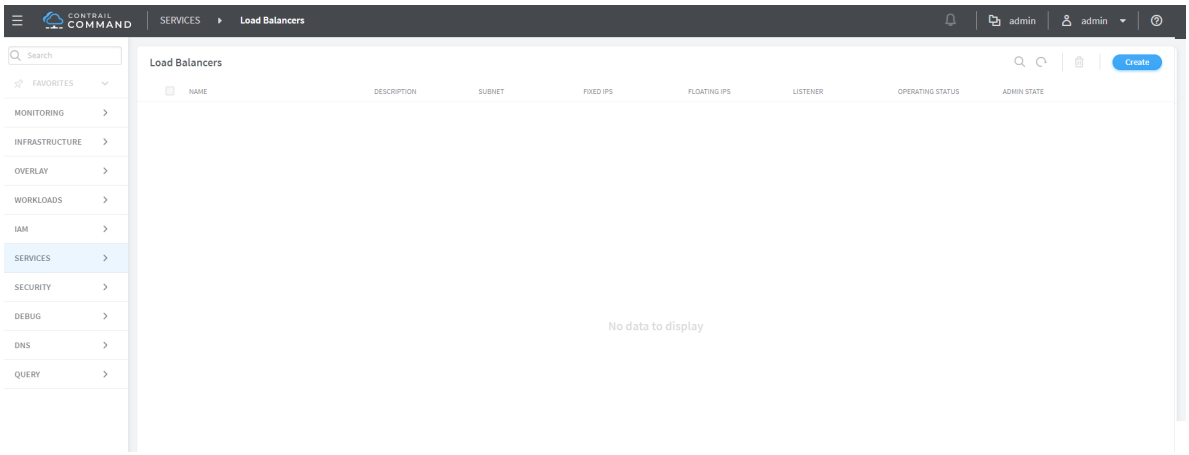
- Each load balancer consists of one or more listeners, pools, pool members, and health monitors.
  - Listener: Port that listens for traffic from a particular load balancer. Multiple listeners can be associated with a single load balancer.
  - Pool: Group of hosts that serves traffic from the load balancer.
  - Pool Member: Server that is specified by the IP address and port for which it uses to serve the traffic it receives from the load balancer.
  - Health Monitor: Health monitors are associated with pools and help divert traffic away from pool members that are temporarily offline.
- Each load balancer can have multiple pools with one or more listeners for each pool.
- The native load balancer has a single pool that is shared among multiple listeners.

Creating a Load Balancer

Use the following steps to create a load balancer with the load balancer wizard.

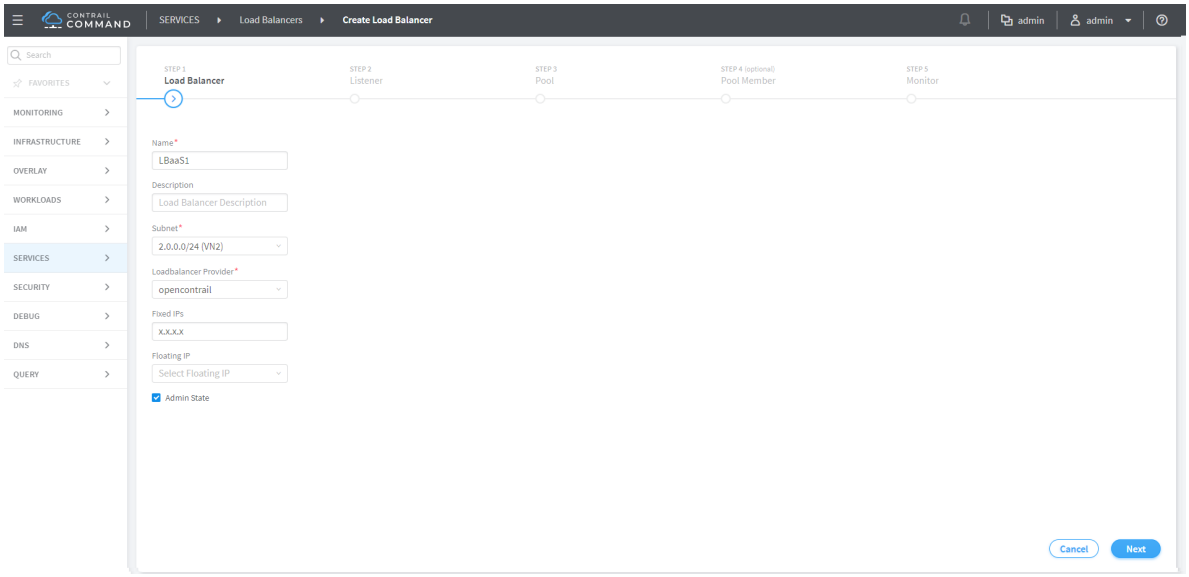
1. Go to **Services > Load Balancers**.

Figure 63: Create Load Balancers



2. To create a load balancer, click **Create**.

Figure 64: Load Balancer Information



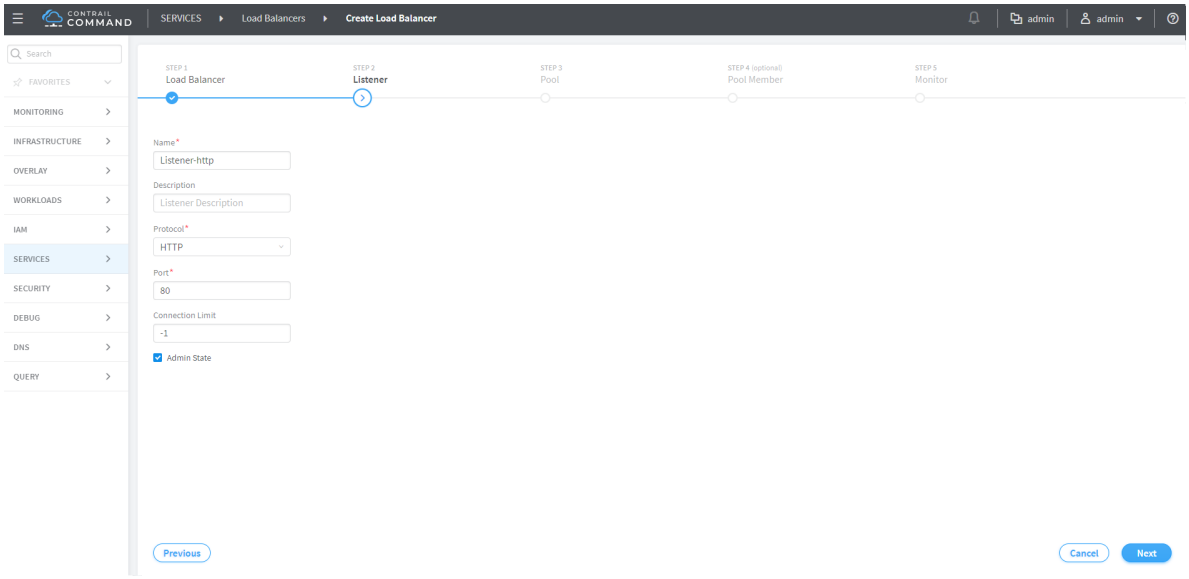
Add the load balancer information:

- **Name**: Name of the load balancer.
- **Description**: (Optional) Description of the load balancer.
- **Subnet**: Drop-down menu displays all subnets from list of all available networks. The subnet is the network on which to allocate the IP address of the load balancer.
- **Loadbalancer Provider**: Drop-down menu includes available options. Default is opencontrail.

- **Fixed IPs**: (Optional) IPv4 or IPv6 address.
- **Floating IP**: (Optional) IPv4 or IPv6 address.
- **Admin State**: Check the checkbox for UP or uncheck the checkbox for DOWN. Default is UP.

3. Click **Next**. The Listener fields are displayed.

Figure 65: Listener Information

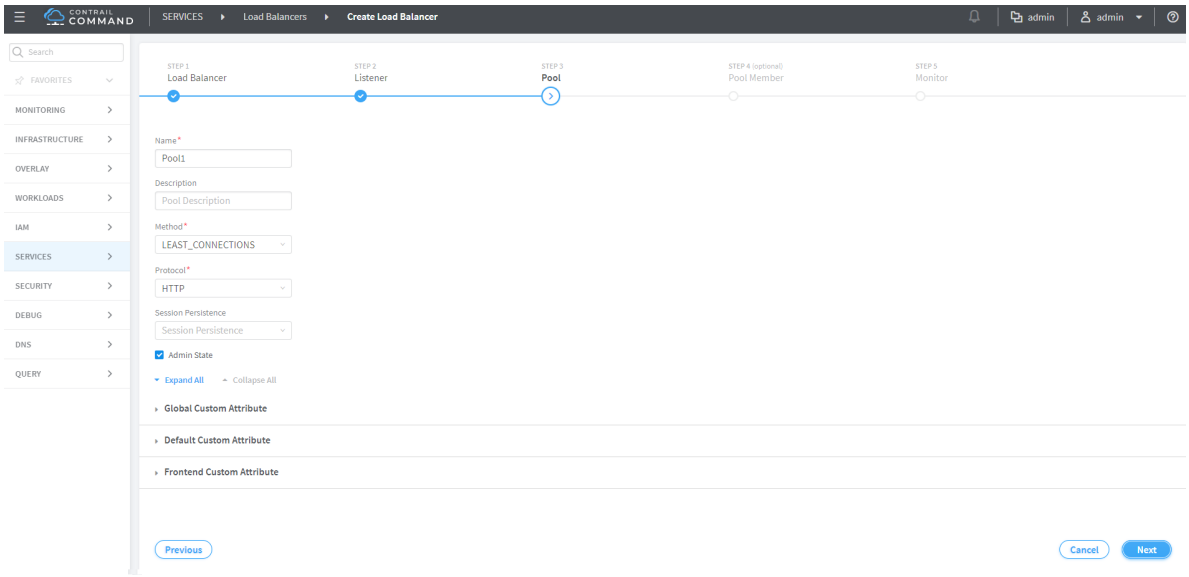


Add the listener information:

- **Name**: Name of the listener.
- **Description**: (Optional) Description of the listener.
- **Protocol**: Dropdown menu includes HTTP and TCP.
- **Port**: Must be an integer in the range of 1 to 65535.
- **Connection Limit**: (Optional). -1indicates an infinite limit.
- **Admin State**: Check the checkbox for UP or uncheck the checkbox for DOWN. Default is UP.

4. Click **Next**. The Pool fields are displayed.

Figure 66: Pool Information



Add the pool information:

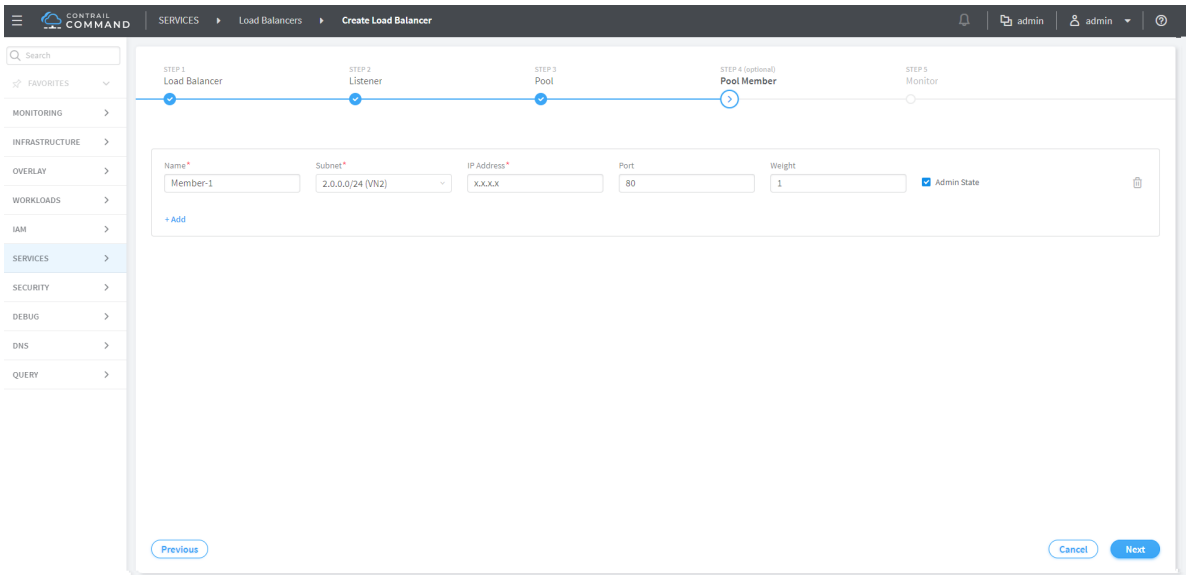
- **Name**: Name of the pool.
- **Description**: (Optional) Description of the pool.
- **Method**: Load balancing method used to distribute incoming requests. Dropdown menu includes LEAST_CONNECTIONS, ROUND_ROBIN, and SOURCE_IP.
- **Protocol**: The protocol used by the pool and its members for the load balancer traffic. Dropdown menu includes TCP and HTTP.
- **Session Persistence**: (Optional) Default value is an empty dictionary.
- **Admin State**: Check the checkbox for UP or uncheck the checkbox for DOWN. Default is UP.

5. Click **Next**. The list of available pool member instances are displayed. To add an external member, click the

   ➕

   Add icon. Each pool member must have a unique IP address and port combination.

Figure 67: Pool Member Information

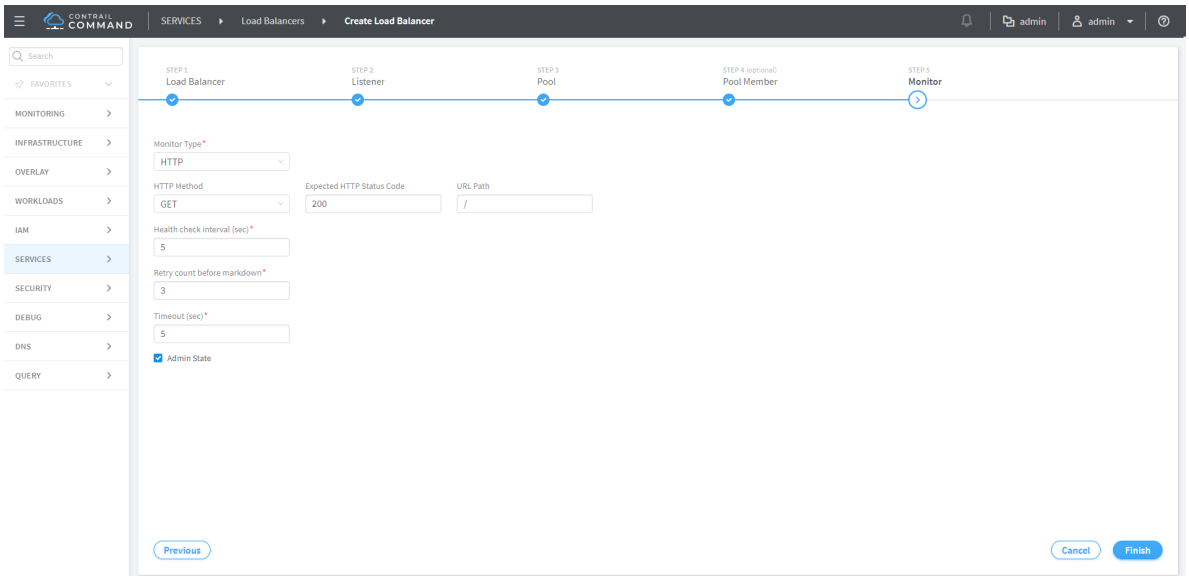


The pool member information includes:

- **Name**: Name of the pool member.
- **Subnet**: The subnet in which to access the member.
- **IP Address**: The IP address of the member that is used to receive traffic from the load balancer.
- **Port**: The port to which the member listens to receive traffic from the load balancer.
- **Weight**: The default value is 1.
- **Admin State**: Check the checkbox for UP or uncheck the checkbox for DOWN. Default is UP.

6. Click **Next**. The Monitor fields are displayed.

Figure 68: Health Monitor Information



Add the health monitor information:

- **Monitor Type**: Dropdown menu includes HTTP, PING, and TCP.
- **HTTP Method**: Required if monitor type is HTTP. Dropdown menu includes GET and HEAD. The default value is GET.
- **Expected HTTP Status Code**: Required if monitor type is HTTP. The default value is 200.
- **URL Path**: Required if monitor type is HTTP. The default value is "/."
- **Health check interval (sec)**: The time interval, in seconds, between each health check. The default value is 5.
- **Retry count before markdown**: The maximum number of failed health checks before the state of a member is changed to OFFLINE. The default value is 3.
- **Timeout (sec)**: The maximum number of seconds allowed for any given health check to complete. The timeout value should always be less than the health check interval. The default value is 5.
- **Admin State**: Check the checkbox for UP or uncheck the checkbox for DOWN. Default is UP.

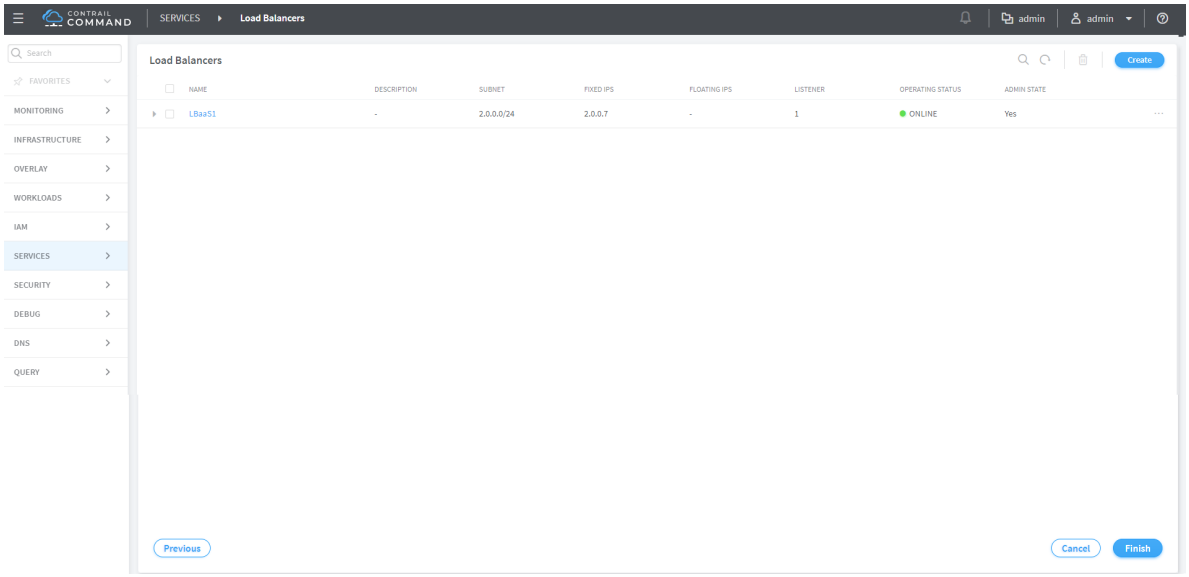
Click **Finish**.

Viewing or Editing Load Balancers

Use the following steps to view or edit existing load balancers.

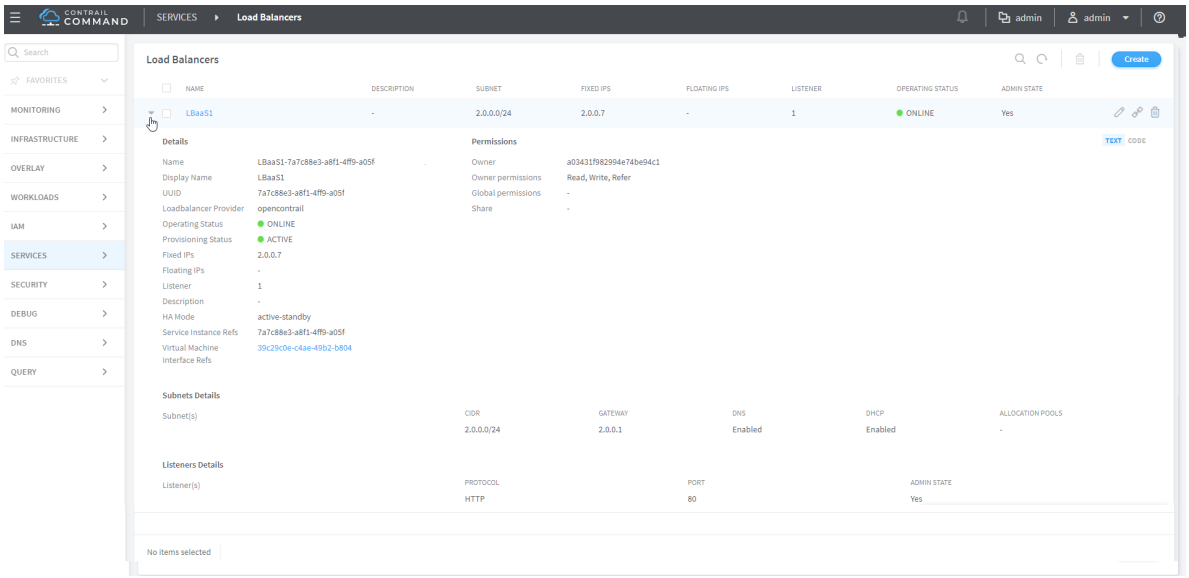
1. Go to **Services > Load Balancers**. A summary screen of the Load Balancers is displayed.

Figure 69: Summary Screen of Load Balancers



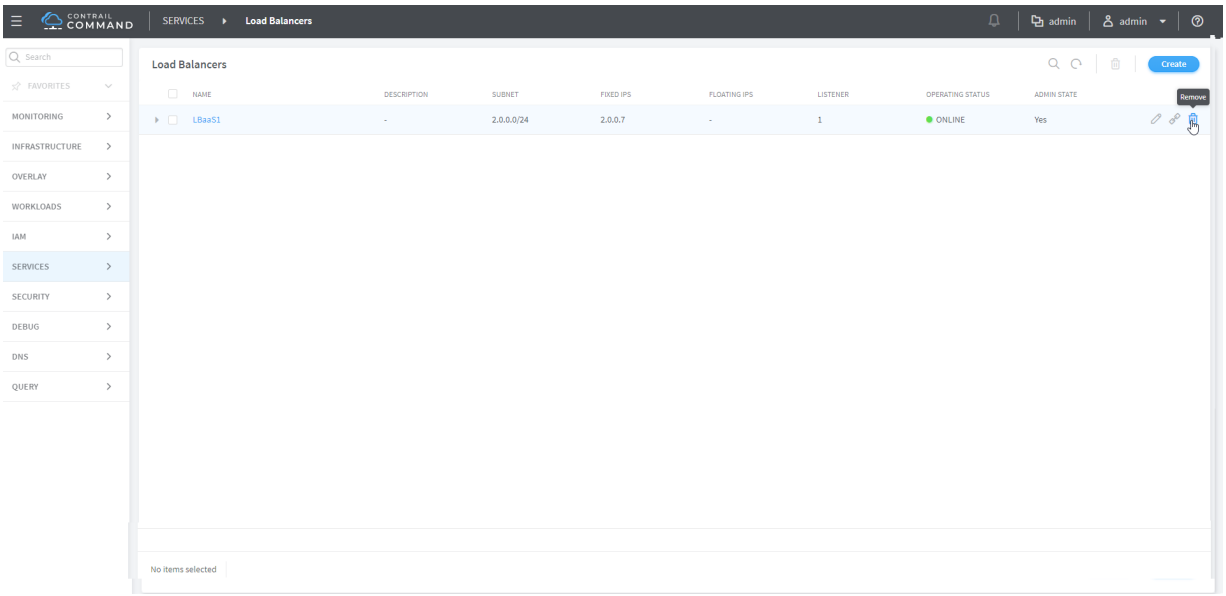
2. To view summary of a load balancer, click the drop down arrow next to a load balancer listed in the summary screen. The Load Balancer Info window is displayed.

Figure 70: Load Balancer Info Window



Deleting a Load Balancer

To delete the load balancer, click the trash can icon.



RELATED DOCUMENTATION

- Installing OpenStack Octavia LBaaS with RHOSP in Contrail Networking | **513**
- Installing OpenStack Octavia LBaaS with Juju Charms in Contrail Networking | **670**
- Using Load Balancers in Contrail | **550**
- Support for OpenStack LBaaS | **564**

CHAPTER 14

Optimizing Contrail Networking

IN THIS CHAPTER

Multiqueue Virtio Interfaces in Virtual Machines

IN THIS SECTION

Contrail 3.2 adds support for multiqueue for the DPDK-based vrouter.

Contrail 3.1 supports multiqueue virtio interfaces for Ubuntu kernel-based router, only.

Multiqueue Virtio Overview

OpenStack Liberty supports the ability to create VMs with multiple queues on their virtio interfaces. Virtio is a Linux platform for I/O virtualization, providing a common set of I/O virtualization drivers. Multiqueue virtio is an approach that enables the processing of packet sending and receiving to be scaled to the number of available virtual CPUs (vCPUs) of a guest, through the use of multiple queues.

Requirements and Setup for Multiqueue Virtio Interfaces

To use multiqueue virtio interfaces, ensure your system meets the following requirements:

- The OpenStack version must be Liberty or greater.

- The maximum number of queues in the VM interface is set to the same value as the number of vCPUs in the guest.
- The VM image metadata property is set to enable multiple queues inside the VM.

Setting Virtual Machine Metadata for Multiple Queues

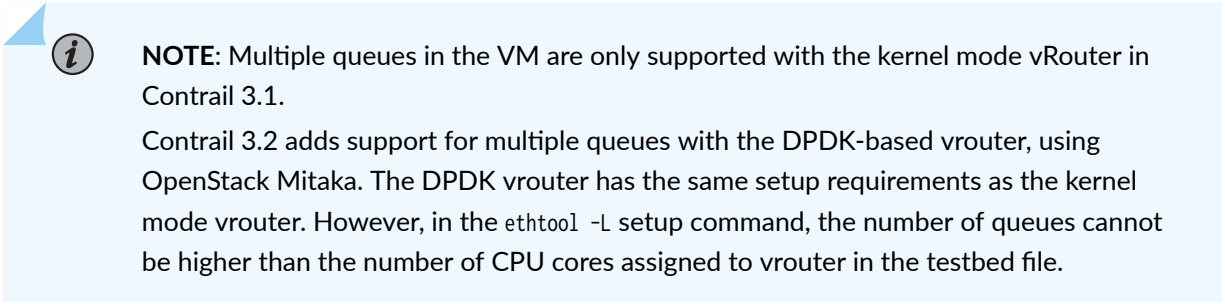
Use the following command on the OpenStack node to enable multiple queues on a VM:

```
source /etc/contrail/openstackrc
nova image-meta <image_name> set hw_vif_multiqueue_enabled="true"
```

After the VM is spawned, use the following command on the virtio interface in the guest to enable multiple queues inside the VM:

```
ethtool -L <interface_name> combined <#queues>
```

Packets will now be forwarded on all queues in the VM to and from the vRouter running on the host.



NOTE: Multiple queues in the VM are only supported with the kernel mode vRouter in Contrail 3.1.

Contrail 3.2 adds support for multiple queues with the DPDK-based vrouter, using OpenStack Mitaka. The DPDK vrouter has the same setup requirements as the kernel mode vrouter. However, in the `ethtool -L` setup command, the number of queues cannot be higher than the number of CPU cores assigned to vrouter in the testbed file.

CHAPTER 15

Contrail Networking OpenStack Analytics

IN THIS CHAPTER

Ceilometer Support in Contrail

IN THIS SECTION

Ceilometer is an OpenStack feature that provides an infrastructure for collecting SDN metrics from OpenStack projects. The metrics can be used by various rating engines to transform events into billable items. The Ceilometer collection process is sometimes referred to as “metering”. The Ceilometer service provides data that can be used by platforms that provide metering, tracking, billing, and similar services. This topic describes how to configure the Ceilometer service for Contrail.

Overview

Contrail Release 2.20 and later supports the OpenStack Ceilometer service, on the OpenStack Juno release on Ubuntu 14.04.1 LTS.

The prerequisites for installing Ceilometer are:

- Contrail Cloud installation

- Provisioned using **enable_ceilometer = True** in the **provisioning** file.

NOTE: Ceilometer services are only installed on the first OpenStack controller node and do not support high availability in Contrail Release 2.20.

Ceilometer Details

Ceilometer is used to reliably collect measurements of the utilization of the physical and virtual resources comprising deployed clouds, persist these data for subsequent retrieval and analysis, and trigger actions when defined criteria are met.

The Ceilometer architecture consists of:

Polling agent	Agent designed to poll OpenStack services and build meters. The polling agents are also run on the compute nodes in addition to the OpenStack controller.
Notification agent	Agent designed to listen to notifications on message queue and convert them to events and samples.
Collector	Gathers and records event and metering data created by the notification and polling agents.
API server	Provides a REST API to query and view data recorded by the collector service.
Alarms	Daemons to evaluate and notify based on defined alarming rules.
Database	Stores the metering data, notifications, and alarms. The supported databases are MongoDB, SQL-based databases compatible with SQLAlchemy, and HBase. The recommended database is MongoDB, which has been thoroughly tested with Contrail and deployed on a production scale.

Verification of Ceilometer Operation

The Ceilometer services are named slightly differently on the Ubuntu and RHEL Server 7.0.

On Ubuntu, the service names are:

Polling agent	`ceilometer-agent-central` and `ceilometer-agent-compute`
Notification agent	`ceilometer-agent-notification`
Collector	`ceilometer-collector`
API Server	`ceilometer-api`

Alarms `ceilometer-alarm-evaluator` and `ceilometer-alarm-notifier`

On RHEL Server 7.0, the service names are:

Polling agent `openstack-ceilometer-central` and `openstack-ceilometer-compute`

Notification agent `openstack-ceilometer-notification`

Collector `openstack-ceilometer-collector`

API server `openstack-ceilometer-api`

Alarms `openstack-ceilometer-alarm-evaluator` and `openstack-ceilometer-alarm-notifier`

To verify the Ceilometer installation, users can verify that the Ceilometer services are up and running by using the `openstack-status` command.

For example, using the **openstack-status** command on an all-in-one node running Ubuntu 14.04.1 LTS with release 2.2 of Contrail installed shows the following Ceilometer services as active:

```
== Ceilometer services ==
ceilometer-api:              active
ceilometer-agent-central:    active
ceilometer-agent-compute:    active
ceilometer-collector:        active
ceilometer-alarm-notifier:   active
ceilometer-alarm-evaluator:  active
ceilometer-agent-notification:active
```

You can issue the `ceilometer meter-list` command on the OpenStack controller node to verify that meters are being collected, stored, and reported via the REST API. The following is an example of the output:

```
user@host:~# (source /etc/contrail/openstackrc; ceilometer meter-list)
+------------------------------+------------+---------+--------------------------------------
+----------------------------------+----------------------------------+
| Name                         | Type       | Unit    | Resource ID                          |
User ID                          | Project ID                       |
+------------------------------+------------+---------+--------------------------------------
+----------------------------------+----------------------------------+
| ip.floating.receive.bytes    | cumulative | B       | a726f93a-65fa-4cad-828b-54dbfcf4a119 |
None                             | None                             |
| ip.floating.receive.packets  | cumulative | packet  | a726f93a-65fa-4cad-828b-54dbfcf4a119 |
None                             | None                             |
```

```
| ip.floating.transmit.bytes   | cumulative | B       | a726f93a-65fa-4cad-828b-54dbfcf4a119 |
None                           | None                           |
| ip.floating.transmit.packets | cumulative | packet  | a726f93a-65fa-4cad-828b-54dbfcf4a119 |
None                           | None                           |
| network                      | gauge      | network | 7fa6796b-756e-4320-9e73-87d4c52ecc83 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network                      | gauge      | network | 9408e287-d3e7-41e2-89f0-5c691c9ca450 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network                      | gauge      | network | b3b72b98-f61e-4e1f-9a9b-84f4f3ddec0b |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network                      | gauge      | network | cb829abd-e6a3-42e9-a82f-0742db55d329 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create               | delta      | network | 7fa6796b-756e-4320-9e73-87d4c52ecc83 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create               | delta      | network | 9408e287-d3e7-41e2-89f0-5c691c9ca450 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create               | delta      | network | b3b72b98-f61e-4e1f-9a9b-84f4f3ddec0b |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create               | delta      | network | cb829abd-e6a3-42e9-a82f-0742db55d329 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| port                         | gauge      | port    | 0d401d96-c2bf-4672-abf2-880eecf25ceb |
01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port                         | gauge      | port    | 211b94a4-581d-45d0-8710-c6c69df15709 |
01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port                         | gauge      | port    | 2287ce25-4eef-4212-b77f-3cf590943d36 |
01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port.create                  | delta      | port    | f62f3732-222e-4c40-8783-5bcbc1fd6a1c |
01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port.create                  | delta      | port    | f8c89218-3cad-48e2-8bd8-46c1bc33e752 |
01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port.update                  | delta      | port    | 43ed422d-b073-489f-877f-515a3cc0b8c4 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet                       | gauge      | subnet  | 09105ed1-1654-4b5f-8c12-f0f2666fa304 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet                       | gauge      | subnet  | 4bf00aac-407c-4266-a048-6ff52721ad82 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet.create                | delta      | subnet  | 09105ed1-1654-4b5f-8c12-f0f2666fa304 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet.create                | delta      | subnet  | 4bf00aac-407c-4266-a048-6ff52721ad82 |
15c0240142084d16b3127d6f844adbd9 | ded208991de34fe4bb7dd725097f1c7e |
+------------------------------+------------+---------+--------------------------------------
+----------------------------------+----------------------------------+
```

NOTE: The `ceilometer meter-list` command lists the meters only if images have been created, or instances have been launched, or if subnet, port, floating IP addresses have been created, otherwise the meter list is empty. You also need to source the **/etc/contrail/openstackrc** file when executing the command.

Contrail Ceilometer Plugin

The Contrail Ceilometer plugin adds the capability to meter the traffic statistics of floating IP addresses in Ceilometer. The following meters for each floating IP resource are added by the plugin in Ceilometer.

```
ip.floating.receive.bytes
ip.floating.receive.packets
ip.floating.transmit.bytes
ip.floating.transmit.packets
```

The Contrail Ceilometer plugin configuration is done in the **/etc/ceilometer/pipeline.yaml** file when Contrail is installed by the Fabric provisioning scripts.

The following example shows the configuration that is added to the file:

```
sources:
    - name: contrail_source
      interval: 600
      meters:
        - "ip.floating.receive.packets"
        - "ip.floating.transmit.packets"
        - "ip.floating.receive.bytes"
        - "ip.floating.transmit.bytes"
      resources:
          - contrail://<IP-address-of-Contrail-Analytics-Node>:8081
      sinks:
        - contrail_sink
sinks:
    - name: contrail_sink
      publishers:
        - rpc://
      transformers:
```

The following example shows the Ceilometer meter list output for the floating IP meters:

```
+------------------------------+------------+-----------
+--------------------------------------------------------------------
+----------------------------------+----------------------------------+
| Name                           | Type       | Unit      | Resource
ID                                                        | User ID
| Project ID                       |
+------------------------------+------------+-----------
+--------------------------------------------------------------------
+----------------------------------+----------------------------------+
| ip.floating.receive.bytes      | cumulative | B         | 451c93eb-
e728-4ba1-8665-6e7c7a8b49e2                                    | None
| None                             |
| ip.floating.receive.bytes      | cumulative | B         | 9cf76844-8f09-4518-a09e-
e2b8832bf894                                  | None                             |
None                              |
| ip.floating.receive.packets    | cumulative | packet    | 451c93eb-
e728-4ba1-8665-6e7c7a8b49e2                                    | None
| None                             |
| ip.floating.receive.packets    | cumulative | packet    | 9cf76844-8f09-4518-a09e-
e2b8832bf894                                  | None                             |
None                              |
| ip.floating.transmit.bytes     | cumulative | B         | 451c93eb-
e728-4ba1-8665-6e7c7a8b49e2                                    | None
| None                             |
| ip.floating.transmit.bytes     | cumulative | B         | 9cf76844-8f09-4518-a09e-
e2b8832bf894                                  | None                             |
None                              |
| ip.floating.transmit.packets   | cumulative | packet    | 451c93eb-
e728-4ba1-8665-6e7c7a8b49e2                                    | None
| None                             |
| ip.floating.transmit.packets   | cumulative | packet    | 9cf76844-8f09-4518-a09e-
e2b8832bf894                                  | None                             |
None                              |
```

In the meter -list output, the Resource ID refers to the floating IP.

The following example shows the output from the `ceilometer resource-show -r 451c93eb-e728-4ba1-8665-6e7c7a8b49e2` command:

```
+-------------+--------------------------------------------------------------------------+
| Property    | Value                                                                    |
+-------------+--------------------------------------------------------------------------+
| metadata    | {u'router_id': u'None', u'status': u'ACTIVE', u'tenant_id':              |
|             | u'ceed483222f9453ab1d7bcdd353971bc', u'floating_network_id':             |
|             | u'6d0cca50-4be4-4b49-856a-6848133eb970', u'fixed_ip_address':            |
|             | u'2.2.2.4', u'floating_ip_address': u'3.3.3.4', u'port_id': u'c6ce2abf- |
|             | ad98-4e56-ae65-ab7c62a67355', u'id':                                     |
|             | u'451c93eb-e728-4ba1-8665-6e7c7a8b49e2', u'device_id':                   |
|             | u'00953f62-df11-4b05-97ca-30c3f6735ffd'}                                 |
| project_id  | None                                                                     |
| resource_id | 451c93eb-e728-4ba1-8665-6e7c7a8b49e2                                     |
| source      | openstack                                                                |
| user_id     | None                                                                     |
+-------------+--------------------------------------------------------------------------+
```

The following example shows the output from the `ceilometer statistics` command and the `ceilometer sample-list` command for the **ip.floating.receive.packets** meter:

```
+--------+---------------------------+---------------------------+-------+-----+-------
+--------+---------------+------------+---------------------------+---------------------------
+
| Period | Period Start               | Period End                 | Count | Min | Max   |
Sum    | Avg            | Duration   | Duration Start             | Duration End               |
+--------+---------------------------+---------------------------+-------+-----+-------
+--------+---------------+------------+---------------------------+---------------------------
+
| 0      | 2015-02-13T19:50:40.795000 | 2015-02-13T19:50:40.795000 | 2892  | 0.0 | 325.0 |
1066.0 | 0.368603042877 | 439069.674 | 2015-02-13T19:50:40.795000 | 2015-02-18T21:48:30.469000 |
+--------+---------------------------+---------------------------+-------+-----+-------
+--------+---------------+------------+---------------------------+---------------------------
+

+--------------------------------------+----------------------------+------------+--------
+--------+---------------------------+
| Resource ID                          | Name                       | Type       | Volume |
Unit   | Timestamp                 |
+--------------------------------------+----------------------------+------------+--------
```

```
+--------+---------------------------+
| 9cf76844-8f09-4518-a09e-e2b8832bf894 | ip.floating.receive.packets | cumulative | 208.0  |
packet | 2015-02-18T21:48:30.469000 |
| 451c93eb-e728-4ba1-8665-6e7c7a8b49e2 | ip.floating.receive.packets | cumulative | 325.0  |
packet | 2015-02-18T21:48:28.354000 |
| 9cf76844-8f09-4518-a09e-e2b8832bf894 | ip.floating.receive.packets | cumulative | 0.0    |
packet | 2015-02-18T21:38:30.350000 |
```

Ceilometer Installation and Provisioning

There are two scenarios possible for Contrail Ceilometer plugin installation.

1. If you install your own OpenStack distribution, you can install the Contrail Ceilometer plugin on the OpenStack controller node.

2. When using Contrail Cloud services, the Ceilometer controller services are installed and provisioned as part of the OpenStack controller node and the compute agent service is installed as part of the compute node when `enable_ceilometer` is set as `True` in the cluster **config** or **testbed** files.

CHAPTER 16

Contrail OpenStack APIs

IN THIS CHAPTER

Working with Neutron

IN THIS SECTION

OpenStack's networking solution, Neutron, has representative elements for Contrail elements for Network (VirtualNetwork), Port (VirtualMachineInterface), Subnet (IpamSubnets), and Security-Group. The Neutron plugin translates the elements from one representation to another.

Data Structure

Although the actual data between Neutron and Contrail is similar, the listings of the elements differs significantly. In the Contrail API, the networking elements list is a summary, containing only the UUID, FQ name, and an href, however, in Neutron, all details of each resource are included in the list.

The Neutron plugin has an inefficient list retrieval operation, especially at scale, because it:

- reads a list of resources (for example. `GET /virtual-networks`), then

- iterates and reads in the details of the resource (`GET /virtual-network/<uuid>`).

As a result, the API server spends most of the time in this type of GET operation just waiting for results from the Cassandra database.

The following features in Contrail improve performance with Neutron:

- An optional detail query parameter is added in the GET of collections so that the API server returns details of all the resources in the list, instead of just a summary. This is accompanied by changes in the Contrail API library so that a caller gets returned a list of the objects.
- The existing Contrail list API takes in an optional `parent_id` query parameter to return information about the resource anchored by the parent.
- The Contrail API server reads objects from Cassandra in a multiget format into `obj_uuid_cf`, where object contents are stored, instead of reading in an xget/get format. This reduces the number of round-trips to and from the Cassandra database.

Network Sharing in Neutron

Using Neutron, a deployer can make a network accessible to other tenants or projects by using one of two attributes on a network:

- Set the `shared` attribute to allow sharing.
- Set the `router:external` attribute, when the plugin supports an `external_net` extension.

Using the Shared Attribute

When a network has the `shared` attribute set, users in other tenants or projects, including non-admin users, can access that network, using:

```
neutron net-list --shared
```

Users can also launch a virtual machine directly on that network, using:

```
nova boot <other-parameters> -nic net-id=<shared-net-id>
```

Using the Router:External Attribute

When a network has the `router:external` attribute set, users in other tenants or projects, including non-admin users, can use that network for allocating floating IPs, using:

```
neutron floatingip-create <router-external-net-id>
```

then associating the IP address pool with their instances.

NOTE: The VN hosting the FIP pool should be marked shared and external.

Commands for Neutron Network Sharing

The following table summarizes the most common Neutron commands used with Contrail.

Table 41: Neutron commands

Action	Command
List all shared networks.	`neutron net-list --shared`
Create a network that has the shared attribute.	`neutron net-create <net-name> -shared`
Set the shared attribute on an existing network.	`neutron net-update <net-name> -shared`
List all `router:external` networks.	`neutron net-list --router:external`
Create a network that has the `router:external`attribute.	`neutron net-create <net-name> -router:external`
Set the `router:external` attribute on an existing network.	`neutron net-update <net-name> -router:external`

Support for Neutron APIs

The OpenStack Neutron project provides virtual networking services among devices that are managed by the OpenStack compute service. Software developers create applications by using the OpenStack Networking API v2.0 (Neutron).

Contrail provides the following features to increase support for OpenStack Neutron:

- Create a port independently of a virtual machine.
- Support for more than one subnet on a virtual network.
- Support for allocation pools on a subnet.
- Per tenant quotas.
- Enabling DHCP on a subnet.

- External router can be used for floating IPs.

For more information about using OpenStack Networking API v2.0 (Neutron), refer to: http://docs.openstack.org/api/openstack-network/2.0/content/ and the OpenStack Neutron Wiki at: http://wiki.openstack.org/wiki/Neutron.

Contrail Neutron Plugin

The Contrail Neutron plugin provides an implementation for the following core resources:

- Network
- Subnet
- Port

It also implements the following standard and upstreamed Neutron extensions:

- Security group
- Router IP and floating IP
- Per-tenant quota
- Allowed address pair

The following Contrail-specific extensions are implemented:

- Network IPAM
- Network policy
- VPC table and route table
- Floating IP pools

The plugin does not implement native bulk, pagination, or sort operations and relies on emulation provided by the Neutron common code.

DHCP Options

In Neutron commands, DHCP options can be configured using extra-dhcp-options in port-create.

Example

```
neutron port-create net1 --extra-dhcp-opt opt_name=<dhcp_option_name>,opt_value=<value>
```

The opt_name and opt_value pairs that can be used are maintained in GitHub: https://github.com/Juniper/contrail-controller/wiki/Extra-DHCP-Options .

Incompatibilities

In the Contrail architecture, the following are known incompatibilities with the Neutron API.

- Filtering based on any arbitrary key in the resource is not supported. The only supported filtering is by `id`, `name`, and `tenant_id`.
- To use a floating IP, it is not necessary to connect the public subnet and the private subnet to a Neutron router. Marking a public network with `router:external` is sufficient for a floating IP to be created and associated, and packet forwarding to it will work.
- The default values for quotas are sourced from **/etc/contrail/contrail-api.conf** and not from **/etc/neutron/neutron.conf.**

CHAPTER 17

Using Contrail with Juju Charms

IN THIS CHAPTER

Installing Contrail with OpenStack by Using Juju Charms

IN THIS SECTION

You can deploy Contrail by using Juju Charms. Juju helps you deploy, configure, and efficiently manage applications on private clouds and public clouds. Juju accesses the cloud with the help of a Juju controller. A Charm is a module containing a collection of scripts and metadata and is used with Juju to deploy Contrail.

Starting in Contrail Networking Release 2011, Contrail Networking supports OpenStack Ussuri with Ubuntu version 18.04 (Bionic Beaver) and Ubuntu version 20.04 (Focal Fossa).

Contrail supports the following charms:

- contrail-agent

- contrail-analytics
- contrail-analyticsdb
- contrail-controller
- contrail-keystone-auth
- contrail-openstack

These topics describe how to deploy Contrail by using Juju Charms.

Preparing to Deploy Contrail by Using Juju Charms

Follow these steps to prepare for deployment:

1. Install Juju.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install juju
```

2. Configure Juju.

   You can add a cloud to Juju, identify clouds supported by Juju, and also manage clouds already added to Juju.

   - **Adding a cloud**—Juju recognizes a wide range of cloud types. You can use any one of the following methods to add a cloud to Juju:

     - **Adding a Cloud by Using Interactive Command**

       Example: Adding an MAAS cloud to Juju

```
juju add-cloud
Cloud Types
  maas
  manual
  openstack
  oracle
  vsphere

Select cloud type: maas

Enter a name for your maas cloud: maas-cloud
```

```
Enter the API endpoint url: http://<ip-address>:<node>/MAAS

Cloud "maas-cloud" successfully added
You may bootstrap with 'juju bootstrap maas-cloud'
```

NOTE: Juju 2.x is compatible with MAAS series 1.x and 2.x.

- **Adding a Cloud Manually**

  You use a YAML configuration file to add a cloud manually. Enter the following command:

```
juju add-cloud <cloud-name>
juju add-credential <cloud name>
```

  For an example, to add the cloud *junmaas*, assuming that the name of the configuration file in the directory is `maas-clouds.yaml`, you run the following command:

```
juju add-cloud junmaas maas-clouds.yaml
```

  The following is the format of the YAML configuration file:

```
clouds:
  <cloud_name>:
    type: <type_of_cloud>
    auth-types: [<authenticaton_types>]
    regions:
      <region-name>:
        endpoint: <http://<ip-address>:<node>/MAAS>
```

NOTE: The `auth-types` for a MAAS cloud type is `oauth1`.

- **Identifying a supported cloud**

Juju recognizes the cloud types given below. You use the `juju clouds` command to list cloud types that are supported by Juju.

```
$ juju clouds
Cloud        Regions  Default          Type        Description
aws               15  us-east-1        ec2         Amazon Web Services
aws-china          1  cn-north-1       ec2         Amazon China
aws-gov            1  us-gov-west-1    ec2         Amazon (USA Government)
azure             26  centralus        azure       Microsoft Azure
azure-china        2  chinaeast        azure       Microsoft Azure China
cloudsigma         5  hnl              cloudsigma  CloudSigma Cloud
google            13  us-east1         gce         Google Cloud Platform
joyent             6  eu-ams-1         joyent      Joyent Cloud
oracle             5  uscom-central-1  oracle      Oracle Cloud
rackspace          6  dfw              rackspace   Rackspace Cloud
localhost          1  localhost        lxd         LXD Container Hypervisor
```

3. Create a Juju controller.

```
juju bootstrap --bootstrap-series=xenial <cloud name> <controller name>
```

NOTE: A Juju controller manages and keeps track of applications in the Juju cloud environment.

Deploying Contrail Charms

IN THIS SECTION

You can deploy Contrail Charms in a bundle or manually.

Deploy Contrail Charms in a Bundle

Follow these steps to deploy Contrail Charms in a bundle.

1. Deploy Contrail Charms.

   To deploy Contrail Charms in a bundle, use the `juju deploy <bundle_yaml_file>` command.

   The following example shows you how to use **bundle_yaml_file** to deploy Contrail on Amazon Web Services (AWS) Cloud.

```
series: bionic

variables:
  openstack-origin:             &openstack-origin              distro
  #vhost-gateway:               &vhost-gateway                 "192.x.40.254"
  data-network:                 &data-network                  "192.x.40.0/24"
  control-network:              &control-network               "192.x.30.0/24"
  virtioforwarder-coremask:     &virtioforwarder-coremask      "1,2"
  agilio-registry:              &agilio-registry               "netronomesystems"
  agilio-image-tag:             &agilio-image-tag              "latest-ubuntu-queens"
  agilio-user:                  &agilio-user                   "<agilio-username>"
  agilio-password:              &agilio-password               "<agilio-password>"
  agilio-insecure:              &agilio-insecure               false
  agilio-phy:                   &agilio-phy                    "nfp_p0"
  docker-registry:              &docker-registry               "<registry-directory>"
  #docker-user:                 &docker-user                   "<docker_username>"
  #docker-password:             &docker-password               "<docker_password>"
  image-tag:                    &image-tag                     "2008.121"
  docker-registry-insecure:     &docker-registry-insecure      "true"
  dockerhub-registry:           &dockerhub-registry            "https://index.docker.io/v1/"

machines:
  "1":
    constraints: tags=controller
    series: bionic
  "2":
    constraints: tags=compute
    series: bionic
  "3":
    constraints: tags=neutron
    series: bionic
services:
  ubuntu:
    charm: cs:ubuntu
    num_units: 1
    to: [ "1" ]
```

```
  ntp:
    charm: cs:ntp
    num_units: 0
    options:
            #source: ntp.ubuntu.com
       source: 10.204.217.158
mysql:
    charm: cs:percona-cluster
    num_units: 1
    options:
      dataset-size: 15%
      max-connections: 10000
      root-password: <password>
      sst-password: <password>
      min-cluster-size: 1
    to: [ "lxd:1" ]
  rabbitmq-server:
num_units: 1
    options:
      min-cluster-size: 1
    to: [ "lxd:1" ]
 heat:
    charm: cs:heat
    num_units: 1
    expose: true
    options:
      debug: true
      openstack-origin: *openstack-origin
    to: [ "lxd:1" ]
keystone:
    charm: cs:keystone
    expose: true
    num_units: 1
    options:
      admin-password: <password>
      admin-role: admin
      openstack-origin: *openstack-origin
      preferred-api-version: 3
nova-cloud-controller:
    charm: cs:nova-cloud-controller
    num_units: 1
    expose: true
    options:
```

```
      network-manager: Neutron
      openstack-origin: *openstack-origin
    to: [ "lxd:1" ]
neutron-api:
    charm: cs:neutron-api
    expose: true
    num_units: 1
    series: bionic
    options:
      manage-neutron-plugin-legacy-mode: false
      openstack-origin: *openstack-origin
    to: [ "3" ]
glance:
    charm: cs:glance
    expose: true
    num_units: 1
    options:
      openstack-origin: *openstack-origin
    to: [ "lxd:1" ]
  openstack-dashboard:
    charm: cs:openstack-dashboard
    expose: true
    num_units: 1
    options:
      openstack-origin: *openstack-origin
    to: [ "lxd:1" ]
  nova-compute:
    charm: cs:nova-compute
    num_units: 0
    expose: true
    options:
      openstack-origin: *openstack-origin
  nova-compute-dpdk:
    charm: cs:nova-compute
    num_units: 0
    expose: true
    options:
      openstack-origin: *openstack-origin
  nova-compute-accel:
    charm: cs:nova-compute
    num_units: 2
    expose: true
    options:
```

```
    openstack-origin: *openstack-origin
  to: [ "2" ]
contrail-openstack:
  charm: ./tf-charms/contrail-openstack
  series: bionic
  expose: true
  num_units: 0
  options:
    docker-registry: *docker-registry
    #docker-user: *docker-user
    #docker-password: *docker-password
    image-tag: *image-tag
    docker-registry-insecure: *docker-registry-insecure
contrail-agent:
  charm: ./tf-charms/contrail-agent
  num_units: 0
  series: bionic
  expose: true
  options:
    log-level: "SYS_DEBUG"
    docker-registry: *docker-registry
    #docker-user: *docker-user
    #docker-password: *docker-password
    image-tag: *image-tag
    docker-registry-insecure: *docker-registry-insecure
    #vhost-gateway: *vhost-gateway
    physical-interface: *agilio-phy
contrail-agent-dpdk:
  charm: ./tf-charms/contrail-agent
  num_units: 0
  series: bionic
  expose: true
  options:
    log-level: "SYS_DEBUG"
    docker-registry: *docker-registry
    #docker-user: *docker-user
    #docker-password: *docker-password
    image-tag: *image-tag
    docker-registry-insecure: *docker-registry-insecure
    dpdk: true
    dpdk-main-mempool-size: "65536"
    dpdk-pmd-txd-size: "2048"
    dpdk-pmd-rxd-size: "2048"
```

```
    dpdk-driver: ""
    dpdk-coremask: "1-4"
    #vhost-gateway: *vhost-gateway
    physical-interface: "nfp_p0"
contrail-analytics:
  charm: ./tf-charms/contrail-analytics
  num_units: 1
  series: bionic
  expose: true
  options:
    log-level: "SYS_DEBUG"
    docker-registry: *docker-registry
    #docker-user: *docker-user
    #docker-password: *docker-password
    image-tag: *image-tag
    control-network: *control-network
    docker-registry-insecure: *docker-registry-insecure
  to: [ "1" ]
contrail-analyticsdb:
  charm: ./tf-charms/contrail-analyticsdb
  num_units: 1
  series: bionic
  expose: true
  options:
    log-level: "SYS_DEBUG"
    cassandra-minimum-diskgb: "4"
    cassandra-jvm-extra-opts: "-Xms8g -Xmx8g"
    docker-registry: *docker-registry
    #docker-user: *docker-user
    #docker-password: *docker-password
    image-tag: *image-tag
    control-network: *control-network
    docker-registry-insecure: *docker-registry-insecure
  to: [ "1" ]
contrail-controller:
  charm: ./tf-charms/contrail-controller
  series: bionic
  expose: true
  num_units: 1
  options:
    log-level: "SYS_DEBUG"
    cassandra-minimum-diskgb: "4"
    cassandra-jvm-extra-opts: "-Xms8g -Xmx8g"
```

```
      docker-registry: *docker-registry
      #docker-user: *docker-user
      #docker-password: *docker-password
      image-tag: *image-tag
      docker-registry-insecure: *docker-registry-insecure
      control-network: *control-network
      data-network: *data-network
      auth-mode: no-auth
    to: [ "1" ]
  contrail-keystone-auth:
    charm: ./tf-charms/contrail-keystone-auth
    series: bionic
    expose: true
    num_units: 1
    to: [ "lxd:1" ]
  agilio-vrouter5:
    charm: ./charm-agilio-vrt-5-37
    expose: true
    options:
      virtioforwarder-coremask: *virtioforwarder-coremask
      agilio-registry: *agilio-registry
      agilio-insecure: *agilio-insecure
      agilio-image-tag: *agilio-image-tag
      agilio-user: *agilio-user
      agilio-password: *agilio-password
relations:
  - [ "ubuntu", "ntp" ]
  - [ "neutron-api", "ntp" ]
  - [ "keystone", "mysql" ]
  - [ "glance", "mysql" ]
  - [ "glance", "keystone" ]
  - [ "nova-cloud-controller:shared-db", "mysql:shared-db" ]
  - [ "nova-cloud-controller:amqp", "rabbitmq-server:amqp" ]
  - [ "nova-cloud-controller", "keystone" ]
  - [ "nova-cloud-controller", "glance" ]
  - [ "neutron-api", "mysql" ]
  - [ "neutron-api", "rabbitmq-server" ]
  - [ "neutron-api", "nova-cloud-controller" ]
  - [ "neutron-api", "keystone" ]
  - [ "nova-compute:amqp", "rabbitmq-server:amqp" ]
  - [ "nova-compute", "glance" ]
  - [ "nova-compute", "nova-cloud-controller" ]
  - [ "nova-compute", "ntp" ]
```

```
  - [ "openstack-dashboard:identity-service", "keystone" ]
  - [ "contrail-keystone-auth", "keystone" ]
  - [ "contrail-controller", "contrail-keystone-auth" ]
  - [ "contrail-analytics", "contrail-analyticsdb" ]
  - [ "contrail-controller", "contrail-analytics" ]
  - [ "contrail-controller", "contrail-analyticsdb" ]
  - [ "contrail-openstack", "nova-compute" ]
  - [ "contrail-openstack", "neutron-api" ]
  - [ "contrail-openstack", "contrail-controller" ]
  - [ "contrail-agent:juju-info", "nova-compute:juju-info" ]
  - [ "contrail-agent", "contrail-controller"]
  - [ "contrail-agent-dpdk:juju-info", "nova-compute-dpdk:juju-info" ]
  - [ "contrail-agent-dpdk", "contrail-controller"]
  - [ "nova-compute-dpdk:amqp", "rabbitmq-server:amqp" ]
  - [ "nova-compute-dpdk", "glance" ]
  - [ "nova-compute-dpdk", "nova-cloud-controller" ]
  - [ "nova-compute-dpdk", "ntp" ]
  - [ "contrail-openstack", "nova-compute-dpdk" ]
  - [ "contrail-agent:juju-info", "nova-compute-accel:juju-info" ]
  - [ "nova-compute-accel:amqp", "rabbitmq-server:amqp" ]
  - [ "nova-compute-accel", "glance" ]
  - [ "nova-compute-accel", "nova-cloud-controller" ]
  - [ "nova-compute-accel", "ntp" ]
  - [ "contrail-openstack", "nova-compute-accel" ]
  - [ "agilio-vrouter5:juju-info", "nova-compute-accel:juju-info"  ]
```

You can create or modify the Contrail Charm deployment bundle YAML file to:

- Point to machines or instances where the Contrail Charms must be deployed.
- Include the options you need.

  Each Contrail Charm has a specific set of options. The options you choose depend on the charms you select. For more information on the options that are available, see "Options for Juju Charms" on page 612.

2. (Optional) Check the status of deployment.

   You can check the status of the deployment by using the `juju status` command.

3. Enable configuration statements.

   Based on your deployment requirements, you can enable the following configuration statements:

   - `contrail-agent`

     For more information, see https://jaas.ai/u/juniper-os-software/contrail-agent/.

- `contrail-analytics`

  For more information, see https://jaas.ai/u/juniper-os-software/contrail-analytics.

- `contrail-analyticsdb`

  For more information, see https://jaas.ai/u/juniper-os-software/contrail-analyticsdb.

- `contrail-controller`

  For more information, see https://jaas.ai/u/juniper-os-software/contrail-controller.

- `contrail-keystone-auth`

  For more information, see https://jaas.ai/u/juniper-os-software/contrail-keystone-auth.

- `contrail-openstack`

  For more information see, https://jaas.ai/u/juniper-os-software/contrail-openstack.

Deploying Juju Charms with OpenStack Manually

Before you begin deployment, ensure that you have:

- Installed and configured Juju
- Created a Juju controller
- Ubuntu 16.04 or Ubuntu 18.04 installed

Follow these steps to deploy Juju Charms manually:

1. Create machine instances for OpenStack, compute, and Contrail.

```
juju add-machine --constraints mem=8G cores=2 root-disk=40G --series=xenial   #for openstack
machine(s) 0
juju add-machine --constraints mem=7G cores=4 root-disk=40G --series=xenial   #for compute
machine(s) 1,(3)
juju add-machine --constraints mem=15G cores=2 root-disk=300G --series=xenial #for contrail
machine 2
```

2. Deploy OpenStack services.

   You can deploy OpenStack services by using any one of the following methods:

   - **By specifying the OpenStack parameters in a YAML file**

The following is an example of a YAML-formatted (**nova-compute-config.yaml**) file.

```
nova-compute:
    openstack-origin: cloud:xenial-ocata
    virt-type: qemu
    enable-resize: True
    enable-live-migration: True
    migration-auth-type: ssh
```

Use this command to deploy OpenStack services by using a YAML-formatted file:

```
juju deploy cs:xenial/nova-compute --config ./nova-compute-config.yaml
```

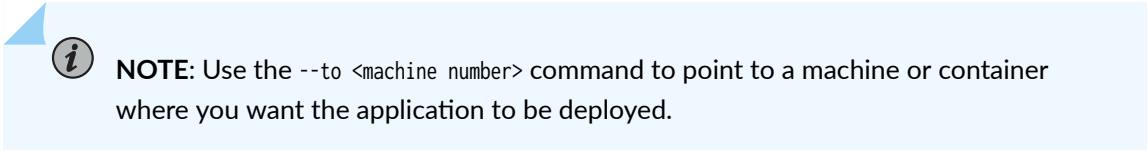
- **By using CLI**

  To deploy OpenStack services through the CLI:

```
juju deploy cs:xenial/nova-cloud-controller --config console-access-protocol=novnc --
config openstack-origin=cloud:xenial-ocata
```

- **By using a combination of YAML-formatted file and CLI**

  To deploy OpenStack services by using a combination of YAML-formatted file and CLI:



NOTE: Use the `--to <machine number>` command to point to a machine or container where you want the application to be deployed.

```
juju deploy cs:xenial/ntp
juju deploy cs:xenial/rabbitmq-server --to lxd:0
juju deploy cs:xenial/percona-cluster mysql --config root-password=<root-password> --
config max-connections=1500 --to lxd:0
juju deploy cs:xenial/openstack-dashboard --config openstack-origin=cloud:xenial-ocata --
to lxd:0
juju deploy cs:xenial/nova-cloud-controller --config console-access-protocol=novnc --
config openstack-origin=cloud:xenial-ocata --config network-manager=Neutron --to lxd:0
juju deploy cs:xenial/neutron-api --config manage-neutron-plugin-legacy-mode=false --
config openstack-origin=cloud:xenial-ocata --config neutron-security-groups=true --to lxd:0
juju deploy cs:xenial/glance --config openstack-origin=cloud:xenial-ocata --to lxd:0
```

```
juju deploy cs:xenial/keystone --config admin-password=<admin-password> --config admin-
role=admin --config openstack-origin=cloud:xenial-ocata --to lxd:0
```

NOTE: You set OpenStack services on different machines or on different containers to prevent HAProxy conflicts from applications.

3. Deploy and configure nova-compute.

```
juju deploy cs:xenial/nova-compute --config ./nova-compute-config.yaml --to 1
```

NOTE: You can deploy nova-compute to more than one compute machine.

(Optional) To add additional computes:

```
juju add-unit nova-compute --to 3 # Add one more unit
```

4. Deploy and configure Contrail services.

```
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-keystone-auth --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-controller --config
auth-mode=rbac --config cassandra-minimum-diskgb=4 --config cassandra-jvm-extra-opts="-Xms1g -
Xmx2g" --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-analyticsdb cassandra-
minimum-diskgb=4 --config cassandra-jvm-extra-opts="-Xms1g -Xmx2g" --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-analytics --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-openstack
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-agent
```

5. Enable applications to be available to external traffic:

```
juju expose openstack-dashboard
juju expose nova-cloud-controller
juju expose neutron-api
juju expose glance
juju expose keystone
```

6. Enable contrail-controller and contrail-analytics services to be available to external traffic if you do not use HAProxy.

```
juju expose contrail-controller
juju expose contrail-analytics
```

7. Apply SSL.

   You can apply SSL if needed. To use SSL with Contrail services, deploy easy-rsa service and `add-relation` command to create relations to contrail-controller service and contrail-agent services.

```
juju deploy cs:~containers/xenial/easyrsa --to 0
juju add-relation easyrsa contrail-controller
juju add-relation easyrsa contrail-agent
```

8. (Optional) HA configuration.

   If you use more than one controller, follow the HA solution given below:

   a. Deploy HAProxy and Keepalived services.

      HAProxy charm is deployed on machines with Contrail controllers. HAProxy charm must have `peering_mode` set to `active-active`. If `peering_mode` is set to `active-passive`, HAProxy creates additional listeners on the same ports as other Contrail services. This leads to port conflicts.

      Keepalived charm does not require `to` option.

```
juju deploy cs:xenial/haproxy --to <first contrail-controller machine> --config
peering_mode=active-active
juju add-unit haproxy --to <another contrail-controller machine>
juju deploy cs:~boucherv29/keepalived-19 --config virtual_ip=<vip>
```

   b. Enable HAProxy to be available to external traffic.

```
juju expose haproxy
```

NOTE: If you enable HAProxy to be available to external traffic, do not follow step 6.

c. Add HAProxy and Keepalived relations.

```
juju add-relation haproxy:juju-info keepalived:juju-info
juju add-relation contrail-analytics:http-services haproxy
juju add-relation contrail-controller:http-services haproxy
juju add-relation contrail-controller:https-services haproxy
```

d. Configure contrail-controller service with VIP.

```
juju set contrail-controller vip=<vip>
```

9. Add other necessary relations.

```
juju add-relation keystone:shared-db mysql:shared-db
juju add-relation glance:shared-db mysql:shared-db
juju add-relation keystone:identity-service glance:identity-service
juju add-relation nova-cloud-controller:image-service glance:image-service
juju add-relation nova-cloud-controller:identity-service keystone:identity-service
juju add-relation nova-cloud-controller:cloud-compute nova-compute:cloud-compute
juju add-relation nova-compute:image-service glance:image-service
juju add-relation nova-compute:amqp rabbitmq-server:amqp
juju add-relation nova-cloud-controller:shared-db mysql:shared-db
juju add-relation nova-cloud-controller:amqp rabbitmq-server:amqp
juju add-relation openstack-dashboard:identity-service keystone

juju add-relation neutron-api:shared-db mysql:shared-db
juju add-relation neutron-api:neutron-api nova-cloud-controller:neutron-api
juju add-relation neutron-api:identity-service keystone:identity-service
juju add-relation neutron-api:amqp rabbitmq-server:amqp

juju add-relation contrail-controller ntp
juju add-relation nova-compute:juju info ntp:juju info

juju add-relation contrail-controller contrail-keystone-auth
juju add-relation contrail-keystone-auth keystone
juju add-relation contrail-controller contrail-analytics
juju add-relation contrail-controller contrail-analyticsdb
juju add-relation contrail-analytics contrail-analyticsdb

juju add-relation contrail-openstack neutron-api
juju add-relation contrail-openstack nova-compute
```

```
juju add-relation contrail-openstack contrail-controller

juju add-relation contrail-agent:juju info nova-compute:juju info
juju add-relation contrail-agent contrail-controller
```

Options for Juju Charms

Each Contrail Charm has a specific set of options. The options you choose depend on the charms you select. The following tables list the various options you can choose:

- Options for **contrail-agent** Charms.

  Table 42: Options for contrail-agent

Option	Default option	Description
physical-interface		Specify the interface where you want to install vhost0 on. If you do not specify an interface, vhost0 is installed on the default gateway interface.
vhost-gateway	auto	Specify the gateway for vhost0. You can enter either an IP address or the keyword (auto) to automatically set a gateway based on the existing vhost routes.
remove-juju-bridge	true	To install vhost0 directly on the interface, enable this option to remove any bridge created to deploy LXD/LXC and KVM workloads.
dpdk	false	Specify DPDK vRouter.
dpdk-driver	uio_pci_generic	Specify DPDK driver for the physical interface.
dpdk-hugepages	70%	Specify the percentage of huge pages reserved for DPDK vRouter and OpenStack instances.
dpdk-coremask	1	Specify the vRouter CPU affinity mask to determine on which CPU the DPDK vRouter will run.

Table 42: Options for contrail-agent *(Continued)*

Option	Default option	Description
dpdk-main-mempool-size		Specify the main packet pool size.
dpdk-pmd-txd-size		Specify the DPDK PMD Tx Descriptor size.
dpdk-pmd-rxd-size		Specify the DPDK PMD Rx Descriptor size.
docker-registry	opencontrailnightly	Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag	latest	Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services. Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.

Table 42: Options for contrail-agent *(Continued)*

Option	Default option	Description
`kernel-hugepages-1g`	Parameter not enabled by default. **NOTE**: 2MB huge pages for kernel-mode vRouters are enabled by default.	Specify the number of 1G huge pages for use with vRouters in kernel mode. You can enable huge pages to avoid compute node reboots during software upgrades. This parameter must be specified at initial deployment. It cannot be modified in an active deployment. If you need to migrate to huge page usage in an active deployment, use 2MB huge pages if suitable for your environment. We recommend allotting 2GB of memory—either using the default 1024x2MB huge page size setting or the 2x1GB size setting—for huge pages. Other huge page size settings should only be set by expert users in specialized circumstances. 1GB and 2MB huge pages cannot be enabled simultaneously in environments using Juju. If you are using this command parameter to enable 1GB huge pages, you must also disable 2MB huge pages. 2MB huge pages can be disabled by entering the **juju config contrail-agent kernel-hugepages-2m=““** command with an empty value. A compute node reboot is required to enable a huge page setting configuration change. After this initial reboot, compute nodes can complete software upgrades without a reboot. Huge pages are disabled for kernel-mode vRouters if the `kernel-hugepages-1g` and the `kernel-hugepages-2m` options are not set. This parameter was introduced in Contrail Networking Release 2005.

Table 42: Options for contrail-agent *(Continued)*

Option	Default option	Description
`kernel-hugepages-2m`	`1024`	Specify the number of 2MB huge pages for use with vRouters in kernel mode. Huge pages in Contrail Networking are used primarily to allocate flow and bridge table memory within the vRouter. Huge pages for kernel-mode vRouters provide enough flow and bridge table memory to avoid compute node reboots to complete future Contrail Networking software upgrades. 1024x2MB huge pages are configured by default starting in Contrail Networking Release 2005. A compute node reboot is required to enable a kernel-mode vRouter huge page setting configuration change, however, so this huge page setting is not enabled on a compute node until the compute node is rebooted. After a compute node is rebooted to enable a vRouter huge page setting, compute nodes can complete software upgrades without a reboot. We recommend allotting 2GB of memory—either using the default 1024x2MB huge page size setting or the 2x1GB size setting—for kernel-mode vRouter huge pages. Other huge page size settings should only be set by expert users in specialized circumstances. 1GB and 2MB huge pages cannot be enabled simultaneously in environments using Juju. If you are using this command parameter to enable 2MB huge pages, you must also disable 1GB huge pages. 1GB huge pages are disabled by default and can also be disabled by entering the **juju config contrail-agent kernel-hugepages-1g=“”** command with an empty value. 1GB huge pages can only be enabled at initial deployment; you cannot initially enable 1GB huge pages in an active deployment. Huge pages are disabled for kernel-mode vRouters if the `kernel-hugepages-1g` and the `kernel-hugepages-2m` options are not set.

Table 42: Options for contrail-agent *(Continued)*

Option	Default option	Description
no_proxy		Specify the list of destinations that must be directly accessed.

- Options for **contrail-analytics** Charms.

Table 43: Options for contrail-analytics

Option	Default option	Description
control-network		Specify the IP address and network mask of the control network.
docker-registry		Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag		Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services. Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.

Table 43: Options for contrail-analytics *(Continued)*

Option	Default option	Description
no_proxy		Specify the list of destinations that must be directly accessed.

- Options for **contrail-analyticsdb** Charms.

Table 44: Options for contrail-analyticsdb

Option	Default option	Description
control-network		Specify the IP address and network mask of the control network.
cassandra-minimum-diskgb	256	Specify the minimum disk requirement.
cassandra-jvm-extra-opts		Specify the memory limit.
docker-registry		Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag		Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services. Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.

Table 44: Options for contrail-analyticsdb *(Continued)*

Option	Default option	Description
`https_proxy`		Specify URL.
`no_proxy`		Specify the list of destinations that must be directly accessed.

- Options for **contrail-controller** Charms.

Table 45: Options for contrail-controller

Option	Default option	Description
`control-network`		Specify the IP address and network mask of the control network.
`auth-mode`	`rbac`	Specify the authentication mode. Options: `rbsc`, `cloud-admin`, `no-auth`. For more information, see https://github.com/Juniper/contrail-controller/wiki/RBAC.
`cassandra-minimum-diskgb`	`20`	Specify the minimum disk requirement.
`cassandra-jvm-extra-opts`		Specify the memory limit.

Table 45: Options for contrail-controller *(Continued)*

Option	Default option	Description
`cloud-admin-role`	`admin`	Specify the role name in keystone for users who have admin-level access. In environments using Canonical orchestration with Contrail Networking, you should change the *cloud-admin-role* to **Admin** with a capital A in most scenarios. The default cloud admin role in Contrail Networking is **admin** and the default cloud admin role in Canonical is **Admin**. These cloud admin role names must match to grant users admin-level access. You can ensure this matching by setting this field to **Admin** in environments using the default settings.
`global-read-only-role`		Specify the role name in keystone for users who have read-only access.
`vip`		Specify if the Contrail API VIP is used for configuring client-side software. If not specified, private IP of the first Contrail API VIP unit will be used.
`use-external-rabbitmq`	`false`	To enable the Charm to use the internal RabbitMQ server, set `use-external-rabbitmq` to `false`. To use an external AMQP server, set`use-external-rabbitmq` to `true`. **NOTE**: Do not change the flag after deployment.
`flow-export-rate`	`0`	Specify how many flow records are exported by vRouter agent to the Contrail Collector when a flow is created or deleted.
`docker-registry`		Specify the URL of the docker-registry.
`docker-registry-insecure`	`false`	Specify if the docker-registry should be configured.

Table 45: Options for contrail-controller *(Continued)*

Option	Default option	Description
`docker-user`		Log in to the docker registry.
`docker-password`		Specify the docker-registry password.
`image-tag`		Specify the docker image tag.
`log-level`	`SYS_NOTICE`	Specify the log level for Contrail services. Options: `SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG`
`http_proxy`		Specify URL.
`https_proxy`		Specify URL.
`no_proxy`		Specify the list of destinations that must be directly accessed.

- Options for **contrail-keystone-auth** Charms.

  Table 46: Options for contrail-keystone-auth

Option	Default option	Description
`ssl_ca`		Specify if the base64-encoded SSL CA certificate is provided to Contrail keystone clients. **NOTE**: This certificate is required if you use a privately signed ssl_cert and ssl_key.

- Options for **contrail-openstack** Charms.

Table 47: Options for contrail-controller

Option	Default option	Description
enable-metadata-server	true	Set enable-metadata-server to true to configure metadata and enable nova to run a local instance of nova-api-metadata for virtual machines
use-internal-endpoints	false	Set use-internal-endpoints to true for OpenStack to configure services to use internal endpoints.
heat-plugin-dirs	/usr/lib64/heat,/usr /lib/heat/usr/lib/ python2.7/dist-packages/vnc_api/gen/heat/resources	Specify the heat plugin directories.
docker-registry		Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag		Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services. Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.

Table 47: Options for contrail-controller *(Continued)*

Option	Default option	Description
no_proxy		Specify the list of destinations that must be directly accessed.

Ironic Support with Juju

Contrail Networking Release 2011.L1 supports new charms for Ironic from OpenStack Train version 15.*x.x*. Ironic is an OpenStack project that manages Bare Metal Servers (BMS) as if they are virtual machines (VM)s. For more information about Contrail and BMS, see Bare Metal Server Management.

Contrail Networking Release 2011.L2 supports OpenStack Ussuri with Ironic deployed on Ubuntu version 20.04 (Focal Fossa).

The updated options are shown in the example `bundle_yaml_file`. Before deploying the updated yaml file, you should have Ceph installed. If not, see Installing Ceph.

For information about deploying the `bundle_yaml_file`, see "Deploying Contrail Charms" on page 599.

Following is an example `bundle_yaml_file` with the additional options highlighted. `ceph-radosgw` and its related options are required to support the new Ironic charms.

```
series: bionic
applications:
  barbican:
    charm: cs:barbican-31
    num_units: 3
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    options:
      openstack-origin: cloud:bionic-train
      region: RegionOne
      use-internal-endpoints: true
      vip: 10.92.76.133 192.168.2.11
      worker-multiplier: 0.25
    bindings:
      "": oam-space
      admin: oam-space
```

```
      amqp: oam-space
      certificates: oam-space
      cluster: oam-space
      ha: oam-space
      hsm: oam-space
      identity-service: oam-space
      internal: oam-space
      public: public-space
      secrets: oam-space
      shared-db: oam-space
  barbican-hacluster:
    charm: cs:hacluster-62
    options:
      cluster_count: 3
    bindings:
      "": alpha
      ha: alpha
      hanode: alpha
      juju-info: alpha
      nrpe-external-master: alpha
      pacemaker-remote: alpha
      peer-availability: alpha
  barbican-vault:
    charm: cs:barbican-vault-12
    bindings:
      "": oam-space
      certificates: oam-space
      juju-info: oam-space
      secrets: oam-space
      secrets-storage: oam-space
  ceph-mon:
    charm: cs:ceph-mon-51
    num_units: 3
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    constraints: spaces=oam-space
    bindings:
      "": alpha
      admin: alpha
      bootstrap-source: alpha
      client: alpha
```

```
      cluster: oam-space
      mds: alpha
      mon: alpha
      nrpe-external-master: alpha
      osd: alpha
      prometheus: alpha
      public: oam-space
      radosgw: alpha
      rbd-mirror: alpha
  ceph-osd:
    charm: cs:ceph-osd-306
    num_units: 3
    to:
    - "17"
    - "21"
    - "19"
    options:
      osd-devices: /dev/sdb
    bindings:
      "": alpha
      cluster: oam-space
      mon: alpha
      nrpe-external-master: alpha
      public: oam-space
      secrets-storage: alpha
  ceph-radosgw:
    charm: cs:ceph-radosgw-292
    num_units: 3
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    options:
      admin-roles: admin
      loglevel: 10
      namespace-tenants: true
      operator-roles: member
      source: cloud:bionic-train/proposed
      vip: 10.92.76.127 192.168.2.190
    constraints: spaces=oam-space,public-space
    bindings:
      "": alpha
      admin: alpha
```

```
      certificates: alpha
      cluster: alpha
      gateway: alpha
      ha: alpha
      identity-service: alpha
      internal: oam-space
      master: alpha
      mon: alpha
      nrpe-external-master: alpha
      object-store: alpha
      public: public-space
      slave: alpha
  contrail-agent:
    charm: local:bionic/contrail-agent-1
    options:
      docker-password: <docker password>
      docker-registry: hub.juniper.net/contrail
      docker-user: JNPR-FieldUser367
      image-tag: "2008.121"
      log-level: SYS_INFO
      physical-interface: bond0.4010
      vhost-gateway: auto
    bindings:
      "": alpha
      agent-cluster: alpha
      contrail-controller: alpha
      juju-info: alpha
      nrpe-external-master: alpha
      tls-certificates: alpha
      vrouter-plugin: alpha
  contrail-agent-csn:
    charm: local:bionic/contrail-agent-3
    options:
      csn-mode: tsn-no-forwarding
      docker-password: <docker password>
      docker-registry: hub.juniper.net/contrail
      docker-user: JNPR-FieldUser367
      image-tag: "2008.121"
      physical-interface: bond0.4010
      vhost-gateway: auto
    bindings:
      "": alpha
      agent-cluster: alpha
```

```
      contrail-controller: alpha
      juju-info: alpha
      nrpe-external-master: alpha
      tls-certificates: alpha
      vrouter-plugin: alpha
  contrail-analytics:
    charm: local:bionic/contrail-analytics-1
    num_units: 4
    to:
    - kvm:0
    - kvm:1
    - kvm:2
    - kvm:13
    options:
      control-network: 192.168.2.0/24
      docker-password: <docker password>
      docker-registry: hub.juniper.net/contrail
      docker-user: JNPR-FieldUser367
      haproxy-http-mode: https
      image-tag: "2008.121"
      log-level: SYS_DEBUG
      min-cluster-size: 3
      vip: 10.92.77.18
    constraints: cpu-cores=16 mem=32768 root-disk=102400 spaces=oam-space,overlay-space
    bindings:
      "": oam-space
      analytics-cluster: oam-space
      contrail-analytics: oam-space
      contrail-analyticsdb: oam-space
      http-services: oam-space
      nrpe-external-master: oam-space
      tls-certificates: oam-space
  contrail-analyticsdb:
    charm: local:bionic/contrail-analyticsdb-1
    num_units: 4
    to:
    - kvm:0
    - kvm:1
    - kvm:2
    - kvm:13
    options:
      cassandra-jvm-extra-opts: -Xms16g -Xmx24g
      cassandra-minimum-diskgb: "4"
```

```
      control-network: 192.168.2.0/24
      docker-password: <docker password>
      docker-registry: hub.juniper.net/contrail
      docker-user: JNPR-FieldUser367
      image-tag: "2008.121"
      log-level: SYS_DEBUG
      min-cluster-size: 3
    constraints: cpu-cores=16 mem=65536 root-disk=512000 spaces=oam-space,overlay-space
    bindings:
      "": oam-space
      analyticsdb-cluster: oam-space
      contrail-analyticsdb: oam-space
      nrpe-external-master: oam-space
      tls-certificates: oam-space
  contrail-command:
    charm: local:bionic/contrail-command-0
    num_units: 1
    to:
    - "9"
    options:
      docker-password: <docker password>
      docker-registry: hub.juniper.net/contrail
      docker-registry-insecure: true
      docker-user: JNPR-FieldUser367
      image-tag: "2008.121"
    constraints: tags=command
    bindings:
      "": alpha
      contrail-controller: alpha
  contrail-controller:
    charm: local:bionic/contrail-controller-1
    num_units: 4
    to:
    - kvm:0
    - kvm:2
    - kvm:1
    - kvm:13
    options:
      auth-mode: rbac
      cassandra-jvm-extra-opts: -Xms16g -Xmx24g
      cassandra-minimum-diskgb: "4"
      control-network: 192.168.2.0/24
      data-network: 172.30.0.0/16
```

```
      docker-password: <docker password>
      docker-registry: hub.juniper.net/contrail
      docker-user: JNPR-FieldUser367
      haproxy-http-mode: https
      haproxy-https-mode: http
      image-tag: "2008.121"
      local-rabbitmq-hostname-resolution: true
      log-level: SYS_DEBUG
      min-cluster-size: 3
      vip: 10.92.77.18
    constraints: cpu-cores=16 mem=65536 root-disk=102400 spaces=oam-space,overlay-space,public-
space
    bindings:
      "": oam-space
      contrail-analytics: oam-space
      contrail-analyticsdb: oam-space
      contrail-auth: oam-space
      contrail-controller: oam-space
      contrail-issu: oam-space
      controller-cluster: oam-space
      http-services: oam-space
      https-services: oam-space
      nrpe-external-master: oam-space
      tls-certificates: oam-space
  contrail-haproxy:
    charm: cs:haproxy-55
    num_units: 4
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    - lxd:13
    options:
      default_timeouts: queue 60000, connect 5000, client 120000, server 120000
      enable_monitoring: true
      peering_mode: active-active
      services: ""
      source: backports
      ssl_cert: SELFSIGNED
      sysctl: '{fs.file-max: 10240}'
    bindings:
      "": oam-space
      local-monitors: oam-space
```

```
    munin: oam-space
    nrpe-external-master: oam-space
    peer: oam-space
    public: public-space
    reverseproxy: oam-space
    statistics: oam-space
    website: public-space
contrail-keepalived:
  charm: cs:~containers/keepalived-28
  options:
    network_interface: eth0
    port: 8143
    virtual_ip: 10.92.77.18
  bindings:
    "": alpha
    juju-info: alpha
    lb-sink: alpha
    loadbalancer: alpha
    website: alpha
contrail-keystone-auth:
  charm: local:bionic/contrail-keystone-auth-1
  num_units: 4
  to:
  - lxd:0
  - lxd:1
  - lxd:2
  - lxd:13
  constraints: spaces=oam-space,overlay-space
  bindings:
    "": oam-space
    contrail-auth: oam-space
    identity-admin: oam-space
    nrpe-external-master: oam-space
contrail-openstack:
  charm: local:bionic/contrail-openstack-3
  options:
    docker-password: <docker password>
    docker-registry: hub.juniper.net/contrail
    docker-user: JNPR-FieldUser367
    image-tag: "2008.121"
    use-internal-endpoints: true
  bindings:
    "": alpha
```

```
      cluster: alpha
      contrail-controller: alpha
      heat-plugin: alpha
      juju-info: alpha
      neutron-api: alpha
      nova-compute: alpha
  dashboard-hacluster:
    charm: cs:hacluster-62
    options:
      cluster_count: 3
    bindings:
      "": alpha
      ha: alpha
      hanode: alpha
      juju-info: alpha
      nrpe-external-master: alpha
      pacemaker-remote: alpha
      peer-availability: alpha
  easyrsa:
    charm: cs:~containers/easyrsa-303
    num_units: 1
    to:
    - lxd:0
    bindings:
      "": oam-space
      client: oam-space
  etcd:
    charm: cs:etcd-521
    num_units: 3
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    options:
      channel: 3.1/stable
    bindings:
      "": oam-space
      certificates: oam-space
      cluster: oam-space
      db: oam-space
      nrpe-external-master: oam-space
      proxy: oam-space
  external-policy-routing:
```

```
    charm: cs:~canonical-bootstack/policy-routing-3
    options:
      cidr: 10.92.76.0/23
      gateway: 10.92.77.254
    bindings:
      "": alpha
      juju-info: alpha
  glance:
    charm: cs:~openstack-charmers-next/glance-442
    num_units: 4
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    - lxd:13
    options:
      openstack-origin: cloud:bionic-train
      region: RegionOne
      restrict-ceph-pools: false
      use-internal-endpoints: true
      vip: 10.92.77.12 192.168.2.12
      worker-multiplier: 0.25
    bindings:
      "": oam-space
      admin: oam-space
      amqp: oam-space
      ceph: oam-space
      certificates: oam-space
      cinder-volume-service: oam-space
      cluster: oam-space
      ha: oam-space
      identity-service: oam-space
      image-service: oam-space
      internal: oam-space
      nrpe-external-master: oam-space
      object-store: oam-space
      public: public-space
      shared-db: oam-space
      storage-backend: oam-space
  glance-hacluster:
    charm: cs:hacluster-62
    options:
      cluster_count: 3
```

```
    bindings:
      "": alpha
      ha: alpha
      hanode: alpha
      juju-info: alpha
      nrpe-external-master: alpha
      pacemaker-remote: alpha
      peer-availability: alpha
  glance-simplestreams-sync:
    charm: cs:glance-simplestreams-sync-33
    num_units: 3
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    options:
      source: ppa:simplestreams-dev/trunk
      use_swift: false
    bindings:
      "": oam-space
      amqp: oam-space
      certificates: oam-space
      identity-service: oam-space
      image-modifier: oam-space
      nrpe-external-master: oam-space
      simplestreams-image-service: oam-space
  heat:
    charm: cs:heat-271
    num_units: 4
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    - lxd:13
    options:
      openstack-origin: cloud:bionic-train
      region: RegionOne
      use-internal-endpoints: true
      vip: 10.92.77.13 192.168.2.13
      worker-multiplier: 0.25
    constraints: cpu-cores=6 mem=32768 root-disk=65536 spaces=oam-space,public-space,overlay-
space
    bindings:
```

```
      "": oam-space
      admin: oam-space
      amqp: oam-space
      certificates: oam-space
      cluster: oam-space
      ha: oam-space
      heat-plugin-subordinate: overlay-space
      identity-service: oam-space
      internal: oam-space
      public: public-space
      shared-db: oam-space
  heat-hacluster:
    charm: cs:hacluster-62
    options:
      cluster_count: 3
    bindings:
      "": alpha
      ha: alpha
      hanode: alpha
      juju-info: alpha
      nrpe-external-master: alpha
      pacemaker-remote: alpha
      peer-availability: alpha
  ironic-api:
    charm: cs:~openstack-charmers-next/ironic-api-8
    num_units: 3
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    options:
      openstack-origin: cloud:bionic-train/proposed
      vip: 10.92.76.130 192.168.2.189
    constraints: spaces=oam-space,public-space
    bindings:
      "": alpha
      admin: alpha
      amqp: alpha
      certificates: alpha
      cluster: alpha
      ha: alpha
      identity-service: alpha
      internal: alpha
```

```
    ironic-api: alpha
    public: alpha
    shared-db: oam-space
ironic-api-hacluster:
  charm: cs:hacluster-72
  options:
    cluster_count: 3
  bindings:
    "": alpha
    ha: alpha
    hanode: alpha
    juju-info: alpha
    nrpe-external-master: alpha
    pacemaker-remote: alpha
    peer-availability: alpha
ironic-conductor:
  charm: cs:~openstack-charmers-next/ironic-conductor-5
  num_units: 1
  to:
  - "14"
  options:
    cleaning-network: ironic
    default-deploy-interface: direct
    default-network-interface: neutron
    disable-secure-erase: true
    enabled-deploy-interfaces: direct
    enabled-network-interfaces: noop,flat,neutron
    max-tftp-block-size: 1418
    openstack-origin: cloud:bionic-train/proposed
    provisioning-network: ironic
    use-ipxe: false
  bindings:
    "": alpha
    amqp: alpha
    certificates: alpha
    cleaning: alpha
    deployment: alpha
    identity-credentials: alpha
    internal: alpha
    ironic-api: alpha
    shared-db: alpha
keystone:
  charm: cs:keystone-309
```

```
    num_units: 4
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    - lxd:13
    options:
      admin-password: c0ntrail123
      admin-role: admin
      openstack-origin: cloud:bionic-train
      preferred-api-version: 3
      region: RegionOne
      token-provider: fernet
      vip: 10.92.77.14 192.168.2.14
      worker-multiplier: 0.25
    bindings:
      "": oam-space
      admin: oam-space
      certificates: oam-space
      cluster: oam-space
      domain-backend: oam-space
      ha: oam-space
      identity-admin: oam-space
      identity-credentials: oam-space
      identity-notifications: oam-space
      identity-service: oam-space
      internal: oam-space
      keystone-fid-service-provider: oam-space
      keystone-middleware: oam-space
      nrpe-external-master: oam-space
      public: public-space
      shared-db: oam-space
      websso-trusted-dashboard: oam-space
  keystone-hacluster:
    charm: cs:hacluster-62
    options:
      cluster_count: 3
    bindings:
      "": alpha
      ha: alpha
      hanode: alpha
      juju-info: alpha
      nrpe-external-master: alpha
```

```
    pacemaker-remote: alpha
    peer-availability: alpha
memcached:
  charm: cs:memcached-26
  num_units: 4
  to:
  - lxd:0
  - lxd:1
  - lxd:2
  - lxd:13
  options:
    allow-ufw-ip6-softfail: true
  constraints: spaces=oam-space
  bindings:
    "": oam-space
    cache: oam-space
    cluster: oam-space
    local-monitors: oam-space
    monitors: oam-space
    munin: oam-space
    nrpe-external-master: oam-space
mysql:
  charm: cs:percona-cluster-281
  num_units: 4
  to:
  - lxd:0
  - lxd:1
  - lxd:2
  - lxd:13
  options:
    enable-binlogs: true
    innodb-buffer-pool-size: 512M
    max-connections: 2000
    min-cluster-size: 3
    performance-schema: true
    source: cloud:bionic-train
    tuning-level: safest
    vip: 192.168.2.17
    wait-timeout: 3600
    wsrep-slave-threads: 48
  bindings:
    "": oam-space
    access: oam-space
```

```
      cluster: oam-space
      db: oam-space
      db-admin: oam-space
      ha: oam-space
      master: oam-space
      nrpe-external-master: oam-space
      shared-db: oam-space
      slave: oam-space
  mysql-hacluster:
    charm: cs:hacluster-62
    options:
      cluster_count: 3
    bindings:
      "": alpha
      ha: alpha
      hanode: alpha
      juju-info: alpha
      nrpe-external-master: alpha
      pacemaker-remote: alpha
      peer-availability: alpha
  ncc-hacluster:
    charm: cs:hacluster-62
    options:
      cluster_count: 3
    bindings:
      "": alpha
      ha: alpha
      hanode: alpha
      juju-info: alpha
      nrpe-external-master: alpha
      pacemaker-remote: alpha
      peer-availability: alpha
  neutron-api:
    charm: cs:neutron-api-281
    num_units: 4
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    - lxd:13
    options:
      default-tenant-network-type: vlan
      dhcp-agents-per-network: 2
```

```
      enable-l3ha: true
      enable-ml2-port-security: true
      global-physnet-mtu: 9000
      l2-population: true
      manage-neutron-plugin-legacy-mode: false
      neutron-security-groups: true
      openstack-origin: cloud:bionic-train
      overlay-network-type: ""
      region: RegionOne
      use-internal-endpoints: true
      vip: 10.92.77.15  192.168.2.15
      worker-multiplier: 0.25
    constraints: cpu-cores=8 mem=32768 root-disk=262144 spaces=oam-space,public-space,overlay-
space
    bindings:
      "": oam-space
      admin: oam-space
      amqp: oam-space
      certificates: oam-space
      cluster: oam-space
      etcd-proxy: oam-space
      external-dns: oam-space
      ha: oam-space
      identity-service: oam-space
      infoblox-neutron: oam-space
      internal: oam-space
      midonet: oam-space
      neutron-api: oam-space
      neutron-load-balancer: oam-space
      neutron-plugin-api: oam-space
      neutron-plugin-api-subordinate: overlay-space
      nrpe-external-master: oam-space
      public: public-space
      shared-db: oam-space
      vsd-rest-api: oam-space
  neutron-hacluster:
    charm: cs:hacluster-62
    options:
      cluster_count: 3
    bindings:
      "": alpha
      ha: alpha
      hanode: alpha
```

```
      juju-info: alpha
      nrpe-external-master: alpha
      pacemaker-remote: alpha
      peer-availability: alpha
  nova-cloud-controller:
    charm: cs:nova-cloud-controller-339
    num_units: 4
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    - lxd:13
    options:
      console-access-protocol: novnc
      console-proxy-ip: local
      cpu-allocation-ratio: 4
      network-manager: Neutron
      openstack-origin: cloud:bionic-train
      ram-allocation-ratio: 0.999999
      region: RegionOne
      use-internal-endpoints: true
      vip: 10.92.77.16 192.168.2.16
      worker-multiplier: 0.25
    bindings:
      "": oam-space
      admin: oam-space
      amqp: oam-space
      amqp-cell: oam-space
      certificates: oam-space
      cinder-volume-service: oam-space
      cloud-compute: oam-space
      cloud-controller: oam-space
      cluster: oam-space
      ha: oam-space
      identity-service: oam-space
      image-service: oam-space
      internal: oam-space
      memcache: oam-space
      neutron-api: oam-space
      nova-cell-api: oam-space
      nova-vmware: oam-space
      nrpe-external-master: oam-space
      placement: oam-space
```

```
      public: public-space
      quantum-network-service: oam-space
      shared-db: oam-space
      shared-db-cell: oam-space
  nova-compute:
    charm: cs:nova-compute-309
    num_units: 5
    to:
    - "3"
    - "4"
    - "5"
    - "6"
    - "15"
    options:
      openstack-origin: cloud:bionic-train
      os-internal-network: 192.168.2.0/24
    bindings:
      "": alpha
      amqp: alpha
      ceph: alpha
      ceph-access: alpha
      cloud-compute: alpha
      cloud-credentials: alpha
      compute-peer: alpha
      ephemeral-backend: alpha
      image-service: alpha
      internal: alpha
      lxd: alpha
      neutron-plugin: alpha
      nova-ceilometer: alpha
      nrpe-external-master: alpha
      secrets-storage: alpha
  nova-ironic:
    charm: cs:~openstack-charmers-next/nova-compute-524
    num_units: 1
    to:
    - "22"
    options:
      enable-live-migration: false
      enable-resize: false
      openstack-origin: cloud:bionic-train/proposed
      virt-type: ironic
    bindings:
```

```
      "": alpha
      amqp: alpha
      ceph: alpha
      ceph-access: alpha
      cloud-compute: alpha
      cloud-credentials: alpha
      compute-peer: alpha
      ephemeral-backend: alpha
      image-service: alpha
      internal: alpha
      ironic-api: alpha
      lxd: alpha
      migration: alpha
      neutron-plugin: alpha
      nova-ceilometer: alpha
      nrpe-external-master: alpha
      secrets-storage: alpha
  ntp:
    charm: cs:ntp-36
    options:
      source: ntp.juniper.net
    bindings:
      "": alpha
      juju-info: alpha
      master: alpha
      nrpe-external-master: alpha
      ntp-peers: alpha
      ntpmaster: alpha
  octavia:
    charm: cs:~apavlov-e/octavia-3
    num_units: 3
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    options:
      amp-ssh-key-name: octavia
      amp-ssh-pub-key:
c3NoLXJzYSBBQUFBQjNOemFDMXljMkVBQUFBREFRQUJBQUFCQVFDa0N0SzJCWk01TC90VGdoM3J3L2FpR2ZEQzlzOEI4alFiS
2VzQzJqTHRWcFNYbGlFellqQTNGNjEyalpwdERjOXdhOEF3eStxbEl0L1Frak5TVjhPMVpvNXZlc2RMREhvQjJrMzV5ZEFvMX
hQRkFmV3lsSjh6VnJrd0U5aU1tWEVYZTd1VjdDdkgyZmdmSnlGeXJKaFR2ZjBWdTZGK1M5RHl1MnkxMUdXWEsrSDAyR3ZneHV
zamZ3QlhoZ3IxNU1kZCt4RkJsbkpYRGtkQjVYVit4azZhYkJsRVJUc0N6c09EdXVOQTg4aVhqeHkvZzJpb2NtNWhtcVhUeDRM
T2gzam9NbEFHUW5RQ2FXdFNSenpWM3dKT3JLeW5zU1pObVRmUnluRDdaaGl1WlZLNUZURWhQaXZUaVAwaHdLeTRMZGZUM0NsS
```

```
kJSWmdMVVJZYUtSYlFwYkQgdWJ1bnR1QGp1bXBob3N0Cg==
      create-mgmt-network: false
      lb-mgmt-controller-cacert: |-
        <certificate>
      lb-mgmt-controller-cert: |-
        <certificate>
      lb-mgmt-issuing-ca-key-passphrase: <passphrase>
      lb-mgmt-issuing-ca-private-key: |-
        <private key>
      lb-mgmt-issuing-cacert: |-
        <certificate>
      loadbalancer-topology: ACTIVE_STANDBY
      openstack-origin: cloud:bionic-train
      region: RegionOne
      use-internal-endpoints: true
      vip: 10.92.76.135 192.168.2.18
      worker-multiplier: 0.25
    bindings:
      "": oam-space
      admin: oam-space
      amqp: oam-space
      certificates: oam-space
      cluster: oam-space
      ha: oam-space
      identity-service: oam-space
      internal: oam-space
      neutron-api: oam-space
      neutron-openvswitch: oam-space
      ovsdb-cms: oam-space
      ovsdb-subordinate: oam-space
      public: public-space
      shared-db: oam-space
  octavia-dashboard:
    charm: cs:octavia-dashboard-17
    bindings:
      "": alpha
      certificates: alpha
      dashboard: alpha
  octavia-diskimage-retrofit:
    charm: cs:octavia-diskimage-retrofit-12
    options:
      amp-image-tag: octavia-amphora
      retrofit-uca-pocket: train
```

```
    bindings:
      "": oam-space
      certificates: oam-space
      identity-credentials: oam-space
      juju-info: oam-space
  octavia-hacluster:
    charm: cs:hacluster-62
    options:
      cluster_count: 3
    bindings:
      "": alpha
      ha: alpha
      hanode: alpha
      juju-info: alpha
      nrpe-external-master: alpha
      pacemaker-remote: alpha
      peer-availability: alpha
  openstack-dashboard:
    charm: cs:openstack-dashboard-295
    num_units: 4
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    - lxd:13
    options:
      cinder-backup: false
      endpoint-type: publicURL
      neutron-network-firewall: false
      neutron-network-l3ha: true
      neutron-network-lb: true
      openstack-origin: cloud:bionic-train
      password-retrieve: true
      secret: encryptcookieswithme
      vip: 10.92.77.11
      webroot: /
    constraints: spaces=oam-space
    bindings:
      "": public-space
      certificates: public-space
      cluster: public-space
      dashboard-plugin: public-space
      ha: public-space
```

```
      identity-service: public-space
      nrpe-external-master: public-space
      public: public-space
      shared-db: oam-space
      website: public-space
      websso-fid-service-provider: public-space
      websso-trusted-dashboard: public-space
  placement:
    charm: cs:placement-11
    num_units: 4
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    - lxd:13
    options:
      openstack-origin: cloud:bionic-train
      region: RegionOne
      use-internal-endpoints: true
      vip: 10.92.77.19 192.168.2.19
    bindings:
      "": oam-space
      admin: oam-space
      amqp: oam-space
      certificates: oam-space
      cluster: oam-space
      ha: oam-space
      identity-service: oam-space
      internal: oam-space
      placement: oam-space
      public: public-space
      shared-db: oam-space
  placement-hacluster:
    charm: cs:hacluster-62
    options:
      cluster_count: 3
    bindings:
      "": alpha
      ha: alpha
      hanode: alpha
      juju-info: alpha
      nrpe-external-master: alpha
      pacemaker-remote: alpha
```

```
    peer-availability: alpha
rabbitmq-server:
  charm: cs:rabbitmq-server-97
  num_units: 4
  to:
  - lxd:0
  - lxd:1
  - lxd:2
  - lxd:13
  options:
    min-cluster-size: 3
    source: cloud:bionic-train
  bindings:
    "": oam-space
    amqp: oam-space
    ceph: oam-space
    certificates: oam-space
    cluster: oam-space
    ha: oam-space
    nrpe-external-master: oam-space
radosgw-hacluster:
  charm: cs:hacluster-72
  options:
    cluster_count: 3
  bindings:
    "": alpha
    ha: alpha
    hanode: alpha
    juju-info: alpha
    nrpe-external-master: alpha
    pacemaker-remote: alpha
    peer-availability: alpha
ubuntu:
  charm: cs:ubuntu-15
  num_units: 4
  to:
  - "0"
  - "1"
  - "2"
  - "13"
  bindings:
    "": alpha
vault:
```

```
    charm: cs:vault-39
    num_units: 3
    to:
    - lxd:0
    - lxd:1
    - lxd:2
    options:
      vip: 192.168.2.20
    bindings:
      "": oam-space
      access: oam-space
      certificates: oam-space
      cluster: oam-space
      db: oam-space
      etcd: oam-space
      external: oam-space
      ha: oam-space
      nrpe-external-master: oam-space
      secrets: oam-space
      shared-db: oam-space
  vault-hacluster:
    charm: cs:hacluster-62
    options:
      cluster_count: 3
    bindings:
      "": alpha
      ha: alpha
      hanode: alpha
      juju-info: alpha
      nrpe-external-master: alpha
      pacemaker-remote: alpha
      peer-availability: alpha
machines:
  "0":
    constraints: tags=controller1
  "1":
    constraints: tags=controller2
  "2":
    constraints: tags=controller3
  "3":
    constraints: tags=compute1
  "4":
    constraints: tags=compute2
```

```
  "5":
    constraints: tags=compute3
  "6":
    constraints: tags=compute4
  "9":
    constraints: tags=command
  "13":
    constraints: tags=controller4
  "14":
    constraints: tags=controller5
  "15":
    constraints: tags=compute5
  "17":
    constraints: tags=CEPH
  "19":
    constraints: tags=CEPH
  "21":
    constraints: tags=CEPH
  "22":
    constraints: tags=CSN
relations:
- - ubuntu:juju-info
  - ntp:juju-info
- - mysql:ha
  - mysql-hacluster:ha
- - keystone:shared-db
  - mysql:shared-db
- - keystone:ha
  - keystone-hacluster:ha
- - glance:shared-db
  - mysql:shared-db
- - glance:identity-service
  - keystone:identity-service
- - nova-cloud-controller:shared-db
  - mysql:shared-db
- - nova-cloud-controller:identity-service
  - keystone:identity-service
- - nova-cloud-controller:image-service
  - glance:image-service
- - nova-cloud-controller:ha
  - ncc-hacluster:ha
- - neutron-api:shared-db
  - mysql:shared-db
```

```
- - neutron-api:neutron-api
  - nova-cloud-controller:neutron-api
- - neutron-api:identity-service
  - keystone:identity-service
- - neutron-api:ha
  - neutron-hacluster:ha
- - nova-compute:image-service
  - glance:image-service
- - nova-compute:cloud-compute
  - nova-cloud-controller:cloud-compute
- - nova-compute:juju-info
  - ntp:juju-info
- - openstack-dashboard:identity-service
  - keystone:identity-service
- - openstack-dashboard:ha
  - dashboard-hacluster:ha
- - heat:shared-db
  - mysql:shared-db
- - heat:identity-service
  - keystone:identity-service
- - heat:ha
  - heat-hacluster:ha
- - placement:shared-db
  - mysql:shared-db
- - placement:identity-service
  - keystone:identity-service
- - placement:placement
  - nova-cloud-controller:placement
- - contrail-controller:contrail-controller
  - contrail-agent:contrail-controller
- - contrail-agent:juju-info
  - nova-compute:juju-info
- - contrail-analytics:contrail-analyticsdb
  - contrail-analyticsdb:contrail-analyticsdb
- - contrail-analytics:contrail-analytics
  - contrail-controller:contrail-analytics
- - contrail-analytics:http-services
  - contrail-haproxy:reverseproxy
- - contrail-analyticsdb:contrail-analyticsdb
  - contrail-controller:contrail-analyticsdb
- - contrail-controller:contrail-auth
  - contrail-keystone-auth:contrail-auth
- - contrail-controller:http-services
```

```
  - contrail-haproxy:reverseproxy
- - contrail-controller:https-services
  - contrail-haproxy:reverseproxy
- - contrail-keystone-auth:identity-admin
  - keystone:identity-admin
- - contrail-openstack:nova-compute
  - nova-compute:neutron-plugin
- - contrail-openstack:neutron-api
  - neutron-api:neutron-plugin-api-subordinate
- - contrail-openstack:heat-plugin
  - heat:heat-plugin-subordinate
- - contrail-openstack:contrail-controller
  - contrail-controller:contrail-controller
- - contrail-haproxy:juju-info
  - contrail-keepalived:juju-info
- - nova-cloud-controller:memcache
  - memcached:cache
- - external-policy-routing:juju-info
  - openstack-dashboard:juju-info
- - external-policy-routing:juju-info
  - glance:juju-info
- - external-policy-routing:juju-info
  - heat:juju-info
- - external-policy-routing:juju-info
  - keystone:juju-info
- - external-policy-routing:juju-info
  - neutron-api:juju-info
- - external-policy-routing:juju-info
  - nova-cloud-controller:juju-info
- - external-policy-routing:juju-info
  - contrail-haproxy:juju-info
- - ntp:juju-info
  - contrail-controller:juju-info
- - ntp:juju-info
  - contrail-analytics:juju-info
- - ntp:juju-info
  - contrail-analyticsdb:juju-info
- - ntp:juju-info
  - neutron-api:juju-info
- - ntp:juju-info
  - heat:juju-info
- - contrail-command:contrail-controller
  - contrail-controller:contrail-controller
```

```
- - glance:ha
  - glance-hacluster:ha
- - placement:ha
  - placement-hacluster:ha
- - mysql:shared-db
  - octavia:shared-db
- - mysql:shared-db
  - barbican:shared-db
- - mysql:shared-db
  - vault:shared-db
- - keystone:identity-service
  - octavia:identity-service
- - keystone:identity-service
  - barbican:identity-service
- - neutron-api:neutron-load-balancer
  - octavia:neutron-api
- - openstack-dashboard:dashboard-plugin
  - octavia-dashboard:dashboard
- - barbican-vault:secrets
  - barbican:secrets
- - vault:secrets
  - barbican-vault:secrets-storage
- - glance-simplestreams-sync:juju-info
  - octavia-diskimage-retrofit:juju-info
- - keystone:identity-service
  - glance-simplestreams-sync:identity-service
- - keystone:identity-credentials
  - octavia-diskimage-retrofit:identity-credentials
- - contrail-openstack:nova-compute
  - octavia:neutron-openvswitch
- - vault:ha
  - vault-hacluster:ha
- - etcd:certificates
  - easyrsa:client
- - etcd:db
  - vault:etcd
- - barbican:ha
  - barbican-hacluster:ha
- - octavia:ha
  - octavia-hacluster:ha
- - rabbitmq-server:amqp
  - barbican:amqp
- - rabbitmq-server:amqp
```

```
  - glance-simplestreams-sync:amqp
- - rabbitmq-server:amqp
  - heat:amqp
- - rabbitmq-server:amqp
  - neutron-api:amqp
- - rabbitmq-server:amqp
  - nova-cloud-controller:amqp
- - rabbitmq-server:amqp
  - nova-compute:amqp
- - rabbitmq-server:amqp
  - octavia:amqp
- - ceph-mon:osd
  - ceph-osd:mon
- - ceph-radosgw:juju-info
  - external-policy-routing:juju-info
- - ceph-radosgw:ha
  - radosgw-hacluster:ha
- - ceph-radosgw:mon
  - ceph-mon:radosgw
- - ceph-radosgw:identity-service
  - keystone:identity-service
- - vault:certificates
  - ceph-radosgw:certificates
- - ceph-radosgw:object-store
  - glance:object-store
- - ceph-mon:client
  - glance:ceph
- - ironic-conductor:amqp
  - rabbitmq-server:amqp
- - ironic-conductor:identity-credentials
  - keystone:identity-credentials
- - ironic-conductor:shared-db
  - mysql:shared-db
- - vault:certificates
  - ironic-conductor:certificates
- - nova-ironic:amqp
  - rabbitmq-server:amqp
- - nova-ironic:image-service
  - glance:image-service
- - nova-ironic:cloud-credentials
  - keystone:identity-credentials
- - nova-ironic:cloud-compute
  - nova-cloud-controller:cloud-compute
```

```
- - ceph-mon:client
  - nova-ironic:ceph
- - nova-ironic:juju-info
  - ntp:juju-info
- - contrail-agent-csn:juju-info
  - nova-ironic:juju-info
- - contrail-agent-csn:contrail-controller
  - contrail-controller:contrail-controller
- - ironic-api:ha
  - ironic-api-hacluster:ha
- - ironic-conductor:ironic-api
  - ironic-api:ironic-api
- - ironic-api:amqp
  - rabbitmq-server:amqp
- - ironic-api:identity-service
  - keystone:identity-service
- - ironic-api:shared-db
  - mysql:shared-db
- - vault:certificates
  - ironic-api:certificates
- - nova-ironic:ironic-api
  - ironic-api:ironic-api
```

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
2011.L2	Contrail Networking Release 2011.L2 supports OpenStack Ussuri with Ironic deployed on Ubuntu version 20.04 (Focal Fossa).
2011.L1	Contrail Networking Release 2011.L1 supports new charms for Ironic from OpenStack Train version 15.*x.x*.
2011	Starting in Contrail Networking Release 2011, Contrail Networking supports OpenStack Ussuri with Ubuntu version 18.04 (Bionic Beaver) and Ubuntu version 20.04 (Focal Fossa).

RELATED DOCUMENTATION

Bundle Reference File

Installing Contrail with Kubernetes by Using Juju Charms

IN THIS SECTION

You can deploy Contrail Networking using Juju Charms. Juju helps you deploy, configure, and efficiently manage applications on private clouds and public clouds. Juju accesses the cloud with the help of a Juju controller. A Charm is a module containing a collection of scripts and metadata and is used with Juju to deploy Contrail.

A Juju Charm helps you deploy Docker containers to the cloud. For more information on containerized Contrail, see "Understanding Contrail Containers" on page 5. Juju Charms simplifies Contrail deployment by providing a simple way to deploy, configure, scale, and manage Contrail operations.

Understanding Juju Charms with Kubernetes

Contrail supports the following charms:

- contrail-agent
- contrail-analytics
- contrail-analyticsdb
- contrail-controller
- contrail-kubernetes-master
- contrail-kubernetes-node

Preparing to Deploy Contrail with Kubernetes by Using Juju Charms

You can deploy Contrail Networking by using Juju bundle.

Follow these steps to prepare for deployment:

1. Install Juju.

```
apt install bridge-utils -y
apt install snapd -y
snap install juju --classic
```

2. Configure Juju.

   You can add a cloud to Juju, identify clouds supported by Juju, and manage clouds already added to Juju.

   Adding a cloud

   Juju already has knowledge of the AWS cloud, which means adding your AWS account to Juju is quick and easy.

```
juju show-cloud --local aws
```

NOTE: In versions prior to Juju v.2.6.0 the `show-cloud` command only operates locally. There is no `--local` option.

   You must ensure that Juju's information is up to date (e.g. new region support). Run the following command to update Juju's public cloud data:

```
juju update-public-clouds
```

   Juju recognizes a wide range of cloud types. You can use any one of the following methods to add a cloud credentials to Juju:

   - **Adding a Cloud Credentials by Using Interactive Command**

     Example: Adding AWS cloud credentials to Juju

```
juju add-credential aws

Enter credential name: jlaurin

Using auth-type "access-key".
```

```
Enter access-key: AKIAIFII5EH5FOCYZJMA

Enter secret-key: ****************************

Credential "jlaurin" added locally for cloud "aws".
```

- **Adding a Cloud Credentials Manually**

  You can use a YAML configuration file to add AWS cloud credentials. Run the following command:

  ```
  juju add-credential aws -f <mycreds.yaml>
  ```

  For details, refer to Juju Adding Credentials from a File.

Identifying a supported cloud

Use the `juju clouds` command to list cloud types that are supported by Juju.

```
$ juju clouds
Cloud        Regions  Default          Type        Description
aws               15  us-east-1        ec2         Amazon Web Services
aws-china          1  cn-north-1       ec2         Amazon China
aws-gov            1  us-gov-west-1    ec2         Amazon (USA Government)
azure             26  centralus        azure       Microsoft Azure
azure-china        2  chinaeast        azure       Microsoft Azure China
cloudsigma         5  hnl              cloudsigma  CloudSigma Cloud
google            13  us-east1         gce         Google Cloud Platform
joyent             6  eu-ams-1         joyent      Joyent Cloud
oracle             5  uscom-central-1  oracle      Oracle Cloud
rackspace          6  dfw              rackspace   Rackspace Cloud
localhost          1  localhost        lxd         LXD Container Hypervisor
```

3. Create a Juju controller.

   ```
   juju bootstrap --bootstrap-series=xenial <cloud name> <controller name>
   ```

   A Juju controller manages and keeps track of applications in the Juju cloud environment.

4. Download the Contrail bundle from JAAS - Contrail Kubernetes.

Deploying Contrail Charms with Kubernetes

IN THIS SECTION

Juju Charms simplifies Contrail deployment by providing a simple way to deploy, configure, scale, and manage Contrail operations.

You can deploy Contrail Charms in a bundle or manually.

Deploying Contrail Charms in a Bundle

Follow these steps to deploy Contrail Charms in a bundle.

1. Deploy Contrail Charms.

   To deploy Contrail Charms in a bundle, use the `juju deploy <bundle_yaml_file>` command.

   The following example shows you how to use a bundle YAML file to deploy Contrail on Amazon Web Services (AWS) Cloud.

```
series: "bionic"

machines:

  # kubernetes pods
  0:
    series: "bionic"
    constraints: mem=8G cores=2 root-disk=60G

  # kubernetes master
  2:
    series: "bionic"
    constraints: mem=8G cores=2 root-disk=60G

  # contrail components
  5:
    series: "bionic"
```

```
    constraints: mem=16G cores=4 root-disk=60G

services:

  # kubernetes

  easyrsa:
    series: "bionic"
    charm: cs:~containers/easyrsa
    num_units: 1
    annotations:
      gui-x: '1168.1039428710938'
      gui-y: '-59.11077045466004'
    to:
    - lxd:2

  etcd:
    series: "bionic"
    charm: cs:~containers/etcd
    annotations:
      gui-x: '1157.2041015625'
      gui-y: '719.1614406201691'
    num_units: 1
    options:
      channel: 3.2/stable
    to: [2]

  kubernetes-master:
    series: "bionic"
    charm: cs:~containers/kubernetes-master-696
    annotations:
      gui-x: '877.1133422851562'
      gui-y: '325.6035540382413'
    expose: true
    num_units: 1
    options:
      channel: '1.14/stable'
      service-cidr: '10.96.0.0/12'
      docker_runtime: 'custom'
      docker_runtime_repo: 'deb [arch={ARCH}] https://download.docker.com/linux/ubuntu {CODE}
stable'
      docker_runtime_key_url: 'https://download.docker.com/linux/ubuntu/gpg'
      docker_runtime_package: 'docker-ce'
```

```
    to: [2]

  kubernetes-worker:
    series: "bionic"
    charm: cs:~containers/kubernetes-worker-550
    annotations:
      gui-x: '745.8510131835938'
      gui-y: '-57.369691124215706'
    num_units: 1
    options:
      channel: '1.14/stable'
      docker_runtime: 'custom'
      docker_runtime_repo: 'deb [arch={ARCH}] https://download.docker.com/linux/ubuntu {CODE}
stable'
      docker_runtime_key_url: 'https://download.docker.com/linux/ubuntu/gpg'
      docker_runtime_package: 'docker-ce'
    to: [0]

  # contrail-kubernetes

  contrail-kubernetes-master:
    series: "bionic"
    charm: cs:~juniper-os-software/contrail-kubernetes-master
    annotations:
      gui-x: '586.8027801513672'
      gui-y: '753.914497641757'
    options:
      log-level: 'SYS_DEBUG'
      service_subnets: '10.96.0.0/12'
      docker-registry: "opencontrailnightly"
      image-tag: "master-latest"

  contrail-kubernetes-node:
    series: "bionic"
    charm: cs:~juniper-os-software/contrail-kubernetes-node
    annotations:
      gui-x: '429.1971130371094'
      gui-y: '216.05209087397168'
    options:
      log-level: 'SYS_DEBUG'
      docker-registry: "opencontrailnightly"
      image-tag: "master-latest"
```

```
# contrail

contrail-agent:
  series: "bionic"
  charm: cs:~juniper-os-software/contrail-agent
  annotations:
    gui-x: '307.5467224121094'
    gui-y: '-24.150856522753656'
  options:
    log-level: 'SYS_DEBUG'
    docker-registry: "opencontrailnightly"
    image-tag: "master-latest"

contrail-analytics:
  series: "bionic"
  charm: cs:~juniper-os-software/contrail-analytics
  annotations:
    gui-x: '15.948270797729492'
    gui-y: '705.2326686475128'
  expose: true
  num_units: 1
  options:
    log-level: 'SYS_DEBUG'
    docker-registry: "opencontrailnightly"
    image-tag: "master-latest"
  to: [5]

contrail-analyticsdb:
  series: "bionic"
  charm: cs:~juniper-os-software/contrail-analyticsdb
  annotations:
    gui-x: '24.427139282226562'
    gui-y: '283.9550754931123'
  num_units: 1
  options:
    cassandra-minimum-diskgb: '4'
    cassandra-jvm-extra-opts: '-Xms1g -Xmx2g'
    log-level: 'SYS_DEBUG'
    docker-registry: "opencontrailnightly"
    image-tag: "master-latest"
  to: [5]

contrail-controller:
```

```
    series: "bionic"
    charm: cs:~juniper-os-software/contrail-controller
    annotations:
      gui-x: '212.01282501220703'
      gui-y: '480.69961284662793'
    expose: true
    num_units: 1
    options:
      auth-mode: 'no-auth'
      cassandra-minimum-diskgb: '4'
      cassandra-jvm-extra-opts: '-Xms1g -Xmx2g'
      log-level: 'SYS_DEBUG'
      docker-registry: "opencontrailnightly"
      image-tag: "master-latest"
    to: [5]

  # misc

  ntp:
    charm: "cs:bionic/ntp"
    annotations:
      gui-x: '678.6017761230469'
      gui-y: '415.27124759750086'

relations:


- [ kubernetes-master:kube-api-endpoint, kubernetes-worker:kube-api-endpoint ]
- [ kubernetes-master:kube-control, kubernetes-worker:kube-control ]
- [ kubernetes-master:certificates, easyrsa:client ]
- [ kubernetes-master:etcd, etcd:db ]
- [ kubernetes-worker:certificates,  easyrsa:client ]
- [ etcd:certificates, easyrsa:client ]

# contrail
- [ kubernetes-master, ntp ]
- [ kubernetes-worker, ntp ]
- [ contrail-controller, ntp ]

- [ contrail-controller, contrail-analytics ]
- [ contrail-controller, contrail-analyticsdb ]
- [ contrail-analytics, contrail-analyticsdb ]
- [ contrail-agent, contrail-controller ]
```

```
# contrail-kubernetes
- [ contrail-kubernetes-node:cni, kubernetes-master:cni ]
- [ contrail-kubernetes-node:cni, kubernetes-worker:cni ]
- [ contrail-kubernetes-master:contrail-controller, contrail-controller:contrail-controller ]
- [ contrail-kubernetes-master:kube-api-endpoint, kubernetes-master:kube-api-endpoint ]
- [ contrail-agent:juju-info, kubernetes-worker:juju-info ]
- [ contrail-agent:juju-info, kubernetes-master:juju-info ]
- [ contrail-kubernetes-master:contrail-kubernetes-config, contrail-kubernetes-node:contrail-
kubernetes-config ]
```

You can create or modify the Contrail Charm deployment bundle YAML file to:

- Point to machines or instances where the Contrail Charms must be deployed.
- Include the options you need.

  Each Contrail Charm has a specific set of options. The options you choose depend on the charms you select. For more information on the options that are available, see *config.yaml* file for each charm located at Contrail Charms.

2. (Optional) Check the status of deployment.

   You can check the status of the deployment by using the `juju status` command.

3. Enable configuration statements.

   Based on your deployment requirements, you can enable the following configuration statements:

   - `contrail-agent`

     For more information, see https://github.com/tungstenfabric/tf-charms/blob/master/contrail-agent/README.md.

   - `contrail-analytics`

     For more information, see https://github.com/tungstenfabric/tf-charms/blob/master/contrail-analytics/README.md.

   - `contrail-analyticsdb`

     For more information, see https://github.com/tungstenfabric/tf-charms/blob/master/contrail-analyticsdb/README.md.

   - `contrail-controller`

     For more information, see https://github.com/tungstenfabric/tf-charms/blob/master/contrail-controller/README.md.

   - `contrail-kubernetes-master`

For more information, see https://github.com/tungstenfabric/tf-charms/blob/master/contrail-kubernetes-master/README.md.

- `contrail-kubernetes-node`

  For more information, see https://github.com/tungstenfabric/tf-charms/blob/master/contrail-kubernetes-node/README.md.

Deploying Juju Charms with Kubernetes Manually

Before you begin deployment, ensure that you have:

- Installed and configured Juju
- Created a Juju controller
- Installed Ubuntu 16.04 or Ubuntu 18.04

Follow these steps to deploy Juju Charms with Kubernetes manually:

1. Create machine instances for Kubernetes master, Kubernetes workers, and Contrail.

```
juju add-machine ssh:<sshusername>@<IP> --constraints mem=8G cores=2 root-disk=32G --
series=xenial  #for Kubernetes worker machine
```

```
juju add-machine ssh:<sshusername>@<IP> --constraints mem=18G cores=2 root-disk=32G --
series=xenial #for Kubernetes master machine
juju add-machine ssh:<sshusername>@<IP> --constraints mem=16G cores=4 root-disk=32G --
series=xenial #for Contrail machine
```

2. Deploy the Kubernetes services.

   Some of the applications may need an additional configuration.

   You can deploy Kubernetes services using any one of the following methods:

   - By specifying the Kubernetes parameters in a YAML file
   - By using CLI
   - By using a combination of YAML-formatted file and CLI

   NOTE: You must use the same docker version for Contrail and Kubernetes.

   For more details, refer to Juju Application Configuration.

3. Deploy and configure `ntp`, `easyrsa`, `etcd`, `kubernetes-master`, `kubernetes-worker`.

```
juju deploy cs:xenial/ntp ntp

juju deploy cs:~containers/easyrsa easyrsa --to lxd:0

juju deploy cs:~containers/etcd etcd \
    --resource etcd=3 \
    --resource snapshot=0
juju set etcd channel="3.2/stable"

juju deploy cs:~containers/kubernetes-master kubernetes-master \
    --resource cdk-addons=0 \
    --resource kube-apiserver=0 \
    --resource kube-controller-manager=0 \
    --resource kube-proxy=0 \
    --resource kube-scheduler=0 \
    --resource kubectl=0
juju set kubernetes-master channel="1.14/stable" \
    enable-dashboard-addons="false" \
    enable-metrics="false" \
    dns-provider="none" \
    docker_runtime="custom" \
    docker_runtime_repo="deb [arch={ARCH}] https://download.docker.com/linux/ubuntu {CODE}
stable" \
    docker_runtime_key_url="https://download.docker.com/linux/ubuntu/gpg" \
    docker_runtime_package="docker-ce"

juju deploy cs:~containers/kubernetes-worker kubernetes-worker \
    --resource kube-proxy="0" \
    --resource kubectl="0" \
    --resource kubelet="0"
juju set kubernetes-worker channel="1.14/stable" \
    ingress="false" \
    docker_runtime="custom" \
    docker_runtime_repo="deb [arch={ARCH}] https://download.docker.com/linux/ubuntu {CODE}
stable" \
    docker_runtime_key_url="https://download.docker.com/linux/ubuntu/gpg" \
    docker_runtime_package="docker-ce"
```

4. Deploy and configure Contrail services.

Deploy `contrail-analyticsdb`, `contrail-analytics`, `contrail-controller`, `contrail-kubernetes-master`, `contrail-kubernetes-node`, `contrail-agent` from the directory where you have downloaded the charms.

> **NOTE**: You must set the *auth-mode* parameter of the contrail-controller charm to **no-auth** if Contrail is deployed without a keystone.

```
juju deploy contrail-analytics contrail-analytics

juju deploy contrail-analyticsdb contrail-analyticsdb
juju set contrail-analyticsdb cassandra-minimum-diskgb="4" cassandra-jvm-extra-opts="-Xms1g -Xmx2g"

juju deploy contrail-controller contrail-controller
juju set contrail-controller cassandra-minimum-diskgb="4" cassandra-jvm-extra-opts="-Xms1g -Xmx2g" auth-mode="no-auth"

juju deploy contrail-kubernetes-master contrail-kubernetes-master

juju deploy contrail-kubernetes-node contrail-kubernetes-node

juju deploy contrail-agent contrail-agent
```

5. Enable applications to be available to external traffic:

```
juju expose kubernetes-master
juju expose kubernetes-worker
```

6. Enable contrail-controller and contrail-analytics services to be available to external traffic if you do not use HAProxy.

```
juju expose contrail-controller
juju expose contrail-analytics
```

7. Apply SSL.

   You can apply SSL if needed. To use SSL with Contrail services, deploy easy-rsa service and `add-relation` command to create relations to contrail-controller service and contrail-agent services.

```
juju add-relation easyrsa contrail-controller
juju add-relation easyrsa contrail-analytics
```

```
juju add-relation easyrsa contrail-analyticsdb
juju add-relation easyrsa contrail-kubernetes-master
juju add-relation easyrsa contrail-agent
```

8. Add other necessary relations.

```
juju add-relation "contrail-controller" "contrail-analytics"
juju add-relation "contrail-controller" "contrail-analyticsdb"
juju add-relation "contrail-analytics" "contrail-analyticsdb"
juju add-relation "contrail-agent" "contrail-controller"
juju add-relation "contrail-controller" "ntp"
juju add-relation “kubernetes-worker”, “ntp”
juju add-relation “kubernetes-master”, “ntp”

juju add-relation "kubernetes-master:kube-api-endpoint" "kubernetes-worker:kube-api-endpoint"
juju add-relation "kubernetes-master:kube-control" "kubernetes-worker:kube-control"
juju add-relation "kubernetes-master:certificates" "easyrsa:client"
juju add-relation "kubernetes-master:etcd" "etcd:db"
juju add-relation "kubernetes-worker:certificates" "easyrsa:client"
juju add-relation "etcd:certificates" "easyrsa:client"

juju add-relation contrail-agent:juju-info, kubernetes-master:juju-info

juju add-relation "contrail-kubernetes-node:cni" "kubernetes-master:cni"
juju add-relation "contrail-kubernetes-node:cni" "kubernetes-worker:cni"
juju add-relation "contrail-kubernetes-master:contrail-controller" "contrail-
controller:contrail-controller"
juju add-relation "contrail-kubernetes-master:kube-api-endpoint" "kubernetes-master:kube-api-
endpoint"
juju add-relation "contrail-agent:juju-info" "kubernetes-worker:juju-info"
juju add-relation "contrail-agent:juju-info" "kubernetes-master:juju-info"
juju add-relation "contrail-kubernetes-master:contrail-kubernetes-config" "contrail-
kubernetes-node:contrail-kubernetes-config"
```

RELATED DOCUMENTATION

https://juju.is/docs/installing

Installing Contrail with Kubernetes in Nested Mode by Using Juju Charms

Contrail Networking Release 1909 and later support provisioning of a Kubernetes cluster inside an OpenStack cluster. Contrail Networking offers a nested control and data plane where a single Contrail control plane and a single network stack can manage and service both the OpenStack and Kubernetes clusters.

In nested mode, a Kubernetes cluster is provisioned in virtual machines of an OpenStack cluster. The CNI plugin and the Contrail-Kubernetes manager of the Kubernetes cluster interface directly with Contrail components that manage the OpenStack cluster.

All Kubernetes features, functions and specifications are supported when used in nested mode.

NOTE: Nested mode deployment is only supported for Contrail with OpenStack cluster.

Before you begin:

- Deploy Contrail with OpenStack either on bare metal server or virtual machines.

  BEST PRACTICE: Public cloud deployment is not recommended because of slow nested virtualization.

- The VMs must have internet connectivity.
- Contrail in underlay network must be configured to support nested mode.

  You must select an unused IP in the cluster to configure *link-local*.

  For example:

  10.10.10.5 is the selected service IP.

LL Service Name	Service IP	Service Port	Fabric IP	Fabric Port
K8s-cni-to-agent	10.10.10.5	9091	127.0.0.1	9091

Follow these steps to deploy Juju Charms with Kubernetes in nested mode using bundle deployment:

Use this method if you want to use the existing machines.

1. Create a Juju controller.

   ```
   juju bootstrap --bootstrap-series=xenial <cloud name> <controller name>
   ```

You can use OpenStack Cloud provider or manually spun-up VMs. For details, refer to Preparing to Deploy Contrail with Kubernetes by Using Juju Charms.

2. Deploy bundle.

```
juju deploy --series xenial cs:~containers/kubernetes-worker-550 --to:0 \ --config channel="1.14/stable" \ --config docker_runtime="custom" \
```

If the machines for the setup are already provisioned, run the following command to deploy bundle:

```
juju deploy --map-machines=existing,0=0,5=1 ./bundle.yaml
```

where `bundle-id=existing-id`

For details, refer to https://jaas.ai/u/juniper-os-software/contrail-k8s-nested/bundle.

or

Follow these steps to deploy Juju Charms with Kubernetes in nested mode manually:

1. Create a Juju controller.

```
juju bootstrap --bootstrap-series=xenial <cloud name> <controller name>
```

You can use OpenStack Cloud provider or manually spun-up VMs. For details, refer to Preparing to Deploy Contrail with Kubernetes by Using Juju Charms.

2. Create machine instances for Contrail components, Kubernetes master and Kubernetes workers.

Sample constraints for minimal deployment:

All-In-One deployment:

```
juju add-machine --constraints mem=32G cores=8 root-disk=150G --series=xenial # for all-in-one machine
```

or

Multinode deployment:

```
juju add-machine --constraints mem=8G cores=2 root-disk=50G --series=xenial # kubernetes workers juju add-machine --constraints mem=8G cores=2 root-disk=50G --series=xenial # kubernetes masters juju add-machine --constraints mem=4G cores=4 root-disk=50G --series=xenial # contrail components
```

You can use any series—*xenial* or *bionic*.

3. Add machines to the cloud.

For details, refer to Using Constraints-Juju.

4. Deploy the Kubernetes services.

Some of the applications may need additional configuration.

You can deploy Kubernetes services using any one of the following methods:

- By specifying the Kubernetes parameters in a YAML file.
- By passing options/values directly on the command line.

NOTE: You must use the same docker version for Contrail and Kubernetes.

For more details, refer to Juju Application Configuration.

5. Deploy and configure `ntp`, `easyrsa`, `etcd`, `kubernetes-master`, `kubernetes-worker`.

```
juju deploy --series xenial cs:ntp ntp

juju deploy --series xenial cs:~containers/easyrsa --to lxd:0

juju deploy --series xenial cs:~containers/etcd --to:0 --config channel="3.2/stable"

juju deploy --series xenial cs:~containers/kubernetes-master-696 --to:0 \
    --config channel="1.14/stable" \
    --config docker_runtime="custom" \
    --config docker_runtime_repo="deb [arch={ARCH}] https://download.docker.com/linux/ubuntu
{CODE} stable" \
    --config docker_runtime_key_url="https://download.docker.com/linux/ubuntu/gpg" \
    --config docker_runtime_package="docker-ce"

juju deploy --series xenial cs:~containers/kubernetes-worker-550 --to:0 \
    --config channel="1.14/stable" \
    --config ingress="false" \
    --config docker_runtime="custom" \
    --config docker_runtime_repo="deb [arch={ARCH}] https://download.docker.com/linux/ubuntu
{CODE} stable" \
    --config docker_runtime_key_url="https://download.docker.com/linux/ubuntu/gpg" \
    --config docker_runtime_package="docker-ce"
```

6. Deploy and configure Contrail services.

Deploy `contrail-kubernetes-master, contrail-kubernetes-node, contrail-agent` from the directory where you have downloaded the charms.

```
contrail-kubernetes-master:
    nested_mode: true
    cluster_project: "{'domain':'default-domain','project':'admin'}"
    cluster_network: "{'domain':'default-domain','project':'admin','name':'juju-net'}"
    service_subnets: '10.96.0.0/12'
    nested_mode_config: |
        {
        "CONTROLLER_NODES": "10.0.12.20",
        "AUTH_MODE": "keystone",
        "KEYSTONE_AUTH_ADMIN_TENANT": "admin",
        "KEYSTONE_AUTH_ADMIN_USER": "admin",
        "KEYSTONE_AUTH_ADMIN_PASSWORD": "password",
        "KEYSTONE_AUTH_URL_VERSION": "/v2.0",
        "KEYSTONE_AUTH_HOST": "10.0.12.122",
        "KEYSTONE_AUTH_PROTO": "http",
        "KEYSTONE_AUTH_PUBLIC_PORT":"5000",
        "KEYSTONE_AUTH_REGION_NAME": "RegionOne",
        "KEYSTONE_AUTH_INSECURE": "True",
        "KUBERNESTES_NESTED_VROUTER_VIP": "10.10.10.5"
        }
juju deploy --series xenial cs:~juniper-os-software/contrail-kubernetes-master \
    --config ./path-to-config.yaml

juju deploy --series xenial cs:~juniper-os-software/contrail-kubernetes-node
```

7. Add the necessary relations.

```
juju add-relation "kubernetes-master:juju-info" "ntp:juju-info"
juju add-relation "kubernetes-worker:juju-info" "ntp:juju-info"

juju add-relation "kubernetes-master:kube-api-endpoint" "kubernetes-worker:kube-api-endpoint"
juju add-relation "kubernetes-master:kube-control" "kubernetes-worker:kube-control"
juju add-relation "kubernetes-master:certificates" "easyrsa:client"
juju add-relation "kubernetes-master:etcd" "etcd:db"
juju add-relation "kubernetes-worker:certificates" "easyrsa:client"
juju add-relation "etcd:certificates" "easyrsa:client"

juju add-relation "contrail-kubernetes-node:cni" "kubernetes-master:cni"
```

```
juju add-relation "contrail-kubernetes-node:cni" "kubernetes-worker:cni"
juju add-relation "contrail-kubernetes-master:kube-api-endpoint" "kubernetes-master:kube-api-endpoint"
juju add-relation "contrail-kubernetes-master:contrail-kubernetes-config" "contrail-kubernetes-node:contrail-kubernetes-config"
```

8. Apply SSL, if needed.

   You must provide the same certificates to the *contrail-kubernetes-master* node if Contrail in underlay cluster has SSL enabled.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
1909	Contrail Networking Release 1909 and later support provisioning of a Kubernetes cluster inside an OpenStack cluster. Contrail Networking offers a nested control and data plane where a single Contrail control plane and a single network stack can manage and service both the OpenStack and Kubernetes clusters.

RELATED DOCUMENTATION

Installing Contrail with Kubernetes by Using Juju Charms | 653

Installing Contrail with OpenStack by Using Juju Charms | 596

Installing OpenStack Octavia LBaaS with Juju Charms in Contrail Networking

Contrail Networking Release 2005 supports Octavia as LBaaS. The deployment supports RHOSP and Juju platforms.

With Octavia as LBaaS, Contrail Networking is only maintaining network connectivity and is not involved in any load balancing functions.

For each OpenStack load balancer creation, Octavia launches a VM known as *amphora VM*. The VM starts the HAPROXY when listener is created for the load balancer in OpenStack. Whenever the load balancer gets updated in OpenStack, *amphora VM* updates the running HAPROXY configuration. The *amphora VM* is deleted on deleting the load balancer.

Contrail Networking provides connectivity to *amphora VM* interfaces. *Amphora VM* has two interfaces; one for management and the other for data. The management interface is used by the Octavia services for the management communication. Since, Octavia services are running in the underlay network and *amphora VM* is running in the overlay network, SDN gateway is needed to reach the overlay network. The data interface is used for load balancing.

Follow the procedure to install OpenStack Octavia LBaaS in Canonical deployment:

1. Prepare Juju setup with OpenStack Train version and Octavia overlay bundle.

   Refer to "No Link Title" on page 677 output.

   ```
   juju deploy --overlay=./octavia-bundle.yaml ./contrail-bundle.yaml
   ```

   or

   Add Octavia service after deploying the main bundle on the existing cluster.

   ```
   juju deploy --overlay=./octavia-bundle.yaml --map-machines=existing ./contrail-bundle.yaml
   ```

2. Prepare ssh key for amphora VM. Add the options in the **octavia-bundle.yaml** file.

   ```
   ssh-keygen -f octavia # generate the key base64 octavia.pub # print public key data
   ```

   Add the following options to Octavia options.

   ```
   amp-ssh-pub-key: # paste public key data here amp-ssh-key-name: octavia
   ```

3. Generate certificates.

   ```
   rm -rf demoCA/
   mkdir -p demoCA/newcerts
   touch demoCA/index.txt
   touch demoCA/index.txt.attr
   openssl genrsa -passout pass:foobar -des3 -out issuing_ca_key.pem 2048
   openssl req -x509 -passin pass:foobar -new -nodes -key issuing_ca_key.pem \
       -config /etc/ssl/openssl.cnf \
       -subj "/C=US/ST=Somestate/O=Org/CN=www.example.com" \
       -days 30 \
       -out issuing_ca.pem
   openssl genrsa -passout pass:foobar -des3 -out controller_ca_key.pem 2048
   openssl req -x509 -passin pass:foobar -new -nodes \
   ```

```
        -key controller_ca_key.pem \
    -config /etc/ssl/openssl.cnf \
    -subj "/C=US/ST=Somestate/O=Org/CN=www.example.com" \
    -days 30 \
    -out controller_ca.pem
openssl req \
    -newkey rsa:2048 -nodes -keyout controller_key.pem \
    -subj "/C=US/ST=Somestate/O=Org/CN=www.example.com" \
    -out controller.csr
openssl ca -passin pass:foobar -config /etc/ssl/openssl.cnf \
    -cert controller_ca.pem -keyfile controller_ca_key.pem \
    -create_serial -batch \
    -in controller.csr -days 30 -out controller_cert.pem
cat controller_cert.pem controller_key.pem > controller_cert_bundle.pem
juju config octavia \
    lb-mgmt-issuing-cacert="$(base64 controller_ca.pem)" \
    lb-mgmt-issuing-ca-private-key="$(base64 controller_ca_key.pem)" \
    lb-mgmt-issuing-ca-key-passphrase=foobar \
    lb-mgmt-controller-cacert="$(base64 controller_ca.pem)" \
    lb-mgmt-controller-cert="$(base64 controller_cert_bundle.pem)"
```

Make sure all the units are in *active* or *blocked* state.

4. Configure vault service.

   a. SSH into the machine where vault service is installed.

   ```
   juju ssh vault/0
   ```

   b. Export vault address and run `init`.

   ```
   export VAULT_ADDR='http://localhost:8200'
   /snap/bin/vault operator init -key-shares=5 -key-threshold=3
   ```

   It will print 5 unseal keys and initial root token.

   c. Call unseal command by using any three of the five printed unseal keys.

   ```
   /snap/bin/vault operator unseal Key1
   /snap/bin/vault operator unseal Key2
   /snap/bin/vault operator unseal Key3
   ```

d. Export initial root token.

```
export VAULT_TOKEN="..."
```

e. Create user token.

```
/snap/bin/vault token create -ttl=10m
```

f. Exit from vault's machine and initialize vault's charm with the user token.

```
juju run-action --wait vault/leader authorize-charm token="..."
```

5. Create amphora image.

```
juju run-action --wait octavia-diskimage-retrofit/leader retrofit-image
```

For more details, refer to https://docs.openstack.org/project-deploy-guide/charm-deployment-guide/latest/app-octavia.html#amphora-image.

6. Install *python-openstackclient* and *python-octaviaclient* and create management network for Octavia.

You must create these objects in *services* project.

```
project=$(openstack project list --domain service_domain | awk '/services/{print $2}')
openstack network create octavia --tag charm-octavia --project $project
openstack subnet create --subnet-range 172.x.0.0/24 --network octavia --tag charm-octavia
octavia
# security group for octavia
openstack security group create octavia --tag charm-octavia --project $project
openstack security group rule create --ingress --ethertype IPv4 --protocol icmp octavia
openstack security group rule create --ingress --ethertype IPv6 --protocol icmp octavia
openstack security group rule create --ingress --ethertype IPv4 --protocol tcp --dst-port
22:22 octavia
openstack security group rule create --ingress --ethertype IPv6 --protocol tcp --dst-port
22:22 octavia
openstack security group rule create --ingress --ethertype IPv6 --protocol tcp --dst-port
9443:9443 octavia
openstack security group rule create --ingress --ethertype IPv4 --protocol tcp --dst-port
9443:9443 octavia
```

```
# security group for octavia-health
openstack security group create octavia-health --tag charm-octavia-health --project $project
openstack security group rule create --ingress --ethertype IPv4 --protocol icmp octavia-health
openstack security group rule create --ingress --ethertype IPv6 --protocol icmp octavia-health
openstack security group rule create --ingress --ethertype IPv4 --protocol udp --dst-port
5555:5555 octavia-health
openstack security group rule create --ingress --ethertype IPv6 --protocol udp --dst-port
5555:5555 octavia-health
```

7. The management network created in step 6 is in overlay network and Octavia services are running in the underlay network. Verify network connectivity between overlay and underlay network via SDN gateway.
8. Configure Octavia with the created network.

   ```
   juju run-action --wait octavia/leader configure-resources
   ```

   Make sure the juju cluster is functional and all units have *active* status.

If you want to run amphora instances on DPDK computes, you have to create your own flavor with the required options and set the ID to configuration of Octavia charm via *custom-amp-flavor-id* option before call configure-resources.
Or
Set the required options to created flavor with name *charm-octavia* by charm

```
openstack flavor set  charm-octavia  --property hw:mem_page_size=any
```

Here is an example for creating and testing load balancer:

Prerequisites:

- You must have connectivity between Octavia controller and amphora instances,
- You must have OpenStack services into LXD containers.
- You must have separate interfaces for control plane and data plane.

1. Create private network.

   ```
   openstack network create private
   openstack subnet create private --network private --subnet-range 10.10.10.0/24 --allocation-
   pool
   start=10.10.10.50,end=10.10.10.70 --gateway none
   ```

2. Create security group.

```
openstack security group create allow_all
openstack security group rule create --ingress --protocol any --prefix '0.0.0.0/0' allow_all
```

3. Check available flavors and images. You can create them, if needed.

```
openstack flavor list
openstack image list
```

4. Create two servers for load balancer.

```
openstack server create --flavor test_flavor --image cirros --security-group allow_all --
network private cirros1
openstack server create --flavor test_flavor --image cirros --security-group allow_all --
network private cirros2
```

5. Create additional server to test load balancer.

```
openstack server create --flavor test_flavor --image cirros --security-group allow_all --
network private cirros-test
```

6. Check status and IP addresses.

```
openstack server list --long
```

7. Create simple HTTP server on every cirros. Login on both the cirros instances and run following commands:

```
MYIP=$(ifconfig eth0|grep 'inet addr'|awk -F: '{print $2}'| awk '{print $1}') while true;
do echo -e "HTTP/1.0 200 OK\r\n\r\nWelcome to $MYIP" | sudo nc -l -p 80 ; done&
```

8. Create load balancer

```
openstack loadbalancer create --name lb1 --vip-subnet-id private
```

Make sure *provisioning_status* is *Active*.

```
openstack loadbalancer show lb1
```

9. Setup load balancer

```
openstack loadbalancer listener create --protocol HTTP --protocol-port 80 --name listener1
lb1
openstack loadbalancer show lb1  # Wait for the provisioning_status to be ACTIVE.
openstack loadbalancer pool create --lb-algorithm ROUND_ROBIN --listener listener1 --
protocol HTTP --name pool1
openstack loadbalancer healthmonitor create --delay 5 --timeout 2 --max-retries 1 --type
HTTP pool1
openstack loadbalancer member create --subnet-id private --address 10.10.10.50 --protocol-
port 80 pool1
openstack loadbalancer member create --subnet-id private --address 10.10.10.51 --protocol-
port 80 pool1
```

IP addresses 10.10.10.50 and 10.10.10.51 belong to VMs created with test http server in step "7" on page 675.

10. Check the status of load balancer.

```
openstack loadbalancer show lb1  # Wait for the provisioning_status to be ACTIVE.
openstack loadbalancer pool list
openstack loadbalancer pool show pool1
openstack loadbalancer member list pool1
openstack loadbalancer listener list
```

11. Login to load balancer client and verify if round robin works.

```
ubuntu@comp-1:~$ ssh cirros@169.x.0.9
The authenticity of host '169.x.0.9 (169.x.0.9)' can't be established.
RSA key fingerprint is SHA256:jv0qgZkorxxxxxxxmykOSVQV3fFl0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '169.x.0.9' (RSA) to the list of known hosts.
cirros@169.x.0.9's password:
$ curl 10.10.10.50
Welcome to 10.10.10.52
$ curl 10.10.10.50
```

```
Welcome to 10.10.10.53
$ curl 10.10.10.50
Welcome to 10.10.10.52
$ curl 10.10.10.50
Welcome to 10.10.10.53
$ curl 10.10.10.50
Welcome to 10.10.10.52
$ curl 10.10.10.50
Welcome to 10.10.10.53
```

Sample octavia-bundle.yaml file

```
# Loadbalancer (LBAASv2) with Octavia - requires Rocky or later
---
applications:
  barbican:
    charm: cs:barbican
    num_units: 1
    options:
      openstack-origin: cloud:bionic-train
    to:
    - lxd:4
  barbican-vault:
    charm: cs:barbican-vault-12
  octavia:
    series: bionic
    charm: cs:~apavlov-e/octavia
    num_units: 1
    options:
      openstack-origin: cloud:bionic-train
      create-mgmt-network: false
    to:
    - lxd:4
  octavia-dashboard:
    charm: cs:octavia-dashboard
  vault:
    charm: cs:vault
    num_units: 1
    to:
    - lxd:4
  glance-simplestreams-sync:
    charm: cs:glance-simplestreams-sync
```

```
    num_units: 1
    options:
      source: ppa:simplestreams-dev/trunk
      use_swift: false
    to:
    - lxd:4
  octavia-diskimage-retrofit:
    charm: cs:octavia-diskimage-retrofit
    options:
      amp-image-tag: 'octavia-amphora'
      retrofit-uca-pocket: train
relations:
- - mysql:shared-db
  - octavia:shared-db
- - mysql:shared-db
  - barbican:shared-db
- - mysql:shared-db
  - vault:shared-db
- - keystone:identity-service
  - octavia:identity-service
- - keystone:identity-service
  - barbican:identity-service
- - rabbitmq-server:amqp
  - octavia:amqp
- - rabbitmq-server:amqp
  - barbican:amqp
- - neutron-api:neutron-load-balancer
  - octavia:neutron-api
- - openstack-dashboard:dashboard-plugin
  - octavia-dashboard:dashboard
- - barbican-vault:secrets
  - barbican:secrets
- - vault:secrets
  - barbican-vault:secrets-storage
- - glance-simplestreams-sync:juju-info
  - octavia-diskimage-retrofit:juju-info
- - keystone:identity-service
  - glance-simplestreams-sync:identity-service
- - rabbitmq-server:amqp
  - glance-simplestreams-sync:amqp
- - keystone:identity-credentials
  - octavia-diskimage-retrofit:identity-credentials
```

```
- - contrail-openstack
  - octavia
```

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
2005	Contrail Networking Release 2005 supports Octavia as LBaaS.

RELATED DOCUMENTATION

- Support for OpenStack LBaaS | 564
- Using Load Balancers in Contrail | 550
- Installing OpenStack Octavia LBaaS with RHOSP in Contrail Networking | 513

Using Netronome SmartNIC vRouter with Contrail Networking and Juju Charms

IN THIS SECTION

NOTE: The Netronome SmartNIC vRouter technology covered in this document is available for evaluation purposes only. It is not intended for deployment in production networks.

You can deploy Contrail Networking by using Juju charms. Juju helps you deploy, configure, and efficiently manage applications on private clouds and public clouds. Juju accesses the cloud with the help of a Juju controller. A charm is a module containing a collection of scripts and metadata and is used with Juju to deploy Contrail.

Starting in Contrail Networking Release 2011, Contrail Networking supports Netronome Agilio CX SmartNICs for Contrail Networking deployment with Juju charms. This feature enables service providers to improve the forwarding performance which includes packets per second (PPS) of vRouter. This optimizes server CPU usage and you can deploy more Virtual Network Functions (VNFs) per server.

Before you begin:

- Equip compute nodes with Netronome Agilio CX SmartNIC. For details, see Agilio CX SmartNICs documentation.
- Retrieve Agilio charm.

  Register on Netronome support site at https://help.netronome.com and provide Docker Hub credentials.

  Netronome will provide the Agilio charm for SmartNIC vRouter deployment on compute nodes. Add the charm version as *charm* variable in the "No Link Title" on page 682. Also, Netronome will authorize Docker Hub registry access.
- Note the *Container Tags* for your Contrail image to customize the *image-tag* variable in the "No Link Title" on page 682. See README Access to Contrail Registry 21XX.
- Note the following version tags:

  agilio-image-tag: 2.48-ubuntu-queens

  maas version: 2.6.2

  Linux kernel: bionic (ga-18.04)

Contrail supports the following charms:

- contrail-agent
- contrail-analytics
- contrail-analyticsdb
- contrail-controller
- contrail-keystone-auth
- contrail-openstack

The following topics describe how to use Netronome SmartNIC vRouter with Contrail Networking and Juju charms.

Prepare to Install Contrail Networking by Using Juju Charms

Follow these steps to prepare for deployment:

1. Install Juju.

```
sudo apt-get update
sudo apt-get upgrade
apt install snapd -y
snap install juju --classic
```

2. Configure Juju.

   You can add a cloud to Juju, and manage clouds already added to Juju. Juju recognizes a wide range of cloud types for adding a cloud.

   This is an example for adding a cloud by using interactive command.

   Example: Adding an MAAS cloud to Juju

```
juju add-cloud
Cloud Types
  maas
  manual
  openstack
  oracle
  vsphere

Select cloud type: maas

Enter a name for your maas cloud: maas-cloud

Enter the API endpoint url: http://<ip-address>:<node>/MAAS

Cloud "maas-cloud" successfully added
You may bootstrap with 'juju bootstrap maas-cloud'
```

NOTE: Juju 2.x is compatible with MAAS series 1.x and 2.x.

3. Create a Juju controller.

```
juju bootstrap --bootstrap-series=xenial <cloud name> <controller name>
```

NOTE: A Juju controller manages and keeps track of applications in the Juju cloud environment.

Deploy Contrail Charms in a Bundle

Follow these steps to deploy Contrail charms in a bundle.

1. Deploy Contrail charms.

   To deploy Contrail charms in a bundle, use the `juju deploy <bundle_yaml_file>` command.

   The following example shows you how to use **bundle_yaml_file** to deploy Contrail Networking with Netronome SmartNIC vRouter on MAAS based deployment.

   Bundle yaml file

```
series: bionic
variables:
  openstack-origin:            &openstack-origin            distro
  #vhost-gateway:              &vhost-gateway               "192.x.40.254"
  data-network:                &data-network                "192.x.40.0/24"
  control-network:             &control-network             "192.x.30.0/24"
  virtioforwarder-coremask:    &virtioforwarder-coremask    "1,2"
  agilio-registry:             &agilio-registry             "netronomesystems"
  agilio-image-tag:            &agilio-image-tag            "2.48-ubuntu-queens"
  agilio-user:                 &agilio-user                 "<agilio-username>"
  agilio-password:             &agilio-password             "<agilio-password>"
  agilio-insecure:             &agilio-insecure             false
  agilio-phy:                  &agilio-phy                  "nfp_p0"
  docker-registry:             &docker-registry             "<registry-directory>"
  #docker-user:                &docker-user                 "<docker_username>"
  #docker-password:            &docker-password             "<docker_password>"
  image-tag:                   &image-tag                   "2011.61"
  docker-registry-insecure:    &docker-registry-insecure    "true"
  dockerhub-registry:          &dockerhub-registry          "https://index.docker.io/
v1/"
machines:
  "1":
    constraints: tags=controller
    series: bionic
  "2":
    constraints: tags=compute
    series: bionic
```

```
  "3":
    constraints: tags=neutron
    series: bionic
services:
  ubuntu:
    charm: cs:ubuntu
    num_units: 1
    to: [ "1" ]
  ntp:
    charm: cs:ntp
    num_units: 0
    options:
            #source: ntp.ubuntu.com
       source: 10.204.217.158
  mysql:
    charm: cs:percona-cluster
    num_units: 1
    options:
      dataset-size: 15%
      max-connections: 10000
      root-password: <password>
      sst-password: <password>
      min-cluster-size: 1
    to: [ "lxd:1" ]
  rabbitmq-server:
    charm: cs:rabbitmq-server
    num_units: 1
    options:
      min-cluster-size: 1
    to: [ "lxd:1" ]
  heat:
    charm: cs:heat
    num_units: 1
    expose: true
    options:
      debug: true
      openstack-origin: *openstack-origin
    to: [ "lxd:1" ]
  keystone:
    charm: cs:keystone
    expose: true
    num_units: 1
    options:
```

```
    admin-password: <password>
    admin-role: admin
    openstack-origin: *openstack-origin
    preferred-api-version: 3
nova-cloud-controller:
  charm: cs:nova-cloud-controller
  num_units: 1
  expose: true
  options:
    network-manager: Neutron
    openstack-origin: *openstack-origin
  to: [ "lxd:1" ]
neutron-api:
  charm: cs:neutron-api
  expose: true
  num_units: 1
  series: bionic
  options:
    manage-neutron-plugin-legacy-mode: false
    openstack-origin: *openstack-origin
  to: [ "3" ]
glance:
  charm: cs:glance
  expose: true
  num_units: 1
  options:
    openstack-origin: *openstack-origin
  to: [ "lxd:1" ]
openstack-dashboard:
  charm: cs:openstack-dashboard
  expose: true
  num_units: 1
  options:
    openstack-origin: *openstack-origin
  to: [ "lxd:1" ]
nova-compute:
  charm: cs:nova-compute
  num_units: 0
  expose: true
  options:
    openstack-origin: *openstack-origin
nova-compute-dpdk:
  charm: cs:nova-compute
```

```
    num_units: 0
    expose: true
    options:
      openstack-origin: *openstack-origin
  nova-compute-accel:
    charm: cs:nova-compute
    num_units: 2
    expose: true
    options:
      openstack-origin: *openstack-origin
    to: [ "2" ]
  contrail-openstack:
    charm: ./tf-charms/contrail-openstack
    series: bionic
    expose: true
    num_units: 0
    options:
      docker-registry: *docker-registry
      #docker-user: *docker-user
      #docker-password: *docker-password
      image-tag: *image-tag
      docker-registry-insecure: *docker-registry-insecure
  contrail-agent:
    charm: ./tf-charms/contrail-agent
    num_units: 0
    series: bionic
    expose: true
    options:
      log-level: "SYS_DEBUG"
      docker-registry: *docker-registry
      #docker-user: *docker-user
      #docker-password: *docker-password
      image-tag: *image-tag
      docker-registry-insecure: *docker-registry-insecure
      #vhost-gateway: *vhost-gateway
      physical-interface: *agilio-phy
  contrail-agent-dpdk:
    charm: ./tf-charms/contrail-agent
    num_units: 0
    series: bionic
    expose: true
    options:
      log-level: "SYS_DEBUG"
```

```
      docker-registry: *docker-registry
      #docker-user: *docker-user
      #docker-password: *docker-password
      image-tag: *image-tag
      docker-registry-insecure: *docker-registry-insecure
      dpdk: true
      dpdk-main-mempool-size: "65536"
      dpdk-pmd-txd-size: "2048"
      dpdk-pmd-rxd-size: "2048"
      dpdk-driver: ""
      dpdk-coremask: "1-4"
      #vhost-gateway: *vhost-gateway
      physical-interface: "nfp_p0"
  contrail-analytics:
    charm: ./tf-charms/contrail-analytics
    num_units: 1
    series: bionic
    expose: true
    options:
      log-level: "SYS_DEBUG"
      docker-registry: *docker-registry
      #docker-user: *docker-user
      #docker-password: *docker-password
      image-tag: *image-tag
      control-network: *control-network
      docker-registry-insecure: *docker-registry-insecure
    to: [ "1" ]
  contrail-analyticsdb:
    charm: ./tf-charms/contrail-analyticsdb
    num_units: 1
    series: bionic
    expose: true
    options:
      log-level: "SYS_DEBUG"
      cassandra-minimum-diskgb: "4"
      cassandra-jvm-extra-opts: "-Xms8g -Xmx8g"
      docker-registry: *docker-registry
      #docker-user: *docker-user
      #docker-password: *docker-password
      image-tag: *image-tag
      control-network: *control-network
      docker-registry-insecure: *docker-registry-insecure
    to: [ "1" ]
```

```
  contrail-controller:
    charm: ./tf-charms/contrail-controller
    series: bionic
    expose: true
    num_units: 1
    options:
      log-level: "SYS_DEBUG"
      cassandra-minimum-diskgb: "4"
      cassandra-jvm-extra-opts: "-Xms8g -Xmx8g"
      docker-registry: *docker-registry
      #docker-user: *docker-user
      #docker-password: *docker-password
      image-tag: *image-tag
      docker-registry-insecure: *docker-registry-insecure
      control-network: *control-network
      data-network: *data-network
      auth-mode: no-auth
    to: [ "1" ]
  contrail-keystone-auth:
    charm: ./tf-charms/contrail-keystone-auth
    series: bionic
    expose: true
    num_units: 1
    to: [ "lxd:1" ]
  agilio-vrouter5:
    charm: ./charm-agilio-vrt-5-37
    expose: true
    options:
      virtioforwarder-coremask: *virtioforwarder-coremask
      agilio-registry: *agilio-registry
      agilio-insecure: *agilio-insecure
      agilio-image-tag: *agilio-image-tag
      agilio-user: *agilio-user
      agilio-password: *agilio-password
relations:
  - [ "ubuntu", "ntp" ]
  - [ "neutron-api", "ntp" ]
  - [ "keystone", "mysql" ]
  - [ "glance", "mysql" ]
  - [ "glance", "keystone" ]
  - [ "nova-cloud-controller:shared-db", "mysql:shared-db" ]
  - [ "nova-cloud-controller:amqp", "rabbitmq-server:amqp" ]
  - [ "nova-cloud-controller", "keystone" ]
```

```
  - [ "nova-cloud-controller", "glance" ]
  - [ "neutron-api", "mysql" ]
  - [ "neutron-api", "rabbitmq-server" ]
  - [ "neutron-api", "nova-cloud-controller" ]
  - [ "neutron-api", "keystone" ]
  - [ "nova-compute:amqp", "rabbitmq-server:amqp" ]
  - [ "nova-compute", "glance" ]
  - [ "nova-compute", "nova-cloud-controller" ]
  - [ "nova-compute", "ntp" ]
  - [ "openstack-dashboard:identity-service", "keystone" ]
  - [ "contrail-keystone-auth", "keystone" ]
  - [ "contrail-controller", "contrail-keystone-auth" ]
  - [ "contrail-analytics", "contrail-analyticsdb" ]
  - [ "contrail-controller", "contrail-analytics" ]
  - [ "contrail-controller", "contrail-analyticsdb" ]
  - [ "contrail-openstack", "nova-compute" ]
  - [ "contrail-openstack", "neutron-api" ]
  - [ "contrail-openstack", "contrail-controller" ]
  - [ "contrail-agent:juju-info", "nova-compute:juju-info" ]
  - [ "contrail-agent", "contrail-controller"]
  - [ "contrail-agent-dpdk:juju-info", "nova-compute-dpdk:juju-info" ]
  - [ "contrail-agent-dpdk", "contrail-controller"]
  - [ "nova-compute-dpdk:amqp", "rabbitmq-server:amqp" ]
  - [ "nova-compute-dpdk", "glance" ]
  - [ "nova-compute-dpdk", "nova-cloud-controller" ]
  - [ "nova-compute-dpdk", "ntp" ]
  - [ "contrail-openstack", "nova-compute-dpdk" ]
  - [ "contrail-agent:juju-info", "nova-compute-accel:juju-info" ]
  - [ "nova-compute-accel:amqp", "rabbitmq-server:amqp" ]
  - [ "nova-compute-accel", "glance" ]
  - [ "nova-compute-accel", "nova-cloud-controller" ]
  - [ "nova-compute-accel", "ntp" ]
  - [ "contrail-openstack", "nova-compute-accel" ]
  - [ "agilio-vrouter5:juju-info", "nova-compute-accel:juju-info"  ]
  - [ "heat:shared-db" , "mysql:shared-db" ]
  - [ "heat:amqp" , "rabbitmq-server:amqp" ]
  - [ "heat:identity-service" , "keystone:identity-service" ]
  - [ "contrail-openstack:heat-plugin" , "heat:heat-plugin-subordinate" ]
```

You can create or modify the Contrail charm deployment bundle YAML file to:

- Point to machines or instances where the Contrail charms must be deployed.

- Include the options you need.

  Each Contrail charm has a specific set of options. The options you choose depend on the charms you select. For more information on the options that are available, see "Options for Juju Charms" on page 612.

2. (Optional) Check the status of deployment.

   You can check the status of the deployment by using the `juju status` command.

3. Enable configuration statements.

   Based on your deployment requirements, you can enable the following configuration statements:

   - `contrail-agent`

     For more information, see https://jaas.ai/u/juniper-os-software/contrail-agent/.

   - `contrail-analytics`

     For more information, see https://jaas.ai/u/juniper-os-software/contrail-analytics.

   - `contrail-analyticsdb`

     For more information, see https://jaas.ai/u/juniper-os-software/contrail-analyticsdb.

   - `contrail-controller`

     For more information, see https://jaas.ai/u/juniper-os-software/contrail-controller.

   - `contrail-keystone-auth`

     For more information, see https://jaas.ai/u/juniper-os-software/contrail-keystone-auth.

   - `contrail-openstack`

     For more information see, https://jaas.ai/u/juniper-os-software/contrail-openstack.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
2011	Starting in Contrail Networking Release 2011, Contrail Networking supports Netronome Agilio CX SmartNICs for Contrail Networking deployment with Juju charms.

RELATED DOCUMENTATION

Installing Contrail with OpenStack by Using Juju Charms | 596

CHAPTER 18

Using Contrail and Contrail Insights with Kolla/ Ocata OpenStack

IN THIS CHAPTER

Contrail, Contrail Insights, and OpenStack Kolla/Ocata Deployment Requirements

IN THIS SECTION

Starting with Contrail Release 5.0.1, the combined installation of Contrail and Contrail Insights allows Contrail monitoring by Contrail Insights. The following topics are referenced for the deployment.

- "Contrail Insights Installation and Configuration for OpenStack" on page 702
- "Contrail Insights Installation for OpenStack in HA" on page 716

The following software and hardware requirements apply to the combined Contrail, Contrail Insights, and Kolla/Ocata deployment.

Software Requirements

- Contrail Release 5.0.x Targets: Centos 7.5 with kernel 3.10.0-862.3.2.el7.x86_64.
- Contrail Insights Targets: Refer to “Software Requirements” in Contrail Insights General Requirements .
- Targets running both Contrail and Contrail Insights: CentOS 7.5 Ansible 2.4.2 for the installer.
- Contrail Insights 2.18.x and later.

Hardware Requirements

- It is strongly recommended that the Contrail Insights controller and Contrail services be installed on separate targets.
- See "Installing a Contrail Cluster using Contrail Command and instances.yml" on page 78 and "Contrail Insights Installation and Configuration for OpenStack" on page 702 for specifics about requirements for installation.

Preparing for the Installation

IN THIS SECTION

In Contrail Release 5.1, nodes on which Contrail, Contrail Insights (formerly AppFormix), or both are installed are referred to as *targets*. The host from which Ansible is run is referred to as the *base host*. A *base host* can also be a *target*, meaning you can install either Contrail, Contrail Insights, or both on a *base host*.

Preparing the Targets

Workflow for preparing the targets consists of the following steps:

1. Image all the Contrail targets with CentOS 7.5 kernel 3.10.0-862.3.2.el7.x86_64.
2. Install the necessary platform software on the targets on which Contrail Insights Controller or Contrail Insights Agent is going to be installed. See the instructions in *Contrail Insights Installation and Configuration for OpenStack*.

Preparing the Base Host using Ansible Installer

Workflow for preparing the base host consists of the following steps:

1. Install Ansible 2.4.2 on the base host. See “Set Up the Bare Host” in Installing Contrail with OpenStack and Kolla Ansible .
2. Set-up the base host. See “Set Up the Base Host” in Installing Contrail with OpenStack and Kolla Ansible . This section includes information about creating the Ansible `instances.yaml` file.
3. On the base host, create a single Ansible `instances.yaml` file that lists inventory for both Contrail and Contrail Insights deployments. An example of the single `instances.yaml` file is provided later in this section.
   - The Contrail inventory section of the `instances.yaml` file is configured according to guidelines in the section “Set Up the Base Host” in Installing Contrail with OpenStack and Kolla Ansible .
   - The Contrail Insights inventory section of the `instances.yaml` file is configured according to guidelines in *Contrail Insights Installation and Configuration for OpenStack*.

TCP/IP Port Conflicts Between Contrail and Contrail Insights

It is strongly recommended that Contrail Insights Controller and Contrail services be installed on separate target nodes. However, if Contrail Insights Controller and Contrail services are installed on the same target, the following configuration is required to resolve port conflicts.

The following Contrail Insights ports must be reconfigured in the Contrail Insights `group-vars` section of the `instances.yaml` file.

- `appformix_datamanager_port_http`

- appformix_datamanager_port_https
- appformix_haproxy_datamanager_port_http
- appformix_haproxy_datamanager_port_https
- appformix_datamanager_port_http:8200

Plugins to Enable for Contrail and Contrail Insights Deployment

Enable the following plugins by including them in the Contrail Insights group-vars section of the instances.yaml file.

```
appformix_plugins: '{{ appformix_contrail_factory_plugins }}'
appformix_openstack_log_plugins: '{{ appformix_openstack_log_factory_plugins }}'
```

Configuring Contrail Monitoring in Contrail Insights

Connections to Contrail are configured by providing complete URLs by which to access the analytics and configuration API services.

- contrail_cluster_name: **Contrail_Clusterxxx**

  A name by which the Contrail instance will be displayed in the Dashboard. If not specified, this variable has a default value of default_contrail_cluster.

- contrail_analytics_url: **http://analytics-api-node-ip-address:8081**

  URL for the Contrail analytics API. The URL should only specify the protocol, address, and optionally port.

- contrail_config_url: **http://contrail-config-api-server-api-address:8082**

  URL for the Contrail configuration API. The URL should only specify the protocol, address, and optionally port.

NOTE: The IP address specified for contrail monitoring corresponds to one of the IPs listed in the Contrail roles for *config and analytics*. Typically, the first active IP address is selected.

Compute Monitoring: Listing IP Addresses to Monitor

The IP addresses to monitor can be added in the `compute` section of Contrail Insights in the `instances.yaml` file. A list of IP addresses with a *vrouter* role in the `instances.yaml` file.

Configuring Openstack_Controller Hosts for Contrail Insights

The Openstack_controller hosts section must be configured with at least one host. An example section is shown.

```
openstack_controller:
      hosts:
        <ip-address>:
          ansible_connection: ssh
          ansible_ssh_user: <root user>
          ansible_sudo_pass: <contrail password>
```

Other Contrail Insights group_vars That Must be Enabled in instances.yaml

The following `group_vars` must be enabled in `instances.yaml`:

- `openstack_platform_enabled:` **true**
- `appformix_remote_host_monitoring_enabled:` **true**

Contrail Insights License

(Required for Contrail Insights and Contrail Insights Flows installations in Release 2003 and earlier.) You must have an appropriate license that supports the combined deployment of Contrail with Contrail Insights for OpenStack. To obtain a license, send an email to mailto:APPFORMIX-KEY-REQUEST@juniper.net. Also, the following `group_vars`Contrail Insights in the `instances.yaml` file must point to this license.

- **appformix_license: /path/appformix-contrail-license-file.sig**

  This is the path where the license is placed on the *bare host* so that the license can be deployed on the target.

RELATED DOCUMENTATION

Installing Contrail with OpenStack and Kolla Ansible

Run the Playbooks

Refer to section “Install Contrail and Kolla requirements” and section “Deploying contrail and Kolla containers” in Installing Contrail with OpenStack and Kolla Ansible and execute the `ansible-playbook`.

Following are examples listing the Contrail play-book invocation from the `contrail-ansible-deployer` directory:

- Configure Contrail OpenStack instances:

```
ansible-playbook -i inventory/ -e config_file=/path/instances.yaml -e
 orchestrator=openstack playbooks/configure_instances.yml (-vvv for debug)
```

- Install OpenStack:

```
ansible-playbook -i inventory/ -e config_file=/path/instances.yaml
 playbooks/install_openstack.yml
```

- Install Contrail:

```
ansible-playbook -i inventory/ -e config_file=/path/instances.yaml -e
 orchestrator=openstack playbooks/install_contrail.yml
```

Source the `/etc/kolla/kolla-toolbox/admin-openrc.sh` file from the OpenStack controller node (`/etc/kolla/kolla-toolbox/ admin-openrc.sh`) to the Contrail Insights Controller to authenticate the OpenStack adapter to access admin privileges over controller services. If the OpenStack control node is different from the base host, either Secure Copy Protocol (SCP) the file over and source it (for example, `execute source /path/admin-openrc.sh`) or manually export the environment enumerated in `/etc/kolla/kolla-toolbox/ admin-openrc.sh` by invoking `export OS_USERNAME=admin` etc. and the remainder as listed in `admin-openrc.sh`

Also at this point, obtain a list of IP addresses to include in the `compute` section of Contrail Insights in the `instances.yaml` file. Refer to `Compute monitoring: Listing IP addresses to monitor` in the `compute`section of Contrail Insights in the `instances.yaml` file.

Refer to Installing AppFormix for OpenStack and validate target configuration requirements and inventory parameters for Contrail Insights Controller and Agent. In place of `-i inventory/`use `-i /absolute-file-path/instances.yaml`.

Following is an example listing the Contrail Insights playbook invocation from the `appformix-2.18.x` directory where `appformix_openstack.yml` is located:

- Install Contrail Insights:

```
ansible-playbook -i /path/instances.yaml appformix_openstack.yml (-vvv for debug)
```

RELATED DOCUMENTATION

> Installing Contrail with OpenStack and Kolla Ansible
>
> Installing a Contrail Cluster using Contrail Command and instances.yml | 78
>
> *Contrail Insights Installation and Configuration for OpenStack*

Accessing Contrail in Contrail Insights Management Infrastructure in UI

Contrail Insights service monitoring Dashboard for a Contrail cluster displays the overall state of the cluster and its components. For more information, see “Dashboard” in “Contrail Monitoring” in the Contrail Insights User Guide.

Open the Dashboard in a Web browser and log in.

`http://<controller-IP-address>:9000`

RELATED DOCUMENTATION

> Contrail Insights User Guide

Notes and Caveats

- Versions of Contrail Insights-2.17 and earlier are not supported with Ansible-2.4.2. The combined Contrail and Contrail Insights installation is not validated on these earlier releases.

- The installation was validated with Contrail Insights-2.18 Agent.
- To view and monitor Contrail in the Contrail Insights Management Infrastructure dashboard, the license used in the deployment must include support for Contrail.
- Verify the datamanager port (re)definitions in the inventory file.
- For Contrail Insights OpenStack HA installation steps, see *Contrail Insights Installation for OpenStack in HA*.

RELATED DOCUMENTATION

> *Contrail Insights Installation for OpenStack in HA*

Example Instances.yml for Contrail and Contrail Insights OpenStack Deployment

See Installing Contrail with OpenStack and Kolla Ansible and "Contrail Insights Installation and Configuration for OpenStack" on page 702 for specific inventory file details:

The following items are part of the `all` section in the `instances.yaml` file for Contrail Insights:

```
all:
      children:
        openstack_controller:
          hosts:
            <ip-address>:
              ansible_connection: ssh
              ansible_ssh_user: <ssh-user>
              ansible_sudo_pass: <sudo-password>
```

The following items are part of the `vars` section in the `instances.yaml` file for Contrail Insights:

```
openstack_platform_enabled: true
##License must support Contrail and Openstack
appformix_license: /path/license-file.sig
contrail_cluster_name: 'Contrail_Cluster'
contrail_analytics_url: 'http://<contrail-analytics-api-server-ip-address>:8081'
contrail_config_url: 'http://<contrail-config-api-server-ip-address>:8082'
```

```
# Defaults from roles/appformix_defaults/defaults/main.yml are overwritten below
appformix_datamanager_port_http: "{{ (appformix_scale_setup_flag|bool) | ternary(28200, 8200) }}"
appformix_datamanager_port_https: "{{ (appformix_scale_setup_flag|bool) | ternary(28201,
8201) }}"
appformix_haproxy_datamanager_port_http: 8200
appformix_haproxy_datamanager_port_https: 8201
appformix_plugins: '{{ appformix_contrail_factory_plugins }} +
{{ appformix_network_device_factory_plugins }}'
```

Following is an example listing of the `instances.yaml`:

There is one `instances.yaml` file for the Contrail and Contrail Insights combined installation.

```
#Contrail inventory section
provider_config:
  bms:
    ssh_pwd: <ssh-password>
    ssh_user: <ssh-user>
    ntpserver: <ntp-server-ip-address>
    domainsuffix: local
instances:
  bms1:
    provider: bms
    ip: <ip-address>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      vrouter:
      openstack:
      openstack_compute:
global_configuration:
  CONTAINER_REGISTRY: <ci-repository-URL>:5000
  REGISTRY_PRIVATE_INSECURE: True
contrail_configuration:
  #UPGRADE_KERNEL: true
  CONTRAIL_VERSION: <contrail-version>
  #CONTRAIL_VERSION: latest
  CLOUD_ORCHESTRATOR: openstack
```

```
  VROUTER_GATEWAY: <gateway-ip-address>
  RABBITMQ_NODE_PORT: 5673
  PHYSICAL_INTERFACE: <interface-name>
  AUTH_MODE: keystone
  KEYSTONE_AUTH_HOST: <keystone-ip-address>
  KEYSTONE_AUTH_URL_VERSION: /v3
  CONFIG_NODEMGR__DEFAULTS__minimum_diskGB: 2
  DATABASE_NODEMGR__DEFAULTS__minimum_diskGB: 2
kolla_config:
  kolla_globals:
    network_interface: <interface-name>
    kolla_internal_vip_address: <ip-address>
    contrail_api_interface_address: <ip-address>
    enable_haproxy: no
    enable_swift: no
  kolla_passwords:
    keystone_admin_password: <password>

# Contrail Insights inventory section
all:
  children:
    appformix_controller:
      hosts:
        <ip-address>:
          ansible_connection: ssh
          ansible_ssh_user: <ssh-user>
          ansible_sudo_pass: <sudo-password>
    openstack_controller:
      hosts:
        <ip-address>:
          ansible_connection: ssh
          ansible_ssh_user: <ssh-user>
          ansible_sudo_pass: <sudo-password>
    compute:
      hosts:
        #List IP addresses of Contrail roles to be monitored here
        <<IP-addresses>>:
          ansible_connection: ssh
          ansible_ssh_user: <ssh-user>
          ansible_sudo_pass: <sudo-password>
    bare_host:
      hosts:
        <ip-address>:
```

```
          ansible_connection: ssh
          ansible_ssh_user: <ssh-user>
          ansible_sudo_pass: <sudo-password>
        #If host is local
        <ip-address>:
          ansible_connection: local
  vars:
    appformix_docker_images:
      - /opt/software/appformix/contrail-insights-platform-images-<version>.tar.gz
      - /opt/software/appformix/contrail-insights-dependencies-images-<version>.tar.gz
      - /opt/software/appformix/contrail-insights-network_device-images-<version>.tar.gz
      - /opt/software/appformix/contrail-insights-openstack-images-<version>.tar.gz
    openstack_platform_enabled: true
    # appformix_license: /opt/software/openstack_appformix/<appformix-contrail-license-file>.sig
    appformix_license: /opt/software/configs/contrail.sig
    appformix_docker_registry: registry.appformix.com/
    appformix_version: <version>      #Must be 2.18.x or above
    appformix_plugins: '{{ appformix_contrail_factory_plugins }} +
{{ appformix_network_device_factory_plugins }} + {{ appformix_openstack_factory_plugins }}'
    appformix_kvm_instance_discovery: true
    # For enabling pre-requisites for package installation
    appformix_network_device_monitoring_enabled: true
    # For running the appformix-network-device-adapter
    network_device_discovery_enabled: true
    appformix_remote_host_monitoring_enabled: true
    appformix_jti_network_device_monitoring_enabled: true
    contrail_cluster_name: 'Contrail_Cluster'
    contrail_analytics_url: 'http://<contrail-analytics-api-server-IP-address>:8081'
    contrail_config_url: 'http://<contrail-config-api-server-IP-address>:8082'
    # Defaults overwritten below were defined in roles/appformix_defaults/defaults/main.yml
    appformix_datamanager_port_http: "{{ (appformix_scale_setup_flag|bool) | ternary(28200,
8200) }}"
    appformix_datamanager_port_https: "{{ (appformix_scale_setup_flag|bool) | ternary(28201,
8201) }}"
    appformix_haproxy_datamanager_port_http: 8200
    appformix_haproxy_datamanager_port_https: 8201
```

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 21XX.

RELATED DOCUMENTATION

Installing a Contrail Cluster using Contrail Command and instances.yml | 78

Contrail Insights Installation and Configuration for OpenStack | 702

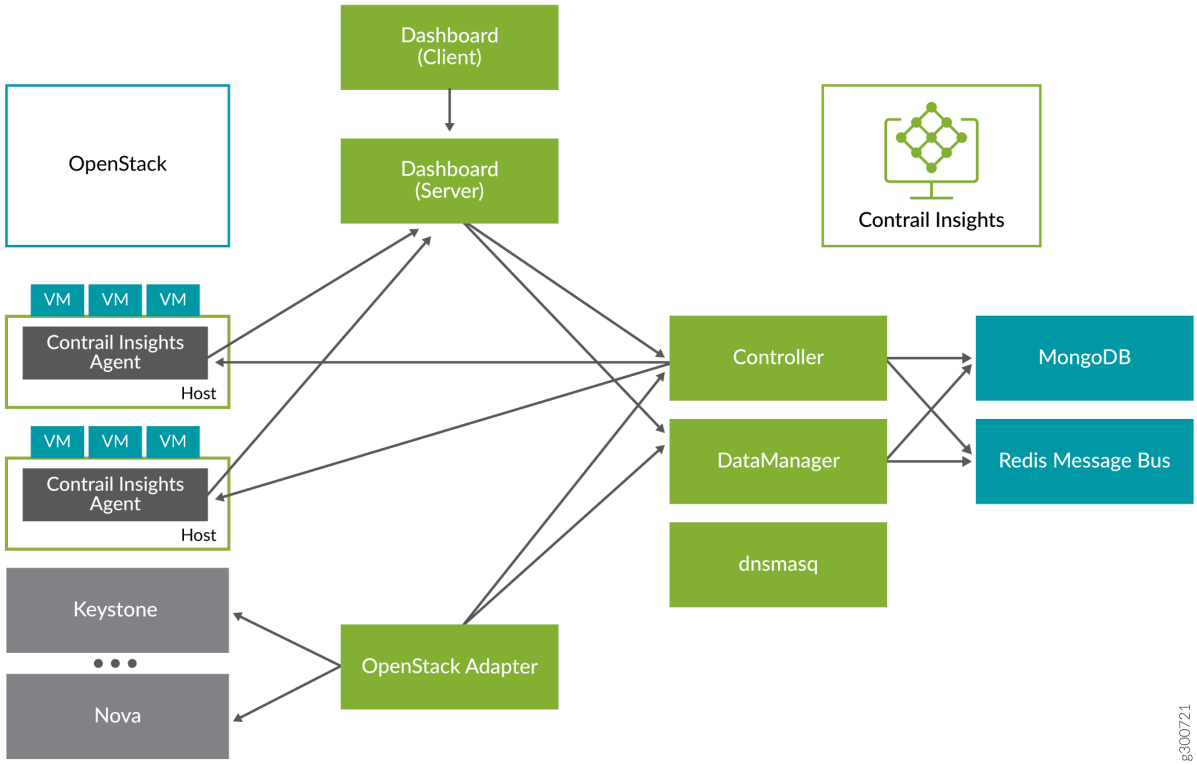
Contrail Insights Installation and Configuration for OpenStack

IN THIS SECTION

Getting Started with OpenStack

Contrail Insights provides resource control and visibility for hosts and virtual machines in an OpenStack cluster. This topic explains how to install Contrail Insights for an OpenStack cluster. Contrail Insights Agent runs on a host to monitor resource consumption of the host itself and the virtual machines executing on that host. Figure 71 on page 703 shows the Contrail Insights architecture with OpenStack.

Figure 71: Contrail Insights Architecture with OpenStack



- Contrail Insights Agent monitors resource usage on the compute nodes.
- Contrail Insights Platform offers REST APIs to configure the system.
- Contrail Insights DataManager stores data from multiple Agents.
- Contrail Insights Dashboard provides a Web-based user interface.
- An adapter discovers platform-specific resources and configures the Contrail Insights Platform.
- Adapters exist for OpenStack and Kubernetes.

Requirements

The following are the requirements for installing Contrail Insights for OpenStack.

- Supported OpenStack versions: Icehouse, Juno, Kilo, Liberty, Mitaka, Newton, Ocata.
- See Contrail Insights General Requirements for software and hardware requirements.
- An administrator account for OpenStack.

- API access to OpenStack services: Cinder, Glance, Heat, Keystone, Neutron, Nova, and Swift. Contrail Insights reads information from these services. The administrator account must provide sufficient permission for read-only API calls. Further, Contrail Insights Platform must be able to open connections to the host and port on which these services listen. Contrail Insights can be configured to use the `admin`, `internal`, or `public` service endpoints. See `OS_ENDPOINT_TYPE` in OpenStack environment variables in the section Installing Contrail Insights.

- 

  NOTE: Upgrade notice: Starting with Contrail Insights 3.2.6, the requirement for a license file is removed. If you are installing a version earlier than 3.2.6, a license is required prior to installation.

  You can obtain a license key from mailto:APPFORMIX-KEY-REQUEST@juniper.net. Provide the following information in your request:

  ```
  Group name:
  Target customers or use:
  Cluster type: OpenStack
  Number of hosts:
  Number of instances:
  ```

Workflow in Three Steps

Installation consists of the following steps:

1. Configure OpenStack.
2. Create Ansible inventory.
3. Install Contrail Insights.

Configure OpenStack

To create an administrator account for Contrail Insights, perform the following steps in the OpenStack Horizon dashboard:

1. Create a user account and name it **appformix**.
2. Select a new project for the user account.
3. Select role as **admin**.

Create Ansible Inventory

Ansible is used to deploy the software to the compute node(s) and the Platform Host. An Ansible inventory file describes groups of hosts in your cluster. Define the inventory in a separate location than the release files, so that the inventory may be reused for an upgrade.

Contrail Insights requires two groups *compute* and *appformix_controller*. Each group lists the hosts in that group. Only the agent is installed on the *compute* hosts. The agent and the Contrail Insights Platform services are installed on the *appformix_controller* host.

Optionally, an *openstack_controller* group can be defined. The agent is installed on hosts in this group to monitor the hosts that execute OpenStack controller services. (New in v2.3.0)

Create a directory `inventory` (or name of your choice) that contains a `hosts` file and a `group_vars/all` file. For example:

```
inventory/
    hosts           # inventory file
    group_vars/
        all         # configuration variables
```

The `inventory/hosts` file contains the list of hosts in each group. For example:

```


[appformix_controller]
appformix01

[compute]
compute01
compute02
compute03

[openstack_controller]
openstack_infra01
openstack_infra02
```

See Ansible inventory documentation.

Ansible Configuration Variables

The Contrail Insights software includes a number of Ansible roles to perform the configuration of Contrail Insights settings. Define the values of variables in the `inventory/group_vars/all` file, in order to be able to use the settings and inventory for future upgrades.

In the `inventory/group_vars/all` file, configure the following variables for installation of Contrail Insights for OpenStack.

```
appformix_docker_images:
    - /path/to/contrail-insights-platform-images-<version>.tar.gz
    - /path/to/contrail-insights-dependencies-images<version>.tar.gz
    - /path/to/contrail-insights-openstack-images-<version>.tar.gz
```

Refer to Platform Dependencies for steps to install dependencies on a Platform Host that cannot fetch files from the Internet.

Configure an HTTP Proxy for Fetching Prerequisites

The Ansible playbook will fetch files from the Internet to install prerequisites on the Platform Host. If the Platform Host requires an HTTP proxy to access the Internet, configure the following variables in `inventory/group_vars/all`:

```
http_proxy_url: 'http://proxy.example.com:1234'

prerequisites_env:

        http_proxy: '{{ http_proxy_url }}'
        https_proxy: '{{ http_proxy_url }}'
```

The `prerequisites_env` is a dictionary that defines environment variables that will be used when invoking commands to install prerequisites. In the above example, the same proxy URL (http://proxy.example.com:1234) is used for both the `http_proxy` and `https_proxy` environment variables because the single proxy can be used to access HTTP and HTTPS URLs. As a convenience, the proxy URL is defined once in the `http_proxy_url` variable. Adjust `prerequisites_env` as necessary for the proxy requirements of your network.

Install Contrail Insights

To install Contrail Insights:

1. Install Ansible on the Contrail Insights Platform node. Ansible will install docker and docker-py on the platform.

```
#Ubuntu
    apt-get install python-pip python-dev                       #Installs Pip
    pip install ansible==2.3.0                                  #Installs Ansible 2.3
    sudo apt-get install build-essential libssl-dev libffi-dev  #Dependencies
    pip install markupsafe httplib2 requests                    #Dependencies

#RHEL/CentOS
    yum install epel-release                                    #Enable EPEL repository
    In case the above command does not work, manually download and install the epel-release
    package with one of the below commands, depending on your system's version.
    yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
    yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm

    yum groupinstall 'Development Tools'                        #Install development tools
    yum install openssl-devel libffi libffi-devel               #Dependencies
    yum install python-pip python-devel                         #Install Pip
    pip install ansible==2.3.0                                  #Install Ansible 2.3
    pip install markupsafe httplib2 requests                    #Dependencies
```

NOTE: For RHEL, the following IPtables rule is needed to access port 9000.

```
iptables -t filter -A IN_public_allow -p tcp --dport 9000 -j ACCEPT
```

2. Install `libvirt` on the compute nodes using the following command:

```
yum -y install libvirt
```

This installs the KVM (Kernel-based Virtual Machine) and associated packages for collecting data from virtual machines running on the compute nodes.

3. On the compute nodes where Contrail Insights Agent runs, verify that python `virtualenv` is installed.

```
#Ubuntu
    apt-get install -y python-pip
    pip install virtualenv
```

```
#RHEL/CentOS
    yum install -y python-pip
    pip install virtualenv
```

4. Enable passwordless login to facilitate Contrail Insights Platform node with Ansible to install agents on the nodes. Create a SSH public key on the node where Ansible playbooks are run, and then copy the key to the `appformix_controller` node.

```
ssh-keygen -t rsa   #Creates Keys

ssh-copy-id -i ~/.ssh/id_rsa.pub <target_host>    #Copies key from the node to other hosts
```

5. Use the `Sample_Inventory` file as a template to create a host file.

```
# Example naming schemes are as below:
#   hostname ansible_ssh_user='username' ansible_sudo_pass='password'

# List all Compute Nodes
[compute]
172.16.70.5
172.16.70.17

# appformix_controller host
#
# Host variables can be defined to control Contrail Insights configuration parameters
# for particular host.  For example, to specify the directory in which MongoDB
# data is stored on hostname1 (the default is /opt/appformix/mongo/data):
#
#   hostname1 appformix_mongo_data_dir=/var/lib/appformix/mongo
#
# For variables with same value for all appformix_controller hosts, set group
# variables below.
#
[appformix_controller]
172.16.70.119

[openstack_controller]
172.16.70.120
```

6. Verify that all the hosts listed in the inventory file are reachable from the Contrail Insights Platform.

```
 export ANSIBLE_HOST_KEY_CHECKING=False   # Eliminates interactive experience prompting for
Known_Hosts

 ansible -i inventory -m ping all         # Pings all the hosts in the inventory file
```

7. At the top-level of the distribution, create a directory named `group_vars`.

```
mkdir group_vars
```

8. Download the Contrail Insights installation packages from software downloads to the Contrail Insights Platform node. Get the following files:

```
contrail-insights-<version>.tar.gz
contrail-insights-dependencies-images-<version>.tar.gz
contrail-insights-openstack-images-<version>.tar.gz
contrail-insights-platform-images-<version>.tar.gz
contrail-insights-network_device-images-<version>.tar.gz
```

If you are installing a version earlier than 3.2.6, copy the Contrail Insights license file to the Contrail Insights Platform node.

9. In `group_vars` directory, create a file named `all`. Add the following:

```
openstack_platform_enabled: true

appformix_manager_version: <version>
appformix_docker_images:
    - /path/to/contrail-insights-platform-images-<version>.tar.gz
    - /path/to/contrail-insights-dependencies-images-<version>.tar.gz
    - /path/to/contrail-insights-openstack-images-<version>.tar.gz
```

If you are installing a version earlier than 3.2.6, include the path to the Contrail Insights license file in `group_vars/all`:

```
appformix_license: path/to/<contrail-insights-license-file>.sig
```

10. Contrail Insights must be configured to communicate with the OpenStack cluster. The Ansible playbooks use OpenStack environment variables to configure Contrail Insights with details of the OpenStack environment.

```
OS_AUTH_URL             Keystone URL (e.g., https://host:5000/)
OS_ENDPOINT_TYPE        endpoint type to communicate with service (default: publicURL)
OS_USERNAME             admin account to be used by Contrail Insights
OS_PASSWORD             password for admin account
OS_PROJECT_NAME         admin project created for Contrail Insights account
OS_PROJECT_DOMAIN_NAME  domain for admin project
OS_USER_DOMAIN_NAME     domain for admin user
OS_DOMAIN_NAME          (optional) use domain-scoped token for admin account
```

11. Source the `openrc` file that contains the (step 10) environment variables and ensure the variables are in the environment of the shell from which the Ansible-playbooks are going to be executed. Then, install Contrail Insights by executing the `appformix_openstack.yml` playbook. Specify the path to the `inventory` directory that you created earlier.

12. Open the Contrail Insights Dashboard in a Web browser. For example:

```
http://<contrail-insights-platform-node-ip>:9000
```

Select **Skip Installation** because the initial configuration was performed by Ansible using the OpenStack environment variables in step 10. Log in to Contrail Insights Dashboard using OpenStack Keystone credentials.

Additional Ansible Configuration

To set up additional Ansible configurations:

1. To install in a Keystone SSL-enabled cluster, include the following variables in the `group_vars/all` file:

```
appformix_keystone_ssl_ca: '/path/to/keystone_ca.crt'
```

Contrail Insights Ansible will distribute this Keystone CA to all of the Contrail Insights Platform nodes and ask Contrail Insights components to talk to Keystone using this CA file with SSL enabled.

NOTE: Deprecation Notice: The `appformix_mongo_cache_size_gb` parameter previously available starting in Contrail Insights 2.19.5 is now deprecated and no longer supported

from Contrail Insights 3.2.0 and going forward. Starting with Contrail Insights version 3.2.0, Mongo will be configured to use a maximum of 40 percent of the available memory on the Contrail Insights Platform nodes.

2. To enable network device monitoring in the cluster, include the following in the `group_vars/all` file:

```
 # For enabling pre-requisites for packdge installation.
  appformix_install_snmp_dependencies: true
  appformix_install_jti_dependencies: true
  # For running the appformix-network-device-adapter
  network_device_discovery_enabled: true
  appformix_plugins: '{{ appformix_network_device_factory_plugins }}'
  # After 3.1, SNMP Traps can be enabled also so appformix_plugins can be specified as below:
  # appformix_plugins: '{{ appformix_network_device_factory_plugins }} +
{{ appformix_snmp_trap_factory_plugins }}'
```

3. To install Contrail Insights certified plug-ins on the cluster, include the following variables in the `group_vars/all` file:

```
appformix_plugins: <list of certified plugins to be installed>
appformix_openstack_log_plugins: <list of OpenStack log plugins to be installed>
```

For example:

```
appformix_plugins:
  - { plugin_info: 'certified_plugins/cassandra_node_usage.json' }
  - { plugin_info: 'certified_plugins/contrail_vrouter.json' }
  - { plugin_info: 'certified_plugins/zookeeper_usage.json' }
  - { plugin_info: 'certified_plugins/heavy_hitters.json' }

appformix_openstack_log_plugins:
  - { plugin_info: 'certified_plugins/cinder_api_logparser.json',
      log_file_path: '/var/log/cinder/cinder-api.log' }
  - { plugin_info: 'certified_plugins/glance_logparser.json',
       log_file_path: '/var/log/glance/glance-api.log' }
  - { plugin_info: 'certified_plugins/keystone_logparser.json',
      log_file_path: '/var/log/apache2/keystone_access.log,/var/log/httpd/
keystone_wsgi_admin_access.log,/var/log/keystone/keystone.log' }
```

For a list of all Contrail Insights certified plug-ins that can be installed, look for the entries starting with `plugin_info` in the file `roles/appformix_defaults/defaults/main.yml`.

The OpenStack log parser plug-ins parse the API log files of each OpenStack service to collect metrics about API calls and response status codes. To install these plug-ins, add them to the variable `appformix_openstack_log_plugins` in `group_vars/all`, as shown above. Each plug-in entry in this list requires a parameter called `log_file_path` to be specified. This parameter should be set to the complete path to the service's API log file on the OpenStack Controller node(s). Multiple comma-separated paths may be specified.

To identify the right log file to be specified in `log_file_path`, look for entries like the following, containing a client IP address, REST call type, and response status code:

```
2019-04-02 06:50:13.103 3465 INFO nova.osapi_compute.wsgi.server [req-d07e953a-6921-4224-a056-
afb6ff69adde 953ea56a96b944b3b170a299af9e87bd 10c9e8809feb4bd1b55955d9c2ed5aba - - -]
172.18.0.6 "GET /v2/10c9e8809feb4bd1b55955d9c2ed5aba/os-hypervisors/detail HTTP/1.1" status:
200 len: 1427 time: 0.0208740
2019-04-02 06:50:13.183 3465 INFO nova.osapi_compute.wsgi.server [req-34b2f686-9eb5-4112-b3fc-
e0b37798a302 953ea56a96b944b3b170a299af9e87bd 10c9e8809feb4bd1b55955d9c2ed5aba - - -]
172.18.0.6 "GET /v2/10c9e8809feb4bd1b55955d9c2ed5aba/servers/detail?
all_tenants=1&status=SHELVED_OFFLOADED HTTP/1.1" status: 200 len: 211 time: 0.0754580
```

Default locations for these files are listed in the variable `appformix_openstack_log_factory_plugins` in `roles/appformix_defaults/defaults/main.yml`.

4. In Contrail Insights version 2.19.8, a timeout value can be configured for connecting to OpenStack services. The default value of this timeout is `10` seconds and can be changed to a value between `5` and `20` seconds (both inclusive). To change the value, add the following variable to `group_vars/all`:

```
appformix_openstack_session_timeout: <number of seconds>
```

To modify the timeout value after the Contrail Insights Platform has been installed, add the variable to the `group_vars/all` file as described above and re-run the Contrail Insights installation playbook. Restart the `appformix-openstack-adapter` Docker container after the playbook has completed:

```
docker restart appformix-openstack-adapter
```

Upgrade Notices

NOTE: In Contrail Insights version 3.2.0, support for discovering OpenStack Octavia Load Balancer services is added. Contrail Insights only collects load balancer state information, such as `provisioning_status` and `operating_status`, as well as flavor information. To enable this service discovery, provide Octavia service's endpoint as variable `appformix_octavia_endpoint_url` in the `group_vars/all` file. For example:

```
appformix_octavia_endpoint_url: http://10.1.1.1:9876
```

Chargeback costs can also be configured for the Octavia Load Balancer services. See Configure Load Balancer Costs.

Run Ansible with the created inventory file.

```
ansible-playbook -i inventory appformix_openstack.yml
```

NOTE: In Contrail Insights version 3.0, the variable `appformix_openstack_factory_plugins` is deprecated. All OpenStack log parser plug-ins have to be specified in the variable `appformix_openstack_log_plugins`. When upgrading from an older version to version 3.0, make sure to move all OpenStack log parser plug-ins defined in `appformix_openstack_factory_plugins` or `appformix_plugins` to `appformix_openstack_log_plugins`. Also, in Contrail Insights version 3.0, all entries in this list have to be specified with a `log_file_path` value, as described in step 3 above.

When upgrading from version 2.18.x to version 3.0, make the following changes in the `group_vars/all` file:

In version 2.18.x:

```
appformix_openstack_factory_plugins:
  - { plugin_info: 'certified_plugins/cinder_api_logparser.json', log_file_path: '/var/log/
cinder/cinder-api.log'}
  - { plugin_info: 'certified_plugins/glance_logparser.json',log_file_path: '/var/log/glance/
api.log'}
  - { plugin_info: 'certified_plugins/heavy_hitters.json' }
  - { plugin_info: 'certified_plugins/keystone_logparser.json', log_file_path: '/var/log/
keystone/keystone.log'}
  - { plugin_info: 'certified_plugins/neutron_logparser.json', log_file_path: '/var/log/neutron/
server.log'}
  - { plugin_info: 'certified_plugins/nova_logparser.json', log_file_path: '/var/log/nova/nova-
```

```
api.log'}

appformix_plugins: {{ appformix_openstack_factory_plugins }} + ...
```

In version 3.0.x:

```
appformix_plugins:
  - { plugin_info: 'certified_plugins/heavy_hitters.json' }

appformix_openstack_log_plugins:
  - { plugin_info: 'certified_plugins/cinder_api_logparser.json', log_file_path: '/var/log/
cinder/cinder-api.log'}
  - { plugin_info: 'certified_plugins/glance_logparser.json',log_file_path: '/var/log/glance/
api.log'}
  - { plugin_info: 'certified_plugins/keystone_logparser.json', log_file_path: '/var/log/
keystone/keystone.log'}
  - { plugin_info: 'certified_plugins/neutron_logparser.json', log_file_path: '/var/log/neutron/
server.log'}
  - { plugin_info: 'certified_plugins/nova_logparser.json', log_file_path: '/var/log/nova/nova-
api.log'}
```

Source the `openrc` file from the OpenStack controller node (`/etc/contrail/openstackrc`) to the Contrail Insights Platform node to authenticate the adapter to access administrator privileges over the controller services. The file should look like the following:

```
export OS_USERNAME=admin
export OS_PASSWORD=<admin-password>
export OS_AUTH_URL=http://172.16.80.2:5000/v2.0/
export OS_NO_CACHE=1
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

Run Ansible with the created inventory file.

```
ansible-playbook -i inventory appformix_openstack.yml
```

Remove a Node from Contrail Insights

To remove a node from Contrail Insights:

1. Edit the inventory file and add `appformix_state=absent` to each node that you want to remove from Contrail Insights.

```
# Example naming schemes are as below:
#   hostname ansible_ssh_user='username' ansible_sudo_pass='password'

# List all Compute Nodes
[compute]
172.16.70.5 appformix_state=absent
172.16.70.17
```

2. Run Ansible with the edited inventory file. This will remove the node and all its resources from Contrail Insights.

```
ansible-playbook -i inventory appformix_openstack.yml
```

Upgrade Contrail Insights for an OpenStack Cluster

Contrail Insights can be easily upgraded by running the `appformix_openstack.yml` playbook of the new release. Follow the same procedure as the installation.

Uninstall Contrail Insights from an OpenStack Cluster

To uninstall Contrail Insights and destroy all data, execute the following command:

```
ansible-playbook -i <inventory_file> clean_appformix_openstack.yml
```

RELATED DOCUMENTATION

- Contrail Insights Agent Requirements

Contrail Insights Installation for OpenStack in HA

IN THIS SECTION

HA Design Overview

Contrail Insights Platform can be deployed to multiple hosts for high availability (HA). Platform services continue to communicate using an API proxy that listens on a virtual IP address. Only one host will have the virtual IP at a time, and so only one API proxy will be the “active” API proxy at a time.

The API proxy is implemented by HAProxy. HAProxy is configured to use services in active-standby or load-balanced active-active mode, depending on the service.

At most, one host will be assigned the virtual IP at any given time. This host is considered the “active” HAproxy. The virtual IP address is assigned to a host by keepalived, which uses VRRP protocol for election.

Services are replicated in different modes of operation. In the “active-passive” mode, HAProxy sends all requests to a single “active” instance of a service. If the service fails, then HAProxy will select a new “active” from the other hosts, and begin to send requests to the new “active” service.In the “active-active” mode, HAProxy load balances requests across hosts on which a service is operational.

Contrail Insights Platform can be deployed in a 3-node, 5-node, or 7-node configuration for high availability.

Requirements

- For each host, on which Contrail Insights Platform is installed, see Contrail Insights General Requirements for hardware and software requirements. For a list of Contrail Insights Agent supported platforms, see Contrail Insights Agent Requirements.

- **NOTE**: Upgrade notice: Starting with Contrail Insights 3.2.6, the requirement for a license file is removed. If you are installing a version earlier than 3.2.6, a license is required prior to installation.

  You can obtain a license key from mailto:APPFORMIX-KEY-REQUEST@juniper.net. Provide the following information in your request:

```
Group name:
Target customers or use:
Cluster type: OpenStack
Number of hosts:
Number of instances:
```

Connectivity

- One virtual IP address to be shared among all the Platform Hosts. This IP address should not be used by any host before installation. It should have reachability from all the Platform Hosts after installation.
- Dashboard client (in browser) must have IP connectivity to the virtual IP.
- IP addresses for each Platform Host for installation and for services running on these hosts to communicate.
- keepalived_vrrp_interface for each Platform Host which would be used for assigning virtual IP address. Details on how to configure this interface is described in the sample_inventory section.

Install Contrail Insights for High Availability

To install Contrail Insights to multiple hosts for high availability:

1. Download the Contrail Insights installation packages from software downloads to the Contrail Insights Platform node. Get the following files:

```
contrail-insights-<version>.tar.gz
contrail-insights-dependencies-images-<version>.tar.gz
contrail-insights-openstack-images-<version>.tar.gz
contrail-insights-platform-images-<version>.tar.gz
contrail-insights-network_device-images-<version>.tar.gz
```

If you are installing a version earlier than 3.2.6, copy the Contrail Insights license file to the Contrail Insights Platform node.

2. Install Ansible on the installer node. Ansible will install docker and the docker Python package on the `appformix_controller`.

```
# sudo apt-get install python-pip python-dev build-essential libssl-dev libffi-dev
# sudo pip install ansible==2.7.6 markupsafe httplib2
```

For Ansible 2.3:

```
# sudo pip install ansible==2.3 markupsafe httplib2 cryptography==1.5
```

3. Install Python and `python-pip` on all the Platform hosts so that Ansible can run between the installer node and the `appformix_controller` node.

```
# sudo apt-get install -y python python-pip
```

4. Install `python pip` package on the hosts where Contrail Insights Agents run.

```
# apt-get install -y python-pip
```

5. To enable passwordless login to all Platform hosts by Ansible, create an SSH public key on the node where Ansible playbooks are run and then copy the key to all the Platform hosts.

```
# ssh-keygen -t rsa                                 #Creates Keys
# ssh-copy-id -i ~/.ssh/id_rsa.pub <platform_host_1>.........#Copies key from the node to
all platform hosts
# ssh-copy-id -i ~/.ssh/id_rsa.pub <platform_host_2>.........#Copies key from the node to
all platform hosts
# ssh-copy-id -i ~/.ssh/id_rsa.pub <platform_host_3>.........#Copies key from the node to
all platform hosts
```

6. Use the sample_inventory file as a template to create a host file. Add all the Platform hosts and compute hosts details.

```
# List all compute hosts which needs to be monitored by Contrail Insights
[compute]
```

```
203.0.113.5
203.0.113.17
# Contrail Insights controller hosts
[appformix_controller]
203.0.113.119 keepalived_vrrp_interface=eth0
203.0.113.120 keepalived_vrrp_interface=eth0
203.0.113.121 keepalived_vrrp_interface=eth0
```

NOTE: Note: In the case of 5-node or 7-node deployment, list all the nodes under `appformix_controller`.

7. At top-level of the distribution, create a directory named `group_vars` and then create a file named `all` inside this directory.

```
# mkdir group_vars
# touch group_vars/all
```

Add the following entries to the newly created `all` file:

```
appformix_vip: <ip-address>
appformix_docker_images:
- /path/to/contrail-insights-platform-images-<version>.tar.gz
- /path/to/contrail-insights-dependencies-images-<version>.tar.gz
- /path/to/contrail-insights-openstack-images-<version>.tar.gz
```

If you are installing a version earlier than 3.2.6, include the path to the Contrail Insights license file in `group_vars/all`:

```
appformix_license: path/to/<contrail-insights-license-file>.sig
```

8. Copy and source the `openrc` file from the OpenStack controller node (`/etc/contrail/openrc`) to the `appformix_controller` to authenticate the adapter to access admin privileges over the controller services.

```
root@installer_node:~# cat /etc/contrail/openrc
export OS_USERNAME=<admin user>
export OS_PASSWORD=<password>
export OS_TENANT_NAME=admin
```

```
export OS_AUTH_URL=http://<openstack-auth-URL>/v2.0/
export OS_NO_CACHE=1
root@installer_node:~# source /etc/contrail/openrc
```

NOTE: In Contrail Insights version 3.2.0, support for discovering OpenStack Octavia Load Balancer services is added. Contrail Insights only collects load balancer state information, such as `provisioning_status` and `operating_status`, as well as flavor information. To enable this service discovery, provide Octavia service's endpoint as variable `appformix_octavia_endpoint_url` in the `group_vars/all` file. For example:

```
appformix_octavia_endpoint_url: http://10.1.1.1:9876
```

Chargeback costs can also be configured for the Octavia Load Balancer services. See Configure Load Balancer Costs.

9. Run Ansible with the created inventory file.

```
ansible-playbook -i inventory appformix_openstack_ha.yml
```

10. If running the playbooks as root user then this step can be skipped. As a non-root user (for example. “ubuntu”), the user “ubuntu” needs access to the `docker` user group. The following command adds the user to the docker group.

```
sudo usermod -aG docker ubuntu
```

CHAPTER 19

Post Installation Tasks

IN THIS CHAPTER

Configuring Role and Resource-Based Access Control

IN THIS SECTION

Contrail Role and Resource-Based Access (RBAC) Overview

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

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

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

API-Level Access Control

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

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

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

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

Where:

object An API resource such as network or subnet.

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

role The Keystone role name.

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

Example: ACL RBAC Object

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

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

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

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

Rule Sets and ACL Objects

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

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

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

Object Level Access Control

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

The `perms2` property has the following fields:

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

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

The `permission` field has the following options:

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

Access is allowed as follows:

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

Configuration

This section describes the parameters used in Contrail RBAC.

Parameter: aaa-mode

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

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

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

  If you are using Contrail Ansible Deployer to provision Contrail Networking, set the value for `AAA_MODE` to `rbac` to enable RBAC by default.

  ```
  contrail_configuration:
    .
    .
    .
    AAA_MODE: rbac
  ```

  If you are installing Contrail Networking from Contrail Command, specify the key and value as `AAA_MODE` and `rbac`, respectively, under the section Contrail Configuration on the **Step 2 Provisioning Options** page.

After enabling RBAC, you must restart the neutron server by running the `service neutron-server restart` command for the changes to take effect.

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

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

Parameter: cloud_admin_role

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

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

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

Configuration Files with Cloud Admin Credentials

The following configuration files contain cloud_admin_role credentials:

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

Changing Cloud Admin Configuration Files

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

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

     ```
     service supervisor-config restart
     ```

   - Neutron server

     ```
     service neutron-server restart
     ```

   - WebUI

     ```
     service supervisor-webui restart
     ```

Global Read-Only Role

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

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

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

Setting the Global Read-Only Role

To set the global read-only role:

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

   /etc/contrail/contrail-api.conf

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

2. Restart the contrail-api service:

   service contrail-api restart

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

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

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

Upgrading from Previous Releases

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

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

Configuring RBAC Using the Contrail User Interface

To use the Contrail UI with RBAC:

1. Set the aaa_mode to no_auth.

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

   aaa_mode = no-auth
   ```

2. Restart the `analytics-api` service.

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

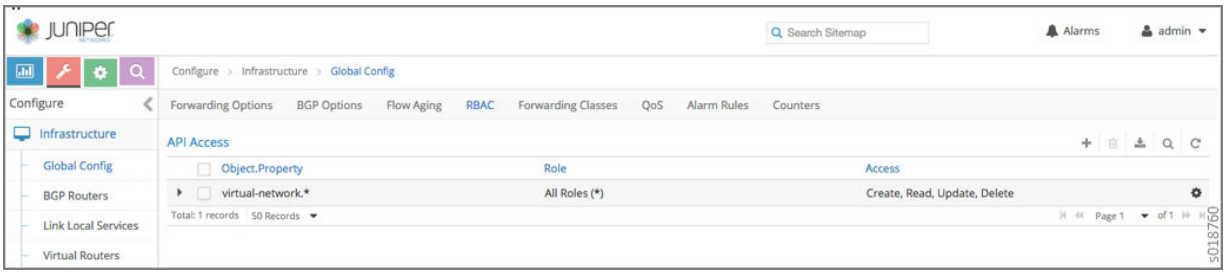
3. Restart services by restarting the container.

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

Configuring RBAC at the Global Level

To configure RBAC at the global level, navigate to **Configure > Infrastructure > Global Config > RBAC**, see Figure 72 on page 727.

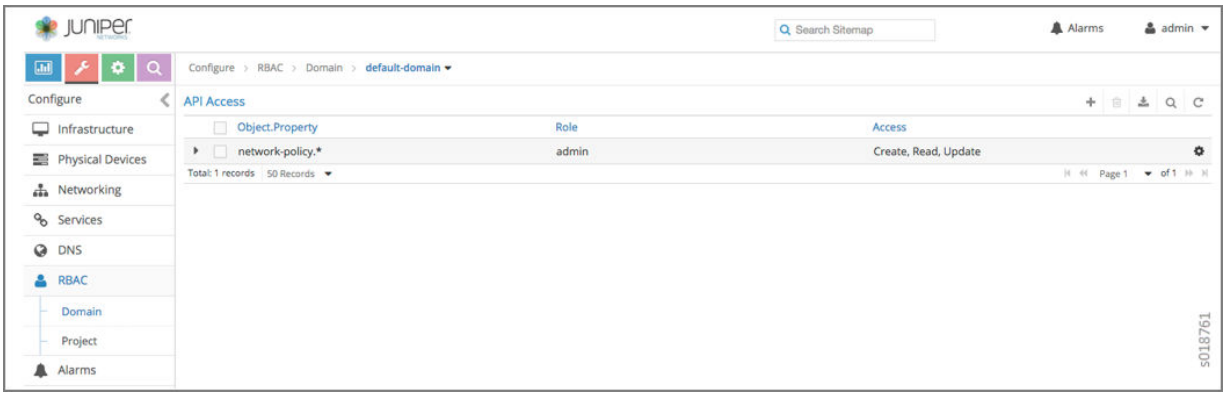
Figure 72: RBAC Global Level



Configuring RBAC at the Domain Level

To configure RBAC at the domain level, navigate to **Configure > RBAC > Domain**, see Figure 73 on page 728.

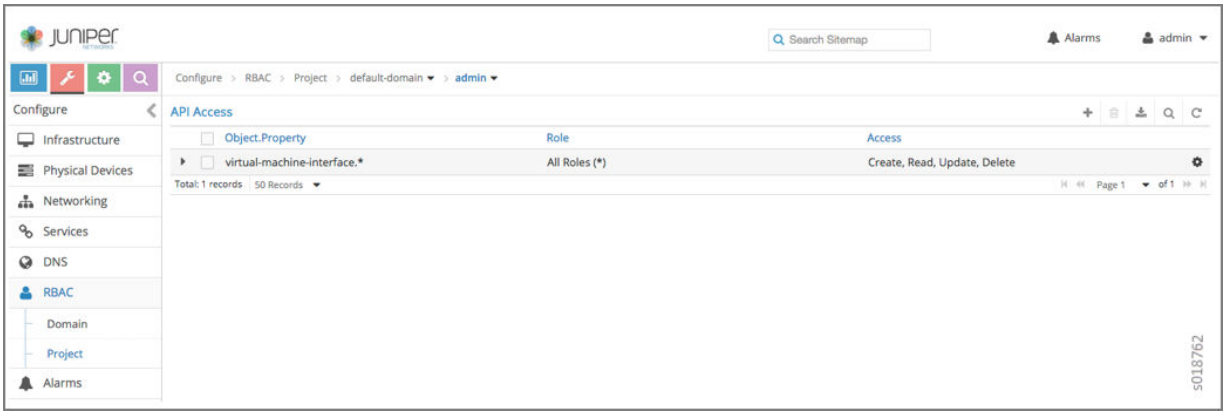
Figure 73: RBAC Domain Level



Configuring RBAC at the Project Level

To configure RBAC at the project level, navigate to **Configure > RBAC > Project**, see Figure 74 on page 728.

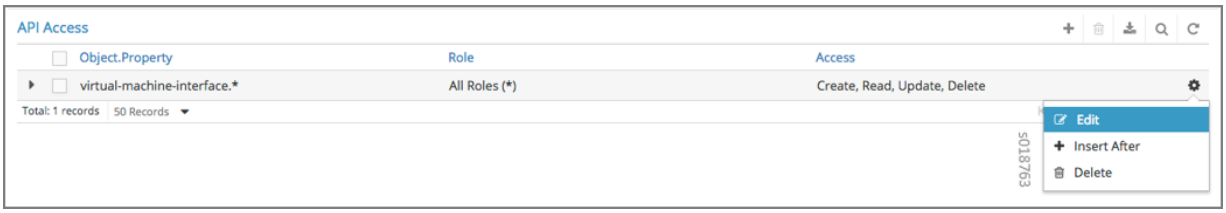
Figure 74: RBAC Project Level



Configuring RBAC Details

Configuring RBAC is similar at all of the levels. To add or edit an API access list, navigate to the global, domain, or project page, then click the plus (+) icon to add a list, or click the gear icon to select from Edit, Insert After, or Delete, see Figure 75 on page 729.

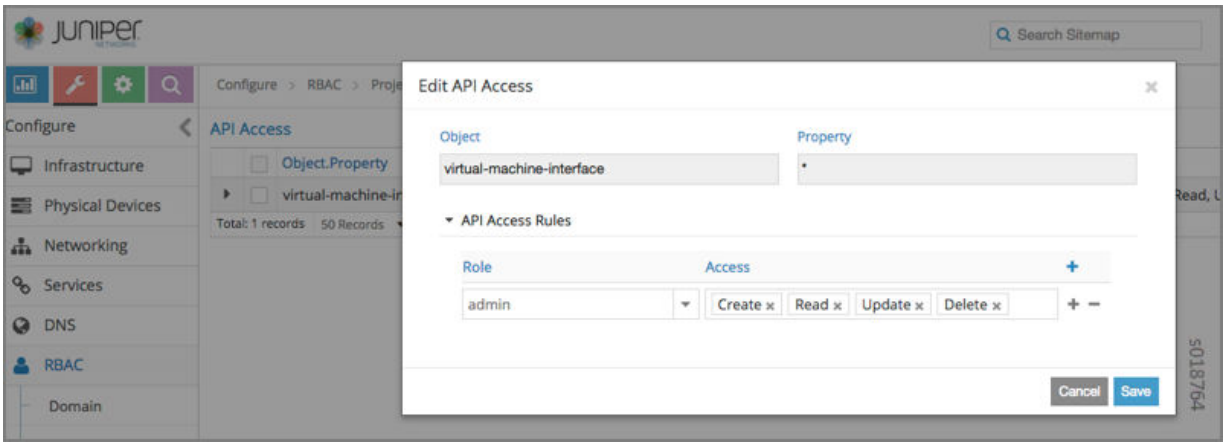
Figure 75: RBAC Details API Access



Creating or Editing API Level Access

Clicking create, edit, or insert after activates the Edit API Access popup window, where you enter the details for the API Access Rules. Enter the user type in the Role field, and use the **+** icon in the Access filed to enter the types of access allowed for the role, including, Create, Read, Update, Delete, and so on, see Figure 76 on page 729.

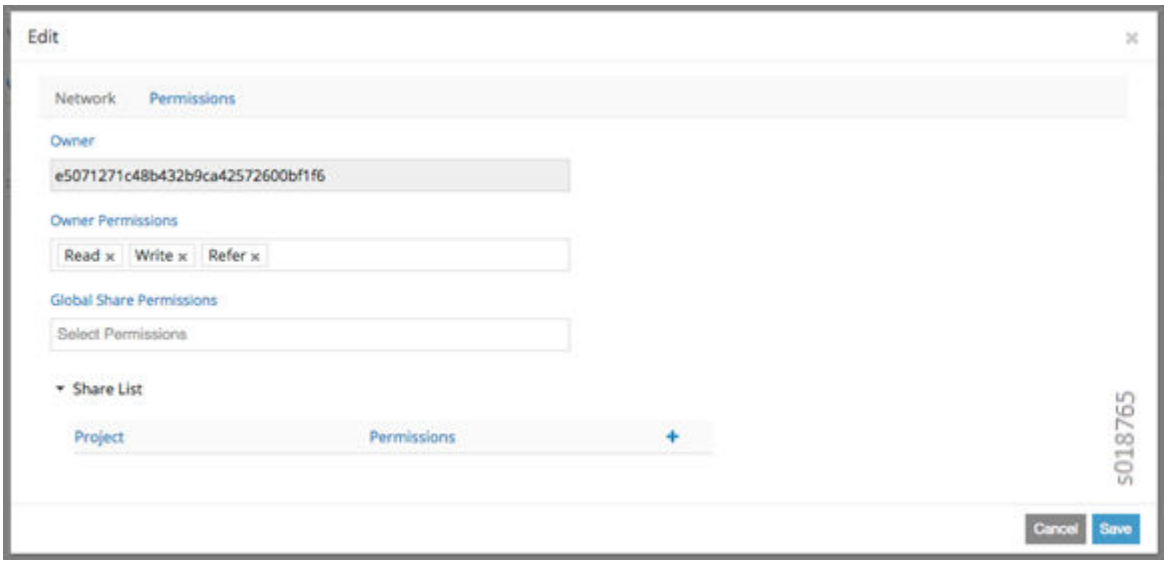
Figure 76: Edit API Access



Creating or Editing Object Level Access

You can configure fine-grained access control by resource. A **Permissions** tab is available on all create or edit popups for resources. Use the **Permissions** popup to configure owner permissions and global share permissions. You can also share the resource to other tenants by configuring it in the **Share List**, see Figure 77 on page 730.

Figure 77: Edit Object Level Access



RBAC Resources

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

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

Configuring Role-Based Access Control for Analytics

The analytics API uses role-based access control (RBAC) to provide the ability to access UVE and query information based on the permissions of the user for the UVE or queried object.

Contrail Networking extends authenticated access so that tenants can view network monitoring information about the networks for which they have read permissions.

The analytics API can map query and UVE objects to configuration objects on which RBAC rules are applied, so that read permissions can be verified using the VNC API.

RBAC is applied to analytics in the following ways:

- For statistics queries, annotations are added to the Sandesh file so that indices and tags on statistics queries can be associated with objects and UVEs. These are used by the contrail-analytics-api to determine the object level read permissions.
- For flow and log queries, the object read permissions are evaluated for each AND term in the where query.

- For UVEs list queries (e.g. analytics/uve/virtual-networks/), the contrail-analytics-api gets a list of UVEs that have read permissions for a given token. For a UVE query for a specific resource (e.g. analytics/uves/virtual-network/vn1), contrail-analytics-api checks the object level read permissions using VNC API.

Tenants cannot view system logs and flow logs, those logs are displayed for cloud-admin roles only.

A non-admin user can see only non-global UVEs, including:

- virtual_network
- virtual_machine
- virtual_machine_interface
- service_instance
- service_chain
- tag
- firewall_policy
- firewall_rule
- address_group
- service_group
- aaplication_policy_set

In `/etc/contrail/contrail-analytics-api.conf`, in the section `DEFAULTS`, the parameter `aaa_mode` now supports `rbac` as one of the values.

Configuring the Control Node with BGP

IN THIS SECTION

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

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

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

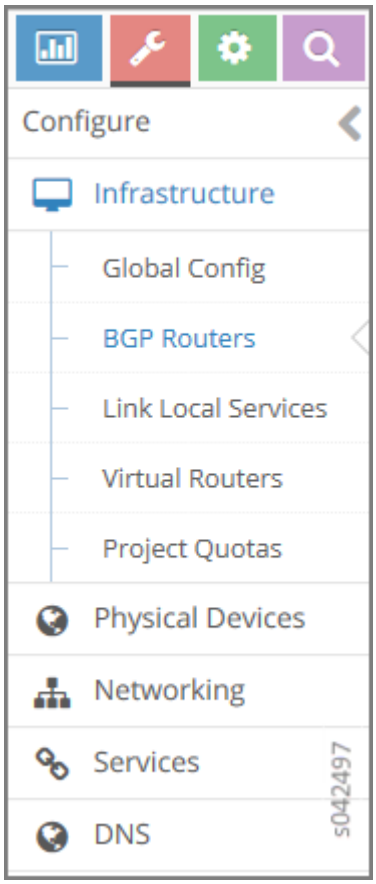
These topics provide instructions to configure the Control Node with BGP.

Configuring the Control Node from Contrail Web UI

To configure BGP peering in the control node:

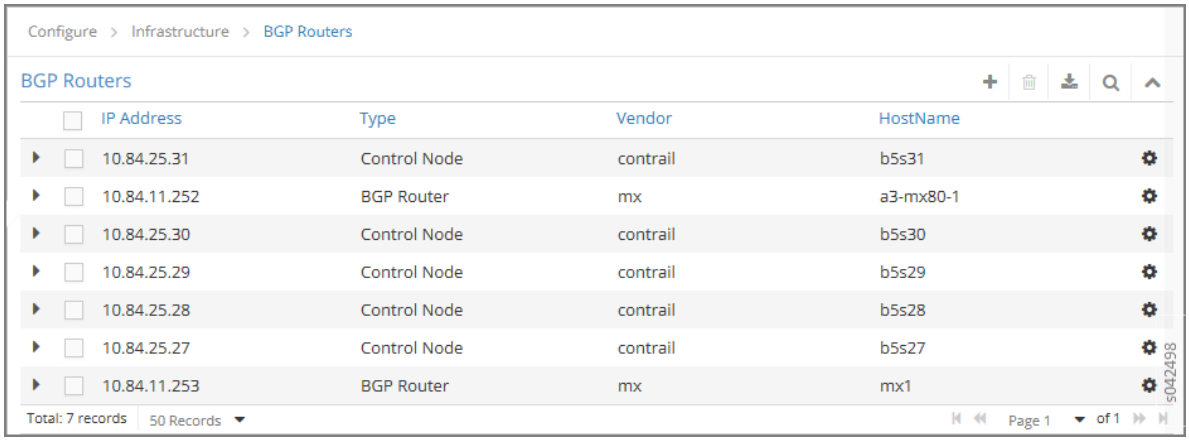
1. From the Contrail Controller module control node (**http://*nn.nn.nn.nn*:8143**), select **Configure > Infrastructure > BGP Routers**; see Figure 78 on page 733.

Figure 78: Configure> Infrastructure > BGP Routers



A summary screen of the control nodes and BGP routers is displayed; see Figure 79 on page 733.

Figure 79: BGP Routers Summary

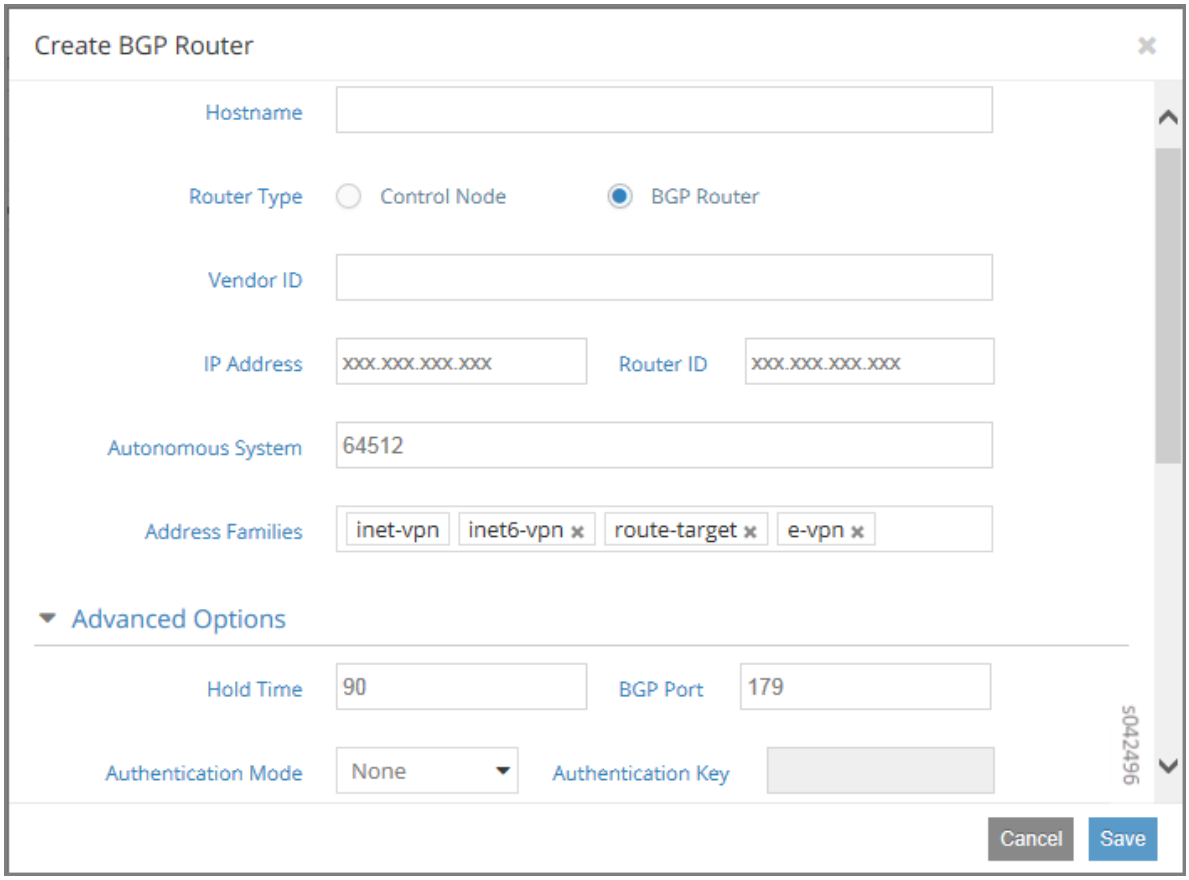


2. (Optional) The global AS number is 64512 by default. To change the AS number, on the **BGP Router** summary screen click the gear wheel and select **Edit**. In the Edit BGP Router window enter the new number.
3. To create control nodes and BGP routers, on the **BGP Routers** summary screen, click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

   icon. The **Create BGP Router** window is displayed; see Figure 80 on page 734.

Figure 80: Create BGP Router



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

   For each node you want to add, populate the fields with values for your system. See Table 48 on page 735.

Table 48: Create BGP Router Fields

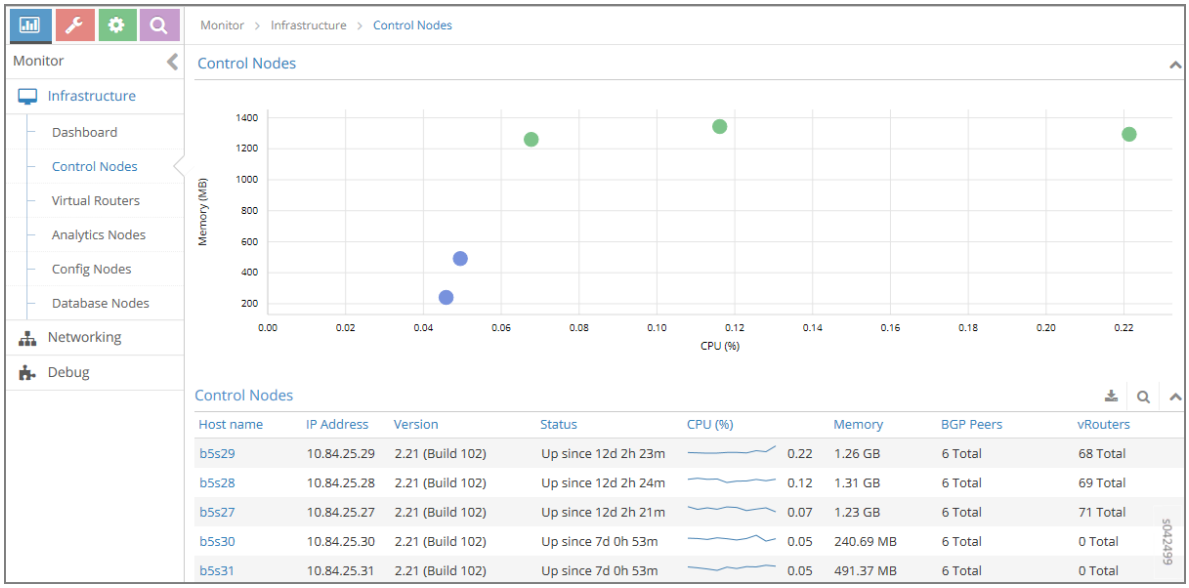
Field	Description
Hostname	Enter a name for the node being added.
Vendor ID	Required for external peers. Populate with a text identifier, for example, “MX-0”. (BGP peer only)
IP Address	The IP address of the node.
Router ID	Enter the router ID.
Autonomous System	Enter the AS number in the range 1-65535 for the node. (BGP peer only)
Address Families	Enter the address family, for example, **inet-vpn**
Hold Time	BGP session hold time. The default is 90 seconds; change if needed.
BGP Port	The default is 179; change if needed.
Authentication Mode	Enable MD5 authentication if desired.
Authentication key	Enter the Authentication Key value.
Physical Router	The type of the physical router.
Available Peers	Displays peers currently available.
Configured Peers	Displays peers currently configured.

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

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

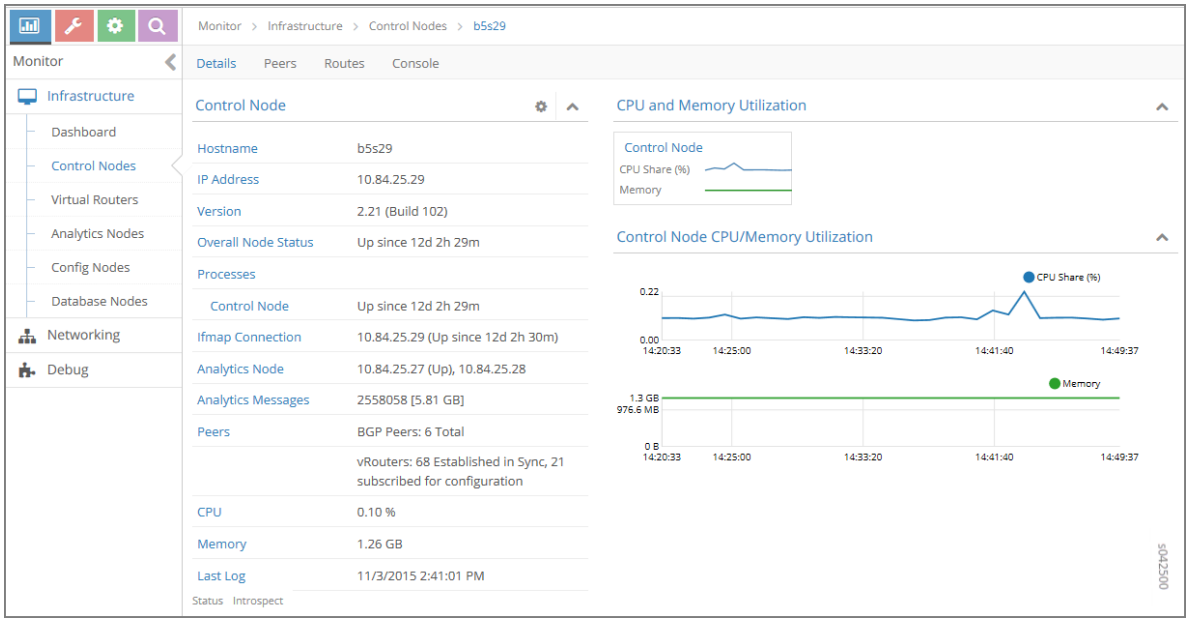
7. You can check for peers by selecting **Monitor > Infrastructure > Control Nodes**; see Figure 81 on page 736.

Figure 81: Control Nodes



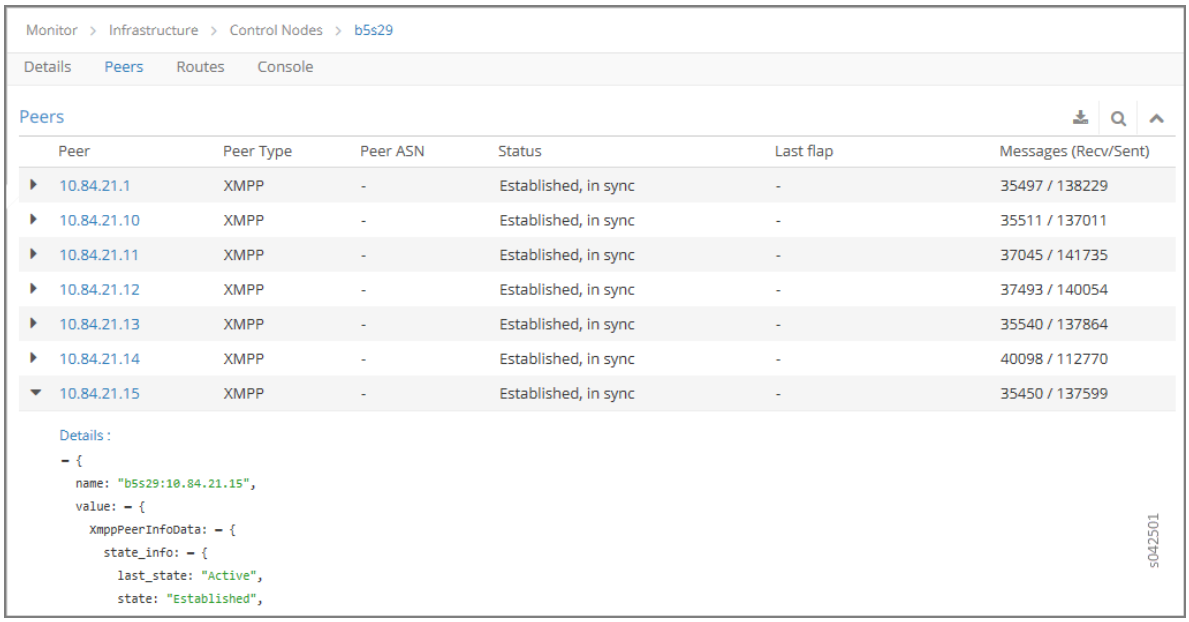
In the **Control Nodes** window, click any hostname in the memory map to view its details; see Figure 82 on page 736.

Figure 82: Control Node Details



8. Click the **Peers** tab to view the peers of a control node; see Figure 83 on page 737.

Figure 83: Control Node Peers Tab



Configuring the Control Node with BGP from Contrail Command

To configure BGP peering in the control node:

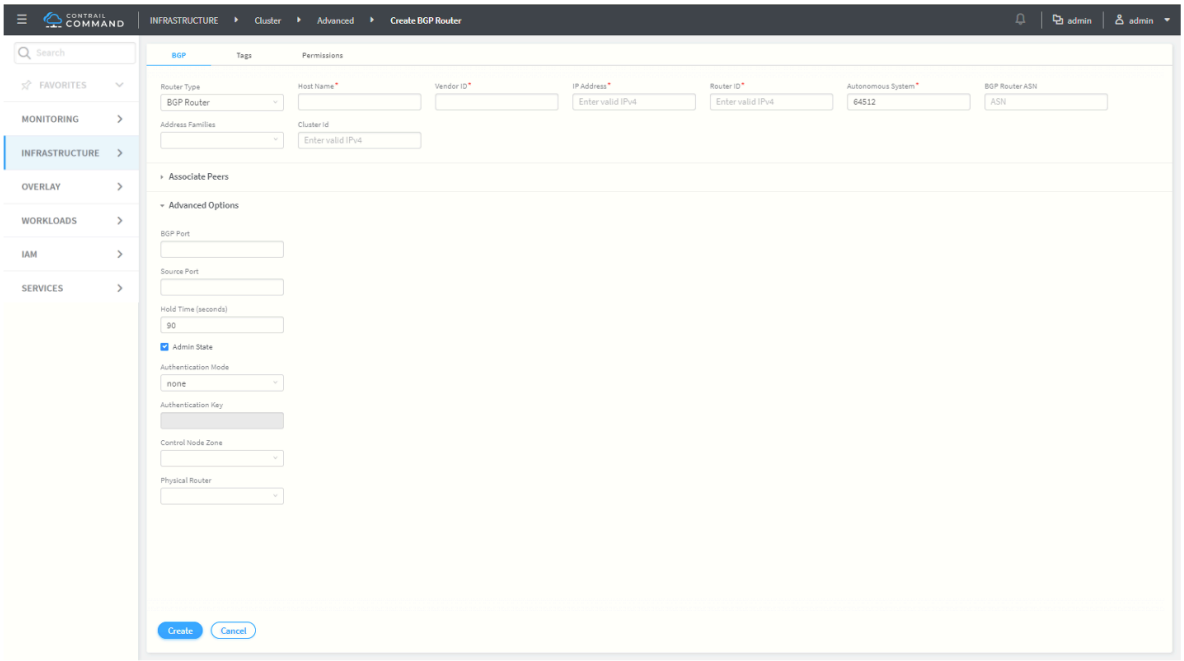
1. From Contrail Command UI select **Infrastructure > Cluster > Advanced** page.

   Click the **BGP Routers** tab. A list of control nodes and BGP routers is displayed. See Figure 84 on page 737.

Figure 84: Infrastructure > Cluster > Advanced > BGP Routers

2. (Optional) The global AS number is 64512 by default. You can change the AS number according to your requirement on the **BGP Router** tab, by clicking the **Edit** icon. In the **Edit BGP Router** tab enter AS number in the range of 1-65,535. You can also enter the AS number in the range of 1-4,294,967,295, when **4 Byte ASN** is enabled in **Global Config**.
3. Click the **Create** button on the **BGP Routers** tab. The **Create BGP Router** window is displayed. See Figure 85 on page 738.

Figure 85: Create BGP Router



4. In the **Create BGP Router** page, populate the fields with values to create your system. See Table 49 on page 738.

Table 49: Create BGP Router

Fields	Description
Router Type	Select the type of router you want create
Hostname	Enter a name for the node being added.
Vendor ID	Required for external peers. Populate with a text identifier, for example, “MX-0”. (BGP peer only)

Table 49: Create BGP Router *(Continued)*

Fields	Description
IP Address	The IP address of the node.
Router ID	Enter the router ID.
Autonomous System (AS)	Enter autonomous system (AS) number in the range of 1-65,535. If you enable **4 Byte ASN** in **Global Config**, you can enter 4-byte AS number in the range of 1-4,294,967,295.
BGP Router ASN	Enter the Local-AS number, specific to the associated peers.
Address Families	Select the Internet Address Family from the list, for example, **inet-vpn**, **inet6-vpn**, and so on.
Cluster ID	Enter the cluster ID, for example, 0.0.0.100.
Associate Peers	
Peer	Select the configured peers from the list.
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.
MD5 Auth Key	Enter the MD5 authentication key value.
State	Select the **state** box when you are associating BGP peers.

Table 49: Create BGP Router *(Continued)*

Fields	Description
Passive	Select the **passive** box to disable the BGP router from advertising any routes. The BGP router can only receive updates from other peers in this state.
Advanced Options	
BGP Port	Enter BGP Port number. The default is 179; change if needed.
Source Port	Enter source port number for client side connection.
Hold Time (seconds)	BGP session hold time. The default is 90 seconds; change if needed.
Admin State	Select the **Admin state** box to enable the state as UP and deselect the box to disable the state to DOWN.
Authentication Mode	Select MD5 from list if required.
Authentication key	Enter the Authentication Key value.
Control Node Zone	Select the required control node zone from the list.
Physical Router	Select the the physical router from the list.

5. Click **Create** to complete add each node.
6. You can check for peers and details about the control nodes by selecting **Infrastructure > Cluster > Control Nodes**. Click the desired node to check the details on **Summary** and **Detailed Stats** page.

RELATED DOCUMENTATION

Creating a Virtual Network with Juniper Networks Contrail

Creating a Virtual Network with OpenStack Contrail | **521**

Configuring MD5 Authentication for BGP Sessions

Contrail supports MD5 authentication for BGP peering based on RFC 2385.

This option allows BGP to protect itself against the introduction of spoofed TCP segments into the connection stream. Both of the BGP peers must be configured with the same MD5 key. Once configured, each BGP peer adds a 16-byte MD5 digest to the TCP header of every segment that it sends. This digest is produced by applying the MD5 algorithm on various parts of the TCP segment. Upon receiving a signed segment, the receiver validates it by calculating its own digest from the same data (using its own key) and compares the two digests. For valid segments, the comparison is successful since both sides know the key.

The following are ways to enable BGP MD5 authentication and set the keys on the Contrail node.

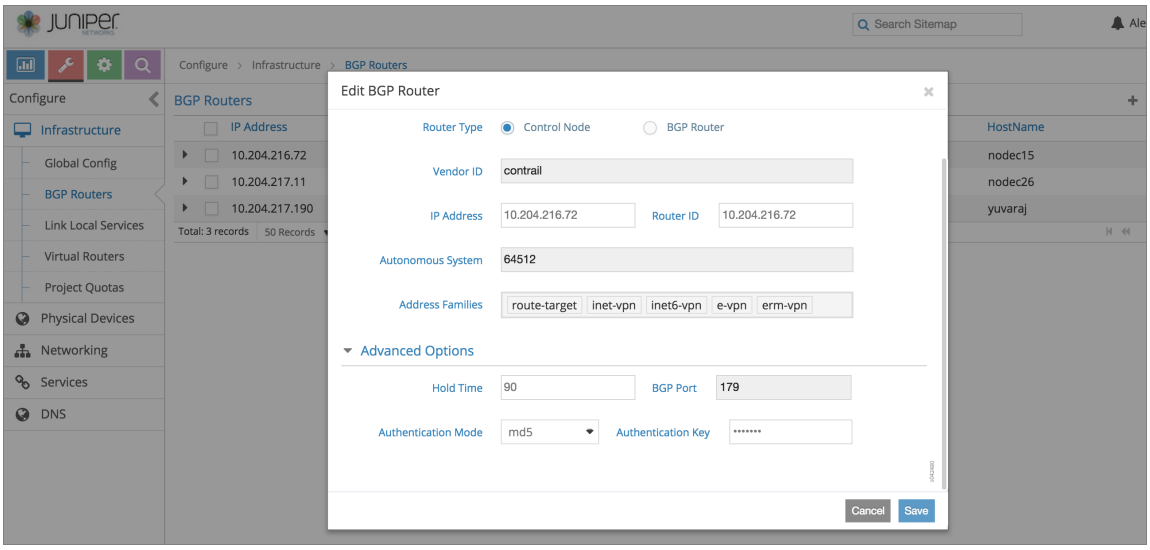
1. If the `md5` key is not included in the provisioning, and the node is already provisioned, you can run the following script with an argument for md5:

```
contrail-controller/src/config/utils/provision_control.py

host@<your_node>:/opt/contrail/utils# python provision_control.py --host_name <host_name> --
host_ip <host_ip> --router_asn <asn> --api_server_ip <api_ip> --api_server_port <api_port> --
oper add --md5 “juniper” --admin_user admin --admin_password <password>  --admin_tenant_name
admin
```

2. You can also use the web user interface to configure MD5.
   - Connect to the node’s IP address at port 8080 (<node_ip>:8080) and select **Configure->Infrastructure->BGP Routers**. As shown in Figure 86 on page 742, a list of BGP peers is displayed.

Figure 86: Edit BGP Router Window



- For a BGP peer, click on the gear icon on the right hand side of the peer entry. Then click **Edit**. This displays the Edit BGP Router dialog box.
- Scroll down the window and select **Advanced Options**.
- Configure the MD5 authentication by selecting **Authentication Mode>MD5** and entering the **Authentication Key** value.

RELATED DOCUMENTATION

Creating a Virtual Network with Juniper Networks Contrail

Creating a Virtual Network with OpenStack Contrail | **521**

Configuring Transport Layer Security-Based XMPP in Contrail

IN THIS SECTION

Overview: TLS-Based XMPP

Transport Layer Security (TLS)-based XMPP can be used to secure all Extensible Messaging and Presence Protocol (XMPP)-based communication that occurs in the Contrail environment.

Secure XMPP is based on *RFC 6120, Extensible Messaging and Presence Protocol (XMPP): Core*.

TLS XMPP in Contrail

In the Contrail environment, the Transport Layer Security (TLS) protocol is used for certificate exchange, mutual authentication, and negotiating ciphers to secure the stream from potential tampering and eavesdropping.

The RFC 6120 highlights a basic stream message exchange format for TLS negotiation between an XMPP server and an XMPP client.

NOTE: Simple Authentication and Security Layer (SASL) authentication is not supported in the Contrail environment.

Configuring XMPP Client and Server in Contrail

In the Contrail environment, XMPP based communications are used in client and server exchanges, between the compute node (as the XMPP client), and:

- the control node (as the XMPP server)
- the DNS server (as the XMPP server)

Configuring Control Node for XMPP Server

To enable secure XMPP, the following parameters are configured at the XMPP server.

On the control node, enable the parameters in the configuration file: `/etc/contrail/contrail-control.conf`.

Parameter	Description	Default
`xmpp_server_cert`	Path to the node's public certificate	`/etc/contrail/ssl/certs/server.pem`

(Continued)

Parameter	Description	Default
`xmpp_server_key`	Path to server's or node's private key	`/etc/contrail/ssl/private/server-privkey.pem`
`xmpp_ca_cert`	Path to CA certificate	`/etc/contrail/ssl/certs/ca-cert.pem`
`xmpp_auth_enable=true`	Enables SSL based XMPP	Default is set to false, XMPP is disabled. **NOTE**: The keyword `true` is case sensitive.

Configuring DNS Server for XMPP Server

To enable secure XMPP, the following parameters are configured at the XMPP DNS server.

On the DNS server control node, enable the parameters in the configuration file: `/etc/contrail/contrail-control.conf`

Parameter	Description	Default
`xmpp_server_cert`	Path to the node's public certificate	`/etc/contrail/ssl/certs/server.pem`
`xmpp_server_key`	Path to server's/node's private key	`/etc/contrail/ssl/certs/server-privkey.pem`
`xmpp_ca_cert`	Path to CA certificate	`/etc/contrail/ssl/certs/ca-cert.pem`
`xmpp_dns_auth_enable=true`	Enables SSL based XMPP	Default is set to false, XMPP is disabled. **NOTE**: The keyword `true` is case sensitive.

Configuring Control Node for XMPP Client

To enable secure XMPP, the following parameters are configured at the XMPP client.

On the compute node, enable the parameters in the configuration file:
`/etc/contrail/contrail-vrouter-agent.conf`

Parameter	Description	Default
`xmpp_server_cert`	Path to the node's public certificate	`/etc/contrail/ssl/certs/server.pem`
`xmpp_server_key`	Path to server's/node's private key	`/etc/contrail/ssl/private/server-privkey.pem`
`xmpp_ca_cert`	Path to CA certificate	`/etc/contrail/ssl/certs/ca-cert.pem`
`xmpp_auth_enable=true` `xmpp_dns_auth_enable=true`	Enables SSL based XMPP	Default is set to false, XMPP is disabled. **NOTE**: The keyword `true` is case sensitive.

Configuring Graceful Restart and Long-lived Graceful Restart

IN THIS SECTION

Graceful restart and long-lived graceful restart BGP helper modes are supported for the Contrail control node and XMPP helper mode.

Application of Graceful Restart and Long-lived Graceful Restart

Whenever a BGP peer session is detected as down, all routes learned from the peer are deleted and immediately withdrawn from advertised peers. This causes instantaneous disruption to traffic flowing end-to-end, even when routes kept in the vrouter kernel in the data plane remain intact.

Graceful restart and long-lived graceful restart features can be used to alleviate traffic disruption caused by downs.

When configured, graceful restart features enable existing network traffic to be unaffected if Contrail controller processes go down. The Contrail implementation ensures that if a Contrail control module restarts, it can use graceful restart functionality provided by its BGP peers. Or when the BGP peers restart, Contrail provides a graceful restart helper mode to minimize the impact to the network. The graceful restart features can be used to ensure that traffic is not affected by temporary outage of processes.

Graceful restart is not enabled by default.

With graceful restart features enabled, learned routes are not deleted when sessions go down, and the routes are not withdrawn from the advertised peers. Instead, the routes are kept and marked as 'stale'. Consequently, if sessions come back up and routes are relearned, the overall impact to the network is minimized.

After a certain duration, if a downed session does not come back up, all remaining stale routes are deleted and withdrawn from advertised peers.

BGP Graceful Restart Helper Mode

The BGP helper mode can be used to minimize routing churn whenever a BGP session flaps. This is especially helpful if the SDN gateway router goes down gracefully, as in an rpd crash or restart on an MX Series Junos device. In that case, the contrail-control can act as a graceful restart helper to the gateway, by retaining the routes learned from the gateway and advertising them to the rest of the network as applicable. In order for this to work, the restarting router (the SDN gateway in this case) must support and be configured with graceful restart for all of the address families used.

The graceful restart helper mode is also supported for BGP-as-a-Service (BGPaaS) clients. When configured, contrail-control can provide a graceful restart or long-lived graceful restart helper mode to a restarting BGPaaS client.

Feature Highlights

The following are highlights of the graceful restart and long-lived graceful restart features.

- Configuring a non-zero restart time enables the ability to advertise graceful restart and long-lived graceful restart capabilities in BGP.

- Configuring helper mode enables the ability for graceful restart and long-lived graceful restart helper modes to retain routes even after sessions go down.
- With graceful restart configured, whenever a session down event is detected and a closing process is triggered, all routes, across all address families, are marked stale. The stale routes are eligible for best-path election for the configured graceful restart time duration.
- When long-lived graceful restart is in effect, stale routes can be retained for a much longer time than that allowed by graceful restart alone. With long-lived graceful restart, route preference is retained and best paths are recomputed. The community marked LLGR_STALE is tagged for stale paths and re-advertised. However, if no long-lived graceful restart community is associated with any received stale route, those routes are not kept, instead, they are deleted.
- After a certain time, if a session comes back up, any remaining stale routes are deleted. If the session does not come back up, all retained stale routes are permanently deleted and withdrawn from the advertised peer.

XMPP Helper Mode

Contrail Networking updated support for long-lived graceful restart (LLGR) with XMPP helper mode in Contrail Networking Release 2011.L2. Starting in Release 2011.L2, the Contrail vRouter datapath agent supports route retention with its controller peer when LLGR with XMPP helper mode is enabled. This route retention allows the datapath agent to retain the last Route Path from the Contrail controller when an XMPP-based connection is lost. The Route Paths are held by the agent until a new XMPP-based connection is established to one of the Contrail controllers. Once the XMPP connection is up and is stable for a predefined duration, the Route Paths from the old XMPP connection are flushed. This support for route retention allows a controller to go down gracefully but with some forwarding interruption when connectivity to a controller is restored.

The following notable behaviors are present when LLGR is used with XMPP helper mode:

- When a local vRouter is isolated from a Contrail controller, the Intra-VN EVPN routes on the remote vRouter are removed.
- During a Contrail vRouter datapath agent restart, forwarding states are not always preserved.

Contrail Networking has limited support for graceful restart and long-lived graceful restart (LLGR) with XMPP helper mode in all Contrail Release 4, 5, and 19 software as well as all Contrail Release 20 software through Contrail Networking Release 2011.L1. Graceful restart and LLGR with XMPP should not be used in most environments and should only be used by expert users in specialized circumstances when running these Contrail Networking releases for reasons described later in this section.

Graceful restart and LLGR can be enabled with XMPP helper mode using Contrail Command, the Contrail Web UI, or by using the provision_control script. The helper modes can also be enabled via

schema, and can be disabled selectively in a contrail-control node for BGP or XMPP sessions by configuring `gr_helper_disable` in the **/etc/contrail/contrail-control.conf** configuration file.

You should be aware of the following dependencies when enabling graceful restart and LLGR with XMPP helper mode:

- You can enable graceful restart and LLGR with XMPP helper mode without enabling the BGP helper. You still have to enable graceful restart, XMPP, and all appropriate timers when graceful restart and LLGR are enabled with XMPP helper mode without the BGP helper.
- LLGR and XMPP sub second timers for fast convergences should not be used simultaneously.
- If a control node fails when LLGR with XMPP helper mode is enabled, vrouters will hold routes for the length of the GR and LLGR timeout values and continue to pass traffic. Routes are removed from the vRouter when the timeout interval elapses and traffic is no longer forwarded at that point.

  If the control node returns to the up state before the timeout interval elapses, a small amount of traffic will be lost during the reconnection.

Graceful restart and LLGR with XMPP should only be used by expert users in specialized circumstances when running Contrail Networking Release 4, 5, and 19 software as well as all Contrail Release 20 software through Contrail Networking Release 2011.L1 due to the following issues:

- Graceful restart is not yet fully supported for the contrail-vrouter-agent.

  Because graceful restart is not yet supported for the contrail-vrouter-agent, the parameter should *not* be set for `graceful_restart_xmpp_helper_enable`. If the vrouter agent restarts, the data plane is reset and the routes and flows are reprogrammed anew. This reprogramming typically results in traffic loss for several seconds for new and existing flows and can result in even longer traffic loss periods.

- The vRouter agent restart caused by enabling graceful restart can cause stale route to be added to the routing table used by the contrail-vrouter-agent.

  This issue occurs after a contrail-vrouter-agent reset. After the reset, previous XMPP control nodes continue to send stale routes to other control nodes. The stale routes sent by the previous XMPP control nodes can eventually get passed to the contrail-vrouter-agent and installed into its routing table as NH1/drop routes, leading to traffic drops. The stale routes are removed from the routing table only after graceful restart is enabled globally or when the timer—which is user configurable but can be set to long intervals—expires.

Configuration Parameters

Graceful restart parameters are configured in the `global-system-config` of the schema. They can be configured by means of a provisioning script or by using the Contrail Web UI.

Configure a non-zero restart time to advertise for graceful restart and long-lived graceful restart capabilities from peers.

Configure helper mode for graceful restart and long-lived graceful restart to retain routes even after sessions go down.

Configuration parameters include:

- `enable` or `disable` for all graceful restart parameters:
  - `restart-time`
  - `long-lived-restart-time`
  - `end-of-rib-timeout`
- `bgp-helper-enable` to enable graceful restart helper mode for BGP peers in contrail-control
- `xmpp-helper-enable` to enable graceful restart helper mode for XMPP peers (agents) in contrail-control

The following shows configuration by a provision script.

```
/opt/contrail/utils/provision_control.py
            --api_server_ip 10.xx.xx.20
            --api_server_port 8082
            --router_asn 64512
            --admin_user admin
            --admin_password <password>
            --admin_tenant_name admin
            --set_graceful_restart_parameters
            --graceful_restart_time 60
            --long_lived_graceful_restart_time 300
            --end_of_rib_timeout 30
            --graceful_restart_enable
            --graceful_restart_bgp_helper_enable
```

The following are sample parameters:

```
-set_graceful_restart_parameters
            --graceful_restart_time 300
            --long_lived_graceful_restart_time 60000
            --end_of_rib_timeout 30
            --graceful_restart_enable
            --graceful_restart_bgp_helper_enable
```

When BGP peering with Juniper Networks devices, Junos must also be explicitly configured for graceful restart/long-lived graceful restart, as shown in the following example:

```
set routing-options graceful-restart
set protocols bgp group <a1234> type internal
set protocols bgp group <a1234> local-address 10.xx.xxx.181
set protocols bgp group <a1234> keep all
set protocols bgp group <a1234> family inet-vpn unicast graceful-restart long-lived restarter
stale-time 20
set protocols bgp group <a1234> family route-target graceful-restart long-lived restarter stale-
time 20
set protocols bgp group <a1234> graceful-restart restart-time 600
set protocols bgp group <a1234> neighbor 10.xx.xx.20 peer-as 64512
```

The graceful restart helper modes can be enabled in the schema. The helper modes can be disabled selectively in the `contrail-control.conf` for BGP sessions by configuring `gr_helper_disable` in the `/etc/contrail/contrail-control.conf` file.

The following are examples:

```
/usr/bin/openstack-config /etc/contrail/contrail-control.conf DEFAULT gr_helper_bgp_disable 1

/usr/bin/openstack-config /etc/contrail/contrail-control.conf DEFAULT gr_helper_xmpp_disable 1

service contrail-control restart
```

For more details about graceful restart configuration, see https://github.com/Juniper/contrail-controller/wiki/Graceful-Restart .

Cautions for Graceful Restart

Be aware of the following caveats when configuring and using graceful restart.

- Using the graceful restart/long-lived graceful restart feature with a peer is effective either to all negotiated address families or to none. If a peer signals support for graceful restart/long-lived graceful restart for only a subset of the negotiated address families, the graceful restart helper mode does not come into effect for any family in the set of negotiated address families.

- Because graceful restart is not yet supported for contrail-vrouter-agent, the parameter should *not* be set for `graceful_restart_xmpp_helper_enable`. If the vrouter agent restarts, the data plane is reset and the routes and flows are reprogrammed anew. This reprogramming typically results in traffic loss for several seconds for new and existing flows and can result in even longer traffic loss periods.

  Additionally, previous XMPP control nodes might continue to send stale routes to other control nodes and these stale routers can be passed to the contrail-vrouter-agent. The contrail-vrouter-agent

can install these stale routes into it's routing table as NH1/ drop routes, causing traffic loss. The stale routes are removed only after graceful restart is enabled globally or when the timer—which is user configurable but can be set to multiple days—expires.

- Graceful restart/long-lived graceful restart is not supported for multicast routes.
- Graceful restart/long-lived graceful restart helper mode may not work correctly for EVPN routes, if the restarting node does not preserve forwarding state for EVPN routes.

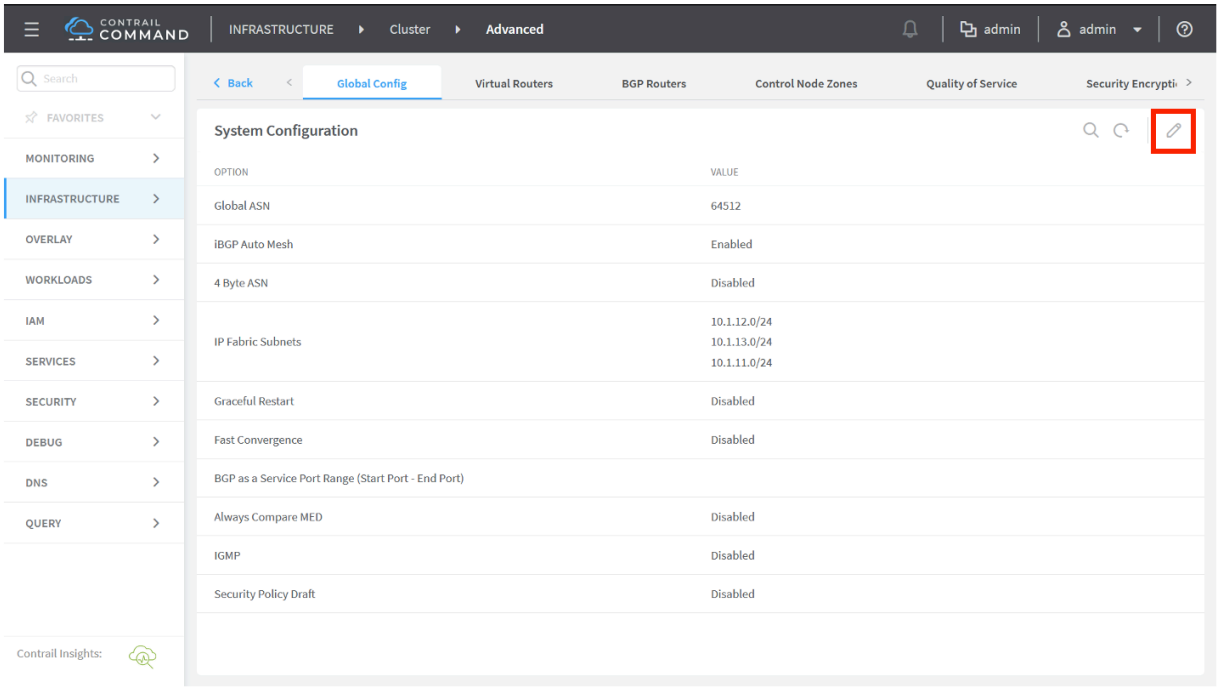
Configuring Graceful Restart

We recommend configuring Graceful Restart using Contrail Command. You can, however, also configure Graceful Restart using the Contrail User Interface in environments not using Contrail Command.

Configuring Graceful Restart using Contrail Command

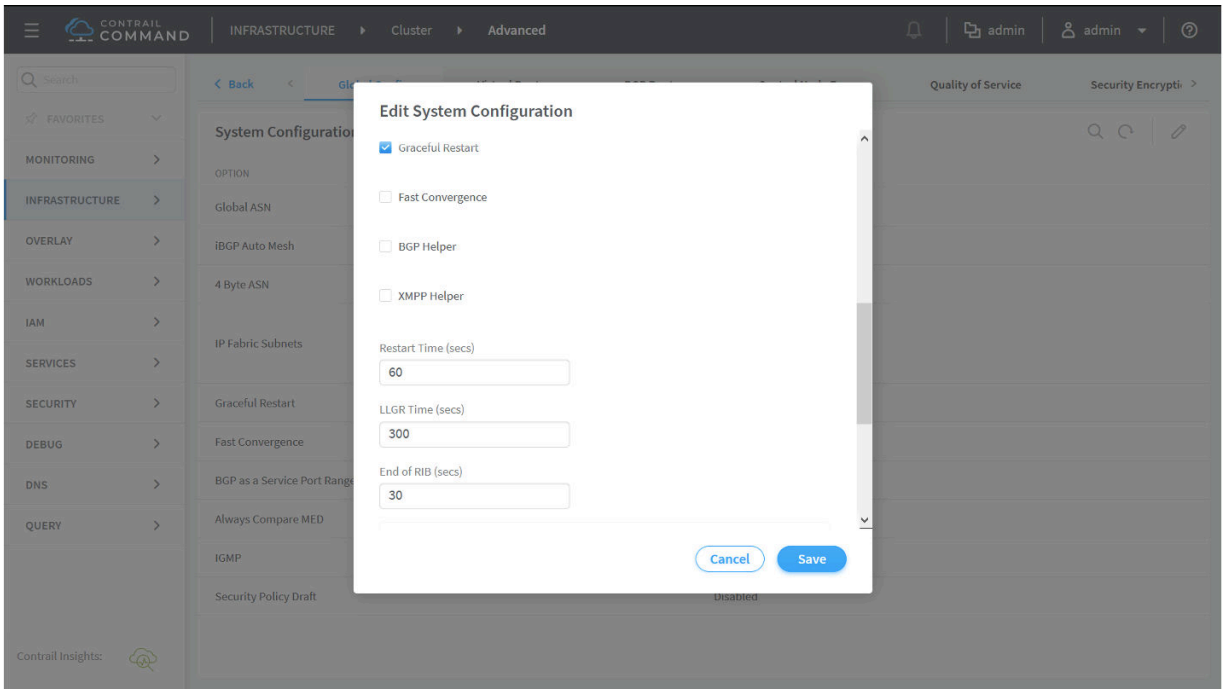
To configure graceful restart in Contrail Command, navigate to **Infrastructure > Cluster > Advanced Options** and select the Edit icon near the top right corner of the screen.

Figure 87: Global Config System Configuration Screen



The **Edit System Configuration** window opens. Click the box for **Graceful Restart** to enable graceful restart, and enter a non-zero number to define the **Restart Time** in seconds. You can also specify the times for the long-lived graceful restart (LLGR) and the end of RIB timers from this window.

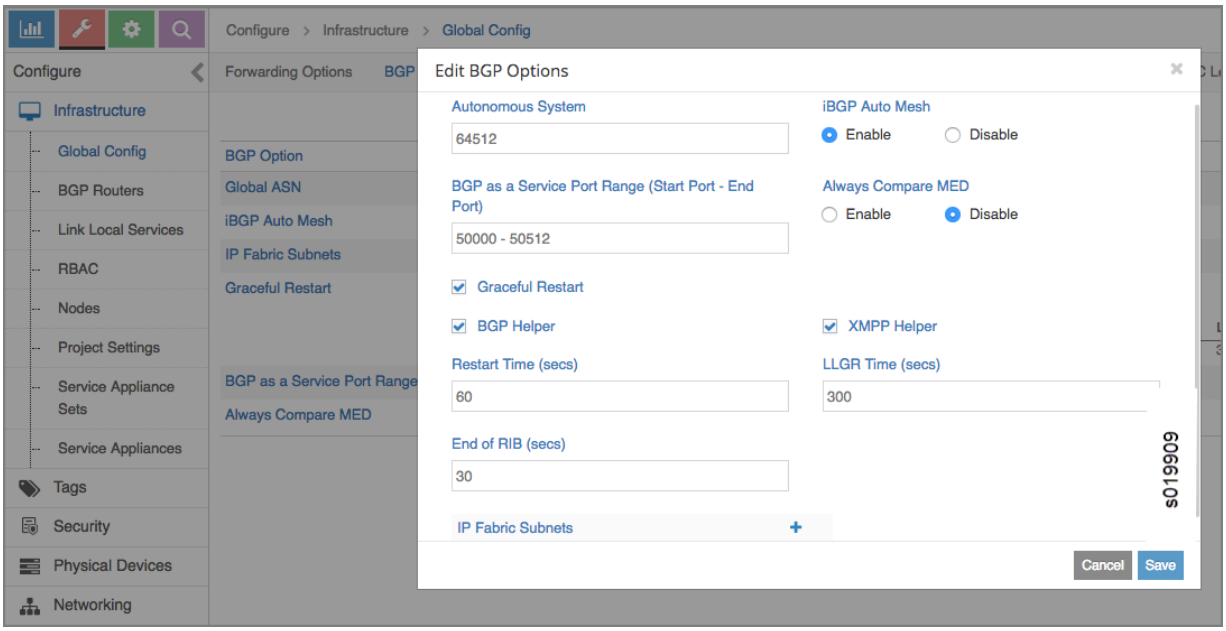
Figure 88: Edit System Configuration Window



Configuring Graceful Restart with the Contrail User Interface

To configure graceful restart in the Contrail UI, go to **Configure > Infrastructure > Global Config**, then select the **BGP Options** tab. The **Edit BGP Options** window opens. Click the box for **Graceful Restart** to enable graceful restart, and enter a non-zero value for the **Restart Time**. Click the helper boxes as needed for BGP Helper and XMPP Helper. You can also enter values for the long-lived graceful restart time in seconds, and for the end of RIB in seconds. See Figure 89 on page 753.

Figure 89: Configuring Graceful Restart



Understanding the Graceful Restart Timers

Table 50 on page 753 provides a summary of the graceful restart timers and their associated behaviors.

Table 50: Graceful Restart Timers

Timer	Description
Restart Time	**BGP helper mode**—Routes advertised by the BGP peer are kept for the duration of the restart time. **XMPP helper mode**—Routes advertised by XMPP peer are kept for the duration of the restart time.
LLGR Time	**BGP helper mode**—Routes advertised by BGP peers are kept for the duration of the LLGR timer when BGP helper mode is enabled. **XMPP helper mode**—Routes advertised by XMPP peers are kept for the duration of the LLGP timer if XMPP helper mode is enabled. When Graceful Restart (GR) and Long-lived Graceful Restart (LLGR) are both configured, the duration of the LLGR timer is the sum of both timers.

Table 50: Graceful Restart Timers *(Continued)*

Timer	Description
End of RIB timer	The End of RIB (EOR) timer specifies the amount of time a control node waits to remove stale routes from a vRouter agent's RIB. When a vRouter agent to Control Node connection is restored, the vRouter agent downloads it's configuration from the control node. An End of Config message is sent from the control node to vRouter agent when this configuration procedure is complete. The EOR timer starts when this End of Config message is received by the vRouter agent. When the EOR timer expires, an EOR message is sent from the vRouter agent to the control node. The control node receives this EOR message then removes the stale routes which were previously advertised by the vRouter agent from it's RIB.

Graceful Restart or Long-Lived Graceful Restart Support for a EVPN Type 2 Route

Today, when a BGP/XXMP peer session restarts or goes down, even if you have configured graceful restart or long-lived graceful restart timers in the Contrail Web UI, the learnt EVPN Type 2 routes are not marked as *stale* and are deleted (control-node explicitly deletes EVPN routes) from the route database. This results in traffic loss for the EVPN family of routes.

Starting in Contrail Networking Release 21.4, the graceful restart or long-lived graceful restart features support the EVPN Type 2 routes and helps in the following ways:

- When a session fails, the learnt EVPN Type 2 routes are not deleted or removed.
- Retains the learnt EVPN Type 2 routes and marks them as *stale* until the configured graceful restart or long-lived graceful restart timers expire.
- Results in resuming the sessions and relearning the routes, reducing overall network impact.

If a downed session remains down after the graceful restart or long-lived graceful restart timer has expired, the stale routes are deleted and removed from the advertised peers.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
R21.4	Starting in Contrail Networking Release 21.4, the graceful restart or long-lived graceful restart features support the EVPN Type 2 routes

Scaling Up Contrail Networking Configuration API Server Instances

Certain deployment scenarios may need running of multiple API configuration server instances for improved performance. One of the methods to achieve this is by increasing the number of API configuration server instances on a node *after deploying* Contrail Networking. This is done by modifying the **docker-compose.yaml** file to allow multiple configuration API containers on the same host.

The steps described in this topic are valid for Contrail Networking Releases 5.0 through 2008. For release 2011 and later, refer the topic "Scaling Up Contrail Networking Configuration API" on page 758.

CAUTION: Any change to the Contrail networking Configuration API must be made only with the help of Juniper Networks Professional Services. We strongly recommend that you contact Juniper Networks Professional Services before you make any change to the Configuration API.

1. Edit the **/etc/contrail/config/docker-compose.yaml** file on each Contrail Configuration node. At the end of the API section, at line 56, append the following chunk of code:

```
api-2:
    image: "hub.juniper.net/contrail/contrail-controller-config-api:1912.32-rhel"
    env_file: /etc/contrail/common_config.env
    environment:
      - CONFIG_API_PORT=18082
      - CONFIG_API_INTROSPECT_PORT=17084
    container_name: config_api_2
    command: ["/usr/bin/contrail-api", "--conf_file", "/etc/contrail/contrail-api.conf", "--
conf_file", "/etc/contrail/contrail-keystone-auth.conf", "--worker_id", "2"]
    network_mode: "host"
    volumes_from:
      - node-init
    depends_on:
```

```
      - node-init
    restart: always
    stdin_open: True
    tty: True
    logging:
      driver: "json-file"
      options:
        max-size: "50m"
        max-file: "10"
```

Make sure that the API port, Introspect port, Worker ID and container_name are unique within the node. The default API section should not be changed to allow other Contrail services (for example, Schema-transformer, SVC-Monitor or contrail-status command) to run without configuration changes. The default API runs with port 8082, introspect port 8084, worker_id 0 and container name config_api_1.

2. Run the following commands on each configuration node to apply the changes:

```
cd /etc/contrail/config
docker-compose down
docker-compose up -d
```

3. Run the following commands on each configuration node to verify the configuration:

```
pgrep -al contrail-api
ss -nlpt | grep contrail-api
```

Sample Output

```
# pgrep -al contrail-api
16922 /usr/bin/python /usr/bin/contrail-api --conf_file /etc/contrail/contrail-api.conf --
conf_file /etc/contrail/contrail-keystone-auth.conf --worker_id 2
17139 /usr/bin/python /usr/bin/contrail-api --conf_file /etc/contrail/contrail-api.conf --
conf_file /etc/contrail/contrail-keystone-auth.conf --worker_id 0
# ss -nlpt | grep contrail-api
LISTEN     0      128          *:17084                    *:*                   users:
(("contrail-api",pid=16922,fd=7))
LISTEN     0      128    10.0.0.16:18082                    *:*                   users:
(("contrail-api",pid=16922,fd=28))
LISTEN     0      128    10.0.0.16:8082                     *:*                   users:
(("contrail-api",pid=17139,fd=28))
```

```
LISTEN     0      128          *:8084                     *:*                   users:
(("contrail-api",pid=17139,fd=7))
```

4. Add new server instances to load-balancer.

   Sample HAProxy Configuration

```
listen contrail_config_internal
  bind 192.168.3.90:8082 transparent
  mode http
  balance leastconn
  option httpchk GET /
  option httplog
  option forwardfor
  timeout client 360s
  timeout server 360s
  server 0contrail-ctl0.local 192.168.3.114:8082 check fall 5 inter 2000 rise 2
  server 1contrail-ctl0.local 192.168.3.114:18081 check fall 5 inter 2000 rise 2
  server 2contrail-ctl0.local 192.168.3.114:18082 check fall 5 inter 2000 rise 2
  server 3contrail-ctl0.local 192.168.3.114:18083 check fall 5 inter 2000 rise 2
  server 0contrail-ctl1.local 192.168.3.139:8082 check fall 5 inter 2000 rise 2
  server 1contrail-ctl1.local 192.168.3.139:18081 check fall 5 inter 2000 rise 2
  server 2contrail-ctl1.local 192.168.3.139:18082 check fall 5 inter 2000 rise 2
  server 3contrail-ctl1.local 192.168.3.139:18083 check fall 5 inter 2000 rise 2
  server 0contrail-ctl2.local 192.168.3.100:8082 check fall 5 inter 2000 rise 2
  server 1contrail-ctl2.local 192.168.3.100:18081 check fall 5 inter 2000 rise 2
  server 2contrail-ctl2.local 192.168.3.100:18082 check fall 5 inter 2000 rise 2
  server 3contrail-ctl2.local 192.168.3.100:18083 check fall 5 inter 2000 rise 2
```

5. Run the following commands to set load-balancer timeouts and balancing method

You cannot configure timeout for the Neutron plugin. Neutron relies on load-balancer to terminate connections. Therefore, it is important that you increase the default timeout value of 30 seconds. This enables Neutron to respond without error even if the Contrail API responds longer than 30 seconds.

Sample timeout options for HAProxy

```
timeout client 360s
timeout server 360s
```

NOTE: It is recommended that you use load balancing methods that distributes the load evenly across the configuration API server instances. The preferred lod balancing methods are `leastconn` and `round-robin`.

CAUTION: Increasing the number of configuration API server instances might result in higher load on RabbitMQ and Schema-transformer.

Scaling Up Contrail Networking Configuration API

IN THIS SECTION

Starting from Contrail Networking Release 2011, config-api can be scaled vertically, to run up to 10 workers in a single container by using uWSGI. However, the recommended maximum number of workers in each config_api container is 5.

The steps described in this topic are valid for Contrail Networking Releases 2011 and later releases. To see these procedures for earlier releases, see"Scaling Up Contrail Networking Configuration API Server Instances" on page 755.

From Contrail Ansible Deployer

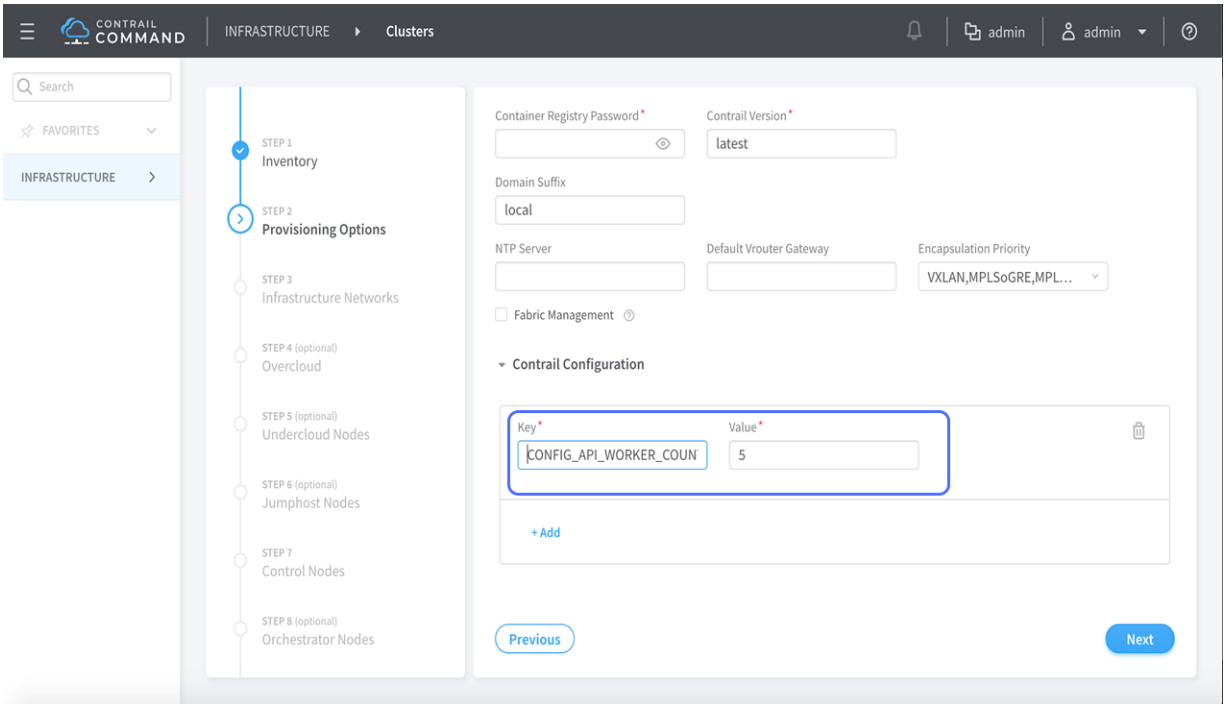
If you are deploying Contrail Networking using Ansible Deployer , you should specify `CONFIG_API_WORKER_COUNT` parameter in the contrail_configuration section of the instances.yml file as shown below.

```
contrail_configuration:
  CONFIG_API_WORKER_COUNT: 5
```

From Contrail Command UI

If you are deploying Contrail Networking from Contrail Command UI, you should specify a new key / value pair (CONFIG_API_WORKER_COUNT / *value*) in the Contrail Configuration section as shown in Figure 90 on page 759.

Figure 90: Scaling API Configuration from Contrail Command



From RHOSP Deployer

If you are deploying Contrail Networking using RHOSP deployer, you should specify the desired value for the parameter `CONFIG_API_WORKER_COUNT` in the `ContrailSettings` section of the contrail-services.yaml file as shown below.

```
ContrailSettings:
   CONFIG_API_WORKER_COUNT: 3
```

From JUJU Deployer

If you are deploying Contrail Networking using JUJU deployer, you must specify the desired value for `config-api-worker-count` in the **config.yaml** file.

```
config-api-worker-count:
  default: 1
  type: int
  description: |
    Number of workers spawned inside config-api container.
```