Engineering Simplicity

Contrail® Networking

Contrail Networking Installation and Upgrade Guide

Published 2023-09-26

RELEASE 1912

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

Contrail® Networking Contrail Networking Installation and Upgrade Guide
1912

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

YEAR 2000 NOTICE

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

END USER LICENSE AGREEMENT

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

Table of Contents

About This Guide

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

- Contrail Command.
- Contrail with AppFormix.
- Contrail with Kubernetes.
- Contrail with Mesos.
- Contrail with VMware vCenter.
- Contrail with Red Hat.
- Contrail and AppFormix with Kolla/Ocata OpenStack.
- Contrail with Juju Charms.

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

Table 1: Contrail Networking Guides

Guide Name	Description
Contrail Networking Installation and Upgrade Guide	Provides step-by-step instructions to install and bring up Contrail and its various components.
Contrail Networking Deployment Guide	Provides information about the next steps to be taken after a successful installation of Contrail.
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.

Table 1: Contrail Networking Guides *(Continued)*

Guide Name	Description
Contrail Networking Service Provider Focused Features Guide	Provides information about the features that are used by service providers.
Contrail Networking Analytics and Troubleshooting Guide	Provides information about AppFormix and Contrail analytics.

RELATED DOCUMENTATION

- README Access to Contrail Networking Registry 19XX
- Contrail Networking Release Notes 1912
- Contrail Networking Configuration API Reference, Release 1912
- Tungsten Fabric Architecture Guide
- Juniper Networks TechWiki: Contrail Networking

1

PART

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.

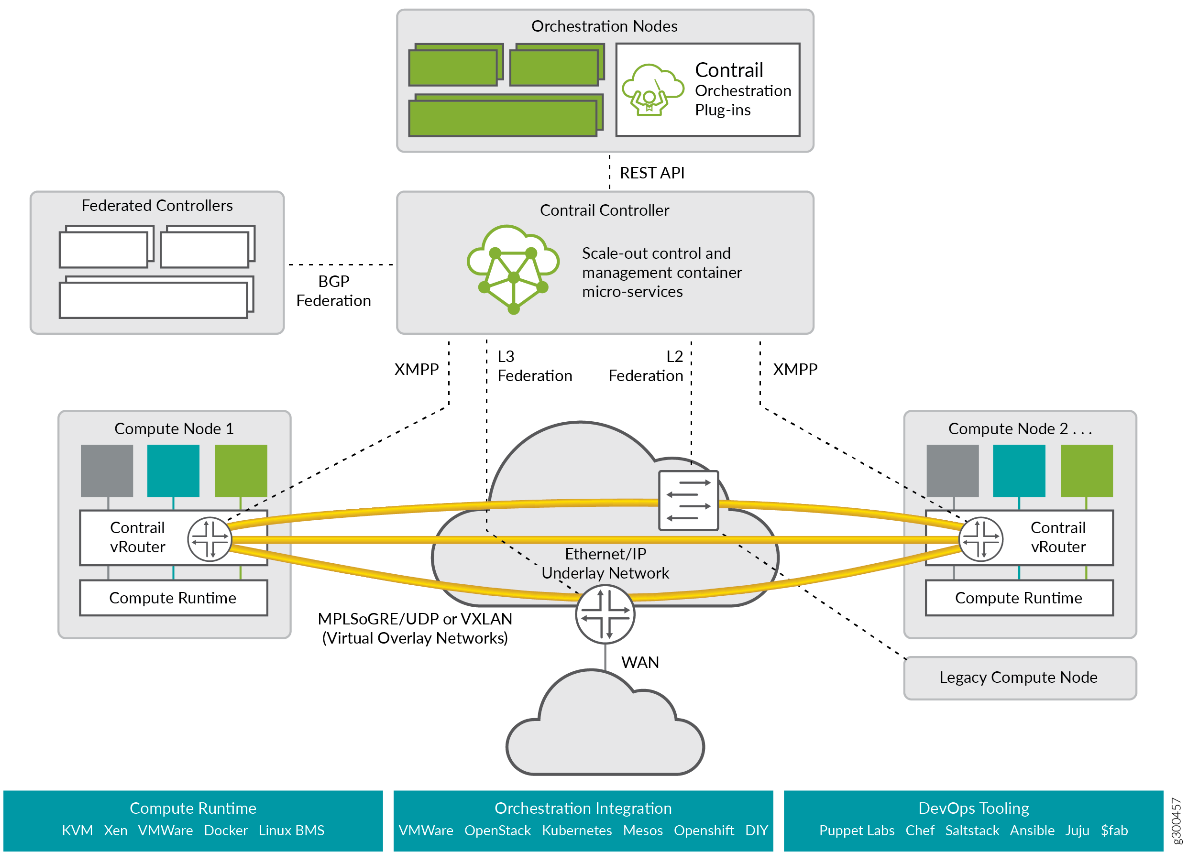
For service providers, Contrail Networking automates network resource provisioning and orchestration to dynamically create highly scalable virtual networks and to chain a rich set of Juniper Networks or third-party virtualized network functions (VNFs) and physical network functions (PNFs) to form differentiated 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 AppFormix 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 systems*, commonly called the controller, have several functions. Few of the major functions are:

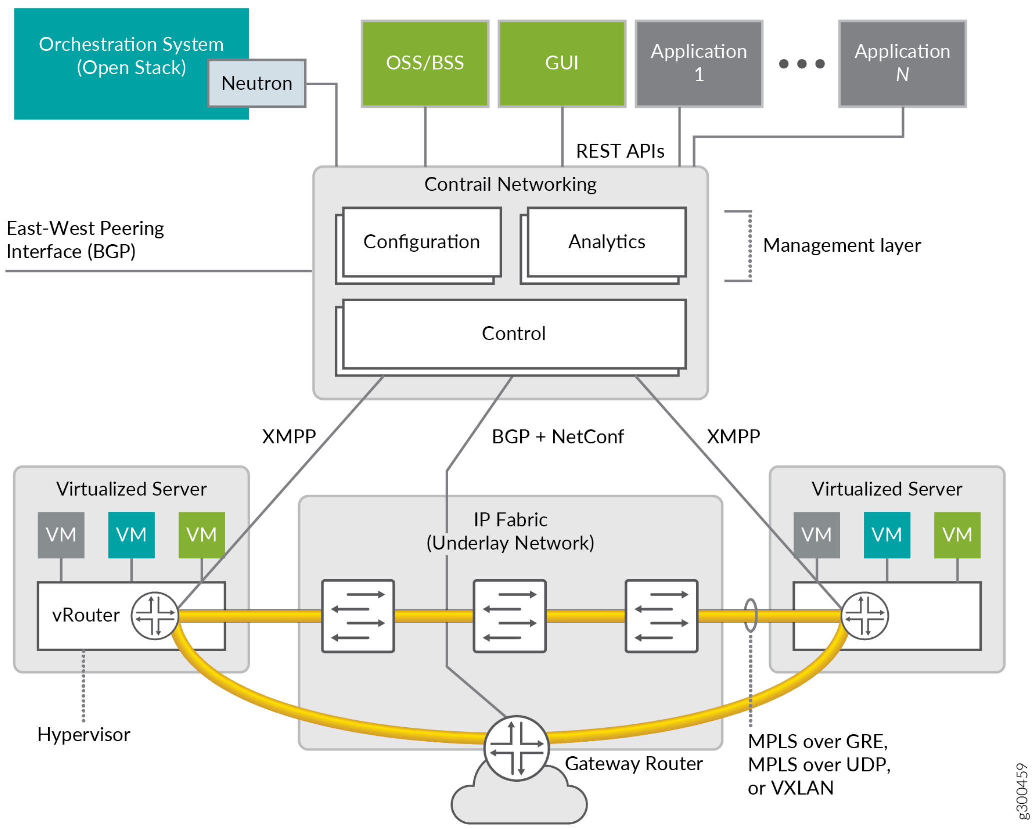
  - *Configuration Nodes*—This function accepts requests from the API to provision workflows like adding new virtual networks, new endpoints, and much more. It converts these abstract high-level requests, with optional detail, into low-level directions that map to the internal data model.

  - *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 new architecture of Contrail containers:

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

Understanding Contrail Microservices Architecture

IN THIS SECTION

What is Contrail Microservices Architecture?

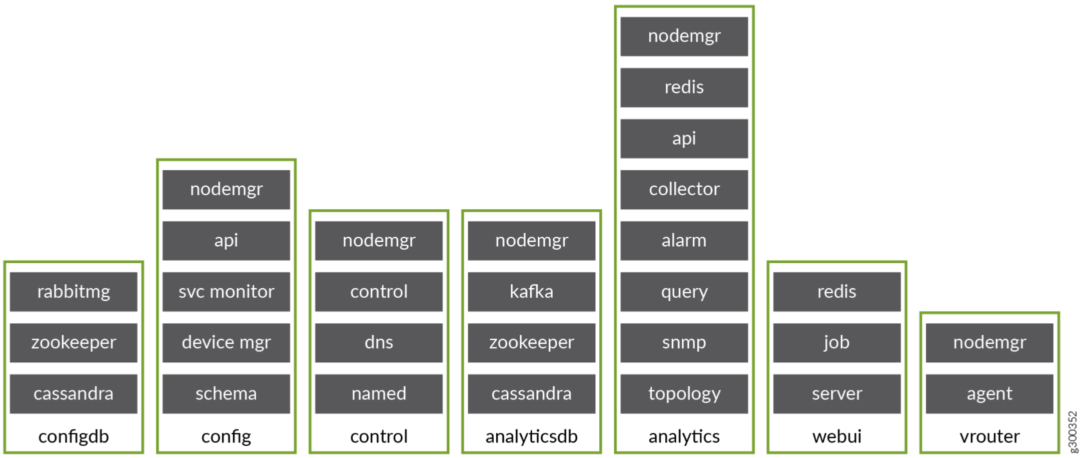
Employing microservices provides a number of benefits which includes:

- 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
- "Installing and Managing Contrail Microservices Architecture Using Helm Charts" on page 85

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 Contrail Cluster using Contrail Command and instances.yml" on page 42.

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 plays:

playbooks/provision_instances.yml

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

- kvm.
- gce.
- aws.

playbooks/configure_instances.yml

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

playbooks/install_contrail.yml

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

Preparing to Install with Contrail 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`.

- CentOS 7.6—Linux Kernel Version 3.10.0-957.12.1
- Ansible 2.4.2.0.
- Name resolution is operational for long and short host names of the cluster nodes, through either DNS or the host file.
- Docker engine (tested version is 18.06.0-ce).
- The docker-compose installed (tested version is 1.17.0).
- The docker-compose Python library (tested version is 1.9.0).
- If using Kubernetes (k8s), the tested version is 1.12
- For high availability (HA), the time must be 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 a Yaml File for Your Environment

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

The configuration has multiple main sections, including:

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

Provider Configuration

The section `provider_config` configures provider-specific settings.

KVM Provider Example

Use this example if you are in a kernel-based virtual machine (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 knobs. The instance configuration specifies which roles are installed on which instance. Additionally, instance-wide and role-specific Contrail and Kolla configurations can be specified, overwriting the parameters from the global Contrail and Kolla configuration settings.

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 Contrail Cluster using Contrail Command and instances.yml" on page 42.

RELATED DOCUMENTATION

Installing Contrail Cluster using Contrail Command and instances.yml | 42

CHAPTER 2

Supported Platforms and Server Requirements

IN THIS CHAPTER

Server Requirements and Supported Platforms

The minimum requirement is three servers, either physical or virtual machines. All non-compute roles can be configured in each controller node. For scalability and availability reasons, it is highly recommended to use physical servers.

Each server must have a minimum of:

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

All installation images are available at 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 19XX.

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

CHAPTER 3

Contrail Command

IN THIS CHAPTER

Installing Contrail Command

IN THIS SECTION

Use this example to install the Contrail Command Docker container images.

Contrail Command is the GUI for Contrail Cloud and Contrail Enterprise Multicloud solutions. It represents the single management touchpoint for the fabric underlay, the overlay networks and virtual endpoints, and the AppFormix performance and resource monitoring application for cloud services.

Contrail Command also simplifies the configuration of OpenStack clusters and the integration of Contrail within those clusters. By providing a workflow to facilitate integration with orchestrators, initially providing support for OpenStack Kolla, Contrail Command makes integration a straightforward task.

After you integrate Contrail Command with your orchestrator, you can use Contrail Command to perform typical tasks such as creating overlay networks, creating flavors, spinning up workloads, attaching workloads to overlay networks, and setting up firewall permissions to control communication paths. For these typical tasks, it is not necessary for you to use the orchestrator's UI (e.g. OpenStack dashboard).

In general, you can use Contrail Command to perform automated workflows such as the following:

- deploy Contrail and Kolla-based OpenStack clusters.
- monitor and manage underlay and physical devices, overlays and virtual endpoints, end to end policy and control.
- orchestrate workloads running on bare metal servers, virtual machines, and containers.
- discover and manage the data center IP fabric.

Most workflows are intent-based, meaning that you configure the workflows using templates and wizards where applicable.

Requirements

The system requirements for the Contrail Command server are:

- A VM or physical x86 server with:
  - 4 vCPUs
  - 32 GB RAM
  - 100 GB disk with all user storage in the "/" partition (that is, remove the "**/home**" partition if it exists, and increase the "/" partition by the amount of freed storage)
- Internet access to and from the VM or physical server, hereafter referred to as the Contrail Command server
- Runs a version of CentOS that supports your version of Contrail Networking.

  We perform regular testing of Contrail Command on CentOS 7 but Contrail Command should work on other common versions of Linux. 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.
- An IP interface attached to the management network. Contrail Command manages Contrail and OpenStack clusters over this interface.
- SSH access

- Access to the *hub.juniper.net* registry. See README Access to Contrail Registry 19XX for more information.

- *<container_tag>* See README Access to Contrail Registry 19XX for more information..

Overview

Installation

IN THIS SECTION

Contrail Command deploys as a pair of Docker containers. You install Contrail Command by downloading and running the `contrail-command-deployer` image, which runs in its own container and exits when the Contrail Command installation is complete.

Before you begin, set up a CentOS 7 server that meets the specified requirements.
For a list of supported platforms, see https://www.juniper.net/documentation/en_US/release-independent/contrail/topics/reference/contrail-supported-platforms.pdf.

Additionally, remove any installed Python Docker (`docker` and `docker-py`) libraries from the server. The `contrail-command-deployer` automatically installs all necessary libraries. If you are using a freshly installed minimal CentOS 7 server, then these Python Docker libraries do not yet exist on the server and you do not have to take any action.

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

There is no harm if you issue the above command when no Python Docker libraries are installed.

Procedure

Step-by-Step Procedure

Perform the following steps on a CentOS 7 server to configure and install Contrail Command.

1. Install and start the Docker Engine. The following set of commands adds the Docker repository and installs and starts Docker Community Edition version 18.06 as an example.

```
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-18.06.0.ce
systemctl start docker
```

2. Pull the `contrail-command-deployer` Docker image from *hub.juniper.net*.

   See README Access to Contrail Registry 19XX for information on how to get credentials to access the secure *hub.juniper.net* registry and for the name of the container tag to use.

   a. Log in to the *hub.juniper.net* registry.

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

   b. Retrieve the `contrail-command-deployer` Docker image.

   ```
   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 want to install. See README Access to Contrail Registry 19XX to obtain the *<container_tag>* for any Contrail Networking Release 19 software.

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

   The **command_servers.yml** file contains information of the server where you plan to install Contrail Command as well as information of the container registry and other configuration parameters. Contrail Command runs on a single server, typically the same server where you run the `contrail-command-deployer` (that is, the server used in this procedure).

   When you run the `contrail-command-deployer`, it reads and processes the **command_servers.yml** file. Examples of this file are shown in "No Link Title" on page 23 and "No Link Title" on page 24.

   NOTE: If you want to deploy AppFormix, add the following two lines to the `command_servers.yml` file. They must be placed 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. The following shows an example where the two lines are added at the top of the file:

```
---
user_command_volumes:
- /opt/software/appformix:/opt/software/appformix
command_servers:
   server1:
      ip:
```

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

   To perform a fresh installation:

```
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 "3" on page 21, and *<container_tag>* is the container tag for the Contrail Command (UI) container deployment for the release that you want to install.

5. (Optional) Track the progress of step "4" on page 22.

```
docker logs -f contrail_command_deployer
```

6. After the installation is complete, 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. Log in to Contrail Command using **https:// *<Contrail-Command-Server-IP-Address>*:9091**. Use the username and password that you specified in the **command_servers.yml** file in step "3" on page 21. If you use the sample **command_servers.yml** files in "Sample command_servers.yml Files" on page 23, the username is **admin** and the password is **contrail123**.

Sample command_servers.yml Files

Minimal command_servers.yml file

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

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.

```
---
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.

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.

```
---
# 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
```

RELATED DOCUMENTATION

Installing a Contrail Cluster Using Contrail Command | 29
Installing Contrail Cluster using Contrail Command and instances.yml | 42
Importing Contrail Cluster Data using Contrail Command | 47
Installing AppFormix and AppFormix Flows using Contrail Command | 77
Upgrading Contrail Command using Backup Restore Procedure | 327

Installing a Contrail Cluster Using Contrail Command

IN THIS SECTION

Use this example procedure to create a Contrail and OpenStack Kolla cluster using Contrail Command. The resulting cluster consists of Contrail containers deployed alongside OpenStack Kolla containers to provide an OpenStack installation that uses Contrail as the SDN.

Requirements

- VMs or physical x86 servers as follows:
  - Contrail Controller — 8 vCPU, 64 GB memory, 300 GB storage
  - OpenStack Controller — 4 vCPU , 32 GB memory, 100 GB storage
  - Contrail Service Node (CSN) — 4 vCPU, 16 GB memory, 100 GB storage
  - Compute nodes — Dependent on the workloads

  For a list of supported platforms including software requirements, see https://www.juniper.net/documentation/en_US/release-independent/contrail/topics/reference/contrail-supported-platforms.pdf.
- User storage for all servers resides in the "/" partition (that is, remove the "**/home**" partition if it exists, and increase the "/" partition by the amount of freed storage).
- An IP interface on each server attached to the management network. Each server is managed by Contrail Command over this interface. Ensure the name for this management interface is the same on all servers.
- An IP interface on each server attached to the user data network. This is the interface that the overlay network will be set up on. Ensure the name for this user data interface is the same on all servers.
- SSH access

Overview

IN THIS SECTION

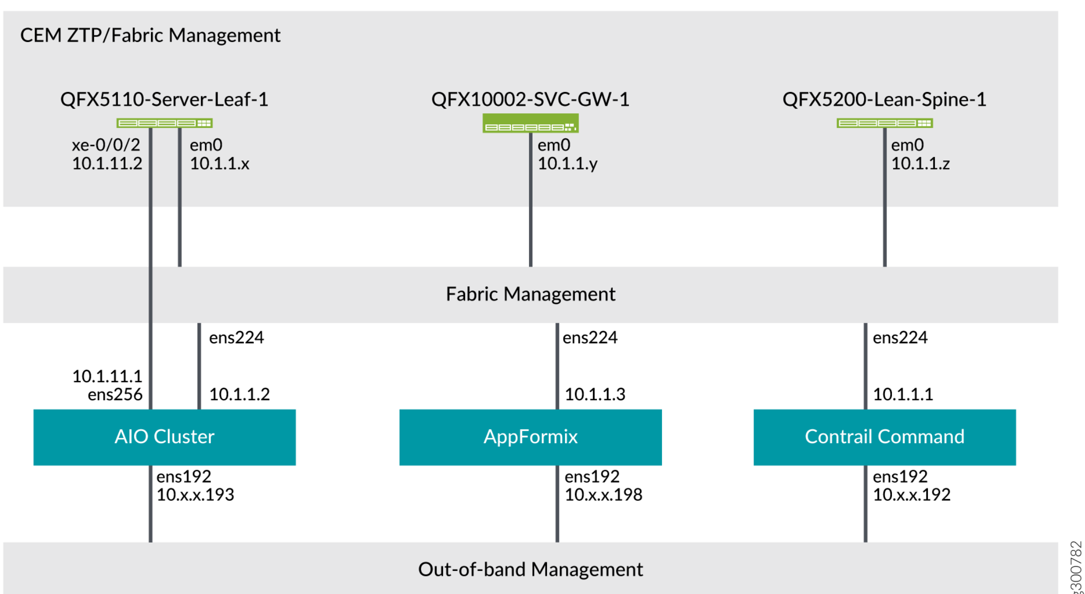
A Contrail cluster consists of hosts that run the Contrail controller, the orchestrator (OpenStack, VMware or Kubernetes), and the compute nodes.

Before you begin, set up servers and/or VMs that meet the specified requirements. Also ensure that the Contrail Command server and all the hosts in the Contrail cluster have **/etc/hosts** entries for each other over the management network.

Topology

Consider a sample cluster topology, with a non-HA environment of one Contrail Controller and one OpenStack Controller, one compute node and one Contrail Service Node (CSN), as displayed in Figure 4 on page 30.

Figure 4: Sample Contrail Cluster Topology



Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

The general workflow is for you to first add the servers and VMs that you want to make available for the cluster you're creating, and then add the cluster.

1. Log in to Contrail Command at **https:// *<Contrail-Command-Server-IP-Address>*:9091**.

   If you used the sample command_servers.yml files when you installed Contrail Command, then the username is **admin** and the password is **contrail123**.

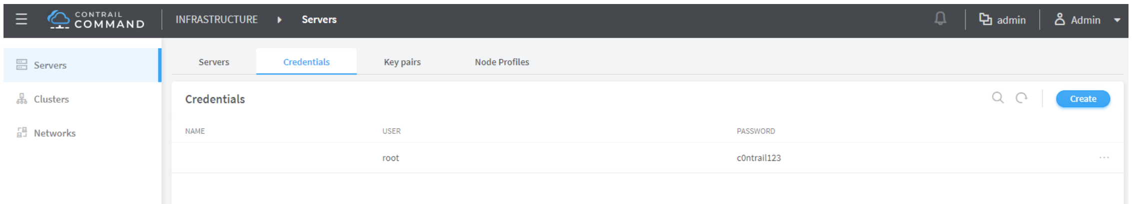
   > **NOTE**: Username and password combinations are provided in this document for illustrative purposes only. We suggest using unique passwords in accordance with your organization's security guidelines in your environment.

   > **NOTE**: If no cluster exists, you are automatically placed into a wizard that guides you to set up a cluster. The left-nav bar shows your progress. If a cluster already exists, you will need to explicitly add a cluster. Regardless of whether you are using the wizard or not, the steps to set up a cluster are very similar. Differences are noted in the steps below.

2. Add the login credentials of the servers and VMs that you're making available. In this step, you are adding the credentials to a floating list. You will explicitly associate each set of credentials to the proper server or VM at a later step. Contrail Command uses these credentials to log in to the servers and VMs when creating the cluster.

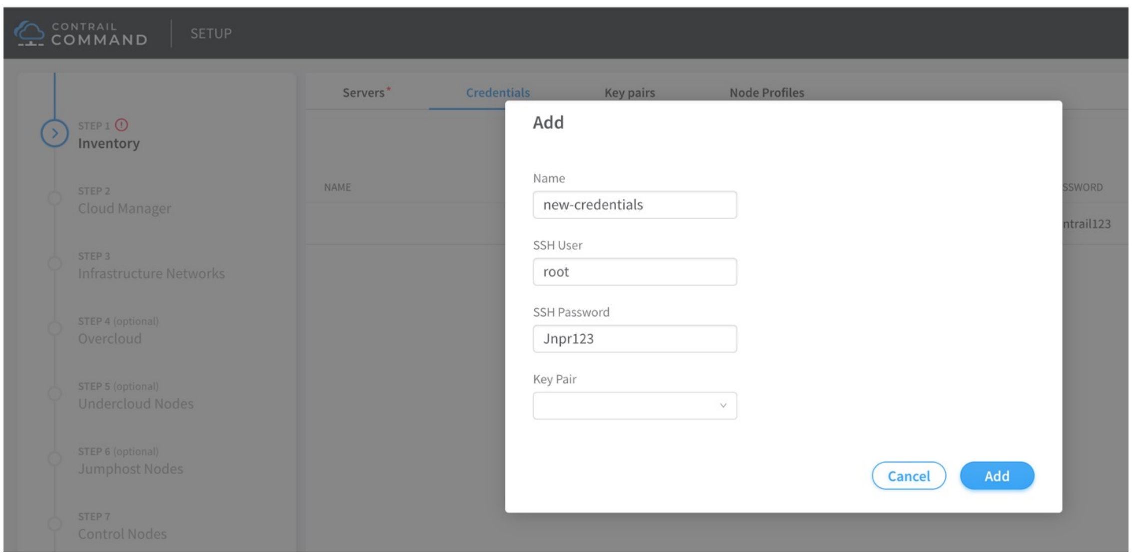
   #### Step-by-Step Procedure

a. To see the list of credentials, navigate to **Servers** > **Credentials.** The set of credentials you specified during Contrail Command installation (in the `server1` section of the command_servers.yml file) is automatically listed.



b. Click **Create** to add a new set of credentials.

Enter **Name**, **SSH User**, and **SSH Password.**



3. Add the servers or VMs, either one at a time or using the bulk import capability.

Navigate to **Servers** > **Create**.

- To add servers one at a time, select **Express** or **Detailed.** In this example, we select **Detailed.**

  Enter the following information:

  - Workload type - Select **Physical/Virtual Node** if you want the workload to run as a VM on the server or **Baremetal** if you want the workload to run directly on the server.
  - Hostname - Enter the hostname part of the FQDN.
  - Management IP - Enter the IP address for the management interface.
  - Management Interface - Enter the interface name for the management interface.

- Credentials - Use the drop-down list to select the correct login credentials for this server. This is the set of credentials you added in step "2" on page 31. Contrail Command uses the credentials you select to log in to the server.
- MAC Address (optional) - Specify the MAC address for the management interface.
- Disk Partition(s) (optional) - Specify the disk partitions you want to use.
- Network Interfaces - Click **Add** to add the interfaces on the server. As a minimum, add the management and the user data interface.

> **NOTE**: If you select the **Express** option, then you will need to edit the server afterwards to add in any missing information.

- To add servers in bulk, select **Bulk Import (csv)**.

Step-by-Step Procedure

a. Click **Download** to download the csv template to use. The downloaded file is a template with sample values.

Here is a sample csv file:

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

physical,5c10s9,10.87.74.69,,enp4s0f0,,,,,,,,,,,2,enp4s0f0,,10.87.74.69,,ens2f0,,10.1.
0.2,

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

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

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

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

b. Fill in the values for your servers, and save and upload the file by clicking **Upload**.

The added servers are now shown in the list of available servers.

4. You can now create the cluster. If you are in the wizard, click **Next**. Otherwise, select **Clusters** > **Add Cluster**.

5. Set the general parameters for the cluster.

Step-by-Step Procedure

a. Select **Contrail Enterprise Multicloud** as the Provisioning Manager.

b. Enter the required information.

- **Cluster Name -** the name that you want to call the cluster
- **Container Registry**, **Container Registry Username**, **Container Registry Password**, **Contrail Version**

  See README Access to Contrail Registry 19XX for the correct values for these fields. The Contrail Version corresponds to the Contrail-Command (UI) container deployment value specified in that document.

- **Provisioner Type -** Ansible
- **Domain Suffix -** the domain name for the cluster
- **NTP Server** - the FQDN or IP address of the NTP server you want to use
- **Default Vrouter Gateway -** the default gateway for the compute nodes

  The default that you specify here is made available in the Default Gateway fields in later steps. If a particular node has a different default gateway, you can always override the default at that later step.

- Encapsulation Priority

  Select `VXLAN`, `MPLSoUDP`, `MPLSoGRE`.

c. Ensure **Insecure** is not selected.

d. Click **Enable ZTP**.

NOTE: From Contrail Networking Release 1911 and later, **Enable ZTP** option is replaced by **Fabric Management**.

e. Click the drop-down arrow for **Contrail Configuration**.

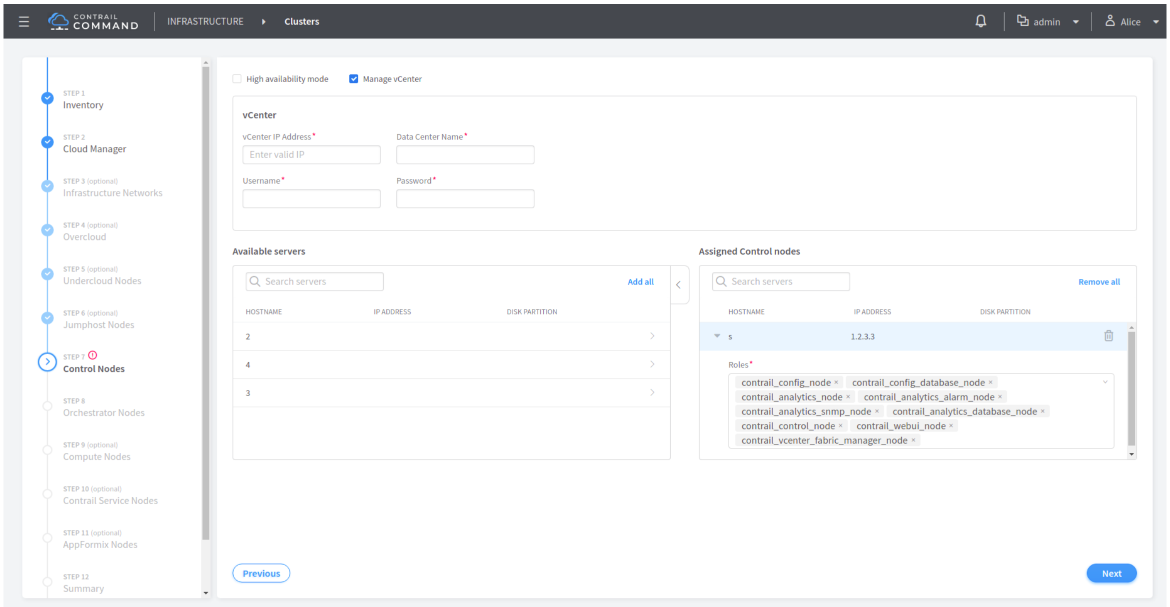
f. Click **Add** and enter the following **Key/Value** pairs.

Key	Value
CONTROL_NODES	List of comma-separated user data interface IP addresses for the controller(s)
PHYSICAL_INTERFACE	The user data interface name
TSN_NODES	List of comma-separated user data interface IP addresses for the Contrail Service Node(s)
CONTRAIL_CONTAINER_TAG	The container tag for the desired Contrail and OpenStack release combination as specified in README Access to Contrail Registry 19XX

g. Click **Next**.

6. Assign the control nodes for the cluster.

Figure 5: Select Control Nodes



Select **High availability mode** if you have HA setup for the controller node. Select all the control nodes from **Available servers** list.

7. Select the orchestrator and assign the orchestrator nodes.

Step-by-Step Procedure

a. Select **Openstack** from the **Orchestrator type** drop-down list.

b. Use the arrows to move one or more servers from the **Available servers** list to the **Assigned Openstack nodes** list.

c. Check the **Show Advanced** option to customize your deployment.

d. Set up the virtual IP addresses if you are deploying an HA cluster.

- **Control & Data Network Virtual IP address -** this is an internal VIP (e.g. 10.87.74.100)
- **Management Network Virtual IP address -** this is an external VIP (e.g. 10.1.0.100)
- **keepalived_virtual_router_id -** (optional) it can be set to any value between 0-255. The default value is 51.

e. Add the following parameters under **Customize configuration** for a VM-based setup:

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

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

f. Click **Add** under **Kolla Globals** and enter the following **Key**/**Value** pairs. These are the standard OpenStack Kolla globals.

Key	Value	Notes
enable_haproxy	no	
enable_ironic	no	Set to **no** if you are not using Life Cycle Management in Contrail Command or PXE boot on Bare Metal Servers (BMS).
enable_swift	yes	Set to **yes** if you are using object store. This parameter is disabled by default.
openstack_release	queens (for example)	This must be one of the supported OpenStack releases.
swift_disk_partition_size	20GB	The default value is 5 GB. If you have two or more images, you must have at least 20 GB allocated for hitless image upload procedure.

g. Click **Add** under **Kolla Passwords** to explicitly add Kolla passwords if desired.

These passwords are placed into the **etc/kolla/passwords.yml** file. By default, all kolla passwords are set to `contrail123`.

NOTE: We suggest changing the default password to a unique password in accordance with your organization's security guidelines.

h. Click **Next**.

8. Assign the compute nodes.

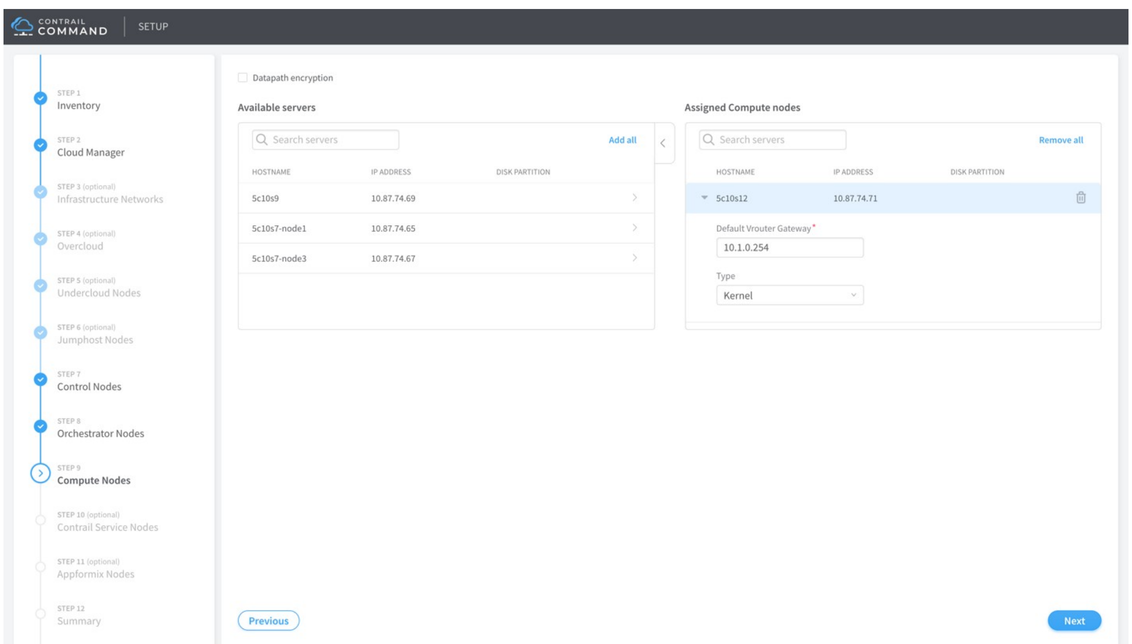
Step-by-Step Procedure

a. Use the arrows to move one or more servers from the **Available servers** list to the **Assigned Compute nodes** list.

b. Enter the **Default Vrouter Gateway** for each node.

c. Select **Kernel** in the **Type** drop-down list. This is the only type supported in the current release.

Figure 6: Select Compute Nodes



d. Click **Next**.

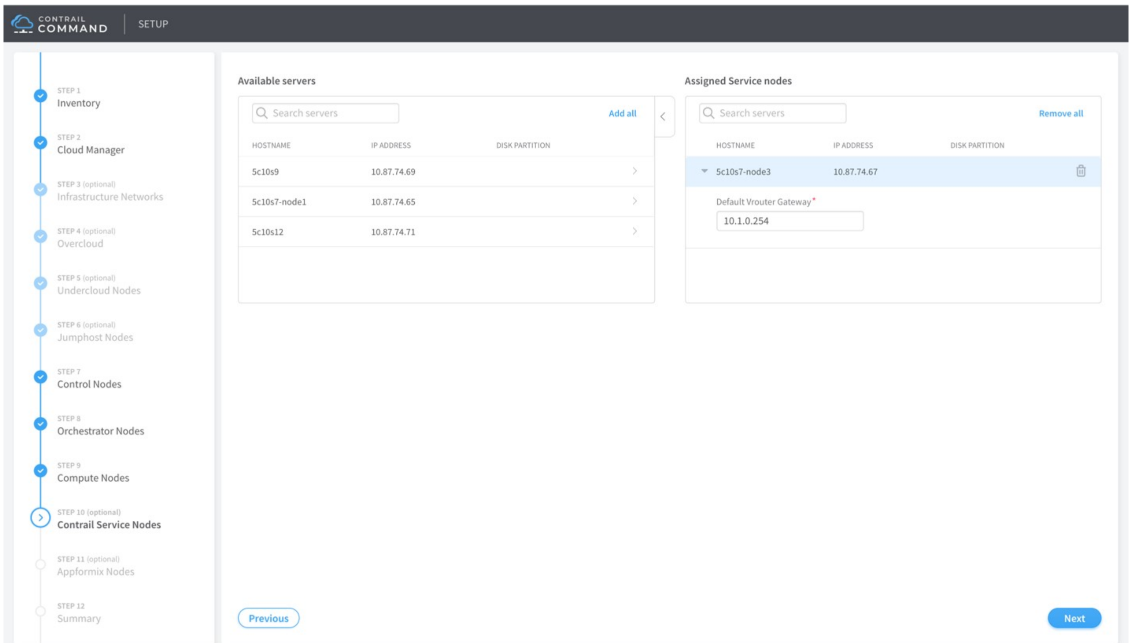
9. Assign the Contrail Service nodes.

Step-by-Step Procedure

a. Use the arrows to move one or more servers from the **Available servers** list to the **Assigned Service nodes** list.

b. Enter the **Default Vrouter Gateway** for each node.

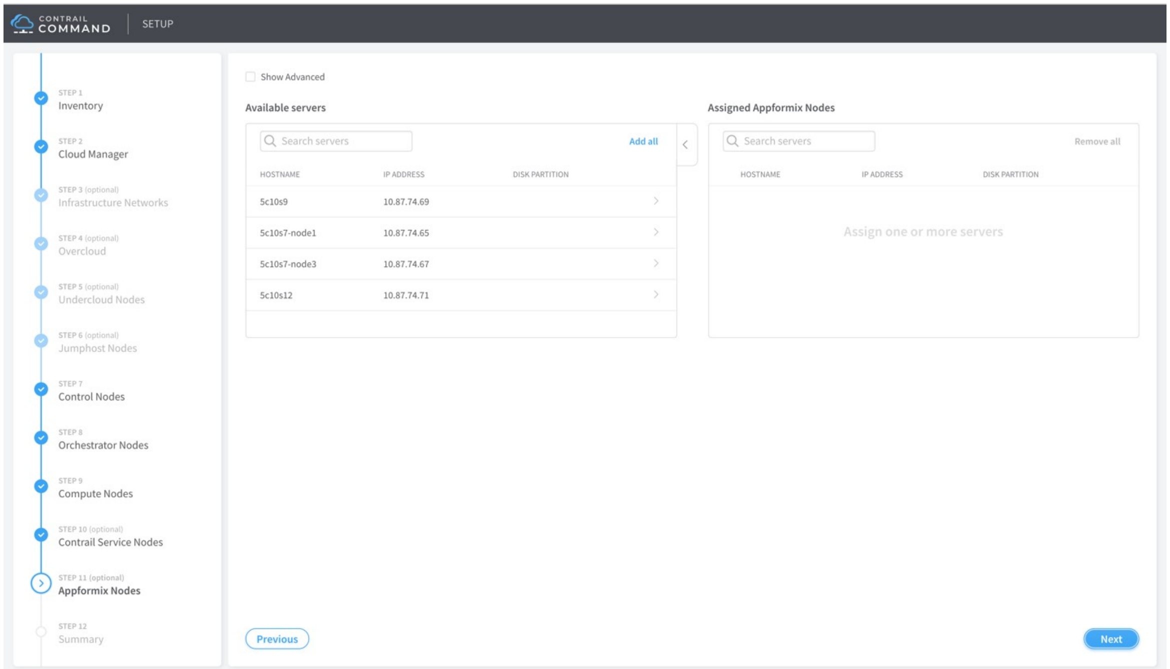
Figure 7: Select Contrail Service Nodes



c. Click **Next**.

10. (Optional) Assign the AppFormix nodes.

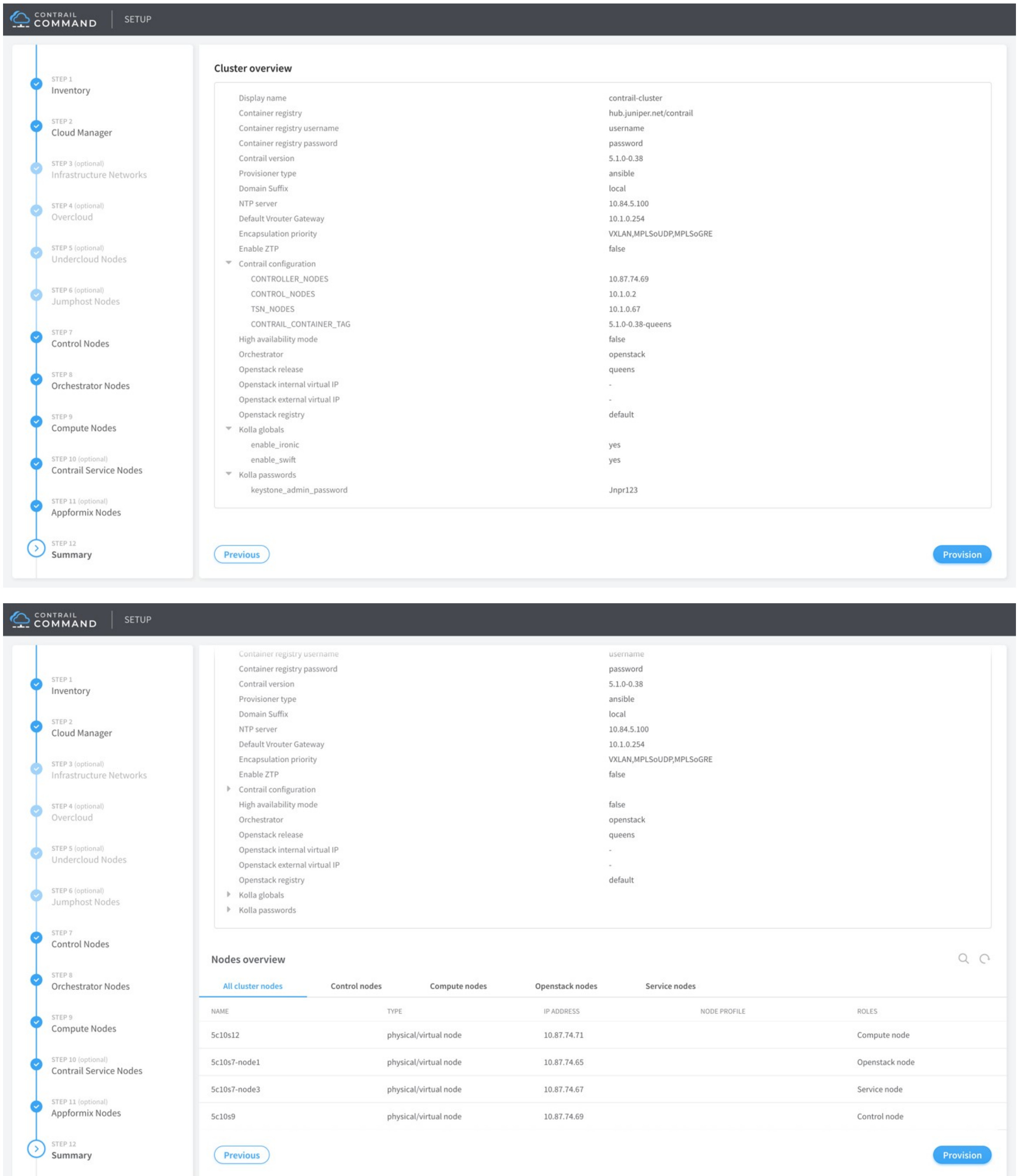
    For details, refer to "Installing AppFormix and AppFormix Flows using Contrail Command" on page 77.



Click **Next**.

11. Verify your cluster configuration in the **Cluster overview** panel and your nodes configuration in the **Nodes overview** panel.

Figure 8: Verify Summary



If the configuration is correct, click **Provision**. If not, click **Previous** to go back to fix any misconfigurations.

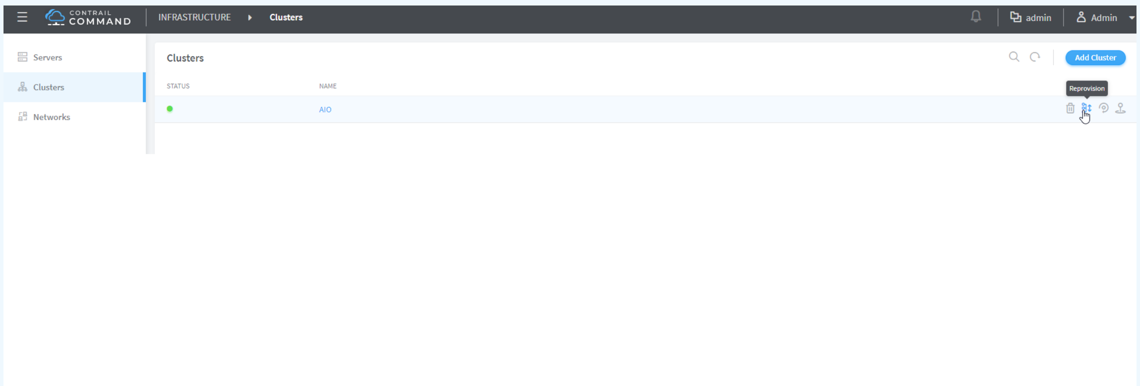
12. You can monitor the progress by running the following commands from Contrail Command server.

```
docker logs -f contrail_command
```

or

```
docker exec contrail_command tail -f /var/log/contrail/deploy.log
```

NOTE: The **Reprovision** button does not support editing Kolla or Contrail configuration parameters. After each failed attempt to provision a fabric, you must destroy and rebuild all the VMs and repeat the process.



RELATED DOCUMENTATION

Installing Contrail Command | 18

Installing Contrail Cluster using Contrail Command and instances.yml | 42

Importing Contrail Cluster Data using Contrail Command | 47

Installing 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.

NOTE: We strongly recommend installing Contrail Command and deploying your Contrail cluster from Contrail Command in most Contrail Networking deployments. You should only use the procedure in this document if you have a strong reason to not use the recommended procedure. See "Installing Contrail Command" on page 18 and "Installing a Contrail Cluster Using Contrail Command" on page 29.

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, see https://www.juniper.net/documentation/en_US/release-independent/contrail/topics/reference/contrail-supported-platforms.pdf.

> **NOTE**: Contrail Release 5.1 does not support AppFormix 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 45 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. The 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" on page 18 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>
```

```
  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)
    swift_disk_partition_size = 10GB
    keepalived_virtual_router_id: <Value between 0-255>
  kolla_passwords:
    keystone_admin_password: <Keystone Admin Password>
```

RELATED DOCUMENTATION

Installing Contrail Command | 18

Installing a Contrail Cluster Using Contrail Command | 29

Importing Contrail Cluster Data using Contrail Command | 47

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, see https://www.juniper.net/documentation/en_US/release-independent/contrail/topics/reference/contrail-supported-platforms.pdf.

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:1911.31, use the following command:

   ```
   docker pull hub.juniper.net/contrail/contrail-command-deployer:1911.31
   ```

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_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>
```

- 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 | 18

Installing a Contrail Cluster Using Contrail Command | 29

Installing Contrail Cluster using Contrail Command and instances.yml | 42

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.

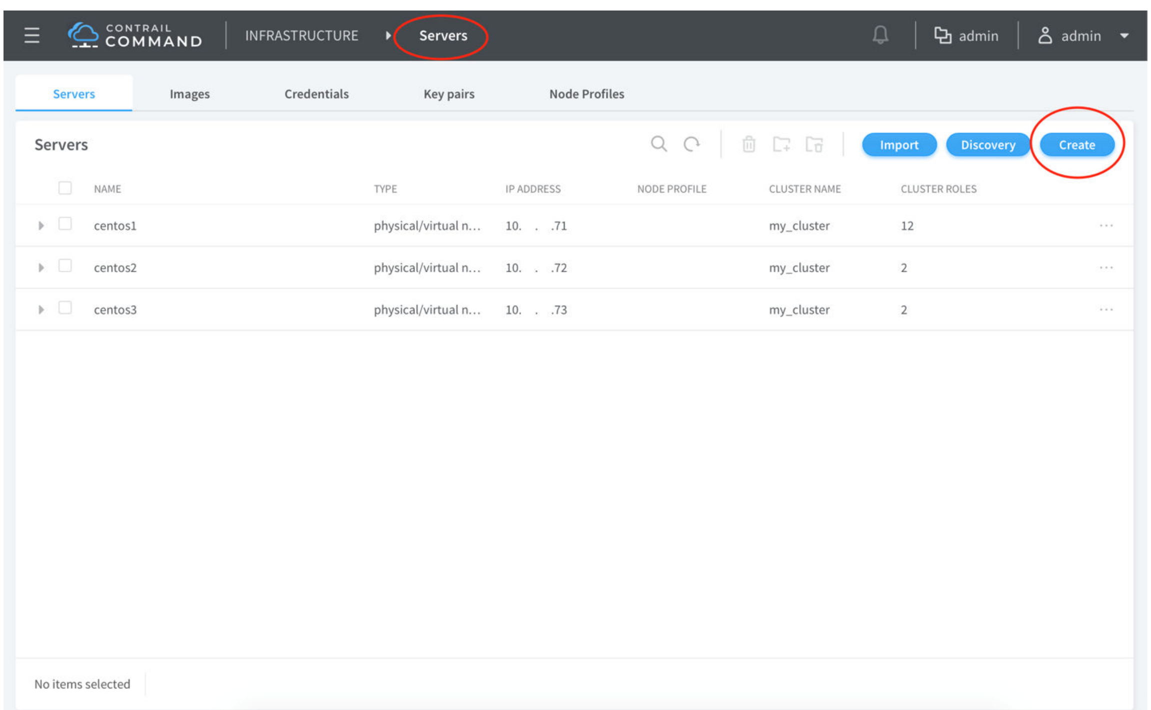
To add a new compute node to an existing Contrail OpenStack cluster:

1. Login to Contrail Command.

   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" on page 18.

2. Click **Servers**.

   a. Click **Create**.

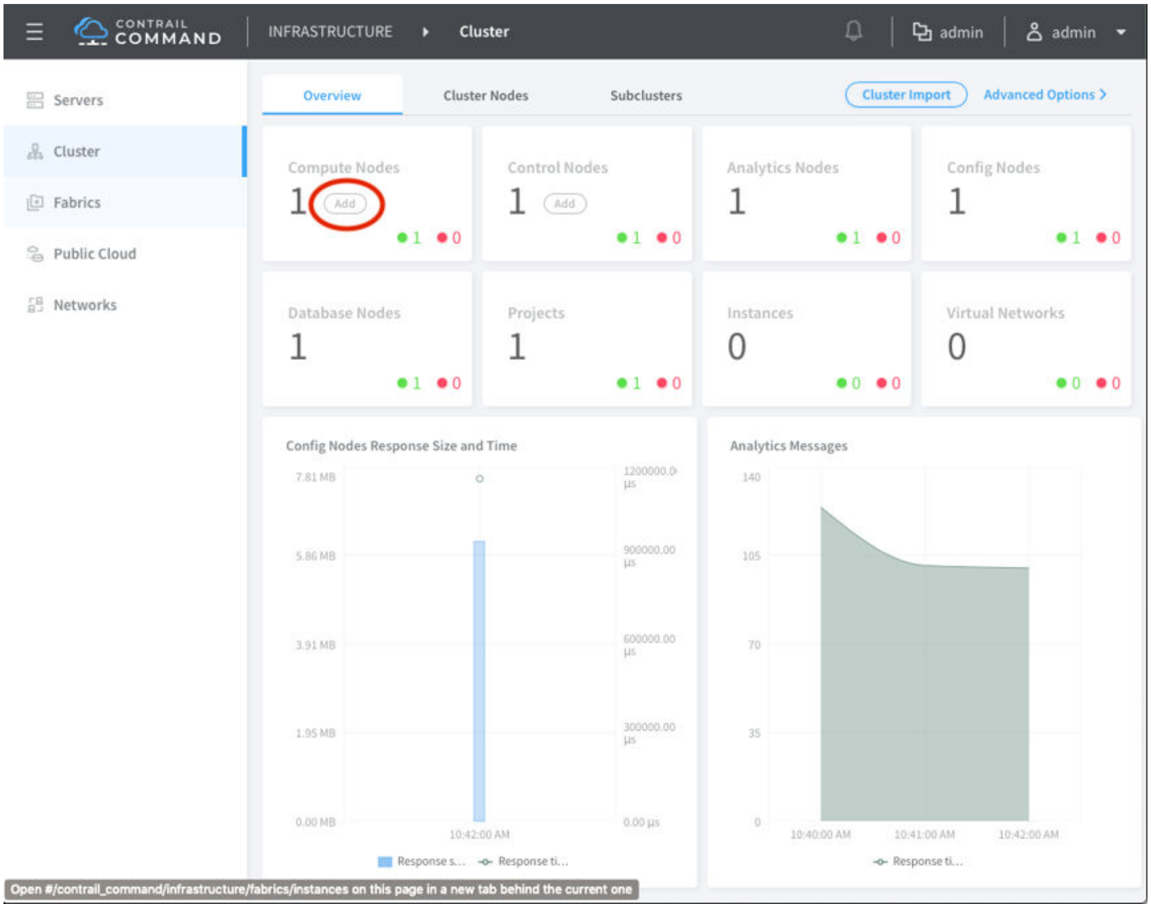


   b. Enter the required details.

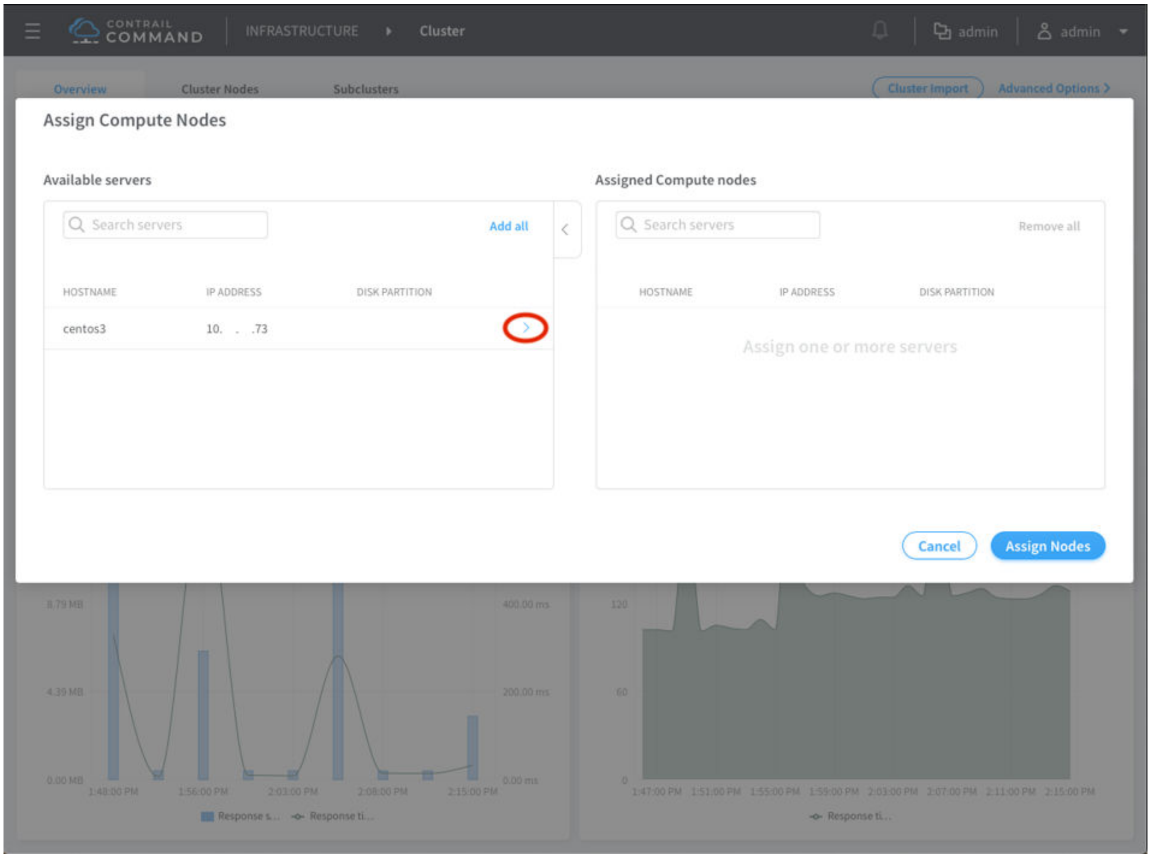
c. Click **Create**.

3. Click **Cluster**.

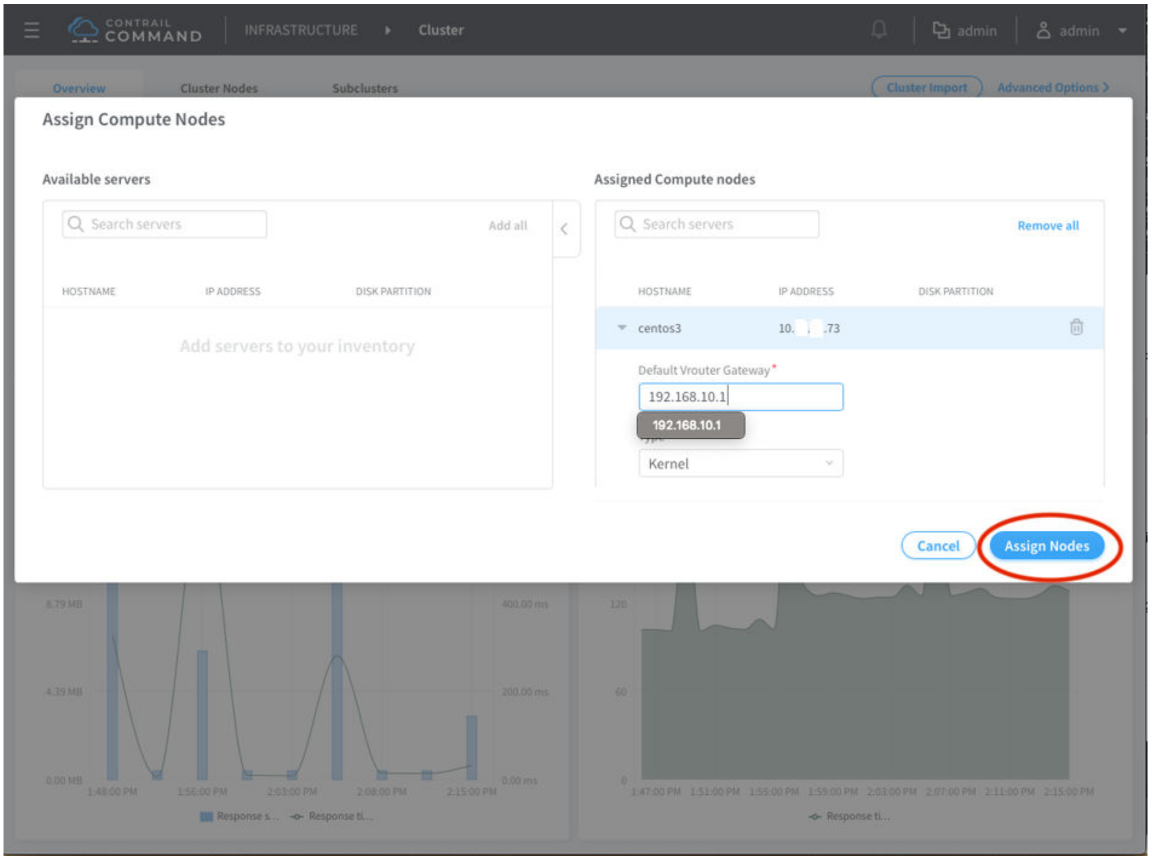
   a. Click **Add** under **Compute Nodes**.



   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.

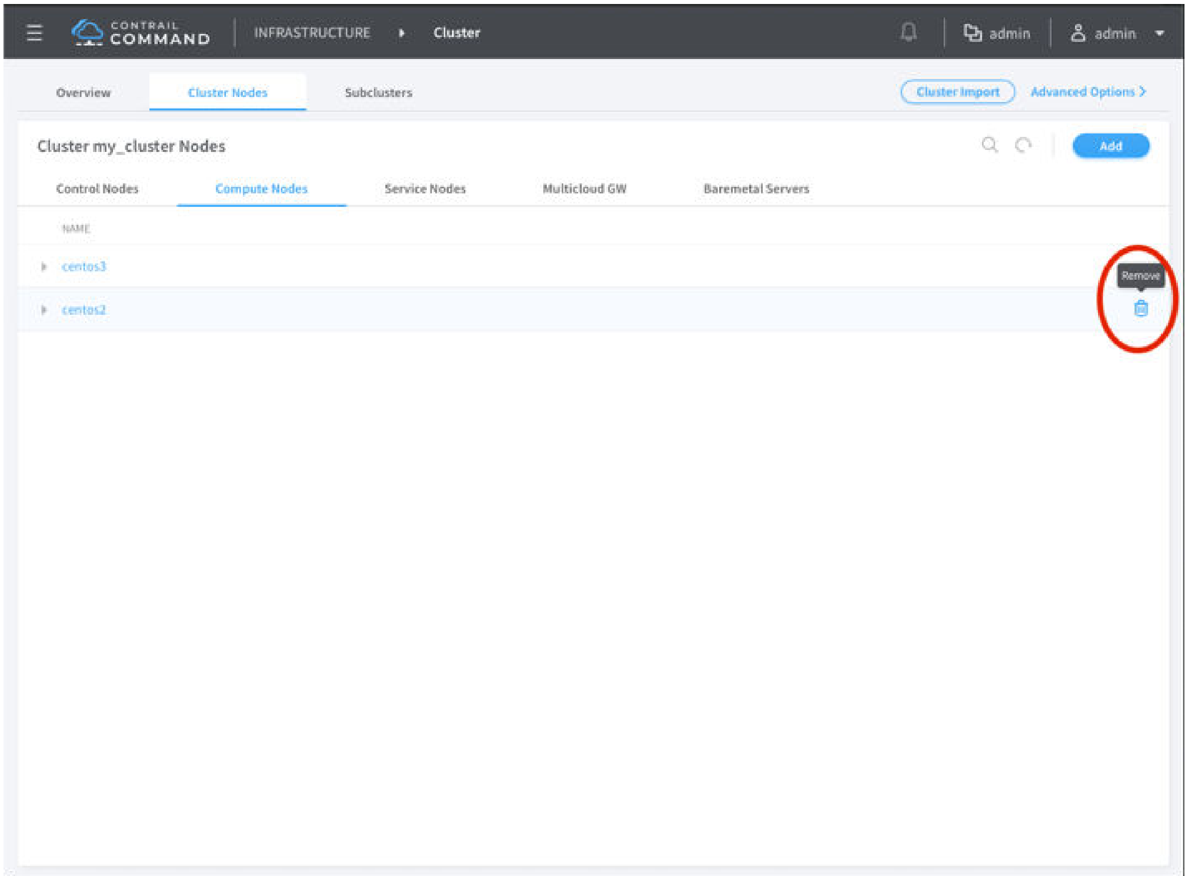
1. Login to Contrail Command.

   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" on page 18.

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 "Adding a New Compute Node to Existing Contrail Cluster" on page 71.

CHAPTER 4

Installing Contrail

IN THIS CHAPTER

Installing Contrail with OpenStack and Kolla Ansible

IN THIS SECTION

NOTE: We recommend using Contrail Command to add compute nodes to existing Contrail clusters in most Contrail Networking deployments. See "Adding a New Compute Node to Existing Contrail Cluster Using Contrail Command" on page 53.

The procedure in this document should only be performed by network administrators with significant YAML file expertise in environments not using the Contrail Command GUI.

The goal of this topic is to install Contrail Networking with OpenStack, using Kolla Ansible playbook `contrail-kolla-ansible`.

Kolla is an OpenStack project which provides tools to build container images for OpenStack services. Kolla Ansible provides Ansible playbooks to deploy the Kolla images.

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

Refer to "Installing Contrail Cluster using Contrail Command and instances.yml" on page 42 to deploy a Contrail cluster using Contrail Command.

Follow the procedure to deploy Kolla containers using `contrail-kolla-ansible` and Contrail Networking containers using `contrail-ansible-deployer`:

Set Up the Base Host

Update CentOS and kernel version. For a list of supported platforms, see https://www.juniper.net/documentation/en_US/release-independent/contrail/topics/reference/contrail-supported-platforms.pdf.

The vRouter has a dependency with the host kernel.

To set up the base host:

1. Download *Ansible Deployer* installer package from the Contrail Downloads page.

2. Install Ansible.

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

   ```
   yum -y install git ansible-2.7.10
   ```

3. Install *python-pip*.

   ```
   yum install -y python-pip
   ```

4. Run the following commands.

   ```
   yum -y remove PyYAML python-requests
   ```

   ```
   pip install PyYAML requests
   ```

5. Untar the tgz file.

   ```
   - tar xvf contrail-ansible-deployer-19<xx>.<NN>.tgz
   ```

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

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

   - The provider configuration (`provider_config`) section refers to the cloud provider where the Contrail cluster will be hosted, and contains all parameters relevant to the provider. For bare metal servers, the provider is `bms`.

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

      If a custom registry is specified, also specify the same registry under `kolla_globals` as `contrail_docker_registry`.
    - `CONTRAIL_VERSION`, if not specified, will default to the "latest" tag.
    - For more information about all possible parameters for `contrail_configuration`, see https://github.com/tungstenfabric/tf-container-builder/blob/master/containers/base/common.sh.
    - You must specify the *roles* in the **instances.yaml** file. Otherwise, the installation procedure will fail.
    - If there are host-specific values per host, for example, if the names of the interfaces used for "network_interface" are different on the servers in your cluster, use the example configuration at *Configuration Sample for Multi Node OpenStack HA and Contrail (multi interface)*.
    - Many of the parameters are automatically derived to sane defaults (how the first configuration works). You can explicitly specify variables to override the derived values if required. Review the code to see the derivation logic.

Example: instances.yaml

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

```
provider_config:
  bms:
    ssh_pwd: <password>
    ssh_user: root
    ntpserver: <IP NTP server>
    domainsuffix: local
instances:
  bms1:
    provider: bms
    ip: <IP BMS>
```

```
roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      analytics_alarm:
      analytics_snmp:
      webui:
      vrouter:
      openstack:
      openstack_compute:
contrail_configuration:
  RABBITMQ_NODE_PORT: 5673
  AUTH_MODE: keystone
  KEYSTONE_AUTH_URL_VERSION: /v3
kolla_config:
  kolla_globals:
    enable_haproxy: no
  kolla_passwords:
    keystone_admin_password: <Keystone admin password>
```

Example: instances.yaml

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

```
provider_config:
  bms:
    ssh_pwd: <password>
    ssh_user: root
    ntpserver: <IP NTP server>
    domainsuffix: local
instances:
  bms1:
    provider: bms
    ip: <IP BMS>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
```

```
        analytics_alarm:
        analytics_snmp:
        webui:
        vrouter:
        openstack:
        openstack_compute:
  global_configuration:
    CONTAINER_REGISTRY: <Registry FQDN/IP>:<Registry Port>
    REGISTRY_PRIVATE_INSECURE: True
  contrail_configuration:
    CONTRAIL_VERSION: latest
    CLOUD_ORCHESTRATOR: openstack
    VROUTER_GATEWAY: <IP gateway>
    RABBITMQ_NODE_PORT: 5673
    PHYSICAL_INTERFACE: <interface name>
    AUTH_MODE: keystone
    KEYSTONE_AUTH_URL_VERSION: /v3
  kolla_config:
    kolla_globals:
      kolla_internal_vip_address: <Internal VIP>
      contrail_api_interface_address: <Contrail API Addr>
      enable_haproxy: no
    kolla_passwords:
      keystone_admin_password: <Keystone Admin Password>
```

7. Run the following commands from the *contrail-ansible-deployer* folder:

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

8. Open web browser and type **https://contrail-server-ip:8143** to access Contrail Web UI.

   The default login user name is **admin**. Use the same password which was entered in step "6" on page 59

Multiple Interface Configuration Sample for Multinode OpenStack HA and Contrail

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

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

Configuration Sample—Multiple Interface

```
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:
      analytics_alarm:
      analytics_snmp:
      webui:
  bms5:
    provider: bms
    ip: <BMS5 IP>
```

```
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      analytics_alarm:
      analytics_snmp:
      webui:
  bms6:
    provider: bms
    ip: <BMS6 IP>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      analytics_alarm:
      analytics_snmp:
      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
  KEYSTONE_AUTH_URL_VERSION: /v3
  IPFABRIC_SERVICE_HOST: <Service Host IP>
  # Add following line for TSN Compute Node
  TSN_NODES: <TSN NODE IP List>
```

```
  # For EVPN VXLAN TSN
  ENCAP_PRIORITY: "VXLAN,MPLSoUDP,MPLSoGRE"
  PHYSICAL_INTERFACE: <Interface name>
kolla_config:
  kolla_globals:
    kolla_internal_vip_address: <Internal VIP>
    kolla_external_vip_address: <External VIP>
    contrail_api_interface_address: <Contrail API IP>
  kolla_passwords:
    keystone_admin_password: <Keystone Admin Password>
```

Single Interface Configuration Sample for Multinode OpenStack HA and Contrail

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

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

Configuration Sample—Single Interface

```
provider_config:
  bms:
    ssh_pwd: <password>
    ssh_user: root
    ntpserver: xx.xx.x.xx
    domainsuffix: local
instances:
  centos1:
    provider: bms
    ip: ip-address
    roles:
      openstack:
  centos2:
    provider: bms
    ip: ip-address
    roles:
      openstack:
  centos3:
    provider: bms
    ip: ip-address
```

```
    roles:
      openstack:
  centos4:
    provider: bms
    ip: ip-address
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      analytics_alarm:
      analytics_snmp:
      webui:
  centos5:
    provider: bms
    ip: ip-address
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      analytics_alarm:
      analytics_snmp:
webui:
  centos6:
    provider: bms
    ip: ip-address
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      analytics_alarm:
      analytics_snmp:
      webui:
  centos7:
    provider: bms
    ip: ip-address
    roles:
      vrouter:
```

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

NOTE: Replace *<contrail_version>* with the correct `contrail_container_tag` value for your Contrail release. The respective `contrail_container_tag` values are listed in README Access to Contrail Registry 19XX.

Frequently Asked Questions

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

Using Host-Specific Parameters

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

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

An extended example is available at: Configuration Sample for Multi Node OpenStack HA and Contrail.

Containers from Private Registry Not Accessible

1. You might have a situation in which containers that are pulled from a private registry named CONTAINER_REGISTRY are not accessible.

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

Error: Failed to insert vrouter kernel module

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

```
 [srvr5] ~ # docker logs vrouter_vrouter-kernel-init_1
    /bin/cp: cannot create regular file '/host/bin/vif': No such file or directory
    INFO: Load kernel module for kver=3.10.0
    INFO: Modprobing vrouter /opt/contrail/vrouter-kernel-modules/3.10.0-957.11.6.el7.x86_64/
vrouter.ko
                  total        used        free      shared  buff/cache   available
    Mem:            62G        999M         55G        9.1M        5.9G         60G
    Swap:            0B          0B          0B
                  total        used        free      shared  buff/cache   available
    Mem:            62G        741M         61G        9.1M        923M         61G
    Swap:            0B          0B          0B
    insmod: ERROR: could not insert module /opt/contrail/vrouter-kernel-modules/
3.10.0-957.11.6.el7.x86_64/vrouter.ko: Unknown symbol in module
    ERROR: Failed to insert vrouter kernel module
```

2. In this release, the vrouter module requires the host kernel version to be 3.10.0-957.11.6.el7.x86_64. To get this kernel version, before running provision, install the kernel version on the target nodes.

```
yum -y install
kernel-3.10.0-957.11.6.el7.x86_64

    yum update
    reboot
```

Fatal Error When Vrouter Doesn't Specify OpenStack

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

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

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

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

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

   For more information about this use case, refer to the bug #1756133.

Need for HAProxy and Virtual IP on a Single OpenStack Cluster

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

Using the kolla_toolbox Container to Run OpenStack Commands

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

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

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

    docker exec -it kolla_toolbox bash
    # NOW YOU ARE INSIDE THE KOLLA_TOOLBOX CONTAINER
    (kolla-toolbox)[ansible@server1 /]$ source /var/lib/kolla/config_files/admin-openrc.sh
    (kolla-toolbox)[ansible@server1 /]$ cd /var/lib/kolla/config_files
    (kolla-toolbox)[ansible@server1 /var/lib/kolla/config_files]$ openstack image create cirros2
--disk-format qcow2 --public --container-format bare --file cirros-0.4.0-x86_64-disk.img
    +------------------+------------------------------------------------------+
    | Field            | Value                                                |
    +------------------+------------------------------------------------------+
    | checksum         | 443b7623e27ecf03dc9e01ee93f67afe                     |
    | container_format | bare                                                 |
    | created_at       | 2018-03-29T21:37:48Z                                 |
    | disk_format      | qcow2                                                |
    | file             | /v2/images/e672b536-0796-47b3-83a6-df48a5d074be/file |
    | id               | e672b536-0796-47b3-83a6-df48a5d074be                 |
    | min_disk         | 0                                                    |
    | min_ram          | 0                                                    |
    | name             | cirros2                                              |
    | owner            | 371bdb766278484bbabf868cf7325d4c                     |
```

```
| protected        | False                                                  |
| schema           | /v2/schemas/image                                      |
| size             | 12716032                                               |
| status           | active                                                 |
| tags             |                                                        |
| updated_at       | 2018-03-29T21:37:50Z                                   |
| virtual_size     | None                                                   |
| visibility       | public                                                 |
+------------------+--------------------------------------------------------+
(kolla-toolbox)[ansible@server1 /var/lib/kolla/config_files]$ openstack image list
+--------------------------------------+---------+--------+
| ID                                   | Name    | Status |
+--------------------------------------+---------+--------+
| e672b536-0796-47b3-83a6-df48a5d074be | cirros2 | active |
| 57e6620e-796a-40ee-ae6e-ea1daa253b6c | cirros2 | active |
+--------------------------------------+---------+--------+
```

RELATED DOCUMENTATION

Installing Contrail Cluster using Contrail Command and instances.yml | 42

Adding a New Compute Node to Existing Contrail Cluster

NOTE: We recommend using Contrail Command to add compute nodes to existing Contrail clusters in most Contrail Networking deployments. See "Adding a New Compute Node to Existing Contrail Cluster Using Contrail Command" on page 53.

The procedure in this document should only be performed by network administrators with significant YAML file expertise in environments not using the Contrail Command GUI.

This is initial process for adding a new compute node to existing Contrail OpenStack cluster.

Assume Contrail cluster is successfully provisioned by the following **instances.yaml** file.

NOTE: The password values in this output are included for illustrative purposes only. We strongly recommend creating a unique username and password combination whenever possible.

```
provider_config:
  bms:
    ssh_pwd: c0ntrail123
    ssh_user: root
    ntpserver: x.x.x.x
    domainsuffix: local
instances:
  srvr1:
    provider: bms
    ip: 192.168.1.51
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      openstack:
  srvr2:
    provider: bms
    ip: 192.168.1.52
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      openstack:
  srvr3:
    provider: bms
    ip: 192.168.1.53
    roles:
      config_database:
      config:
      control:
```

```
      analytics_database:
      analytics:
      webui:
      openstack:
  srvr4:
    provider: bms
    ip: 192.168.1.54
    roles:
      vrouter:
      openstack_compute:
contrail_configuration:
  CONTRAIL_VERSION: 5.1.0-0.40-ocata
  CONTROL_DATA_NET_LIST: 192.168.10.0/24
  RABBITMQ_NODE_PORT: 5673
  VROUTER_GATEWAY: 192.168.10.1
  IPFABRIC_SERVICE_HOST: 192.168.10.150
  KEYSTONE_AUTH_URL_VERSION: /v3
kolla_config:
  kolla_globals:
    kolla_internal_vip_address: 192.168.10.150
    kolla_external_vip_address: 192.168.1.150
```

Run the following commands to add a new compute node to an existing Contrail OpenStack cluster.

1. Edit the **instances.yaml** file to add a compute node, *srvr5*.

   NOTE: The password values in this output are included for illustrative purposes only. We strongly recommend creating a unique username and password combination whenever possible.

```
provider_config:

  bms:
    ssh_pwd: c0ntrail123
    ssh_user: root
    ntpserver: x.x.x.x
    domainsuffix: local
instances:
  srvr1:
    provider: bms
    ip: 192.168.1.51
```

```
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      openstack:
  srvr2:
    provider: bms
    ip: 192.168.1.52
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      openstack:
  srvr3:
    provider: bms
    ip: 192.168.1.53
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      openstack:
  srvr4:
    provider: bms
    ip: 192.168.1.54
    roles:
      vrouter:
      openstack_compute:
  srvr5:
    provider: bms
    ip: 192.168.1.55
    roles:
      vrouter:
      openstack_compute:
```

```
contrail_configuration:
  CONTRAIL_VERSION: 5.1.0-0.38-ocata
  CONTROL_DATA_NET_LIST: 192.168.10.0/24
  RABBITMQ_NODE_PORT: 5673
  VROUTER_GATEWAY: 192.168.10.1
  IPFABRIC_SERVICE_HOST: 192.168.10.150
  KEYSTONE_AUTH_URL_VERSION: /v3
kolla_config:
  kolla_globals:
    kolla_internal_vip_address: 192.168.10.150
    kolla_external_vip_address: 192.168.1.150
```

2. Run the **configure_instances.yml** playbook with the new **instances.yaml** file.

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

   It will install the required software and also, prepare the new node for running the relevant containers.

3. Run playbooks.

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

> **NOTE**: The *--tags nova* option runs only the *nova* role so that the other containers are not affected.
>
> It is not recommended to omit the above option. If the option is omitted, especially when multiple OpenStack nodes are running with HA, the *MariaDB Galera* cluster will go out of sync and will not converge. In such situation, the only solution is to re-provision the entire OpenStack cluster.

You can also add or remove a compute node to existing Contrail cluster using Contrail Command UI. For details, refer to "Adding a New Compute Node to Existing Contrail Cluster Using Contrail Command" on page 53.

CHAPTER 5

Using Contrail with AppFormix

IN THIS CHAPTER

Contrail and AppFormix Deployment Requirements

IN THIS SECTION

Starting with Contrail Release 5.1, the combined installation of Contrail and AppFormix using the Contrail Command UI is supported. For information about server requirements, supported platforms, and installation see:

- "Server Requirements and Supported Platforms" on page 16.
- "Installing AppFormix and AppFormix Flows using Contrail Command" on page 77.

Software Requirements

Table 2 on page 76 specifies which AppFormix release to use with each applicable Contrail release.

Table 2: AppFormix Releases > Contrail Releases

Contrail Release	AppFormix Release
1912	3.1.11

Table 2: AppFormix Releases > Contrail Releases *(Continued)*

Contrail Release	AppFormix Release
1911	3.1.9
1910	3.1.6
1909	3.1.2
1908	3.0.1
1907	2.19.11
5.1	2.19.10

Installing AppFormix and AppFormix Flows using Contrail Command

IN THIS SECTION

AppFormix Release to Use with Contrail Release

Table 3 on page 78 specifies which AppFormix release to use with the Contrail release. For previous releases, check the Supported Platforms in the Contrail Release Notes for the applicable Contrail release.

Table 3: AppFormix Release

Contrail Networking Release	AppFormix Release	Operating System
Contrail Networking Release 1912.L1	3.1.11 - AppFormix 1.0.6 - AppFormix Flows	CentOS 7.7
Contrail Networking Release 1912	3.1.11 - AppFormix 1.0.6 - AppFormix Flows	CentOS 7.7

NOTE: Install AppFormix and AppFormix Flows during the initial installation along with the Contrail and OpenStack installation. AppFormix and AppFormix Flows cannot be installed after the Contrail and OpenStack cluster is already deployed and imported into Contrail Command.

4-Node Setup

4-Node setup includes:

Node 1	Contrail Command
Node 2	OpenStack and Contrail
Node 3	AppFormix
Node 4	AppFormix Flows

Hardware Requirements

The Contrail Command server, on which AppFormix Platform is installed has the following minimum requirements.

- CPU: 16 cores (virtual or physical)
- Memory: 64 GB
- Storage: 2 TB (recommended)

Requirements

- Install Centos 7.7 on all nodes.
- Have your AppFormix license from Juniper available. This will need to be added on the Contrail Command server in the same folder together with Appformix.

Server requirements and supported platforms are listed in the installation guide and release notes.

Download and Install AppFormix and AppFormix Flows on the Contrail-Command Node

To install AppFormix using Contrail Command:

1. Download AppFormix from

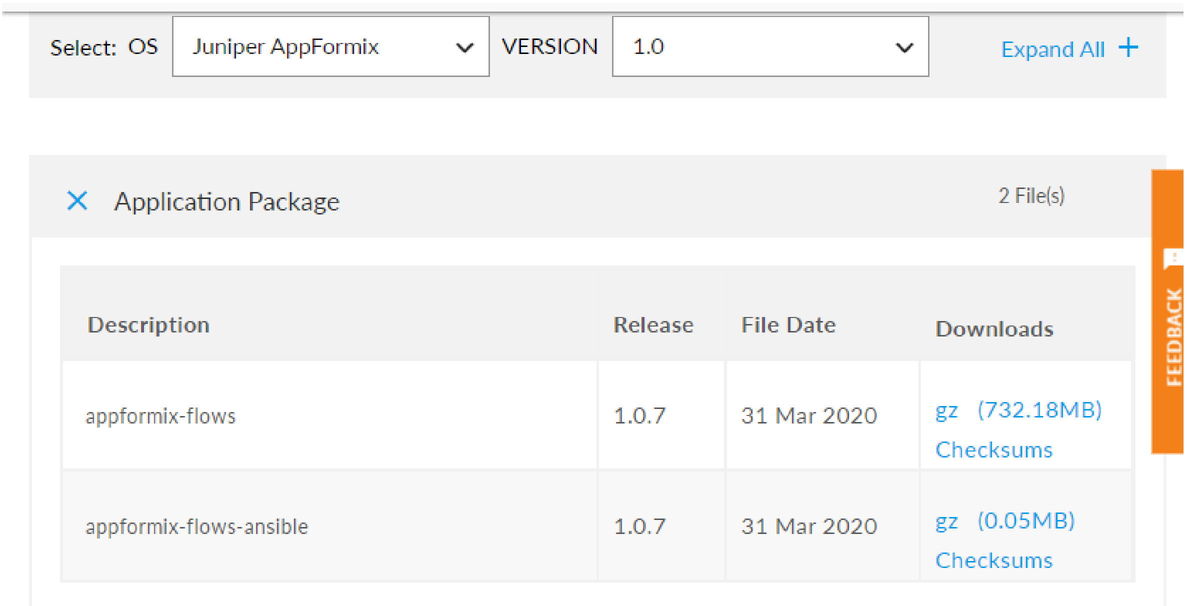
   https://support.juniper.net/support/downloads/.

   > **NOTE**: See Table 3 on page 78 for Contrail, AppFormix, and AppFormix Flows version mapping.

   ```
   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
   ```

   Download the AppFormix Flows images by selecting 1.0 from the drop-down list:

Figure 9: Download AppFormix Flows Images



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

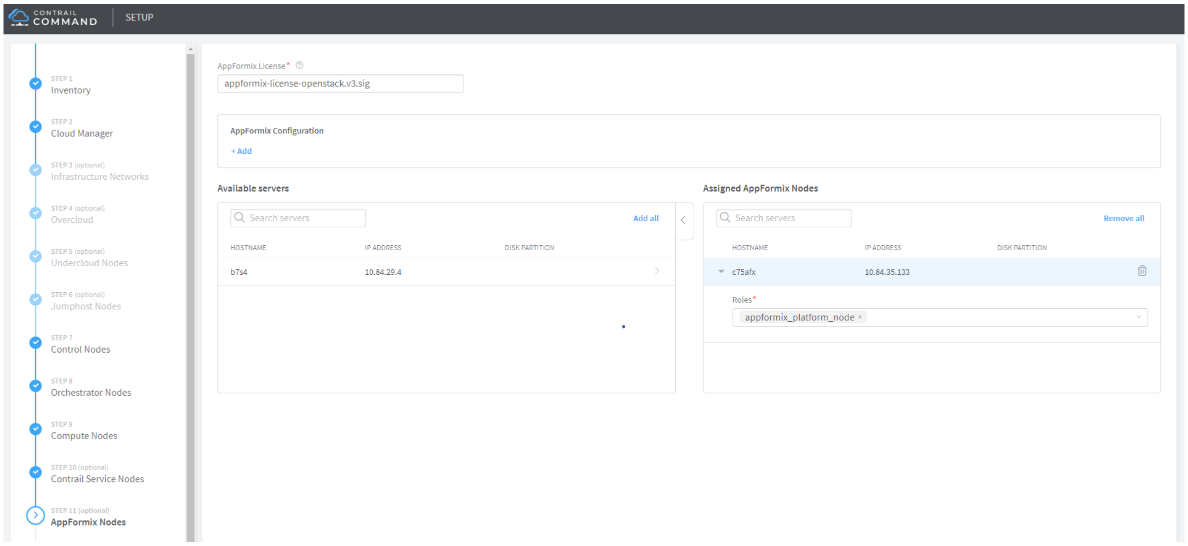
2. Verify the `command_servers.yml` file was added before installing Contrail Command as specified in "Installing Contrail Command" on page 18. This makes `/opt/software/appformix` available in `contrail-command` Docker.

   Add the following statements to the `command_servers.yml` file. They must be placed after the "---" at the very top of the file or as the last two lines at the very bottom of the file.

```
---
user_command_volumes:
- /opt/software/appformix:/opt/software/appformix
- /opt/software/xflow:/opt/software/xflow
command_servers:
    server1:
        ip: 192.168.100.129
```

3. From Setup, select **AppFormix Nodes**.

Figure 10: AppFormix Nodes Setup

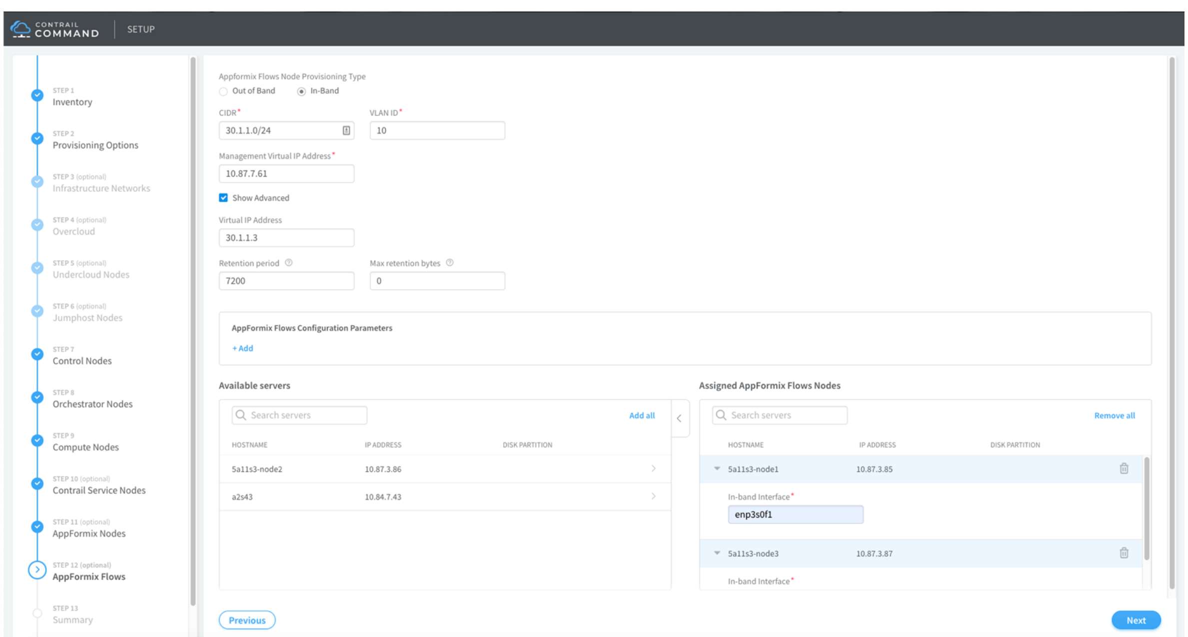


4. From AppFormix Nodes, the `appformix_platform` role is selected by default.

   Optionally, select `appformix_bare_host` and `appformix_network_agents` roles as needed.

5. Starting with Contrail networking release 1910, AppFormix Flows is integrated in the Contrail Command UI. For the AppFormix node, keep the `appformix_platform` role default.

Figure 11: Installing AppFormix Flows



Enter the following values:

Out-of-Band Provisioning:

- Virtual IP Address—This is the IP address of the load balancer node for AppFormix Flow collectors.
- Available Servers—Select the server on which AppFormix Flows node is to be installed.

In-Band Provisioning:

- CIDR—Enter the underlay telemetry infrastructure subnet. In-band interface on the AppFormix Flows node is assigned an IP address from this subnet.
- VLAN-ID—Enter the VLAN ID used for the telemetry infrastructure network.
- Management Virtual IP Address—Enter an unused IP address which will be used as the management IP Address. Contrail Command uses this IP address to connect to the AppFormix Flows node.
- Virtual IP Address—The IP address is populated by default and is usually the third IP Address from the CIDR range (telemetry subnet). However, you can change the IP address if needed. This IP address is used as the collector destination IP address for the sFlow target on TOR switches.
- (Optional) Retention Period—Time duration in seconds that you want to keep the collected data. Default is `7200 s`.
- (Optional) Max Retention Bytes—Maximum size of the data to be collected. Default is `0` which indicates unlimited size.

6. Click **Next** to continue to Summary.
7. Verify the summary of your configuration and click **Provision**.
8. Use the following commands to check and track deployment progress:

```
vim /var/tmp/contrail_cluster/<cluster-id>/instances.yml
tail -f /var/log/contrail/deploy.log
```

NOTE: After the AppFormix installation, you can view monitoring by selecting **Monitoring > External Apps > AppFormix**.

Enable LLDP and Analytics To Collect

In the AppFormix software, enable LLDP for each device and any analytics that you want to collect.

1. In the AppFormix Dashboard, select the menu in the upper-right corner, then select **Settings**.

2. Select **Network Devices** > **Add Device**.

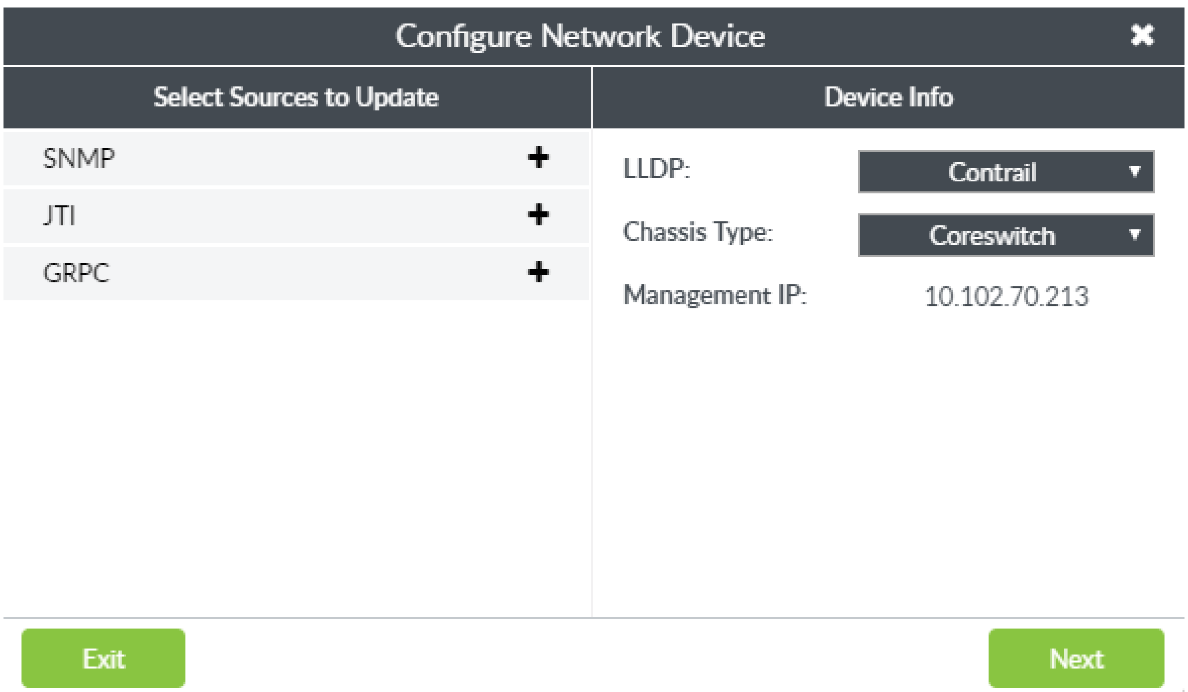
3. In the LLDP field, complete the following:

   - Select **Contrail** in Device Info for LLDP. The default is **Contrail**.

   - Add the Management IP address, then click **Next**.

   - Select **SNMP** > **+** in Device Sources to input the SNMP community string. The default community string for each Junos device provisioned is **public**.

Figure 12: Enable LLDP and Add Management IP for Network Device



4. In the Resource field, select the **Resource** from the list, then click **Add**.

Figure 13: Add Selected Resource for Network Device MIB Configurations

5. Click **Submit** to complete.

Release History Table

Release	Description
1910	Starting with Contrail networking release 1910, AppFormix Flows is integrated in the Contrail Command UI.

RELATED DOCUMENTATION

Contrail Insights Flows in Contrail Command

Configuring Instances in Contrail Insights

Configuring Contrail Insights Alarms using Contrail Command

Viewing Cluster Node Details and Metric Values

Metrics Collected by Contrail Insights

CHAPTER 6

Using Contrail with Kubernetes

IN THIS CHAPTER

Installing and Managing Contrail Microservices Architecture Using Helm Charts

IN THIS SECTION

NOTE: Starting in Contrail Release 1912.L1, Helm support is unavailable in Contrail Networking. The Helm support content in this document supports Contrail Networking Releases 1907 through 1912.

This section provides an overview of using Helm charts when installing Contrail with a microservices architecture. Contrail Helm charts work together with OpenStack Helm for an OpenStack Contrail deployment. For an introduction to Contrail microservices, refer to "Understanding Contrail Microservices Architecture" on page 6.

Understanding Helm Charts

Helm is the package manager for Kubernetes, an open source software for managing containerized systems. The packaging format used by Helm is a chart, a collection of files that describe a related set of Kubernetes resources. Helm charts enable you to define, install, and configure your Kubernetes application. A chart can be used to deploy something simple, like a memcached pod, or something complex, like a full web application stack complete with HTTP servers, databases, and the like.

Contrail Helm charts give you complete life cycle management of installation, update, and delete of Contrail Docker-based containers in a microservices architecture.

The Contrail Helm deployer supports deploying Contrail for OpenStack.

Contrail Helm Deployer Charts

The Contrail Helm deployer uses the following charts.

- **helm-toolkit chart**

  Contains common templates and functions that are used by every other Contrail Helm chart.

- **contrail-thirdparty chart**

  Defines and deploys third party containers as Kubernetes resources for Contrail, including:

  - RabbitMQ
  - ZooKeeper
  - Cassandra
  - Kafka
  - Redis

- **contrail-controller chart**

  Deploys and manages Contrail components as Kubernetes resources, including:

  - control
  - config
  - webui

- **contrail-analytics chart**

  Deploys and manages Contrail analytics components as Kubernetes resources.

- **contrail-vrouter chart**

  Deploys and manages Contrail vrouter components as Kubernetes resources.

- **contrail-superset chart**

  A superset of all other Contrail Helm charts, can be used to install all Kubernetes resources defined in other Contrail charts.

Contrail Kubernetes Resource implementation

All Contrail Helm charts follow a similar approach to implementing Kubernetes resources. For each of the Contrail Release 5.0 containers, configuration input is given as an environment variable in the file `values.yaml`. Use the variable `.Values.contrail_env` to define environment variables for the containers.

```
contrail_env:
  CONTROLLER_NODES: <Controller-Nodes-IP-Address>
  LOG_LEVEL: SYS_NOTICE
  CLOUD_ORCHESTRATOR: openstack
  AAA_MODE: cloud-admin
```

All of the environment variables are stored in Kubernetes resources called configmaps. The configmaps are loaded into specific containers as environment variables.

Because Contrail is an infrastructure-level application, every pod of Contrail is hosted on the host network namespace. Consequently, the daemonset controller is used to define all Contrail pods, so that each of the Contrail pods are brought up on different nodes to avoid port conflicts.

Example: Contrail Pods Deployment Options

NOTE: By default, the `contrail-thirdparty` Helm chart creates a separate pod for each of the third party services.

```
pods:
    - contrail-control
      containers:
        - contrail-control
        - contrail-dns
        - contrail-named
        - control-nodemgr
    - contrail-config
      containers:
        - config-api
        - schema-transformer
        - svc-monitor
        - device-manager
        - config-nodemgr
    - contrail-webui
      containers:
        - contrail-webui
        - contrail-middleware
    - contrail-analytics
      containers:
        - analytics-api
        - analytics-colletor
        - snmp-collector
        - query-engine
        - alarm-gen
        - contrail-topology
    - contrail-vrouter
      containers:
        - vrouter-kernel/vrouter-dpdk/vrouter-sriov
        - vrouter-agent
        - vrouter-nodemgr
```

Installing Contrail Using Helm Charts

Use one of the following procedures to install Contrail with OpenStack Ocata using Helm charts:

- "Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability " on page 103
- "Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata " on page 114

RELATED DOCUMENTATION

Provisioning of Kubernetes Clusters

IN THIS SECTION

Contrail Networking supports the following ways of provisioning Kubernetes clusters:

Provisioning of a Standalone Kubernetes Cluster

You can provision a standalone Kubernetes cluster using contrail-ansible-deployer.

Perform the following steps to install one Kubernetes cluster and one Contrail cluster and integrate them together.

1. See Supported Platforms Contrail Release for a list of supported platforms.
2. Install the necessary tools.

```
yum -y install epel-release git ansible net-tools
```

3. Download the `contrail-ansible-deployer-19<xx>.<NN>.tgz` Ansible Deployer application tool package onto your provisioning host from Contrail Downloads page and extract the package.

   ```
   - tar xvf contrail-ansible-deployer-19<xx>.<NN>.tgz
   ```

4. Navigate to the **contrail-ansible-deployer** directory.

   ```
   cd contrail-ansible-deployer
   ```

5. Edit the **config/instances.yaml** and enter the necessary values. See "Understanding contrail-ansible-deployer used in Contrail Command" on page 7 for a sample **config/instances.yaml** file.

6. Turn off the `swap` functionality on all nodes.

   ```
   swapoff -a
   ```

7. Configure the nodes.

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

8. Install Kubernetes and Contrail.

   ```
   ansible-playbook -e orchestrator=kubernetes -i inventory/ playbooks/install_k8s.yml
   ```

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

9. Turn on the `swap` functionality on all nodes.

   ```
   swapon -a
   ```

Provisioning of Nested Contrail Kubernetes Clusters

IN THIS SECTION

When Contrail provides networking for a Kubernetes cluster that is provisioned on the workloads of a Contrail-OpenStack cluster, it is called a nested Kubernetes cluster. Contrail components are shared between the two clusters.

Prerequisites

Ensure that the following prerequisites are met before provisioning a nested Kubernetes cluster:

1. Ensure that you have an operational Contrail-OpenStack cluster based on Contrail Networking Release 19<xx>..

2. Ensure that you have an operational Kubernetes v1.12.9 cluster on virtual machines created on an Contrail-OpenStack cluster.

3. Update the **/etc/hosts** file on the Kubernetes master node with entries for each node of the cluster.

   For example, if the Kubernetes cluster is made up of three nodes such as master1 (IP: x.x.x.x), minion1 (IP: y.y.y.y), and minion2 (IP: z.z.z.z). The **/etc/hosts** on the Kubernetes master node must have the following entries:

   ```
   x.x.x.x master1
   y.y.y.y minion1
   z.z.z.z minion2
   ```

4. If Contrail container images are stored in a secure docker registry, a Kubernetes secret must be created and referenced during "Generate a single yaml file to create a Contrail-k8s cluster" on page 93, with credentials of the private docker registry.

   ```
   kubectl create secret docker-registry name --docker-server=registry --docker-
   username=username --docker-password=password --docker-email=email -n namespace
   ```

   Command options:

   - *name*—Name of the secret.
   - *registry*—Name of the registry. Example: hub.juniper.net/contrail.
   - *username*—Username to log in to the registry.
   - *password*—Password to log in to the registry.
   - *email*—Registered email of the registry account.
   - *namespace*—Kubernetes namespace where the secret must be created. This should be the namespace where you intend to create the Contrail pods.

The following steps describe how to provision a nested Contrail Kubernetes cluster.

Configure network connectivity to Contrail configuration and data plane functions.

A nested Kubernetes cluster is managed by the same Contrail control processes that manage the underlying OpenStack cluster.

The kube-manager is essentially a part of the Contrail Config function. In a nested deployment, one kube-manager instance will is provisioned in each overlay cluster. This necessitates the need The kube-manager running in the overlay must have network reachability to Contrail config functions of the underlay OpenStack cluster.

Network connectivity for the following Contrail config functions are required:

- Contrail Config
- Contrail Analytics
- Contrail Msg Queue
- Contrail VNC DB
- Keystone

In addition to config connectivity, the CNI for the Kubernetes cluster needs network reachability to the vRouter on its Compute node. Network connectivity for the vRouter data plane function is also required.

You can use the link local service feature or a combination of link local service with fabric Source Network Address Translation (SNAT) feature of Contrail to provide IP reachability to and from the overlay Kubernetes cluster config and data components to corresponding config and data compoeenents of the underlay OpenStack cluster.

To provide IP reachability to and from the Kubernetes cluster using the fabric SNAT with link local service, perform the following steps.

1. Enable fabric SNAT on the virtual network of the VMs.

   The fabric SNAT feature must be enabled on the virtual network of the virtual machines on which the Kubernetes master and minions are running.
2. Create a link local service for the Container Network Interface (CNI) to communicate with its vRouter Agent. This link local service should be configured using the Contrail GUI, in the following example:

Contrail Process	Service IP	Service Port	Fabric IP	Fabric Port
vRouter	*Service-IP for the active node*	9091	127.0.0.1	9091

NOTE: Fabric IP address is 127.0.0.1 since you must make the CNI communicate with the vRouter on its underlay node.

For example, the following link local services must be created:

Link Local Service Name	Service IP	Service Port	Fabric IP	Fabric Port
K8s-cni-to-agent	10.10.10.5	9091	127.0.0.1	9091

NOTE: Here 10.10.10.5 is the Service IP address that you chose. This can be any unused IP in the cluster. This IP address is primarily used to identify link local traffic and has no other significance.

Generate a single yaml file to create a Contrail-k8s cluster

Contrail components are installed on the Kubernetes cluster as pods. The configuration to create these pods in Kubernetes is encoded in a yaml file.

This file can be generated as follows:

1. Download the `contrail-ansible-deployer-19<xx>.<NN>.tgz` Ansible Deployer application tool package onto your provisioning host from Juniper Networks and extract the package.

   `- tar xvf contrail-ansible-deployer-19<xx>.<NN>.tgz`

2. Navigate to the **contrail-container-builder** directory.

   `cd contrail-container-builder`

3. Populate the **common.env** file located in the top directory of the cloned contrail-container-builder repo with information corresponding to your cluster and environment.

   For you reference, see a sample **common.env** file with required bare minimum configurations here https://github.com/Juniper/contrail-container-builder/blob/master/kubernetes/sample_config_files/common.env.sample.nested_mode.

   NOTE: If Contrail container images are stored in a secure docker registry, a Kubernetes secret must be created and referenced as documented in "4" on page 91 of Prerequisites. Populate the variable KUBERNETES_SECRET_CONTRAIL_REPO=<*secret-name*> with the name of the generated Kubernetes secret, in the **common.env** file.

4. Generate the yaml file as following in your shell:

   ```
   cd contrail-container-build-repo/kubernetes/manifests

   ./resolve-manifest.sh contrail-kubernetes-nested.yaml  > nested-contrail.yml
   ```

5. Copy the output (or file) generated from 4 to the master node in your Kubernetes cluster.

Instantiate the Contrail-k8s cluster

Create contrail components as pods on the Kubernetes cluster.

```
root@k8s:~# kubectl get pods -n kube-system
NAME                                 READY     STATUS    RESTARTS   AGE
contrail-kube-manager-lcjbc          1/1       Running   0          3d
contrail-kubernetes-cni-agent-w8shc  1/1       Running   0          3d
```

You will see the following pods running in the kube-system namespace:

contrail-kube-manager-xxxxxx—This is the manager that acts as conduit between Kubernetes and OpenStack clusters

contrail-kubernetes-cni-agent-xxxxx—This installs and configures Contrail CNI on Kubernetes nodes

Provisioning of Non-Nested Contrail Kubernetes Clusters

Prerequisites

Ensure that the following prerequisites are met before provisioning a non-nested Kubernetes cluster:

1. You must have an installed and operational Contrail OpenStack cluster based on the Contrail Networking Release 19*xx* release.

2. You must have an installed and operational Kubernetes cluster on the server where you want to install the non-nested Contrail Kubernetes cluster.

3. Label the Kubernetes master node with the Contrail controller label:

   ```
   kubectl label node node node-role.opencontrail.org/config=true
   ```

4. Ensure that the Kubelet running on the Kubernetes master node is not run with network plugin options. If kubelet is running with network plugin option, then disable or comment out the KUBELET_NETWORK_ARGS option in the **/etc/systemd/system/kubelet.service.d/10-kubeadm.conf** configuration file.

   > **NOTE**: It is recommended that the Kubernetes master should not be configured with a network plugin, so as to not install vRouter kernel module on the control node. However, this is optional.

5. Restart the kubelet service:

```
systemctl daemon-reload;
systemctl restart kubelet.service
```

In non-nested mode, a Kubernetes cluster is provisioned side by side with an OpenStack cluster with networking provided by the same Contrail components of the OpenStack cluster.

Provisioning a Contrail Kubernetes Cluster

Follow these steps to provision Contrail Kubernetes cluster.

1. Download the contrail-ansible-deployer-19*<xx>*.*<NN>*.tgz Ansible Deployer application tool package onto your provisioning host from Juniper Networks and extract the package.

   - tar xvf contrail-ansible-deployer-19<xx>.<NN>.tgz

2. Navigate to the **contrail-container-builder** directory.

   cd contrail-container-builder

3. Populate the **common.env** file located in the top directory of the cloned contrail-container-builder repo with information corresponding to your cluster and environment.

   For a sample **common.env** file with required bare minimum configurations see https://github.com/Juniper/contrail-container-builder/blob/master/kubernetes/sample_config_files/common.env.sample.non_nested_mode.

   > **NOTE**: If Config API is not secured by keystone, ensure that *AUTH_MODE* and *KEYSTONE_ ** variables are not configured or present while populating the **common.env** file.

4. Generate the yaml file as shown below:

```
cd contrail-container-build-repo/kubernetes/manifests

./resolve-manifest.sh contrail-kubernetes-nested.yaml  > non-nested-contrail.yml
```

5. Copy the file generated from 4 to the master node in your Kubernetes cluster.
6. Create contrail components as pods on the Kubernetes cluster as follows:

```
kubectl apply -f non-nested-contrail.yml
```

7. Create the following Contrail pods on the Kubernetes cluster. Ensure that contrail-agent pod is created only on the worker node.

```
[root@b4s403 manifests]# kubectl get pods --all-namespaces -o wide
       NAMESPACE     NAME                            READY     STATUS    RESTARTS
AGE       IP            NODE
       kube-system   contrail-agent-mxkcq            2/2       Running   0
1m        <x.x.x.x>     b4s402
       kube-system   contrail-kube-manager-glw5m     1/1       Running   0
1m        <x.x.x.x>     b4s403
```

RELATED DOCUMENTATION

Contrail Integration with Kubernetes

Installing Standalone Kubernetes Contrail Cluster using the Contrail Command UI

IN THIS SECTION

Starting with Contrail Release 5.1, you can use Contrail Command to initiate Kubernetes Contrail cluster deployment. This example topic describes how to use the Contrail Command User interface (UI) to deploy a standalone Kubernetes Contrail cluster.

Requirements

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

Overview

You can use Contrail Command to initiate a standalone Kubernetes Contrail cluster deployment. You must install the controller and compute nodes first. When the host nodes are operational, Contrail Command uses the underlying Ansible deployer to install a standalone Kubernetes Contrail cluster. Contrail Command supports the management and provisioning of Contrail components. To provision Kubernetes resources, such as pods, services, and so on, use the Kubernetes API server or the `kubectl` CLI on the Kubernetes master node.

Configuration

IN THIS SECTION

Deploying a Kubernetes Contrail Cluster

Step-by-Step Procedure

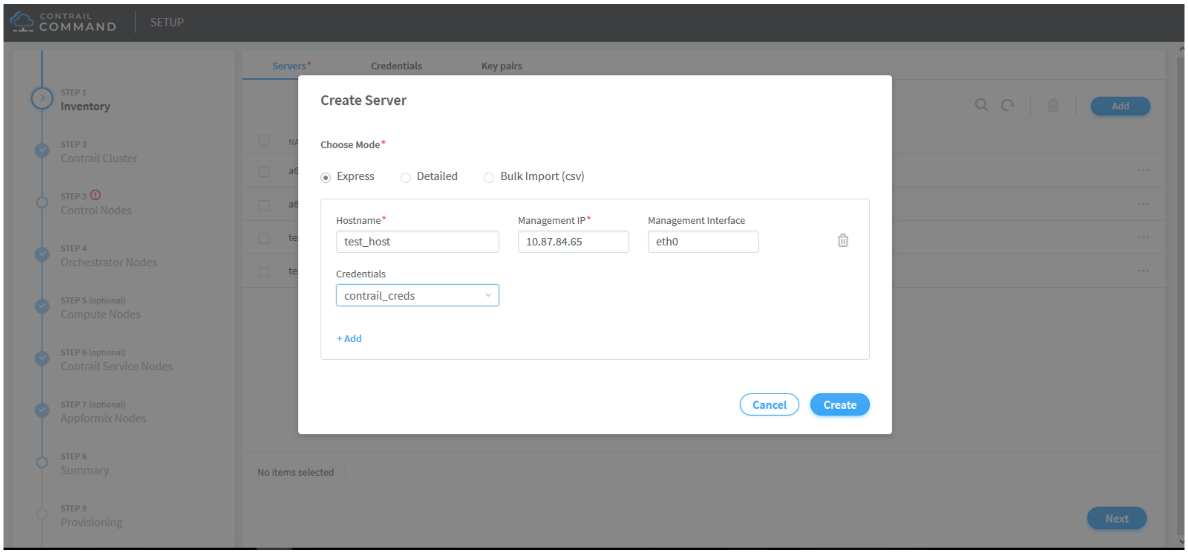
To deploy a Kubernetes Contrail cluster using Contrail Command, perform the following steps.

1. Click the **Create** button on the **Setup > Servers** tab to add physical servers. The **Create Server** page is displayed. You can add a server in the following ways:

   - Express
   - Detailed
   - Bulk Import (csv)

   > **NOTE**: Create server login credentials before adding the servers.

Figure 14: Create Server



Click **Create** to create the server. The list of servers is displayed in the **Inventory** page. Click **Next** to continue creating a cluster. The **Contrail Cluster** page appears.

2. Create a Contrail cluster.

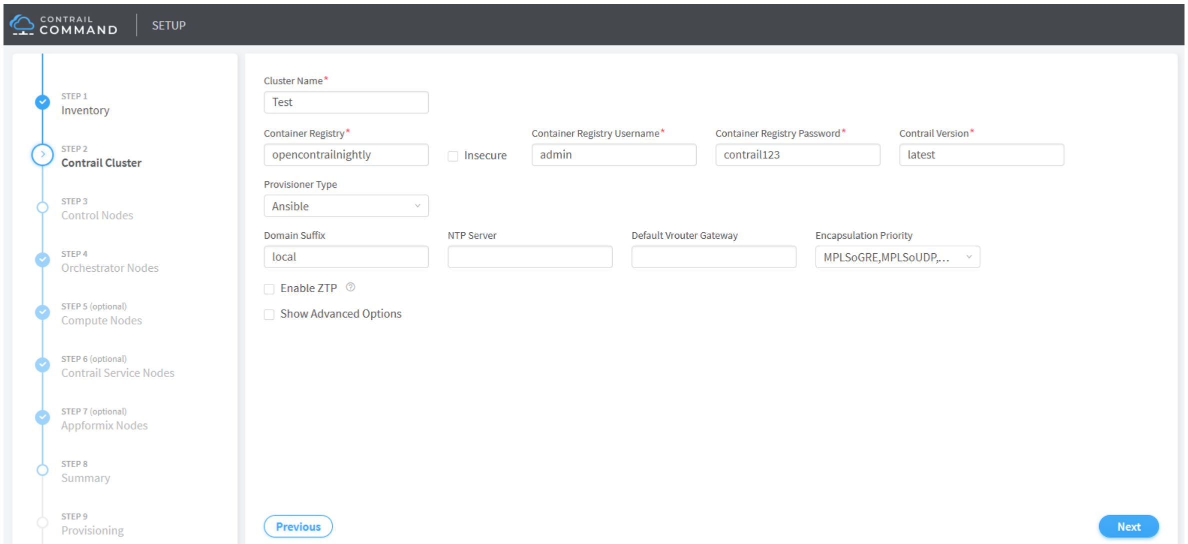
   If **Container registry** = hub.juniper.net/contrail . This registry is secure. Unselect the **Insecure** box. Also, **Contrail version** = *contrail_container_tag* for your release of Contrail as listed in README Access to Contrail Registry 19XX.

   Default vRouter Gateway = Default gateway for the compute nodes. If any one of the compute nodes has a different default gateway than the one provided here, enter that gateway in "5" on page 100 and "6" on page 101 for service nodes.

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

```
VXLAN, MPLSoUDP, MPLSoGRE
```

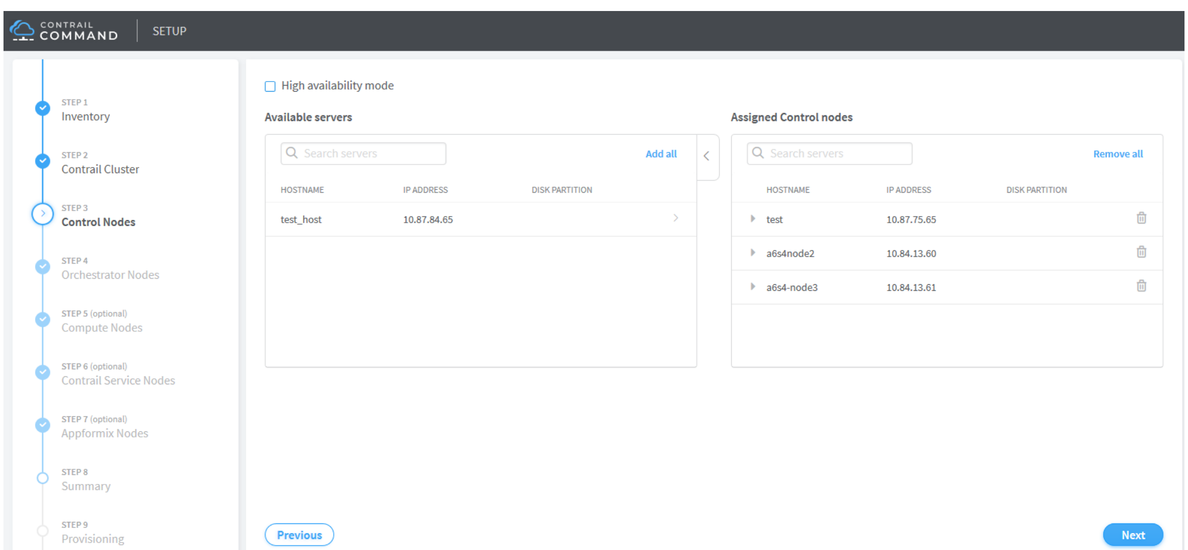
Figure 15: Contrail Cluster



Click **Next**. The **Control Nodes** page appears.

3. Select the Contrail control nodes.

Figure 16: Control Nodes



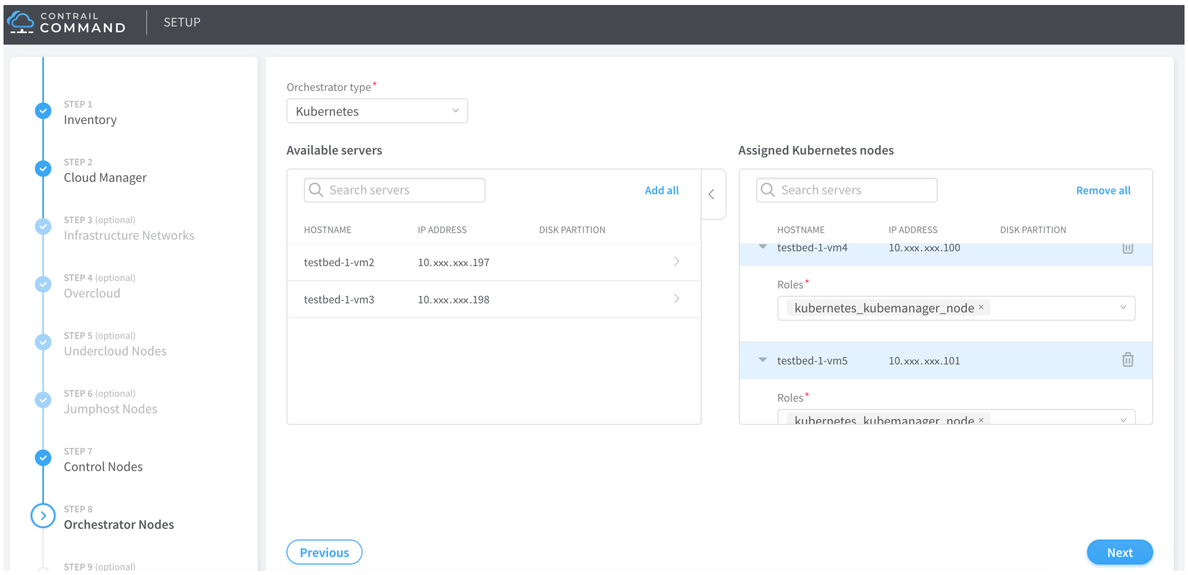
Click **Next**. The **Orchestrator Nodes** page appears.

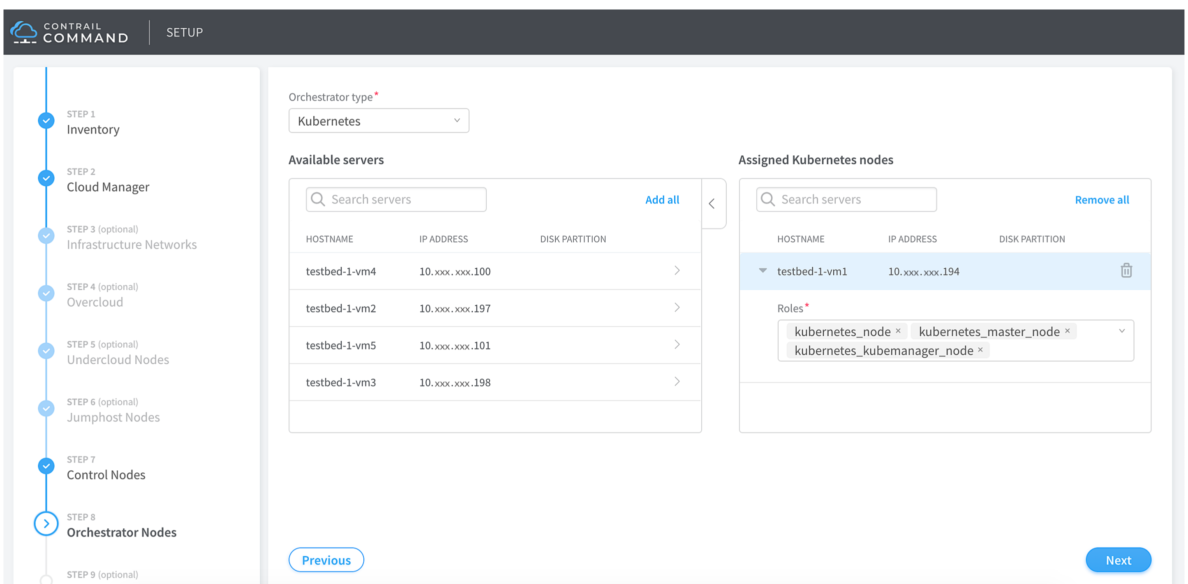
4. Select the Kubernetes orchestration type.

   Select the Kubernetes nodes from the list of available servers.

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.

Figure 17: Orchestrator Nodes

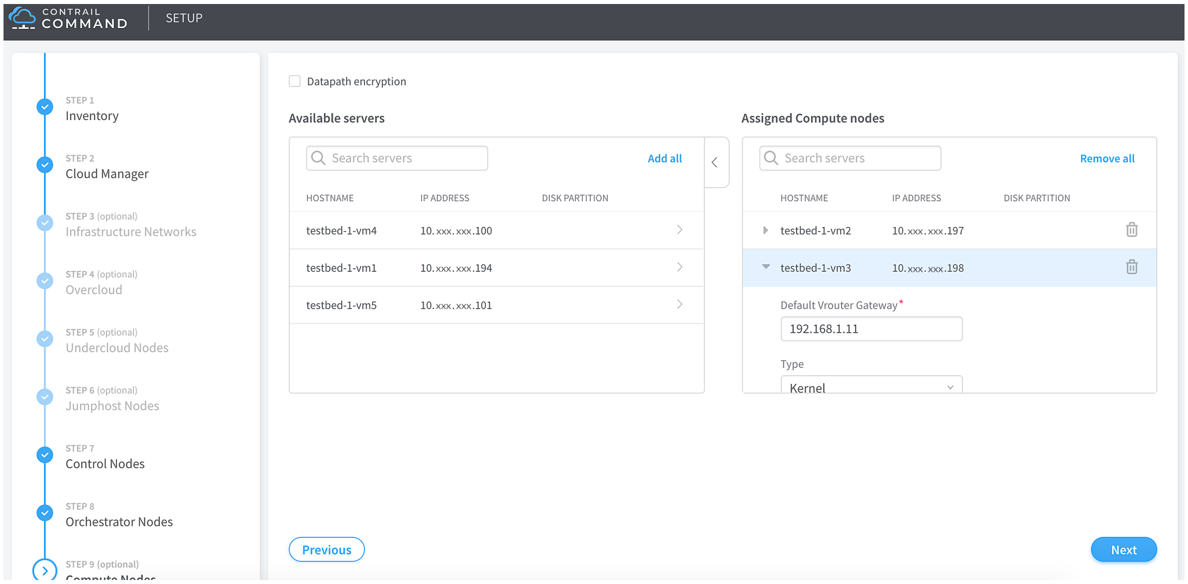




Click **Next**. The **Compute Nodes** page appears.

5. Select the compute node associated with the kunernetes_node role from the list of available servers, .

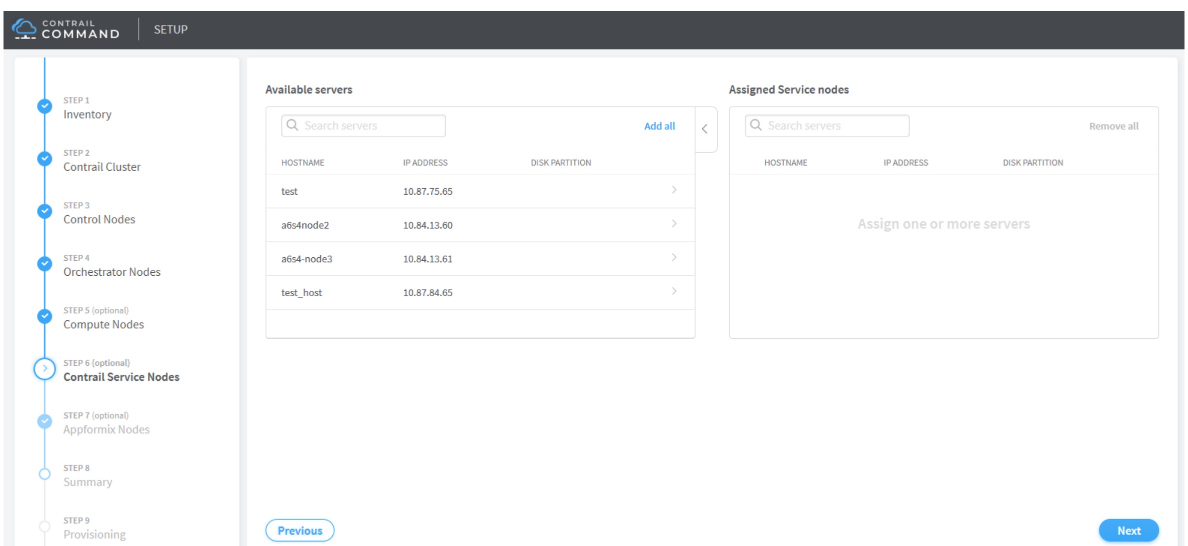
Figure 18: Compute Nodes



Click **Next**. The **Contrail Service Nodes** page appears.

6. (Optional) Select the Contrail service nodes from the list of available servers.

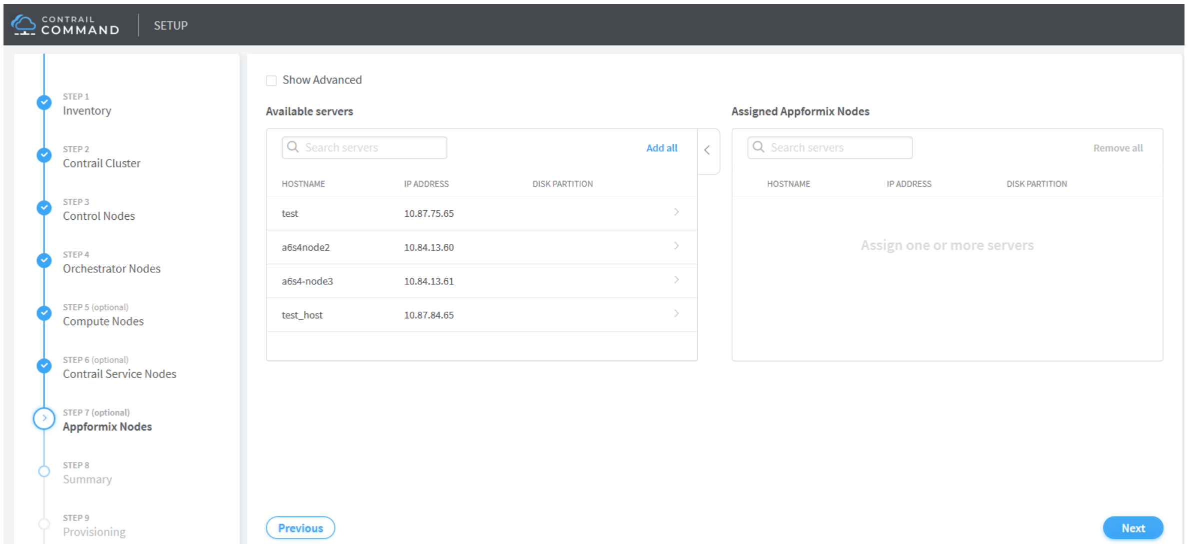
Figure 19: Contrail Service Nodes



Click **Next**. The **Appformix Nodes** page appears.

7. (Optional) Select the AppFormix nodes from the list of available nodes.

Figure 20: Appformix Nodes



Click **Next**. The **Summary** page appears.

8. The summary page displays the cluster details as well as the node details. Verify the summary of your cluster configuration and click **Provision**.

Figure 21: Summary - Cluster Overview

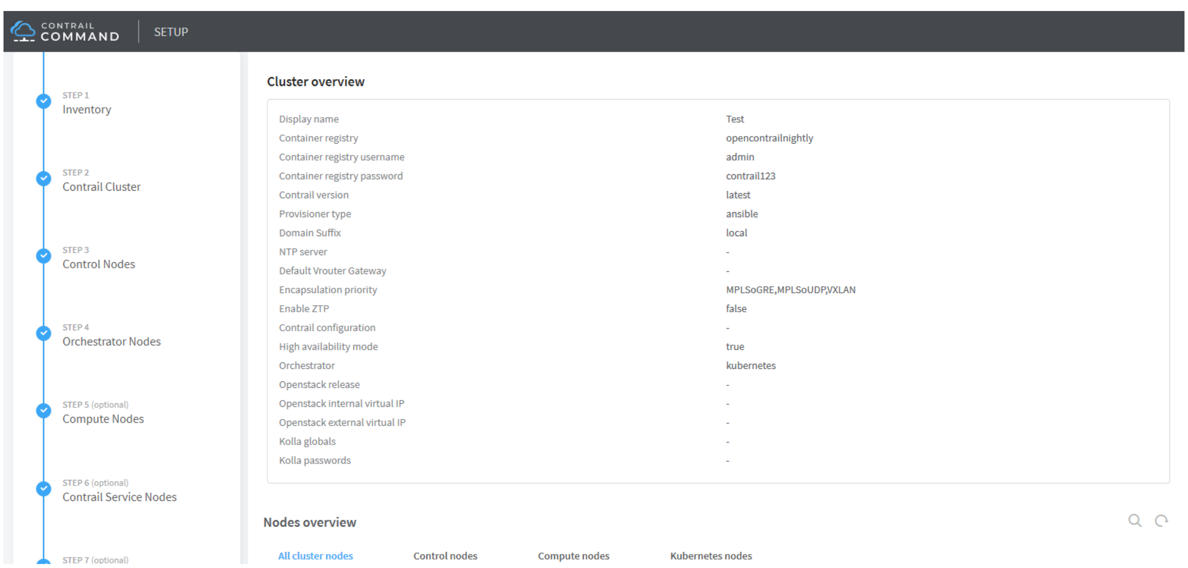
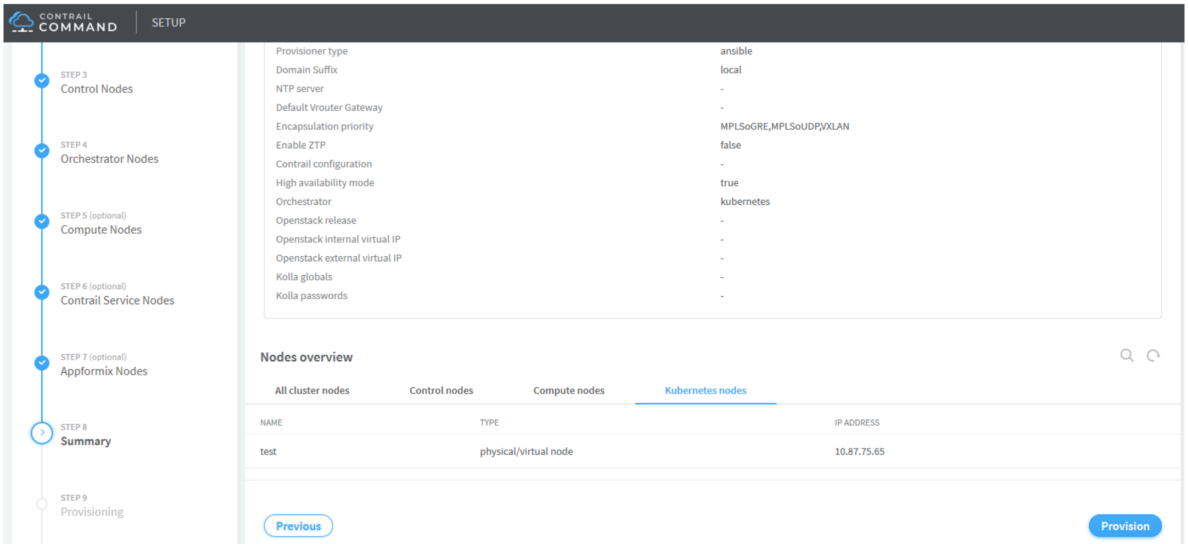


Figure 22: Summary - Nodes Overview



Sample command_servers.yml File

RELATED DOCUMENTATION

Installing Contrail Command | 18

Installing Contrail Cluster using Contrail Command and instances.yml | 42

Importing Contrail Cluster Data using Contrail Command | 47

Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability

IN THIS SECTION

NOTE: Starting in Contrail Release 1912.L1, Helm support is unavailable in Contrail Networking. The Helm support content in this document supports Contrail Networking Releases 1907 through 1912.

This is the installation procedure for using Helm charts to provision a multinode Contrail system with OpenStack Ocata and high availability.

NOTE: Contrail Command is not supported for Helm deployed Contrail clusters.

System Specifications

This procedure uses Juniper OpenStack Helm infrastructure and the OpenStack Helm repository to provision an OpenStack Ocata Contrail multinode deployment.

This procedure is tested with:

- Operating system: Ubuntu 17.03.2 LTS
- Kernel: 4.4.0-112-generic
- Docker: 1.13.1-cs9
- Helm: v2.7.2
- Kubernetes: v1.9.3
- OpenStack: Ocata

For a list of supported platforms, see https://www.juniper.net/documentation/en_US/release-independent/contrail/topics/reference/contrail-supported-platforms.pdf.

Preparing to Install

This section is the prerequisites needed to prepare your system before provisioning multinode Contrail with OpenStack Ocata and high availability.

1. Generate SSH key on master node and copy to all nodes, in below example three nodes with IP addresses 10.13.82.43, 10.13.82.44, and 10.13.82.45 are used.

```
(k8s-master)> ssh-keygen

(k8s-master)> ssh-copy-id -i ~/.ssh/id_rsa.pub 10.13.82.43
(k8s-master)> ssh-copy-id -i ~/.ssh/id_rsa.pub 10.13.82.44
(k8s-master)> ssh-copy-id -i ~/.ssh/id_rsa.pub 10.13.82.45
```

2. Make sure NTP is configured in all nodes and each node is synched to the time-server in your environment. In below example the NTP server IP is "10.84.5.100".

```
(k8s-all-nodes)> ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*10.84.5.100     66.129.255.62    2 u   15   64  377   72.421  -22.686   2.628
```

3. Get the contrail-helm-deployer.

   From Juniper Networks, download `contrail-helm-deployer-5.1.0-0.38.tgz` onto your provisioning host.

   - `scp contrail-helm-deployer-5.1.0-0.38.tgz` to all nodes on your cluster.
   - Untar contrail-helm-deployer-5.1.0-0.38.tgz on all nodes.

     ```
     tar -zxf contrail-helm-deployer-5.1.0-0.38.tgz -C /opt/
     ```

4. Export required variables.

```
(k8s-master)> cd /opt
(k8s-master)> export BASE_DIR=$(pwd)
(k8s-master)> export OSH_PATH=${BASE_DIR}/openstack-helm
(k8s-master)> export OSH_INFRA_PATH=${BASE_DIR}/openstack-helm-infra
(k8s-master)> export CHD_PATH=${BASE_DIR}/contrail-helm-deployer
```

5. Install necessary packages and deploy Kubernetes.

NOTE: If you want to install a different version of Kubernetes, CNI, or Calico, edit `${OSH_INFRA_PATH}/tools/gate/devel/local-vars.yaml` to override the default values in `${OSH_INFRA_PATH}/tools/gate/playbooks/vars.yaml`.

```
(k8s-master)> cd ${OSH_PATH}
(k8s-master)> ./tools/deployment/developer/common/001-install-packages-opencontrail.sh
```

6. Create an inventory file on the master node for Ansible base provisioning. In the following output, 10.13.82.43/.44/.45 are the IP addresses of the nodes, and will use the SSK-key generated in step "1" on page 105.

```
#!/bin/bash
(k8s-master)> set -xe
(k8s-master)> cat > /opt/openstack-helm-infra/tools/gate/devel/multinode-inventory.yaml <<EOF
all:
 children:
   primary:
     hosts:
       node_one:
         ansible_port: 22
         ansible_host: 10.13.82.43
         ansible_user: root
         ansible_ssh_private_key_file: /root/.ssh/id_rsa
         ansible_ssh_extra_args: -o StrictHostKeyChecking=no
   nodes:
     hosts:
       node_two:
         ansible_port: 22
         ansible_host: 10.13.82.44
         ansible_user: root
         ansible_ssh_private_key_file: /root/.ssh/id_rsa
         ansible_ssh_extra_args: -o StrictHostKeyChecking=no
       node_three:
         ansible_port: 22
         ansible_host: 10.13.82.45
         ansible_user: root
         ansible_ssh_private_key_file: /root/.ssh/id_rsa
```

```
        ansible_ssh_extra_args: -o StrictHostKeyChecking=no
EOF
```

7. Create an environment file on the master node for the cluster.

> **NOTE**: By default. Kubernetes v1.9.3, Helm v2.7.2, and CNI v0.6.0 are installed. If you want to install a different version, edit the `${OSH_INFRA_PATH}/tools/gate/devel/multinode-vars.yaml` file to override the values given in `${OSH_INFRA_PATH}/playbooks/vars.yaml`.

Sample `multinode-vars.yaml` :

```
(k8s-master)> cat > /opt/openstack-helm-infra/tools/gate/devel/multinode-vars.yaml <<EOF
# version fields
version:
 kubernetes: v1.9.3
 helm: v2.7.2
 cni: v0.6.0

kubernetes:
 network:
   # enp0s8 is your control/data interface, to which kubernetes will bind to
   default_device: enp0s8
 cluster:
   cni: calico
   pod_subnet: 192.168.0.0/16
   domain: cluster.local
docker:
 # list of insecure_registries, from where you will be pulling container images
 insecure_registries:
   - "10.87.65.243:5000"
 # list of private secure docker registry auth info, from where you will be pulling container
images
 #private_registries:
 #  - name: <docker-registry-name>
 #    username: username@abc.xyz
 #    email: username@abc.xyz
 #    password: password
 #    secret_name: contrail-image-secret
 #    namespace: openstack
EOF
```

8. Run playbooks on the master node.

```
(k8s-master)> set -xe
(k8s-master)> cd ${OSH_INFRA_PATH}
(k8s-master)> make dev-deploy setup-host multinode
(k8s-master)> make dev-deploy k8s multinode
```

9. Verify the `kube-dns` connection from all nodes. Use `nslookup` to verify that you are able to resolve Kubernetes cluster-specific names.

```
(k8s-all-nodes)> nslookup
 > kubernetes.default.svc.cluster.local
 Server:         10.96.0.10
 Address:        10.96.0.10#53

 Non-authoritative answer:
 Name:   kubernetes.default.svc.cluster.local
 Address: 10.96.0.1
```

Installation of OpenStack Helm Charts

Use this procedure to install the OpenStack Helm charts.

1. Before installing the OpenStack Helm charts, review the default labels for the nodes.

   The default nodes have the labels `openstack-control-plane` and `openstack-compute-node`.The default configuration creates OpenStack Helm (OSH) pods on all the nodes. Use the following commands to check the default OpenStack labels.

```
(k8s-master)>  kubectl get nodes -o wide -l openstack-control-plane=enabled
(k8s-master)> kubectl get nodes -o wide -l openstack-compute-node=enabled
```

   If you need to restrict the creation of OSH pods on specific nodes, disable the OpenStack labels. The following example shows how to disable the `openstack-compute-node` label on the `ubuntu-contrail-9` node.

```
(k8s-master)> kubectl label node ubuntu-contrail-9 --overwrite openstack-compute-node=disabled
```

2. Deploy OpenStack Helm charts.

```
(k8s-master)> set -xe
  (k8s-master)> cd ${OSH_PATH}

  (k8s-master)> ./tools/deployment/multinode/010-setup-client.sh
  (k8s-master)> ./tools/deployment/multinode/021-ingress-opencontrail.sh
  (k8s-master)> ./tools/deployment/multinode/030-ceph.sh
  (k8s-master)> ./tools/deployment/multinode/040-ceph-ns-activate.sh
  (k8s-master)> ./tools/deployment/multinode/050-mariadb.sh
  (k8s-master)> ./tools/deployment/multinode/060-rabbitmq.sh
  (k8s-master)> ./tools/deployment/multinode/070-memcached.sh
  (k8s-master)> ./tools/deployment/multinode/080-keystone.sh
  (k8s-master)> ./tools/deployment/multinode/090-ceph-radosgateway.sh
  (k8s-master)> ./tools/deployment/multinode/100-glance.sh
  (k8s-master)> ./tools/deployment/multinode/110-cinder.sh
  (k8s-master)> ./tools/deployment/multinode/131-libvirt-opencontrail.sh
  # Edit ${OSH_PATH}/tools/overrides/backends/opencontrail/nova.yaml and
  # ${OSH_PATH}/tools/overrides/backends/opencontrail/neutron.yaml
  # to make sure that you are pulling init container image from correct registry and tag
  (k8s-master)> ./tools/deployment/multinode/141-compute-kit-opencontrail.sh
  (k8s-master)> ./tools/deployment/developer/ceph/100-horizon.sh
```

Installation of Contrail Helm Charts

Use this procedure to install the Contrail Helm charts.

1. Label the Contrail pods. All Contrail pods are to be deployed in the namespace `contrail`, using the following labels:

   - Controller components—config, control, analytics
   - vRouter kernel—opencontrail.org/vrouter-kernel
   - vRouter DPDK—opencontrail.org/vrouter-dpdk

   The following example shows how to label `ubuntu-contrail-11` as DPDK and label `ubuntu-contrail-10`as kernel vrouter.

```
(k8s-master)> kubectl label node  ubuntu-contrail-11 opencontrail.org/vrouter-dpdk=enabled
(k8s-master)> kubectl label node ubuntu-contrail-10 opencontrail.org/vrouter-kernel=enabled
```

```
(k8s-master)> kubectl label nodes ubuntu-contrail-9 ubuntu-contrail-10 ubuntu-contrail-11
opencontrail.org/controller=enabled
```

2. Create Kubernetes ClusterRoleBinding for Contrail.

```
(k8s-master)> cd $CHD_PATH
(k8s-master)> kubectl replace -f ${CHD_PATH}/rbac/cluster-admin.yaml
```

3. Set up the Contrail Helm charts and set the configuration settings specific to your system in the values.yaml file for each of the charts.

```
(k8s-master)> cd $CHD_PATH
(k8s-master)> make

# Please note in below example, 192.168.1.0/24 is "Control/Data" network
# Export variables
(k8s-master)> export CONTROLLER_NODES="192.168.1.43,192.168.1.44,192.168.1.45"
(k8s-master)> export VROUTER_GATEWAY="192.168.1.1"
(k8s-master)> export CONTROL_DATA_NET_LIST="192.168.1.0/24"
(k8s-master)> export BGP_PORT="1179"

# [Optional] By default, it will pull latest image from opencontrailnightly

(k8s-master)> export CONTRAIL_REGISTRY="opencontrailnightly"
(k8s-master)> export CONTRAIL_TAG="latest"

# [Optional] only if you are pulling images from a private docker registry
export CONTRAIL_REG_USERNAME="abc@abc.com"
export CONTRAIL_REG_PASSWORD="password"

tee /tmp/contrail-env-images.yaml << EOF
global:
  contrail_env:
    CONTROLLER_NODES: ${CONTROLLER_NODES}
    CONTROL_NODES: ${CONTROL_NODES:-CONTROLLER_NODES}
    LOG_LEVEL: SYS_NOTICE
    CLOUD_ORCHESTRATOR: openstack
    AAA_MODE: cloud-admin
    VROUTER_GATEWAY: ${VROUTER_GATEWAY}
    BGP_PORT: ${BGP_PORT}
  contrail_env_vrouter_kernel:
```

```
      CONTROL_DATA_NET_LIST: ${CONTROL_DATA_NET_LIST}
      AGENT_MODE: nic
    contrail_env_vrouter_dpdk:
      AGENT_MODE: dpdk
    images:
      tags:
        kafka: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-external-kafka:$
{CONTRAIL_TAG:-latest}"
        cassandra: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-external-cassandra:$
{CONTRAIL_TAG:-latest}"
        redis: "redis:4.0.2"
        zookeeper: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-external-zookeeper:$
{CONTRAIL_TAG:-latest}"
        contrail_control: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-
control-control:${CONTRAIL_TAG:-latest}"
        control_dns: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-control-
dns:${CONTRAIL_TAG:-latest}"
        control_named: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-control-
named:${CONTRAIL_TAG:-latest}"
        config_api: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-config-api:$
{CONTRAIL_TAG:-latest}"
        config_devicemgr: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-
config-devicemgr:${CONTRAIL_TAG:-latest}"
        config_schema_transformer: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-
controller-config-schema:${CONTRAIL_TAG:-latest}"
        config_svcmonitor: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-
config-svcmonitor:${CONTRAIL_TAG:-latest}"
        webui_middleware: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-webui-
job:${CONTRAIL_TAG:-latest}"
        webui: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-webui-web:$
{CONTRAIL_TAG:-latest}"
        analytics_api: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-api:$
{CONTRAIL_TAG:-latest}"
        contrail_collector: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-
collector:${CONTRAIL_TAG:-latest}"
        analytics_alarm_gen: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-
alarm-gen:${CONTRAIL_TAG:-latest}"
        analytics_query_engine: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-
query-engine:${CONTRAIL_TAG:-latest}"
        analytics_snmp_collector: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-
analytics-snmp-collector:${CONTRAIL_TAG:-latest}"
        contrail_topology: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-snmp-
topology:${CONTRAIL_TAG:-latest}"
```

```
        build_driver_init: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-kernel-
build-init:${CONTRAIL_TAG:-latest}"
        vrouter_agent: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-agent:$
{CONTRAIL_TAG:-latest}"
        vrouter_init_kernel: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-
kernel-init:${CONTRAIL_TAG:-latest}"
        vrouter_dpdk: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-agent-dpdk:$
{CONTRAIL_TAG:-latest}"
        vrouter_init_dpdk: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-kernel-
init-dpdk:${CONTRAIL_TAG:-latest}"
        nodemgr: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-nodemgr:${CONTRAIL_TAG:-
latest}"
        contrail_status: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-status:$
{CONTRAIL_TAG:-latest}"
        node_init: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-node-init:$
{CONTRAIL_TAG:-latest}"
        dep_check: quay.io/stackanetes/kubernetes-entrypoint:v0.2.1
EOF
```

NOTE: If any other environment variables need to be added, add them in the values.yaml file of the respective charts.

```
# [Optional] only if you are pulling contrail images from a private registry
tee /tmp/contrail-registry-auth.yaml << EOF
global:
  images:
    imageCredentials:
      registry: ${CONTRAIL_REGISTRY:-opencontrailnightly}
      username: ${CONTRAIL_REG_USERNAME}
      password: ${CONTRAIL_REG_PASSWORD}
EOF

# [Optional] only if you are pulling images from a private registry
export CONTRAIL_REGISTRY_ARG="--values=/tmp/contrail-registry-auth.yaml "
```

4. Use Helm install commands to deploy each of the Contrail Helm charts.

```
 (k8s-master)> helm install --name contrail-thirdparty ${CHD_PATH}/contrail-thirdparty \
  --namespace=contrail \
```

```
  --values=/tmp/contrail-env-images.yaml \
  ${CONTRAIL_REGISTRY_ARG}

  (k8s-master)> helm install --name contrail-controller ${CHD_PATH}/contrail-controller \
  --namespace=contrail \
  --values=/tmp/contrail-env-images.yaml \
  ${CONTRAIL_REGISTRY_ARG}

  (k8s-master)> helm install --name contrail-analytics ${CHD_PATH}/contrail-analytics \
  --namespace=contrail \
  --values=/tmp/contrail-env-images.yaml \
  ${CONTRAIL_REGISTRY_ARG}

  # Edit contrail-vrouter/values.yaml and make sure that
global.images.tags.vrouter_init_kernel is right. Image tag name will be different depending
upon your linux. Also set the global.node.host_os to ubuntu or centos depending on your system

  (k8s-master)> helm install --name contrail-vrouter ${CHD_PATH}/contrail-vrouter \
  --namespace=contrail \
  --values=/tmp/contrail-env-images.yaml \
  ${CONTRAIL_REGISTRY_ARG}
```

5. When the Contrail pods are up and running, deploy the OpenStack Heat chart.

```
# Edit ${OSH_PATH}/tools/overrides/backends/opencontrail/nova.yaml and
# ${OSH_PATH}/tools/overrides/backends/opencontrail/heat.yaml
# to make sure that you are pulling the right opencontrail init container image
(k8s-master)> ./tools/deployment/multinode/151-heat-opencontrail.sh
```

6. When finished, run the compute kit test.

```
(k8s-master)> ./tools/deployment/multinode/143-compute-kit-opencontrail-test.sh
```

Basic Testing OpenStack Helm Contrail Cluster

Use the following commands to perform basic testing on the virtual network and the virtual machines in your OpenStack Helm Contrail cluster.

```
(k8s-master)> export OS_CLOUD=openstack_helm

(k8s-master)> openstack network create MGMT-VN
```

```
(k8s-master)> openstack subnet create --subnet-range 172.16.1.0/24 --network MGMT-VN MGMT-VN-
subnet

(k8s-master)> openstack server create --flavor m1.tiny --image 'Cirros 0.3.5 64-bit' \
--nic net-id=MGMT-VN \
Test-01

(k8s-master)> openstack server create --flavor m1.tiny --image 'Cirros 0.3.5 64-bit' \
--nic net-id=MGMT-VN \
Test-02
```

Accessing the Contrail OpenStack Helm Cluster

Use the following topic to access the OpenStack and Contrail Web UI and prepare the OpenStack client for command-line interface (CLI):

"Accessing a Contrail OpenStack Helm Cluster" on page 118

RELATED DOCUMENTATION

Installing and Managing Contrail Microservices Architecture Using Helm Charts | 85
Frequently Asked Questions About Contrail and Helm Charts | 121
Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata | 114
Accessing a Contrail OpenStack Helm Cluster | 118

Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata

IN THIS SECTION

NOTE: Starting in Contrail Release 1912.L1, Helm support is unavailable in Contrail Networking. The Helm support content in this document supports Contrail Networking Releases 1907 through 1912.

This is the installation procedure for using Helm charts to provision an all-in-one Contrail system with OpenStack Ocata. This is not a high availability configuration.

NOTE: All-in-one systems are only used for testing or for demonstration purposes.

System Specifications

This procedure uses Helm to provision an OpenStack Ocata Contrail all-in-one cluster without high availability.

This procedure is tested with:

- Operating system: Ubuntu 16.04.3 LTS
- Kernel: 4.4.0-87-generic
- Docker: 1.13.1-cs9
- Helm: v2.7.2
- Kubernetes: v1.8.3
- OpenStack: Ocata

This setup was tested on a system with the following specifications:

- CPU: 8
- RAM: 32 GB
- HDD: 120 GB

Installation Steps

1. Get the contrail-helm-deployer.

   From Juniper Networks, download `contrail-helm-deployer-5.1.0-0.38.tgz` onto your provisioning host.

- Untar contrail-helm-deployer-5.1.0-0.38.tgz.

  ```
  tar -zxf contrail-helm-deployer-5.1.0-0.38.tgz -C /opt/
  ```

2. Export required variables.

```
export BASE_DIR=$(pwd)
export OSH_PATH=${BASE_DIR}/openstack-helm
export OSH_INFRA_PATH=${BASE_DIR}/openstack-helm-infra
export CHD_PATH=${BASE_DIR}/contrail-helm-deployerExport variables
```

3. Install necessary packages and deploy Kubernetes.

NOTE: If you want to install a different version of Kubernetes, CNI, or Calico, edit `${OSH_INFRA_PATH}/tools/gate/devel/local-vars.yaml` to override the default values in `${OSH_INFRA_PATH}/tools/gate/playbooks/vars.yaml`.

```
cd ${OSH_PATH}
./tools/deployment/developer/common/001-install-packages-opencontrail.sh
./tools/deployment/developer/common/010-deploy-k8s.sh
```

4. Install OpenStack and the Heat client.

```
./tools/deployment/developer/common/020-setup-client.sh
```

5. Deploy OpenStack Helm-related charts.

```
./tools/deployment/developer/nfs/031-ingress-opencontrail.sh
./tools/deployment/developer/nfs/040-nfs-provisioner.sh
./tools/deployment/developer/nfs/050-mariadb.sh
./tools/deployment/developer/nfs/060-rabbitmq.sh
./tools/deployment/developer/nfs/070-memcached.sh
./tools/deployment/developer/nfs/080-keystone.sh
./tools/deployment/developer/nfs/100-horizon.sh
./tools/deployment/developer/nfs/120-glance.sh
./tools/deployment/developer/nfs/151-libvirt-opencontrail.sh
./tools/deployment/developer/nfs/161-compute-kit-opencontrail.sh
```

6. Deploy Contrail Helm charts.

```
cd $CHD_PATH

make

# Set the IP of your CONTROL_NODES (specify your control data ip, if you have one)
export CONTROL_NODES=10.87.65.245
# set the control data network cidr list separated by comma and set the respective gateway
export CONTROL_DATA_NET_LIST=10.87.65.128/25
export VROUTER_GATEWAY=10.87.65.129

kubectl label node opencontrail.org/controller=enabled --all
kubectl label node opencontrail.org/vrouter-kernel=enabled --all

kubectl replace -f ${CHD_PATH}/rbac/cluster-admin.yaml

tee /tmp/contrail.yaml << EOF
global:
  contrail_env:
    CONTROLLER_NODES: 172.17.0.1
    CONTROL_NODES: ${CONTROL_NODES}
    LOG_LEVEL: SYS_NOTICE
    CLOUD_ORCHESTRATOR: openstack
    AAA_MODE: cloud-admin
    CONTROL_DATA_NET_LIST: ${CONTROL_DATA_NET_LIST}
    VROUTER_GATEWAY: ${VROUTER_GATEWAY}
EOF

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

7. Deploy Heat charts.

```
cd ${OSH_PATH}
./tools/deployment/developer/nfs/091-heat-opencontrail.sh
```

Accessing the Contrail OpenStack Helm Cluster

Use the following topic to access the OpenStack and Contrail Web UI and prepare the OpenStack client for command-line interface (CLI):

"Accessing a Contrail OpenStack Helm Cluster" on page 118

RELATED DOCUMENTATION

Accessing a Contrail OpenStack Helm Cluster

IN THIS SECTION

NOTE: Starting in Contrail Release 1912.L1, Helm support is unavailable in Contrail Networking. The Helm support content in this document supports Contrail Networking Releases 1907 through 1912.

When the provisioning of Contrail with Helm charts is completed, use this topic to access the OpenStack and Contrail Web UI and prepare the OpenStack client for command-line interface (CLI).

Overview

This topic assumes you have already installed Contrail and OpenStack using Helm charts, typically by using these procedures:

- "Installing and Managing Contrail Microservices Architecture Using Helm Charts" on page 85
- "Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability " on page 103
- "Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata " on page 114
- "Frequently Asked Questions About Contrail and Helm Charts" on page 121

Installing the OpenStack Client

Use this procedure to install the OpenStack CLI tool.

1. Install the OpenStack client CLI tool on the primary Ubuntu host.

```
apt install python-dev python-pip -y
pip install --upgrade pip
pip install python-openstackclient    OR
apt-get install python-openstackclient
```

2. If you have problems installing the python-dev package, add another repository.

```
Add following repo to source "/etc/apt/sources.list"
deb http://archive.ubuntu.com/ubuntu/ xenial-updates main universe multiverse
apt-get update
apt-get install python-dev
```

Create openstackrc File and Test OpenStack Client

1. Create an openstackrc file.

```
cat > /root/openstackrc << EOF
export OS_USERNAME=admin
export OS_PASSWORD=password
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://keystone-api.openstack:35357/v3
```

```
# The following lines can be omitted
#export OS_TENANT_ID=tenantIDString
#export OS_REGION_NAME=regionName
export OS_IDENTITY_API_VERSION=3
export OS_USER_DOMAIN_NAME=${OS_USER_DOMAIN_NAME:-"Default"}
export OS_PROJECT_DOMAIN_NAME=${OS_PROJECT_DOMAIN_NAME:-"Default"}
EOF
```

2. Test the OpenStack client.

```
source openstackrc
openstack server list
openstack stack list
openstack --help
```

Accessing the Contrail Web UI

1. Access the Contrail Web UI using port 8143. Use the IP address of the host where the contrail-webui pod is running, with the port 8143.

```
https://<IP address host with contrail-webui>:8143
```

2. At the Contrail login screen, enter the default username and password: admin, password.

Accessing OpenStack Horizon

The OpenStack Web UI (GUI) service is exposed by the Kubernetes service, using the IP address of the node port and the default port 31000.

1. Check the NodePort used for the OpenStack Web UI pod.

```
kubectl get svc -n openstack | grep horizon-int
horizon-int           NodePort    10.99.150.28     <none>        80:31000/TCP         4d
```

2. Access the OpenStack Web UI and log in with the default username and password: admin, password.

```
http://<IP address NodePort>:31000/auth/login/?next=/
```

Accessing the Virtual Machine Console from Horizon

To access the virtual machine (VM) console, add the nova novncproxy fully-qualified domain name (FQDN) in the /etc/hosts file, using the host-ip where the osh-ingress pod is running.

The following example for MAC-OS shows the ingress pod running on the host with IP address 10.13.82.233.

```
/private/etc/
hosts

127.0.0.1    localhost
255.255.255.255    broadcasthost
::1             localhost
10.13.82.233 nova-novncproxy.openstack.svc.cluster.local
```

NOTE: If you don't want to make changes in /etc/hosts, you can replace the nova-novncproxy.openstack.svc.cluster.local portion in the URL with the IP address where the OSH ingress pod is running.

OpenStack References

For more information about accessing and using OpenStack, see the following OpenStack resources:

- *Create OpenStack client environment scripts*
- *Install the OpenStack command-line clients*
- *External DNS to FQDN/Ingress*

Frequently Asked Questions About Contrail and Helm Charts

IN THIS SECTION

NOTE: Starting in Contrail Release 1912.L1, Helm support is unavailable in Contrail Networking. The Helm support content in this document supports Contrail Networking Releases 1907 through 1912.

This topic presents frequently asked questions and answers about Contrail and Helm Charts.

How do I set up the vhost0 interface for the vrouter on the non-management interface of the compute node?

NOTE: Some Contrail versions assume a single name for all of the non-management interfaces in your cluster.

If your non-management interface is `eth1`, in the `contrail-vrouter/values.yaml` set the `contrail_env.PHYSICAL_INTERFACE` to `eth1` and set the `contrail_env.VROUTER_GATEWAY` to the IP address of the non-management gateway.

```
# Sample config
contrail_env:
  CONTROLLER_NODES: 1.1.1.10
  LOG_LEVEL: SYS_NOTICE
  CLOUD_ORCHESTRATOR: openstack
  AAA_MODE: cloud-admin
```

```
  PHYSICAL_INTERFACE: eth1
  VROUTER_GATEWAY: 1.1.1.1
```

How do I configure the Contrail control BGP server to listen on a different port?

To configure a non-default BGP port, in the `contrail-controller/values.yaml` set the `contrail_env.BGP` to the desired port.

```
# Sample config
contrail_env:
  CONTROLLER_NODES: 1.1.1.10
  LOG_LEVEL: SYS_NOTICE
  CLOUD_ORCHESTRATOR: openstack
  AAA_MODE: cloud-admin
  BGP_PORT: 1179
```

How can I pass additional parameters to services in Contrail by using the configuration file in INI format?

The following example configures the minimum_diskGB parameter for the node manager of the analytics database.

```
# Sample config
contrail_env:
  DATABASE_NODEMGR__DEFAULTS__minimum_diskGB: "2"
```

How do I configure services for the vrouter agent?

The following is an example configuration for the vrouter agent.

```
# Sample config
contrail_env:
  VROUTER_AGENT__FLOWS__thread_count: "2"
  VROUTER_AGENT__METADATA__metadata_use_ssl = True
  VROUTER_AGENT__METADATA__metadata_client_cert = /usr/share/ca-certificates/contrail/
client_cert.pem
  VROUTER_AGENT__METADATA__metadata_client_key = /usr/share/ca-certificates/contrail/
```

```
client_key.pem
  VROUTER_AGENT__METADATA__metadata_ca_cert = /usr/share/ca-certificates/contrail/cacert.pem
```

What are the Contrail services that can be configured?

Configurable services at this time include the following:

- Configurable services for config node:
  - SVC_MONITOR
  - API
  - DEVICE_MANAGER
  - SCHEMA
  - CONFIG_NODEMGR
- Configurable services for control:
  - CONTROL
  - DNS
  - CONTROL_NODEMGR
- Configurable services for analytics:
  - ALARM_GEN
  - TOPOLOGY
  - ANALYTICS_API
  - COLLECTOR
  - SNMP_COLLECTOR
  - QUERY_ENGINE
  - ANALYTICS_NODEMGR
  - Configurable services for database:
    - DATABASE_NODEMGR
  - Configurable services for vrouter:
    - VROUTER_AGENT

- VROUTER_AGENT_NODEMGR

How can I pass additional parameters to the Contrail Web UI services a with configuration file in JS format?

Define the exact variable in the environment. Available configuration settings can be found in the source code, see https://github.com/Juniper/contrail-container-builder/blob/master/containers/controller/webui/base/entrypoint.sh#L31-L199 .

```
# Sample config
contrail_env:
  WEBUI_SSL_CIPHERS: "ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384"
```

How can I verify all pods of Contrail are up and running?

Use the following command to list all pods of Contrail.

```
kubectl get pods -n openstack -o wide | grep contrail-
```

How can I see the logs of each of the containers?

Contrail logs are stored under /var/log/contrail/ on each node. To check for the standard output (stdout) log for each container:

```
kubectl logs -f <contrail-pod-name> -n openstack
```

How can I enter into a pod?

Use the kubectl command.

```
kubectl exec -it <contrail-pod> -n openstack -- bash
```

Installing Contrail Networking for Kubernetes using Helm

NOTE: Starting in Contrail Release 1912.L1, Helm support is unavailable in Contrail Networking. The Helm support content in this document supports Contrail Networking Releases 1907 through 1912.

This procedure describes how to deploy Contrail with Helm charts, but without OpenStack.

NOTE: Nodes should be configured so the master can `ssh` into Minion. If ssh keys are needed, these should be specified in the inventory file.

Follow these steps to deploy Contrail with Helm:

1. Download the file **contrail-helm-deployer-*release-tag*.tgz** onto your provisioning host. It contains the required two required Helm repositories: **/opt/openstack-helm-infra** (which contains code to deploy k8s) and **/opt/contrail-helm-deployer**.

2. Run the command `scp contrail-helm-deployer-`*`release-tag`*`.tgz` for all nodes in the cluster.

3. Untar **contrail-helm-deployer-*release-tag*.tgz** on all nodes:

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

4. Using any node in the cluster, export the following variables:

   ```
   export BASE_DIR=/opt
   export OSH_INFRA_PATH=${BASE_DIR}/openstack-helm-infra
   export CHD_PATH=${BASE_DIR}/contrail-helm-deployer
   ```

5. In this step, all the required packages are installed and Kubernetes is deployed. If you want to install a different version of Kubernetes or CNI, edit the file **${OSH_INFRA_PATH}/tools/gate/devel/multinode-vars.yaml**. Doing this overrides the default values in **${OSH_INFRA_PATH}/playbooks/vars.yaml**. Following is an example **multinode-vars.yaml** file, with sample values indicated for the `private_registries` section:

   ```
   version:
     kubernetes: v1.9.3
   ```

```
  helm: v2.7.2
  cni: v0.6.0
docker:
  # list of insecure_registries, from where you will be pulling container images
  insecure_registries:
    - "10.87.65.243:5000"
  # list of private secure docker registry auth info, from where you will be pulling
container images
  #private_registries:
  #  - name: docker-registry-name
  #    username: username@abc.xyz
  #    email: username@abc.xyz
  #    password: password
  #    secret_name: contrail-image-secret
  #    namespace: openstack
kubernetes:
  network:
    default_device: ens3
  cluster:
    cni: calico
    pod_subnet: 192.168.0.0/16
    domain: cluster.local
```

6. Install the dependent packages using `sudo apt-get`.

```
sudo apt-get update
sudo apt-get install --no-install-recommends -y ca-certificates make jq nmap curl uuid-
runtime ipcalc linux-headers-$(uname -r)
```

7. Prepare the nodes definition in **$OSH_INFRA_PATH/tools/gate/devel/multinode-inventory.yaml**, similar to this example:

```
all:
  children:
    primary:
      hosts:
        controller1:
          ansible_port: 22
          ansible_host: 10.10.0.1
          ansible_user: root
```

```
      ansible_ssh_extra_args: -o StrictHostKeyChecking=no
      ansible_ssh_private_key_file: /path/to/ssh/key/file
nodes:
  hosts:
    controller2:
      ansible_port: 22
      ansible_host: 10.10.0.2
      ansible_user: root
      ansible_ssh_extra_args: -o StrictHostKeyChecking=no
      ansible_ssh_private_key_file: /path/to/ssh/key/file
```

8. Deploy k8s to the nodes and use the `kubectl get nodes` command to verify the deployment is successful.

```
cd ${OSH_INFRA_PATH}
make dev-deploy setup-host multinode
make dev-deploy k8s multinode

nslookup kubernetes.default.svc.cluster.local || /bin/true
kubectl get nodes -o wide
```

9. Set the correct labels for the nodes.

```
kubectl label node controller1.localdomain --overwrite openstack-compute-node=disable
kubectl label node controller1.localdomain opencontrail.org/controller=enabled
kubectl label node controller2.localdomain --overwrite openstack-compute-node=disable
kubectl label node controller2.localdomain opencontrail.org/controller=enabled
```

10. Deploy the OpenContrail charts.

```
cd $CHD_PATH
    make
# Change k8s rbac settings
    kubectl replace -f ${CHD_PATH}/rbac/cluster-admin.yaml
```

11. Prepare the values for Contrail in **/tmp/contrail.yml**, similar to the following example.

NOTE: This example uses bash variables you should replace with exact values using any preferred means (sed, eval, cat, and so on). Similarly, replace the other variables with actual values where indicated, including `IPDATA_SERVICE_HOST`, `METADATA_PROXY_SECRET`, and keystone IP/VIP details.

```
global:
  images:
    tags:
      kafka: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-external-kafka:$
{CONTRAIL_TAG:-latest}"
      cassandra: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-external-cassandra:$
{CONTRAIL_TAG:-latest}"
      redis: "redis:4.0.2"
      zookeeper: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-external-zookeeper:$
{CONTRAIL_TAG:-latest}"
      contrail_control: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-
control-control:${CONTRAIL_TAG:-latest}"
      control_dns: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-control-
dns:${CONTRAIL_TAG:-latest}"
      control_named: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-control-
named:${CONTRAIL_TAG:-latest}"
      config_api: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-config-
api:${CONTRAIL_TAG:-latest}"
      config_devicemgr: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-
config-devicemgr:${CONTRAIL_TAG:-latest}"
      config_schema_transformer: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-
controller-config-schema:${CONTRAIL_TAG:-latest}"
      config_svcmonitor: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-
config-svcmonitor:${CONTRAIL_TAG:-latest}"
      webui_middleware: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-
webui-job:${CONTRAIL_TAG:-latest}"
      webui: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-controller-webui-web:$
{CONTRAIL_TAG:-latest}"
      analytics_api: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-api:$
{CONTRAIL_TAG:-latest}"
      contrail_collector: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-
collector:${CONTRAIL_TAG:-latest}"
      analytics_alarm_gen: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-
alarm-gen:${CONTRAIL_TAG:-latest}"
      analytics_query_engine: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-
```

```
query-engine:${CONTRAIL_TAG:-latest}"
      analytics_snmp_collector: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-
analytics-snmp-collector:${CONTRAIL_TAG:-latest}"
      contrail_topology: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-analytics-
topology:${CONTRAIL_TAG:-latest}"
      build_driver_init: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-kernel-
build-init:${CONTRAIL_TAG:-latest}"
      vrouter_agent: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-agent:$
{CONTRAIL_TAG:-latest}"
      vrouter_init_kernel: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-
kernel-init:${CONTRAIL_TAG:-latest}"
      vrouter_dpdk: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-agent-dpdk:$
{CONTRAIL_TAG:-latest}"
      vrouter_init_dpdk: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-vrouter-kernel-
init-dpdk:${CONTRAIL_TAG:-latest}"
      nodemgr: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-nodemgr:${CONTRAIL_TAG:-
latest}"
      contrail_status: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-status:$
{CONTRAIL_TAG:-latest}"
      node_init: "${CONTRAIL_REGISTRY:-opencontrailnightly}/contrail-node-init:$
{CONTRAIL_TAG:-latest}"
      dep_check: quay.io/stackanetes/kubernetes-entrypoint:v0.2.1

  contrail_env:
    CONTROLLER_NODES: 10.10.0.1,10.10.0.2
    LOG_LEVEL: SYS_DEBUG
    CLOUD_ORCHESTRATOR: openstack
    JVM_EXTRA_OPTS: "-Xms1g -Xmx2g"
    BGP_PORT: "1179"
    CONFIG_DATABASE_NODEMGR__DEFAULTS__minimum_diskGB: "2"
    DATABASE_NODEMGR__DEFAULTS__minimum_diskGB: "2"
    IPFABRIC_SERVICE_HOST: metadata IP of old OpenStack setup
    METADATA_PROXY_SECRET: metadata proxy secret of old OpenStack setup
endpoints:
  keystone:
    auth:
      username: admin
      password: password
      project_name: admin
      user_domain_name: admin_domain
      project_domain_name: admin_domain
      region_name: RegionOne
    hosts:
```

```
      default: keystone IP/VIP
    path:
      default: /v3
    port:
      admin:
        default: 35357
      api:
        default: 5000
    scheme:
      default: http
    host_fqdn_override:
      default: keystone IP/VIP
    namespace: null
```

12. If you are using a private registry, add the username and password under the `imageCredentials` section as follows:

```
global:
  images:
    imageCredentials:
      registry: ${CONTRAIL_REGISTRY:-opencontrailnightly}
      username: ${CONTRAIL_REG_USERNAME}
      password: ${CONTRAIL_REG_PASSWORD}
```

13. Finally, deploy the Contrail charts:

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

After all containers are deployed, you can check cluster status using the `contrail-status` command. You can also use the Contrail web browser interface to view and verify the cluster status.

Verifying Configuration for CNI for Kubernetes

IN THIS SECTION

Use the verification steps in this topic to view and verify your configuration of Contrail Container Network Interface (CNI) for Kubernetes.

View Pod Name and IP Address

Use the following command to view the IP address allocated to a pod.

```
[root@device ~]# kubectl get pods --all-namespaces -o wide
NAMESPACE         NAME                 READY     STATUS     RESTARTS    AGE
IP                        NODE
default               client-1            1/1           Running       0
19d       10.47.25.247    k8s-minion-1-3
default               client-2            1/1          Running       0
19d       10.47.25.246    k8s-minion-1-1
default               client-x             1/1           Running       0
19d       10.84.21.272    k8s-minion-1-1
```

Verify Reachability of Pods

Perform the following steps to verify if the pods are reachable to each other.

1. Determine the IP address and name of the pod.

```
[root@device ~]# kubectl get pods --all-namespaces -o wide
NAME                           READY     STATUS     RESTARTS    AGE       IP                NODE
```

```
example1-36xpr   1/1       Running   0          43s       10.47.25.251   b3s37
example2-pldp1   1/1       Running   0          39s       10.47.25.250   b3s37
```

2. Ping the destination pod from the source pod to verify if the pod is reachable.

```
root@device ~]# kubectl exec -it example1-36xpr  ping 10.47.25.250
PING 10.47.25.250 (10.47.25.250): 56 data bytes
64 bytes from 10.47.25.250: icmp_seq=0 ttl=63 time=1.510 ms
64 bytes from 10.47.25.250: icmp_seq=1 ttl=63 time=0.094 ms
```

Verify If Isolated Namespace-Pods Are Not Reachable

Perform the following steps to verify if pods in isolated namespaces cannot be reached by pods in non-isolated namespaces.

1. Determine the IP address and name of a pod in an isolated namespace.

```
[root@device ~]# kubectl get pod -n test-isolated-ns -o wide
NAME                        READY     STATUS    RESTARTS   AGE       IP              NODE
example3-bvqx5   1/1       Running   0          1h        10.47.25.249   b3s37
```

2. Determine the IP address of a pod in a non-solated namespace.

```
[root@device ~]# kubectl get pods
NAME                        READY     STATUS    RESTARTS   AGE
example1-36xpr   1/1       Running   0          15h
example2-pldp1   1/1       Running   0          15h
```

3. Ping the IP address of the pod in the isolated namespace from the pod in the non-isolated namespace.

```
[root@device ~]# kubectl exec -it example1-36xpr ping 10.47.25.249
        --- 10.47.255.249 ping statistics ---
 2 packets transmitted, 0 packets received, 100% packet loss
```

Verify If Non-Isolated Namespace-Pods Are Reachable

Perform the following steps to verify if pods in non-isolated namespaces can be reached by pods in isolated namespaces.

1. Determine the IP address of a pod in a non-isolated namespace.

```
[root@device ~]# kubectl get pods -o wide
NAME                        READY     STATUS     RESTARTS   AGE       IP                NODE
example1-36xpr    1/1       Running    0          15h       10.47.25.251    b3s37
example2-pldp1    1/1       Running    0          15h       10.47.25.250    b3s37
```

2. Determine the IP address and name of a pod in an isolated namespace.

```
[root@device ~]# kubectl get pod -n test-isolated-ns -o wide
NAME                        READY     STATUS     RESTARTS   AGE       IP                NODE
example3-bvqx5    1/1       Running    0          1h        10.47.25.249    b3s37
```

3. Ping the IP address of the pod in the non-isolated namespace from a pod in the isolated namespace.

```
[root@device ~]# kubectl exec -it example3-bvqx5 -n test-isolated-ns ping 10.47.25.251
PING 10.47.25.251 (10.47.25.251): 56 data bytes
64 bytes from 10.47.25.251: icmp_seq=0 ttl=63 time=1.467 ms
64 bytes from 10.47.25.251: icmp_seq=1 ttl=63 time=0.137 ms
^C--- 10.47.25.251 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.137/0.802/1.467/0.665 ms
```

Verify If a Namespace is Isolated

Namespace annotations are used to turn on isolation in a Kubernetes namespace. In isolated Kubernetes namespaces, the namespace metadata is annotated with the `opencontrail.org/isolation : true` annotation.

Use the following command to view annotations on a namespace.

```
[root@a7s16 ~]#
kubectl describe namespace test-isolated-ns
Name:        test-isolated-ns
Labels:      <none>
Annotations:     opencontrail.org/isolation : true     Namespace is isolated
Status:      Active
```

RELATED DOCUMENTATION

Contrail Integration with Kubernetes

CHAPTER 7

Using Contrail with Mesos

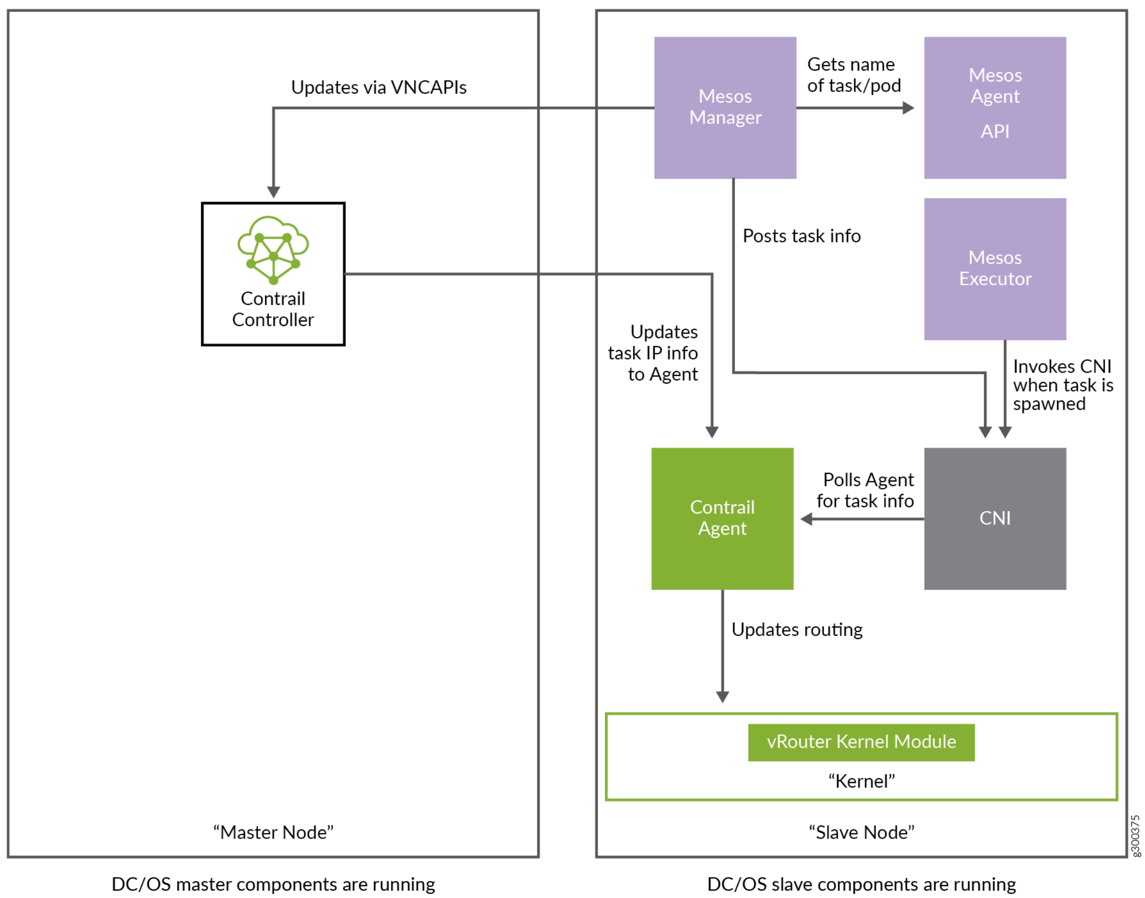
IN THIS CHAPTER

Understanding Contrail with Mesos Architecture

IN THIS SECTION

From Contrail Release 5.1.*x*, Contrail overlay and non-overlay network virtualization features are also available in Apache Mesos environment. The features are available in the commercial version of Mesosphere DC/OS.

Contrail with Mesos Architecture Diagram



Setup information

Setup is performed in two parts:

- DC/OS installation.

  For DC/OS setup, refer to https://dcos.io/install.

- Contrail installation.

  For Contrail installation, refer to https://github.com/Juniper/contrail-ansible-deployer.

NOTE: You must update the **inventory** file and set the orchestrator as *mesos*.

Master nodes consists of:

- DC/OS master components.

  For details, refer to https://docs.mesosphere.com/1.11/overview/architecture/components/.

- Contrail master components including Contrail Controller, Analytics, Config, and UI.

Slave/Agent nodes consists of:

- Contrail Agent.
- Contrail vRouter kernel module.
- Contrail CNI.
- Contrail Mesos Manager.
- DC/OS slave components.

  For details, refer to https://docs.mesosphere.com/1.11/overview/architecture/components/.

Components

The following components are a part of the architecture:

Contrail Controller

Contrail controller performs all the decision making. It includes config management, analytics, UI and control plane components for network virtualization. For further details, refer to https://github.com/Juniper/contrail-controller.

Contrail controller exposes APIs for creating configuration and updating virtual network components. In Mesos, mesos manager updates the task (universal docker) information to the Contrail controller via API server. All Contrail controller components are microservice docker containers.

Mesos Manager

Mesos manager consists of two sub modules:

- VNC server.
- Interaction with CNI and mesos agent.

```
+---------------+
|               |
|               |          Talks to Contrail VNC
|               +--------->  API server
| Mesos Manager |
|               |
|               <---------+  Gets task info from CNI
|               |
|               |
|               +--------->  Polls agent to get task name
|               |
+---------------+
```

Mesos manager application runs inside a docker on every slave node.

Mesos manager creates two networking by default: *mesos-default-pod-task* network and *ip-fabric* network.

All the pods and tasks are created in the *mesos-default-pod-task* network.

ip-fabric network is created in the respective domains of *mesos-default* and *project-default*.

CNI receives the task information and posts it to the Mesos manager. On receiving the task information, the Mesos manager creates the *contrail-vnc* objects.

Configuration information for the Mesos manager is present in **contrail-mesos.conf** file. The file is located at **/etc/contrail/contrail-mesos.conf** in the mesos manager docker.

Sample of **contrail-mesos.conf** file:

```
[MESOS]
listen_ip_addr=127.0.0.1
listen_port=6991
pod_task_subnets=10.x.x.0/12
ip_fabric_subnets=10.x.x.0/12

[VNC]
vnc_endpoint_ip=127.0.0.1
vnc_endpoint_port=8082
admin_user=admin
admin_password=admin
admin_tenant=admin
rabbit_server=127.0.0.1
rabbit_port=5673
cassandra_server_list=127.0.0.1:9161
```

```
[DEFAULTS]
disc_server_ip=127.0.0.1
disc_server_port=5998
log_local=1
log_level=SYS_NOTICE
log_file=/var/log/contrail/contrail-mesos-manager.log

[SANDESH]
#sandesh_ssl_enable=False
#introspect_ssl_enable=False
#sandesh_keyfile=/etc/contrail/ssl/private/server-privkey.pem
#sandesh_certfile=/etc/contrail/ssl/certs/server.pem
#sandesh_ca_cert=/etc/contrail/ssl/certs/ca-cert.pem
```

You can add the network to pod or task through annotation. You can set the network using labels.

Sample task/pod input json file:

```
networks": [
    {
      "name": "contrail-cni-plugin",
      "mode": "container",
      "labels": {
        "networks": "default-domain:default:blue-network",
        "pod-subnets": "default-domain:default:blue-network"
      }
    }
```

Introspect for *mesos-manager* objects on the port *8109*.

Contrail Container Network Interface (CNI)

The Container Network Interface (CNI) is located at **/opt/mesosphere/active/cni/contrail-cni-plugin**. It is a run to completion executable file.

The config file is located at **/opt/mesosphere/etc/dcos/network/cni/contrail-cni-plugin.conf**.

Sample **contrail-cni-plugin.conf** file:

```
{
    "cniVersion": "0.2.0",
```

```
    "contrail" : {
        "vrouter-ip"    : "slave-ip",
        "vrouter-port"  : 9091,
        "cluster-name"  : "slave-hostname",
        "config-dir"    : "/var/lib/contrail/ports/vm",
        "poll-timeout"  : 15,
        "poll-retries"  : 5,
        "log-file"      : "/var/log/contrail/cni/opencontrail.log",
        "log-level"     : "debug",
        "mesos-ip"      : "localhost",
        "mesos-port"    : "6991",
        "mode"          : "mesos"
    },

    "name": "contrail-cni-plugin",
    "type": "contrail-cni-plugin"
}
```

Mesos agent invokes Contrail CNI when custom/host network provider is mentioned as *contrail-cni-plugin* in the task description.

Installing Contrail with Mesos

The setup process is a 2-step process including DC/OS setup and Contrail setup.

Refer to Mesosphere DC/OS website to set up DC/OS.

Contrail Networking supports DC/OS version—1.11.0.

Contrail Setup

contrail-container-builder is added with two new containers: Mesos manager and mesos-node-init

- mesos-node-init installs Mesos CNI.
- mesos manager and mesos-node-init runs on the worker node.

NOTE: Orchestration is set to Mesos but no Mesos components will be installed through *contrail-ansible-deployer*.

Run the following commands to set up Contrail.

1. Install ansible and clone ansible deployer.

```
yum install -y ansible-2.4.2.0 git vim
git clone http://github.com/Juniper/contrail-ansible-deployer
cd contrail-ansible-deployer
ssh-copy-id <all-nodes>
```

   For more details, refer to contrail-ansible-deployer GitHub repository.

2. Verify **configure_instances.yml** file.

   Sample configure_instances.yml file

```
provider_config:
  bms:
    ssh_pwd: <password>
    ssh_user: root
    ntpserver: <ntp_server>
    domainsuffix: local
instances:
  bms1:
    provider: bms
    ip: <ip-address-master>
    roles:
        config_database:
        config:
        webui:
        control:
        analytics_database:
        analytics:
  bms2:
    provider: bms
    ip: <ip-address-agent>
    roles:
        mesos_master:
  bms3:
    provider: bms
    ip: <ip-address>
    roles:
        vrouter:
        mesosmanager:
        mesos_agent_private:
  bms4:
```

```
    provider: bms
    ip: <ip-address>
    roles:
        vrouter:
        mesosmanager:
        mesos_agent_public:
global_configuration:
  CONTAINER_REGISTRY: <contrail-registry>
  REGISTRY_PRIVATE_INSECURE: true
contrail_configuration:
  CLOUD_ORCHESTRATOR: mesos
  CONTRAIL_VERSION: queens-master-latest
  RABBITMQ_NODE_PORT: 5673
```

For more details, refer to Contrail deployment on Mesosphere DC/OS orchestrator.

3. Run Contrail Ansible playbooks.

NOTE: You can specify orchestrator as Mesos in **instance.yaml** or run in ansible-playbook as -e orchestrator=mesos.

```
ansible-playbook -i inventory/ playbooks/configure_instances.yml
ansible-playbook -i inventory/ playbooks/install_contrail.yml
```

You can also import *Mesos* cluster in Contrail command. For details, refer to "Importing Contrail Cluster Data using Contrail Command" on page 47.

RELATED DOCUMENTATION

Understanding Contrail with Mesos Architecture | 136
Importing Contrail Cluster Data using Contrail Command | 47

CHAPTER 8

Using VMware vCenter with Containerized Contrail

IN THIS CHAPTER

Integrating vCenter for Contrail

IN THIS SECTION

These topics provide instructions for integrating Contrail Release 5.1.x and microservices with VMware vCenter.

Prerequisites

Before you start the integration, ensure that the contrail controller meets the prerequisites given in "Server Requirements and Supported Platforms" on page 16.

Follow these steps to prepare Contrail controller(s):

```
yum update -y

yum install -y yum-plugin-priorities https://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm

yum install -y python-pip git gcc python-devel sshpass

yum install -y git

pip install “ansible==2.5.0” pyvmomi
```

ESX Agent Manager

VMware provides a standard vCenter solution called vSphere ESX Agent Manager (EAM), that allows you to deploy, monitor, and manage ContrailVMs on ESXi hosts.

The ContrailVM is deployed as an Agent VM that is monitored by EAM. With this integration, ContrailVMs are marked as more critical and privileged than other tenant VMs on the host.

The following are the benefits of running ContrailVM as an AgentVM from EAM:

- Auto-deploy ContrailVMs on ESXi hosts in scope (clusters).
- Manage and Monitor ContrailVMs through EAM in the vSphere web client.
- Integrate with other vCenter features like AddHos, Maintenance Mode, vSphere DRS, vSphere DPM, and VMWare HA.

These topics provide instructions for integrating Contrail Release 5.1.x and microservices with VMware vCenter.

Set Up vCenter Server

Follow these steps to set up the vCenter server.

1. Download the Contrail Ansible Deployer (**contrail-ansible-deployer-< >.tgz**) onto your provisioning host. You can download the deployer from https://www.juniper.net/support/downloads/?p=contrail#sw.
2. Untar the **tgz**.

   ```
   - tar xvf contrail-ansible-deployer-< >.tgz
   ```

3. Prepare a **vcenter_vars.yml** file populated with vCenter server and ESXI hosts parameters. You can download the CentOS 7.5 and ESXi VM Host from https://www.juniper.net/support/downloads/?p=contrail#sw.

   NOTE: You can see a sample of the vcenter_vars.yml file in the **contrail-ansible-deployer/playbooks /roles/vcenter/vars/vcenter_vars.yml** after you extract the image files.

   NOTE: The ContrailVM's Open Virtualization Format (OVF) image must be hosted on an http or https server which runs on and is reachable from the vCenter server. The location of the OVF is provided as a URL path for `vmdk:` as shown in the example given below.

   Example: Enabling HA and DRS in the cluster

```
vcenter_servers:
  - SRV1:
      hostname:
      username:
      password:
      # Optional: defaults to False
      #validate_certs: False
      datacentername:
      clusternames:
      #path to the ovf, is needed for ESX Agent Manager to deploy ContrailVMs
      vmdk: http://<ip-address>/centos-7.5/LATEST/ContrailVM.ovf
      # Optional: If not specified HA and DRS are turned off on the clusters.
      enable_ha: yes
      enable_drs: yes
```

   For definition examples, refer **contrail-ansible-deployer/playbooks/roles/vcenter/vars/vcenter_vars.yml.sample**.

   To enable HA and DRS in the cluster, set `enable_ha` and `enable_drs` to `yes` in the **vcenter_vars.yml** file. If these flags are not enabled, HA and DRS is turned off by default for newly created and existing clusters.

   Example instances.yaml File

Username and password combinations are provided in this output for illustrative purposes only. We suggest using unique username and password combinations in accordance with your organization's security guidelines in your environment.

```
provider_config:
  bms:
    ssh_pwd: password
    ssh_user: root
    ntpserver: 8.8.8.8
    domainsuffix: blah.net

instances:
  bms1:
    provider: bms
    ip: <ip-address>
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      webui:
      vcenter_plugin:
  bms2:
    provider: bms
    esxi_host: <ip-address>
    ip: <ip-address>
    roles:
      vrouter:
      vcenter_manager:
        ESXI_USERNAME: root
        ESXI_PASSWORD: password
  bms3:
    provider: bms
    esxi_host: <ip-address>
    ip: <ip-address>
    roles:
      vrouter:
      vcenter_manager:
        ESXI_USERNAME: root
        ESXI_PASSWORD: password
  bms4:
```

```
    provider: bms
    esxi_host: <ip-address>
    ip: <ip-address>
    roles:
      vrouter:
      vcenter_manager:
        ESXI_USERNAME: root
        ESXI_PASSWORD: password


global_configuration:
  CONTAINER_REGISTRY: hub.juniper.net
  CONTAINER_REGISTRY_USERNAME: username
  CONTAINER_REGISTRY_PASSWORD: password
  REGISTRY_PRIVATE_INSECURE: False

contrail_configuration:
  CLOUD_ORCHESTRATOR: vcenter
  CONTROLLER_NODES: <ip-address>
  CONTRAIL_VERSION: 5.1.0-0.360
  RABBITMQ_NODE_PORT: 5673
  VCENTER_SERVER: <ip-address>
  VCENTER_USERNAME: administrator@vsphere.net
  VCENTER_PASSWORD: password
  VCENTER_DATACENTER: <DC name here>
  VCENTER_DVSWITCH: overlay
  VCENTER_WSDL_PATH: /usr/src/contrail/contrail-web-core/webroot/js/vim.wsdl
  VCENTER_AUTH_PROTOCOL: https
```

NOTE: The default login credentials for Contrail OVF:

- Username: *root*
- Password: *c0ntrail123*

We suggest using unique usernames and passwords in accordance with your organization's security guidelines.

Example vcenter_vars.yml File

```
---
vcenter_servers:
  - SRV1:
      hostname: <host-ip-address>
      username: administrator@vsphere.net
      password: password
      # Optional: defaults to False
      #validate_certs: False
      datacentername: "<your DC name here>"
      clusternames:
        - "<your cluster name here>"
      vmdk: http://<ip-address>/contrail/images/ContrailVM.ovf
      dv_switch:
        dv_switch_name: overlay
      dv_port_group:
        dv_portgroup_name: VM_pg
        number_of_ports: 1800

esxihosts:
  - name: <ip-address>
    username: root
    password: password
    datastore: <your local datastore here>
    datacenter: "<your DC name here>"
    cluster: "<your cluster name here>"
    contrail_vm:
      networks:
        - mac: 00:77:56:aa:bb:01
    vcenter_server: SRV1 #leave this
  - name: <ip-address>
    username: root
    password: password
    datastore: <your local datastore here>
    datacenter: "<your DC name here>"
    cluster: "<your cluster name here>"
    contrail_vm:
      networks:
        - mac: 00:77:56:aa:bb:02
    vcenter_server: SRV1 #leave this
  - name: <ip-address>
```

```
    username: root
    password: password
    datastore: <your local datastore here>
    datacenter: "<your DC name here>"
    cluster: "<your cluster name here>"
    contrail_vm:
      networks:
        - mac: 00:77:56:aa:bb:77
    vcenter_server: SRV1 #leave this
```

4. Run the Contrail vCenter playbook.

```
ansible-playbook playbooks/vcenter.yml
```

> **NOTE**: Verify that the hostnames for the contrail controller(s) and the ContrailVMs (vRouters) are unique in **/etc/hostname** file.
>
> You can verify hostname from either the DHCP options (if the management network uses DHCP) or manually (if the management network uses static IP allocation).

Configure Contrail Parameters

Populate the file **config/instances.yaml** with Contrail roles.

For an example file, see **contrail-ansible-deployer/confing/instances.yaml.vcenter_example**.

Install Contrail

Install Contrail by running the following Contrail playbooks:

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

Monitor and Manage ContrailVM from ESX Agent Manager

ContrailVMs can be monitored from EAM by using ContrailVM-Agency.

Follow these steps to monitor and manage Contrail VM from EAM:

1. Resolve issues from the ContrailVM-Agency.

   The ContrailVM-Agency is in an alert state when the ContrailVM in any host is powered off or is deleted.

Click **Resolve All Issues** from the ContrailVM-Agency to correct the issue. The ContrailVM-Agency will attempt to correct the issue by bringing the ContrailVM back online or by spawning a ContrailVM from the OVF on the ESXi host.

Figure 23: vCenter Server Extensions

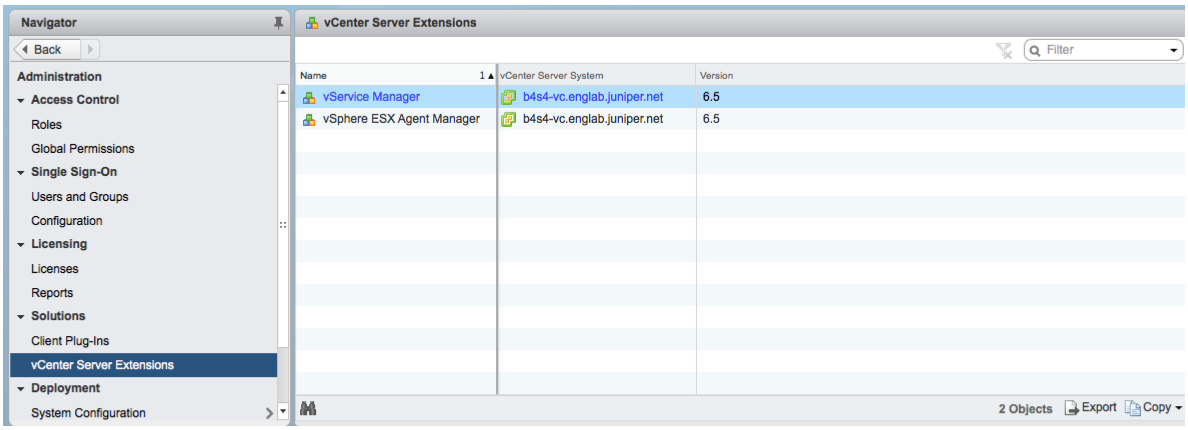
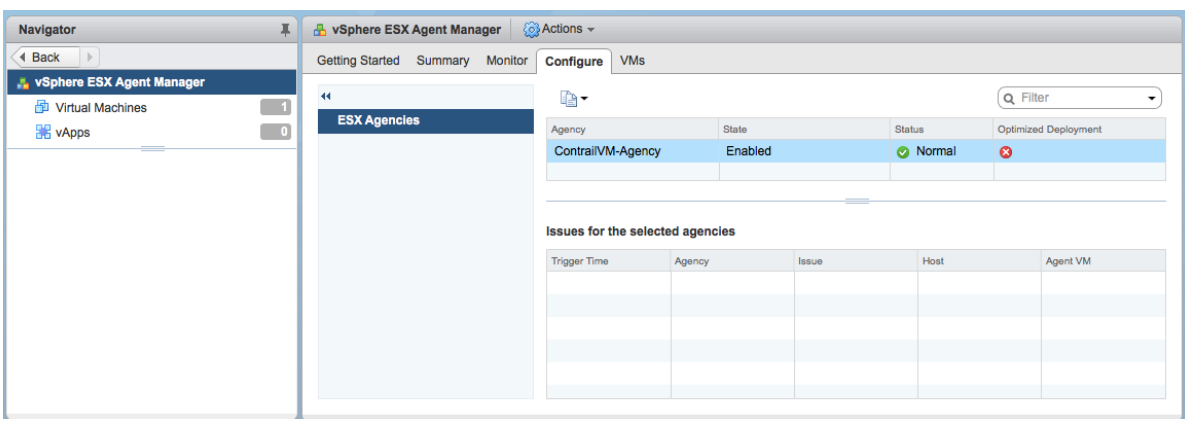


Figure 24: ESX Agencies

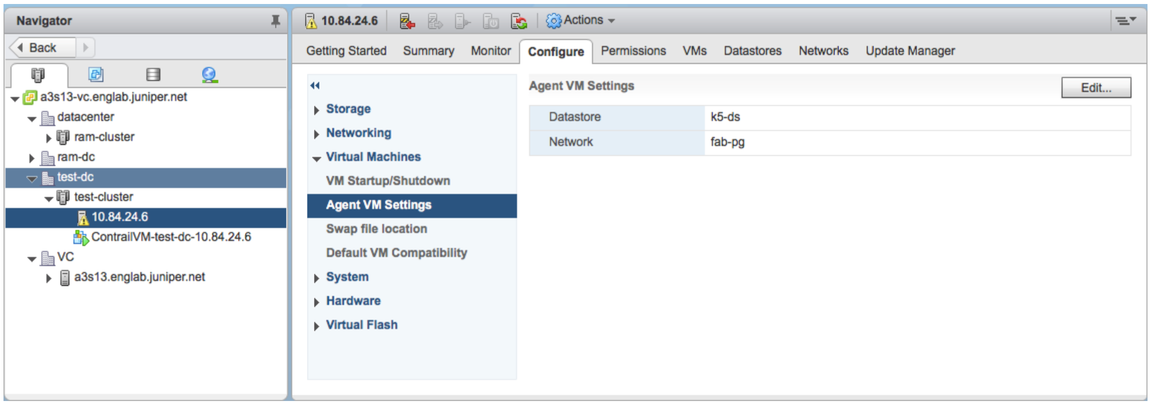


2. Add host.

   a. Add ESXi host to the cluster.

   b. Configure **Agent VM Settings** for the ESXI host.

Figure 25: Configure Agent VM Settings



For more information on configuring Agent VM, network, and datastore settings, see Configure Agent VM Settings.

EAM deploys a ContrailVM (from the base OVF) on the ESXi host.

c. Add ESXi host details to **vcenter_vars.yml** and repeat step 4 to add appropriate interfaces to the ContrailVM and to configure necessary settings in the vCenter server.

d. Add ContrailVM details to **instances.yaml** and provision Contrail on the newly added ContrailVm (router). For more information on provisioning Contrail, see "Install Contrail" on page 150.

3. Clean up the ContrailVM-Agency.

   Delete **ContrailVM-Agency** from the EAM user interface to delete ContrailVM and the agency.

RELATED DOCUMENTATION

Configuring Underlay Network for ContrailVM

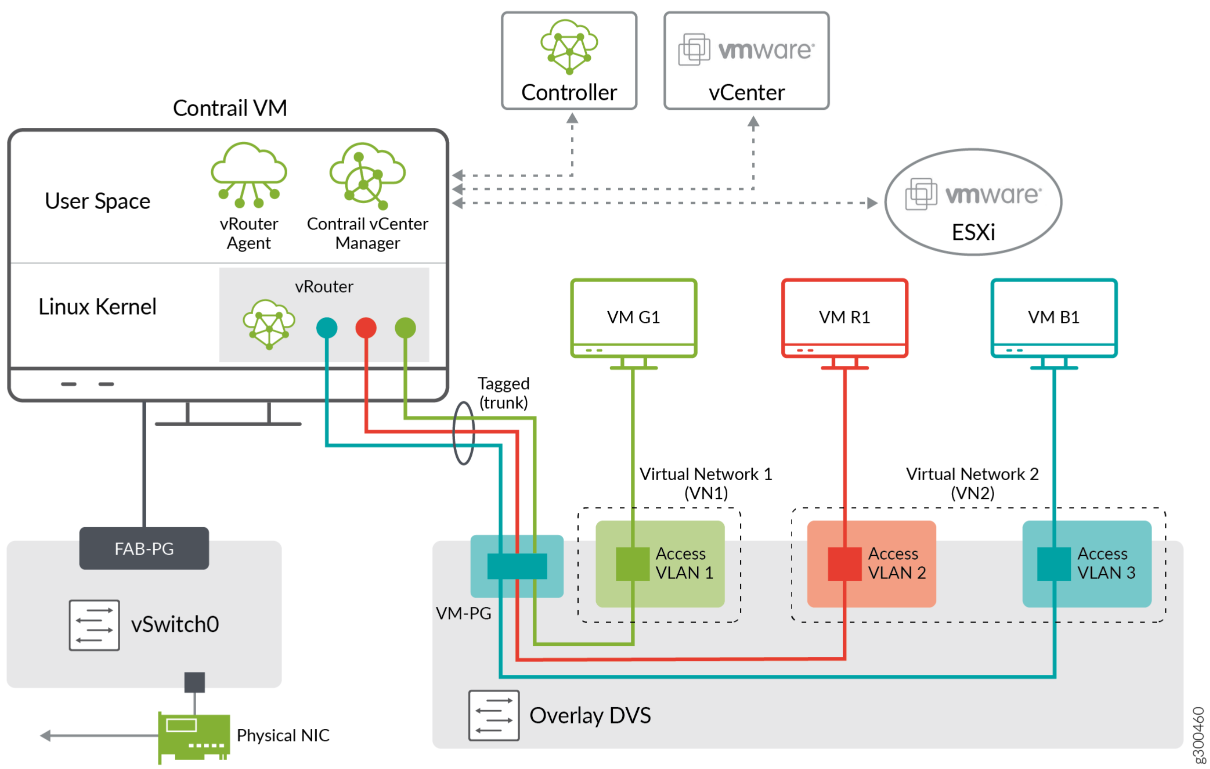
IN THIS SECTION

The ContrailVM can be configured in several different ways for the underlay (`ip-fabric`) connectivity:

Standard Switch Setup

In the standard switch setup, the ContrailVM is provided an interface through the standard switch port group that is used for management and control data, see Figure 26 on page 153.

Figure 26: Standard Switch Setup



To set up the ContrailVM in this mode, the standard switch and port group must be configured in `vcenter_vars.yml`.

If switch name is not configured, the default values of `vSwitch0` are used for the standard switch.

The ContrailVM supports multiple NICs for management and `control_data` interfaces. The management interface must have the DHCP flag as `true` and the `control_data` interface can have DHCP set as `false`. When DHCP is set to false, the IP address of the `control_data` interface must be configured by the user and ensure connectivity. Additional configuration such as static routes and bond interface must be configured by the user.

The following is an example of configuration with standard switch.

```
-   name: <esxi_host>
    username: <username>
    password: <password>
    datastore: <datastore>
    vcenter_server: <server>
    datacenter: <datacenter>
    cluster: <cluster>
    std_switch_list:
      - pg_name: mgmt-pg
        switch_name: vSwitch0
    contrail_vm:
       networks:
         - mac: 00:77:56:aa:bb:03
           sw_type: standard
           switch_name: vSwitch0
           pg: mgmt-pg
```

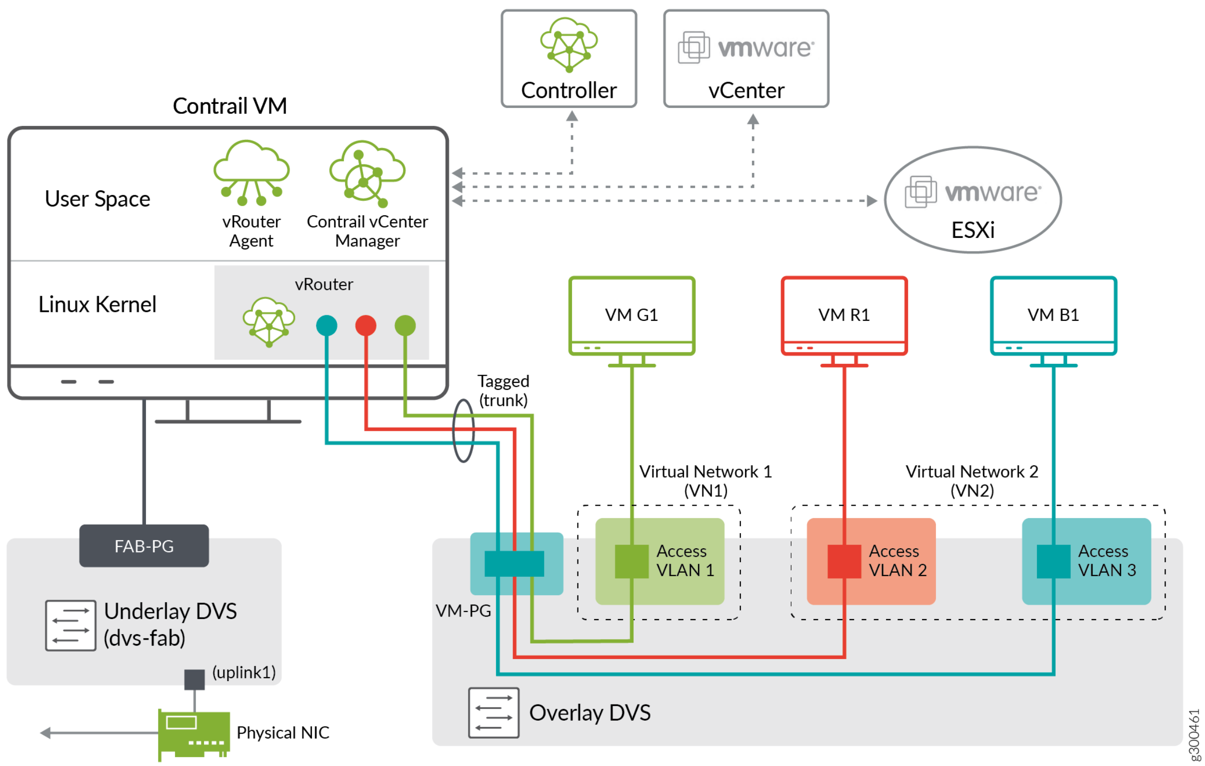
Distributed Switch Setup

A distributed switch functions as a single virtual switch across associated hosts.

In the distributed switch setup, the ContrailVM is provided an interface through the distributed switch port group that is used for management and control data, see Figure 27 on page 155.

The ContrailVM can be configured to use the management and control_data NICs from DVS. When the DVS configuration is specified, the standard switch configuration is ignored.

Figure 27: Distributed Switch Setup



To set up the ContrailVM in this mode, configure the distributed switch, port group, number of ports in the port group, and the uplink in the `vcenter_servers` section in **vcenter_servers.yml**.

NOTE: The uplink can be a link aggregation group (LAG). If you use LAG, then DVS and LAG should be preconfigured.

The following is an example distributed switch configuration in `vcenter_vars.yml`.

```
vcenter_servers:
 - SRV1:
     hostname: <server>
     username: <username>
     password: <password>
     datacentername: <datacenter>
     clusternames:
       - <cluster>
```

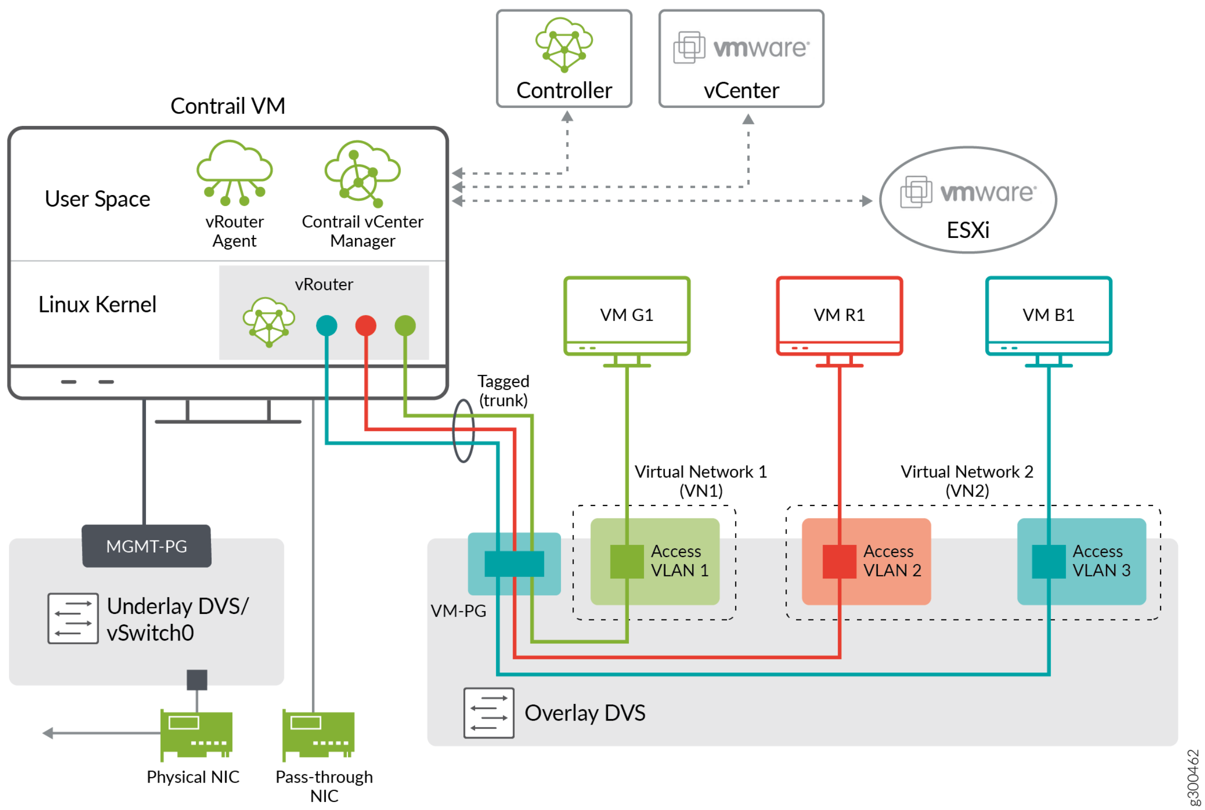
```
dv_switch:
  dv_switch_name: <dvs_name>
dv_port_group:
  dv_portgroup_name: <pg_name>
  number_of_ports: <num_of_ports>
dv_switch_control_data:
  dv_switch_name: <ctrl_dvs_name>
dv_port_group_control_data:
  dv_portgroup_name: <ctrl_pg_name>
  number_of_ports: <num_of_ports>
  uplink:
   - 'vmnic3'
```

PCI Pass-Through Setup

PCI pass-through is a virtualization technique in which a physical Peripheral Component Interconnect (PCI) device is directly connected to a virtual machine, bypassing the hypervisor. Drivers in the VM can directly access the PCI device, resulting in a high rate of data transfer.

In the pass-through setup, the ContrailVM is provided management and control data interfaces. Pass-through interfaces are used for control data. Figure 28 on page 157 shows a PCI pass-through setup with a single `control_data` interface.

Figure 28: PCI Pass-Through with Single Control Data Interface



When setting up the ContrailVM with pass-through interfaces, upon provisioning ESXi hosts in the installation process, the PCI pass-through interfaces are exposed as Ethernet interfaces in the ContrailVM, and are identified in the `control_data` device field.

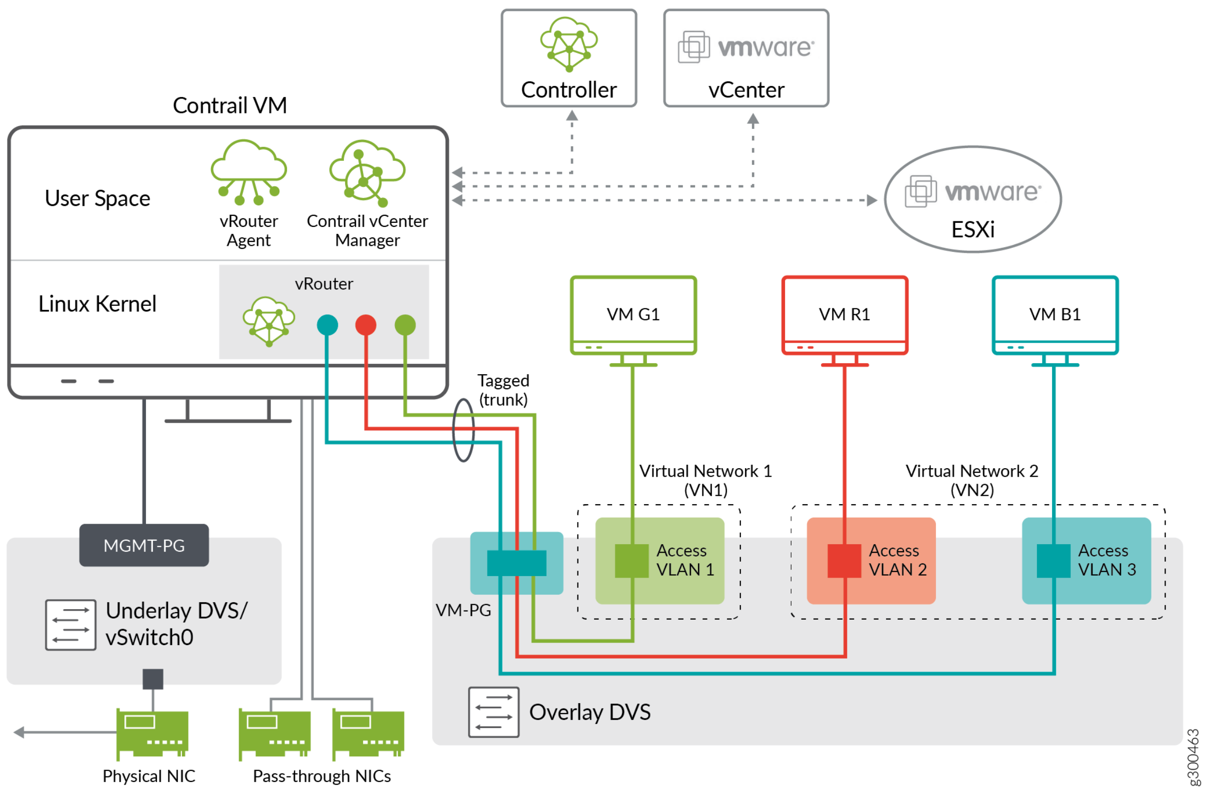
The following is an example PCI pass-through configuration with a single `control_data` interface:

```
esxihosts:
  - name: <esxi_host>
    username: <username>
    password: <password>
    datastore: <datastore>
    vcenter_server: <server>
    datacenter: <datacenter>
    cluster: <cluster>
    contrail_vm:
      networks:
        - mac: <mac_addr>
```

```
pci_devices:
 - '0000:04:00.0'
```

Figure 29 on page 158 shows a PCI pass-through setup with a bond_control data interface, which has multiple pass-through NICs.

Figure 29: PCI Pass-Through Setup with Bond Control Interface



Update the ContrailVM section in `vcenter_vars.yml` with `pci_devices` as shown in the following example:

```
esxihosts:
  - name: <esxi_host>
    username: <username>
    password: <password>
    datastore: <datastore>
    vcenter_server: <server>
    datacenter: <datacenter>
    cluster: <cluster>
    contrail_vm:
```

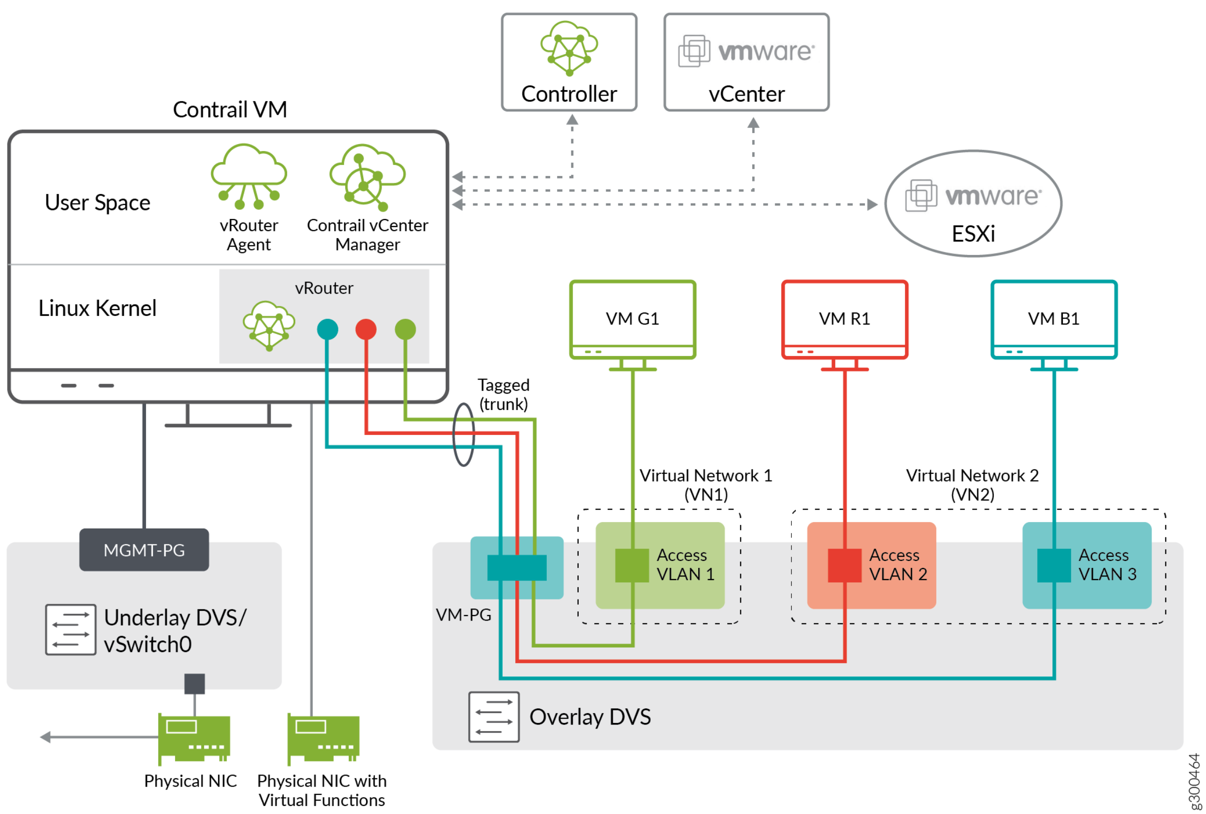
```
networks:
  - mac: <mac_addr>
pci_devices:
 - '0000:04:00.0'
 - '0000:04:00.1'
```

SR-IOV Setup

A single root I/O virtualization (SR-IOV) interface allows a network adapter device to separate access to its resources among various hardware functions.

In the SR-IOV setup, the ContrailVM is provided management and control data interfaces. SR-IOV interfaces are used for control data. See Figure 30 on page 159.

Figure 30: SR-IOV Setup



In VMware, the `port-group` is mandatory for SR-IOV interfaces because the ability to configure the networks is based on the active policies for the port holding the virtual machines.

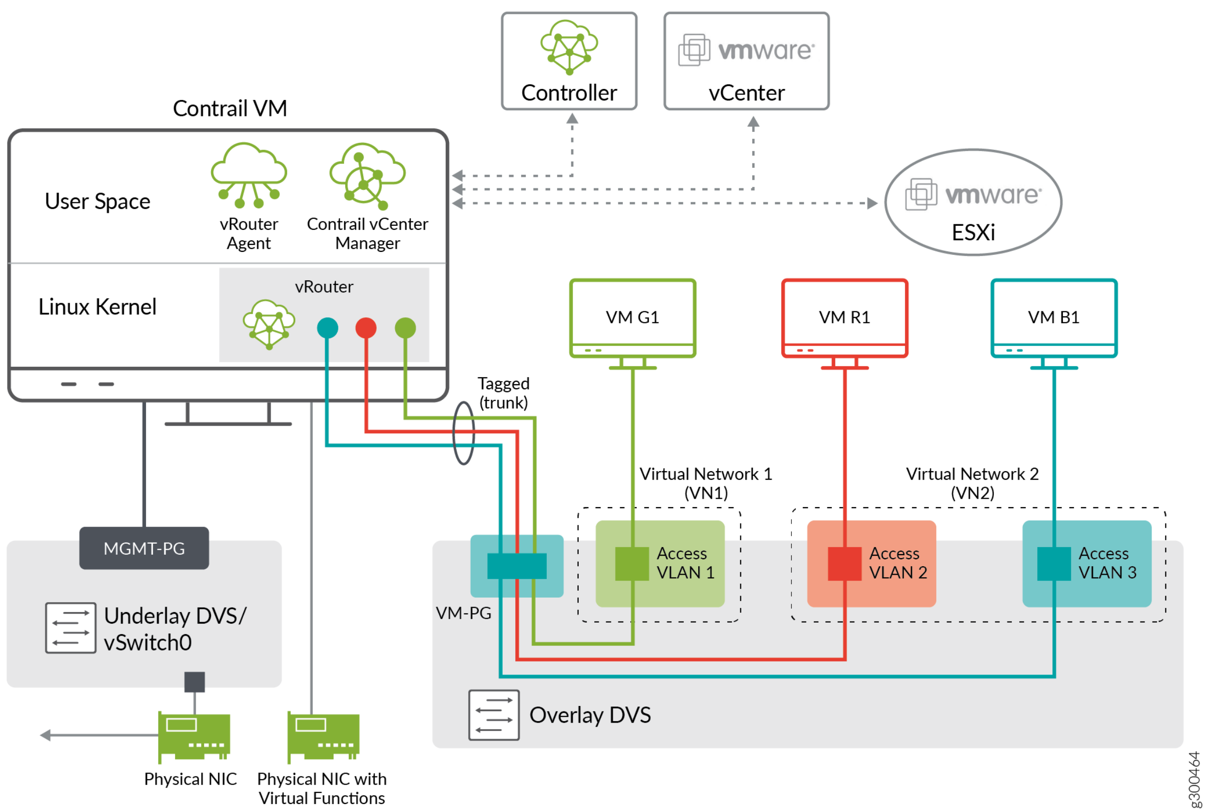
To set up the ContrailVM with SR-IOV interfaces, all configurations used for the standard switch setup are also used for the pass-through setup, providing management connectivity to the ContrailVM.

To provide the `control_data` interfaces, configure the SR-IOV-enabled physical interfaces in the `contrail_vm` section, and configure the `control_data` in the global section of `vcenter_vars.yml`.

Upon provisioning ESXi hosts in the installation process, the SR-IOV interfaces are exposed as Ethernet interfaces in the ContrailVM.

Figure 31 on page 160 shows a SR-IOV setup with a single `control_data` interface.

Figure 31: SR-IOV With Single Control Data Interface



The following is an example SR-IOV configuration for the cluster and server configuration.

The cluster configuration:

```
vcenter_servers:
  - SRV1:
      hostname: <server>
```

```
    username: <username>
    password: <password>
    datacentername: <datacenter>
    clusternames:
      - <cluster>


    dv_switch:
      dv_switch_name: <dvs_name>
    dv_port_group:
      dv_portgroup_name: <pg_name>
      number_of_ports: <num_of_ports>
    dv_switch_sr_iov:
      dv_switch_name: <sriov_dvs_name>
    dv_port_group_sriov:
      dv_portgroup_name: <sriov_pg_name>
      number_of_ports:
```

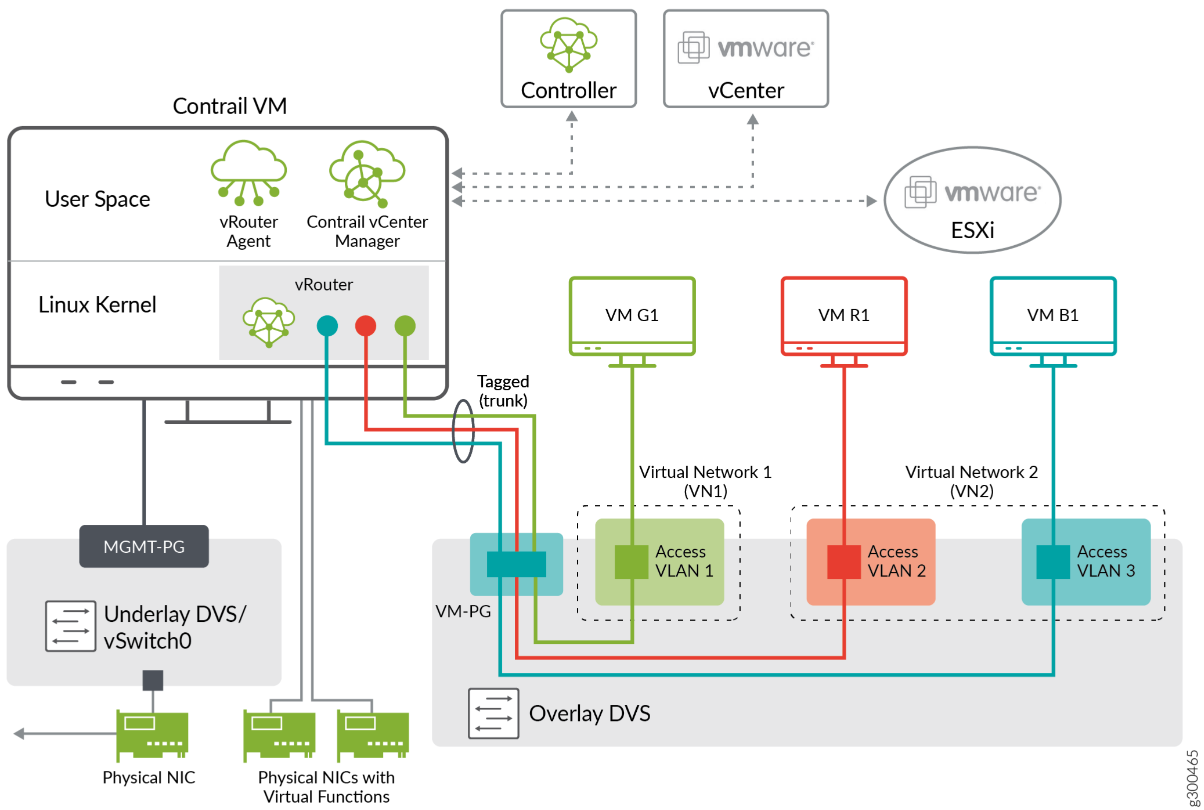
The server configuration:

```
esxihosts:
  - name: <esxi_host>
    username: <username>
    password: <password>
    datastore: <datastore>
    vcenter_server: <server>
    datacenter: <datacenter>
    cluster: <cluster>
    contrail_vm:

      networks:
        - mac: <mac_addr>
      sr_iov_nics:
       - 'vmnic0'
```

Figure 32 on page 162 shows an SR-IOV configuration with a bond `control_data` interface, which has multiple SR-IOV NICs.

Figure 32: SR-IOV With Bond Control Data Interface



For Bond interface-configuration specify multiple NICs in `sr_iov_nics`, and add required configuration for multi-interface and bond configuration in `vcenter_vars.yml`.

The cluster configuration:

```
vcenter_servers:
  - SRV1:
      hostname: <server>
      username: <username>
      password: <password>
      datacentername: <datacenter>
      clusternames:
        - <cluster>


      dv_switch:
        dv_switch_name: <dvs_name>
      dv_port_group:
```

```
        dv_portgroup_name: <pg_name>
        number_of_ports: <num_of_ports>
      dv_switch_sr_iov:
        dv_switch_name: <sriov_dvs_name>
      dv_port_group_sriov:
        dv_portgroup_name: <sriov_pg_name>
        number_of_ports:
```

The server configuration:

```
esxihosts:
  - name: <esxi_host>
    username: <username>
    password: <password>
    datastore: <datastore>
    vcenter_server: <server>
    datacenter: <datacenter>
    cluster: <cluster>
    contrail_vm:

      networks:
        - mac: <mac_addr>
      sr_iov_nics:
       - 'vmnic0'
       - 'vmnic1'
```

RELATED DOCUMENTATION

Managing Networks From Contrail Command and VMware vCenter User Interfaces

Installing and Provisioning Contrail VMware vRealize Orchestrator Plugin

IN THIS SECTION

A dedicated Contrail plugin is used to connect to VMware vRealize Orchestrator (vRO). Contrail Release 5.0 supported a Beta version of the plugin. Starting with Contrail Release 5.1, a fully supported version of the plugin is available.

You must install the Contrail VMware vRealize Orchestrator (vRO) plugin to connect to the vRO server.

Before you begin installation, ensure the following:

- You have administrator-level access to the Control Center of a deployed vRO appliance.
- You know the host name (*{vRO}*) of the deployed vRO Appliance.
- You have the login credentials of the vCenter SSO service.
- You have downloaded the vRO plugin package file to your local system.

  You can download the plugin from https://www.juniper.net/support/downloads/?p=contrail.

You can deploy the Contrail plugin in any Java Virtual Machine (JVM) compatible environment and load it on an active vRO instance.

The following topics describe how to install and provision the Contrail vRO plugin.

Accessing vRO Control Center

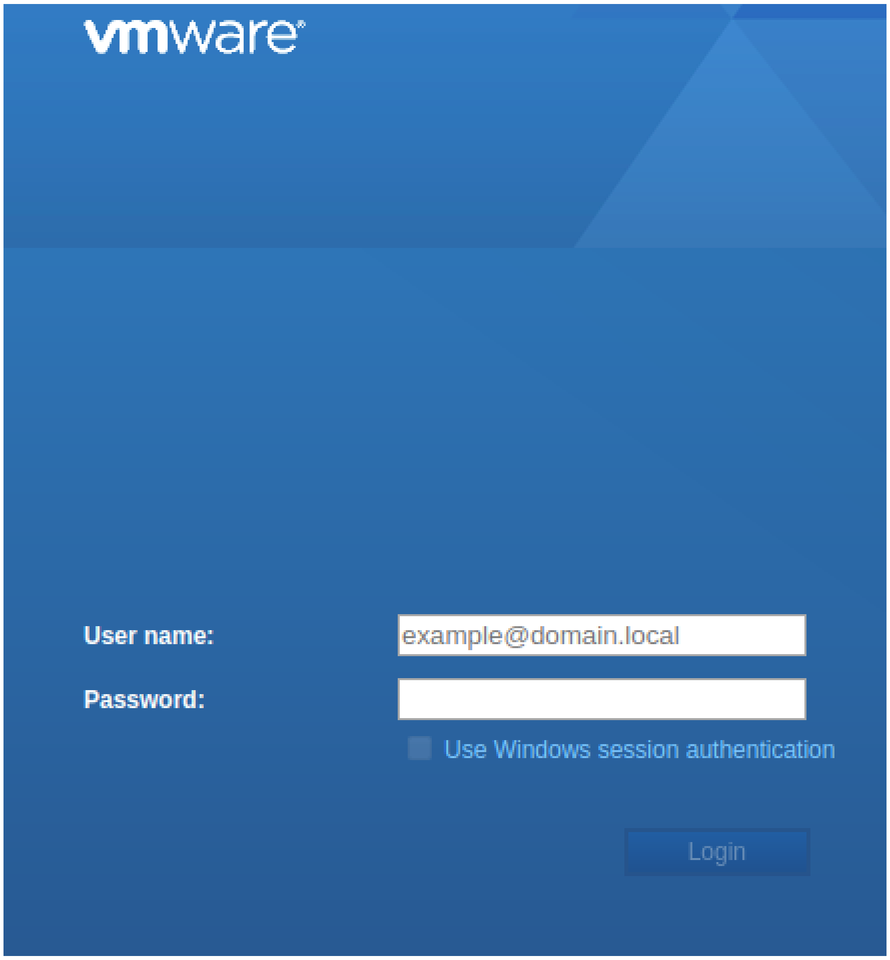
Follow the steps given below to access and log in to vRO Control Center:

1. To access vRO Control Center through a Web browser, navigate to the **https://{vRO}:8283/vco-controlcenter** URL.

   > **NOTE**: Replace *{vRO}* given in the URL with the *host name* of the deployed vRO Appliance. The *host name* is the IP address or the FQDN of the vRO node.

   The **vCenter SSO** service page is displayed.

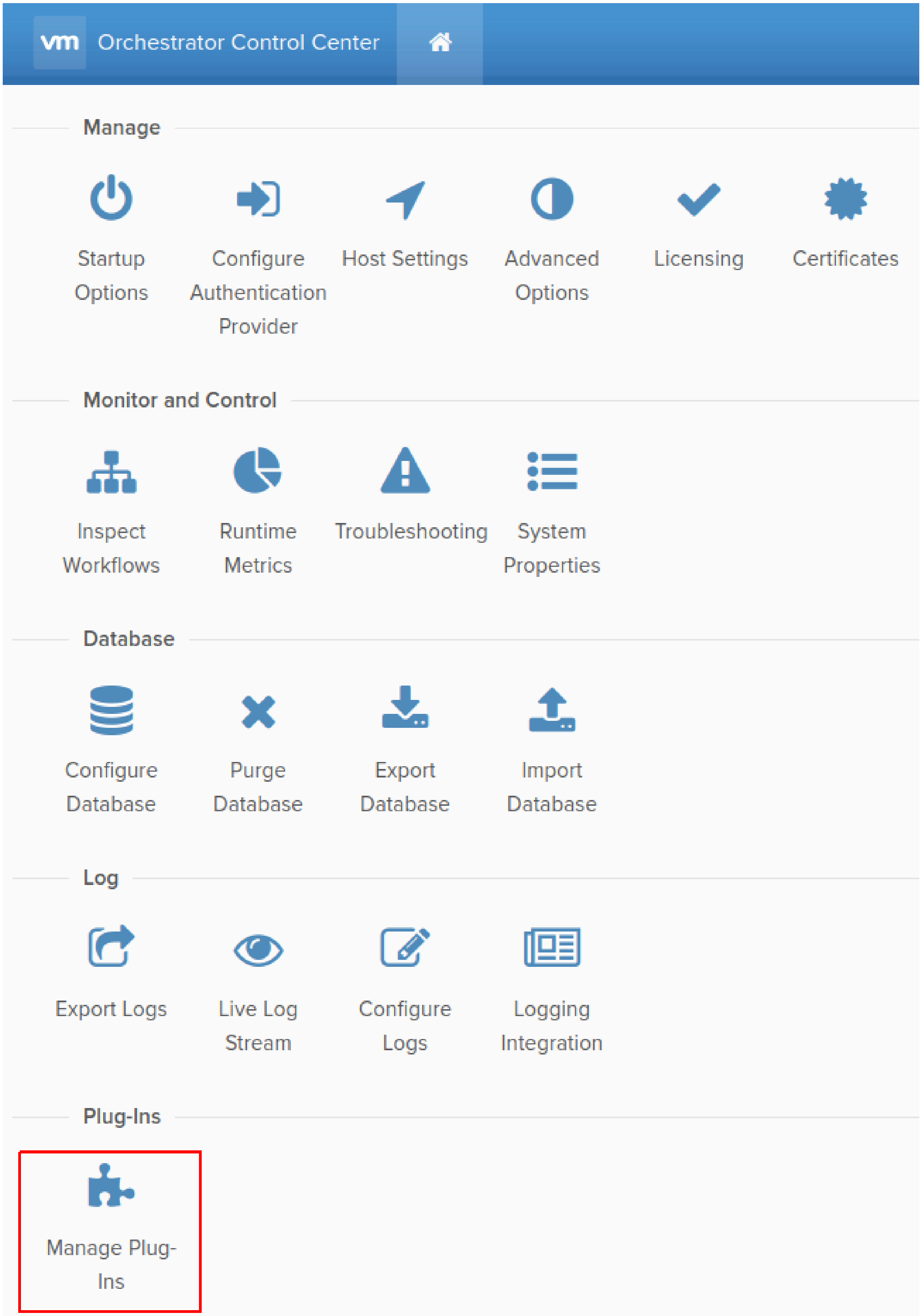
Figure 33: vCenter SSO service page



2. On the vCenter SSO service page, enter the **User name** and **Password** in the respective fields and click **Login**. See Figure 33 on page 165.

   The **Orchestrator Control Center** home page is displayed.

Figure 34: Orchestrator Control Center



Installing vRO Plugin

Perform the following steps to install the vRO plugin:

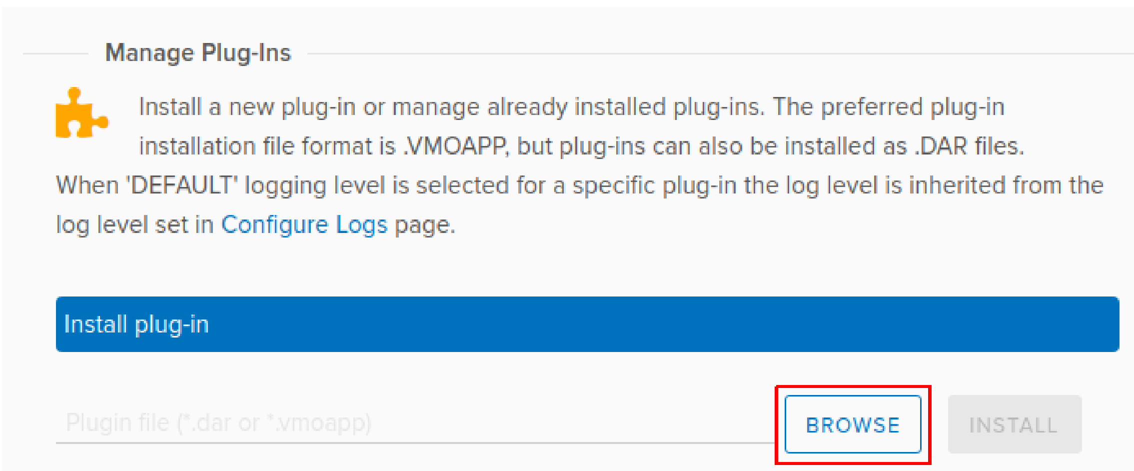
1. Upload vRO plugin package.

   To upload vRO plugin package:

   - From the Orchestrator Control Center home page, click **Manage Plug-Ins** under the **Plug-Ins** section.

     The **Manage Plug-Ins** page is displayed.

     Figure 35: Manage Plug-Ins page

     NOTE: You can install a new plugin or manage an already installed plugin from the Manage Plug-Ins page.

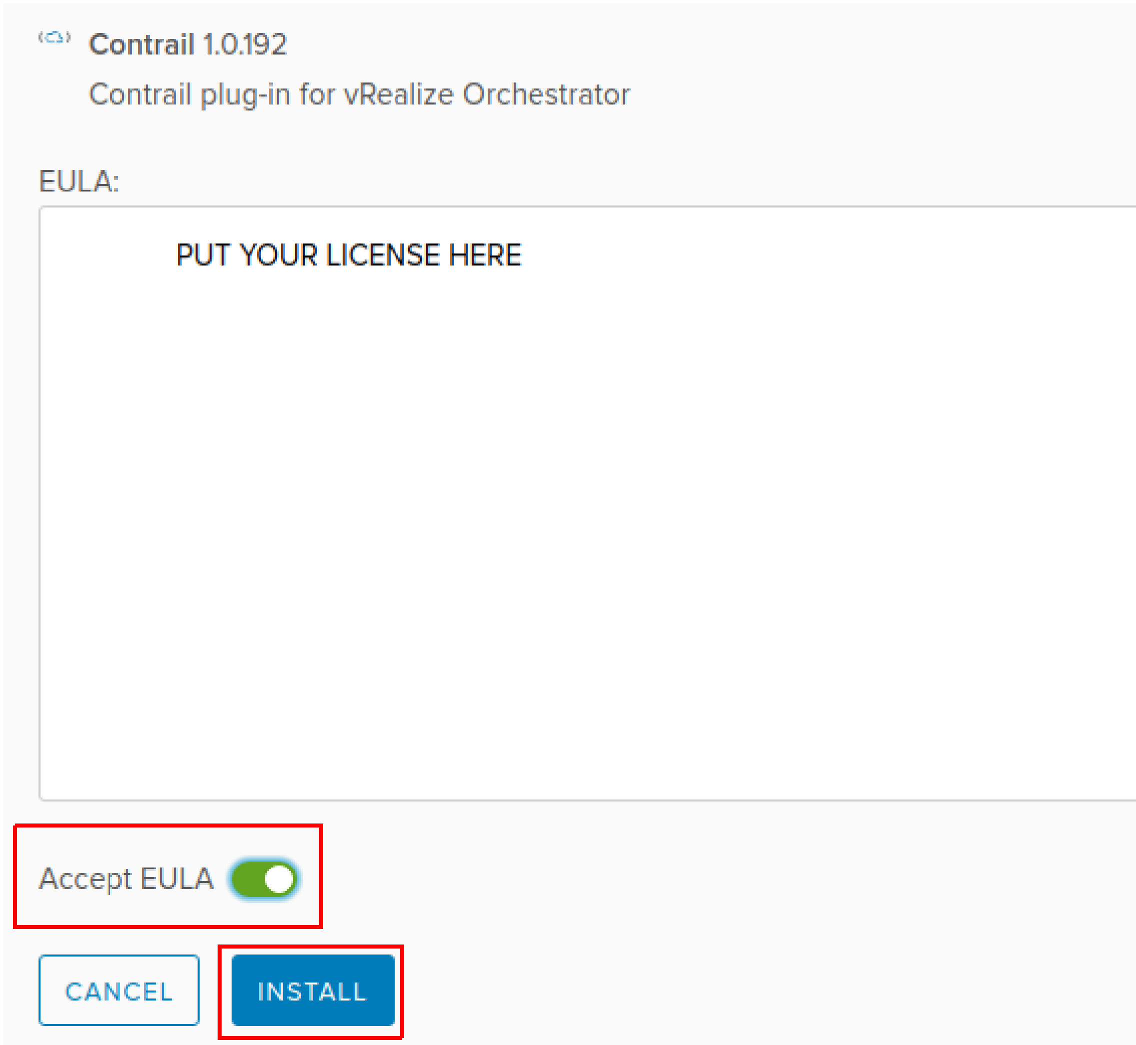
     NOTE: ***.vmoapp** or ***.dar** file format can be used. Also, the version in this example may be different from the version you have downloaded.

   - Click **Browse** in the **Install plug-in** pane and select the downloaded vRO plugin package file on your local system.

   - After you select vRO plugin package file, click **Install** to upload the vRO plugin package to the vRO server.

     The **EULA** page is displayed.

Figure 36: EULA page



2. Install vRO plugin.

   After you upload the vRO plugin package, select **Accept EULA** on the **EULA** page and then click **Install**.

   > **NOTE**: If you use ***.vmoapp** file format, you are directed to the Accept EULA page before you proceed with the installation.
   > If you use ***.dar** file format, you can directly proceed with installation.

   The vRO plugin is installed.

Accessing vRO Desktop Client

After you install the VMware vRealize Orchestrator (vRO) plugin, download vRealize Orchestrator Client version 7.3.0 to access the vRO server.

To download and install the vRO desktop client application, click **https://{vRO}:8281/vco/**.

> **NOTE**: Replace *{vRO}* given in the URL with the *host name* of the deployed vRO Appliance.

Figure 37: Getting Started with vRealize Orchestrator

You can download vRO desktop client applications for Windows, Mac OS X, and Linux operating systems.

Connecting to vRO using the Desktop Client

You connect to the vRO server by using the vRO desktop client.

1. Start the vRO desktop client.

   The **VMware vRealize Orchestrator Login** page is displayed.

Figure 38: VMware vRealize Orchestrator Login page

2. In the VMware vRealize Orchestrator Login page, enter **Host name**, **User name**, and **Password**.

> **NOTE**: The **Host name** also includes the port number and must be in the **{vRO}:8281** format.

3. Click **Login** to connect to the vRO server. See Figure 38 on page 170.

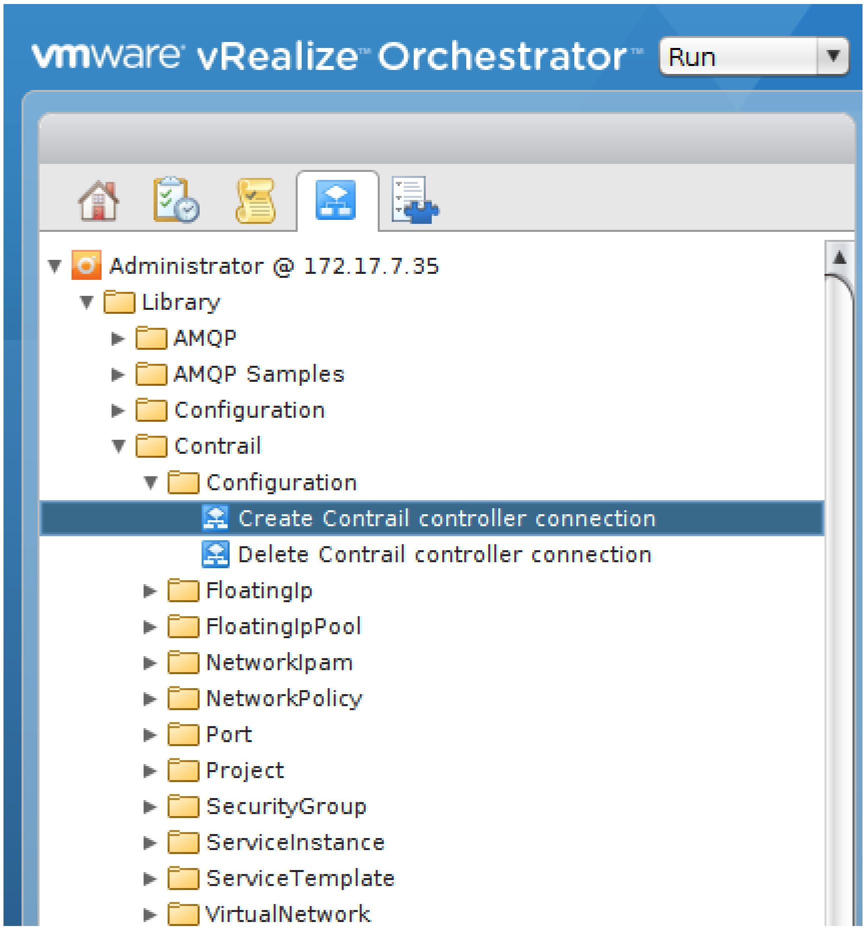
Connecting to Contrail Controller

To connect Contrail vRO to the Contrail Controller:

1. Navigate to the **Contrail > Configuration** folder in the workflow library. See Figure 39 on page 171.

2. Select **Create Contrail controller connection**.

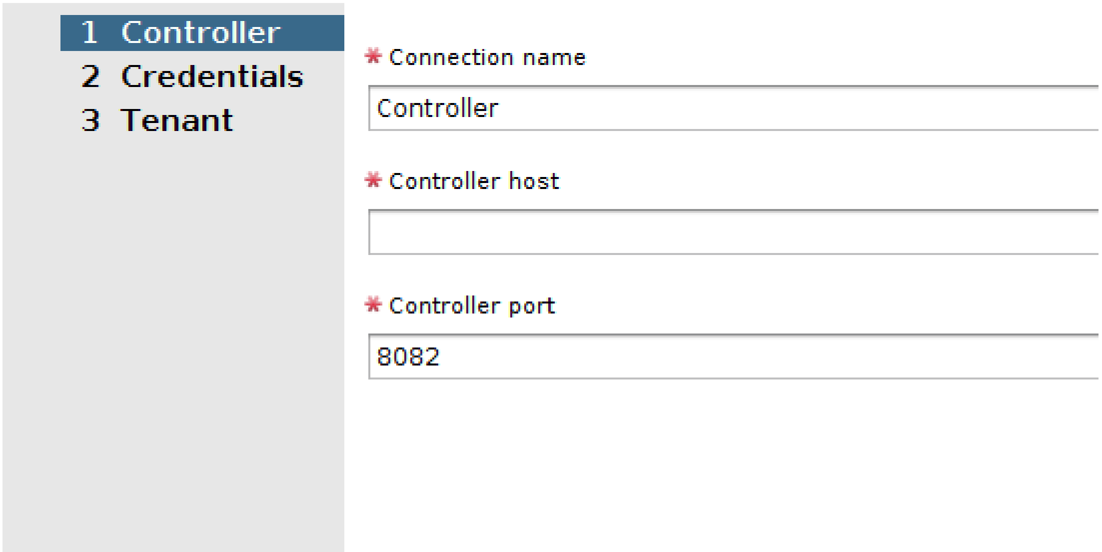
Figure 39: Workflow Library



3. Click the **Controller** tab and enter the following information:

   - **Connection name**—a unique name to identify the connection
   - **Controller host**—host name of the Contrail Connector
   - **Controller port**—port used to access the Contrail Controller

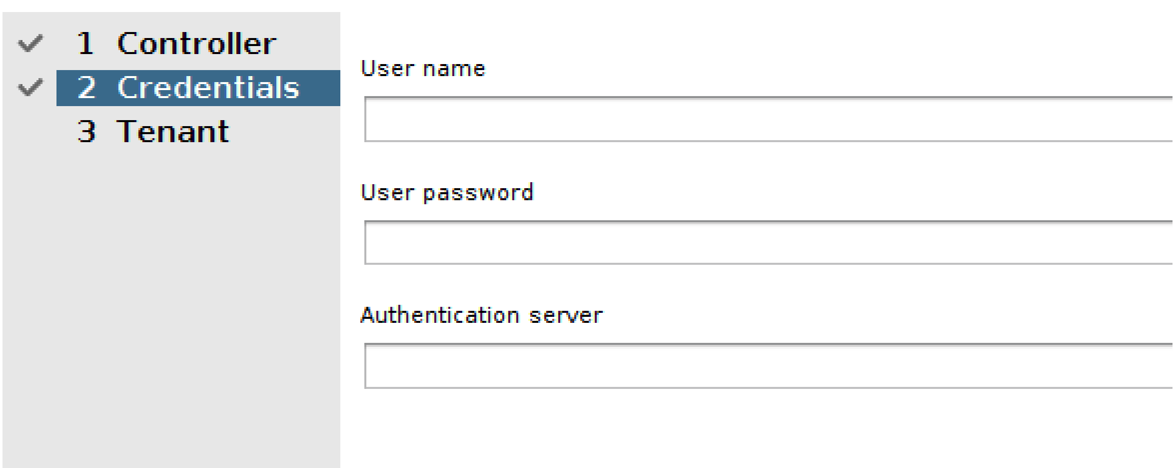
Figure 40: Controller Tab



4. Click the **Credentials** tab and enter the following credentials to manage the Contrail Controller:

   - **User name**—user name to access the Contrail Controller
   - **User password**—password to access the Contrail Controller
   - **Authentication server**—URL of the authentication server

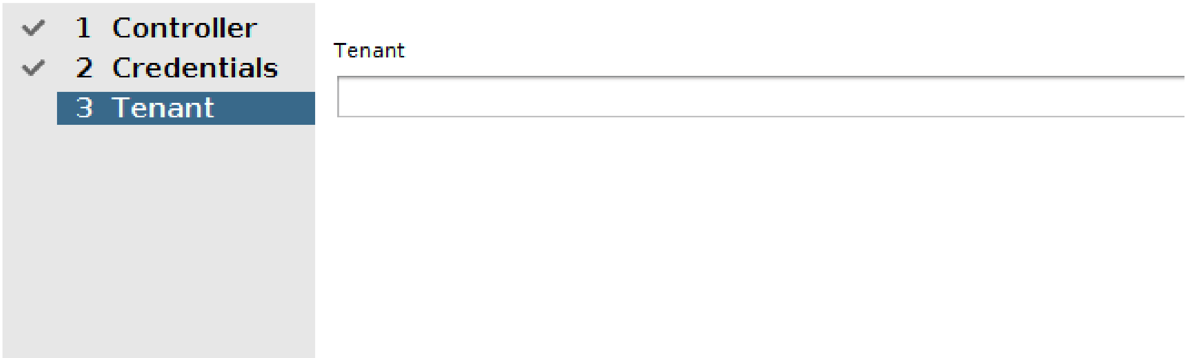
Figure 41: Credentials Tab



5. Click the **Tenant** tab to define tenant information.

   In the **Tenant** field, enter the name of the Contrail tenant.

Figure 42: Tenant Tab



6. Click **Submit** to establish connection.

   Once you connect Contrail vRO to the Contrail Controller, you use Contrail workflows to make configuration changes to Contrail.

Deploying Contrail vRO Plugin

You can deploy the Contrail plugin in any Java Virtual Machine (JVM) compatible environment and load it on an active vRO instance.

RELATED DOCUMENTATION

Integrating Contrail Release 5.0.X with VMware vRealize Orchestrator

CHAPTER 9

Using Contrail with Red Hat OpenStack

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

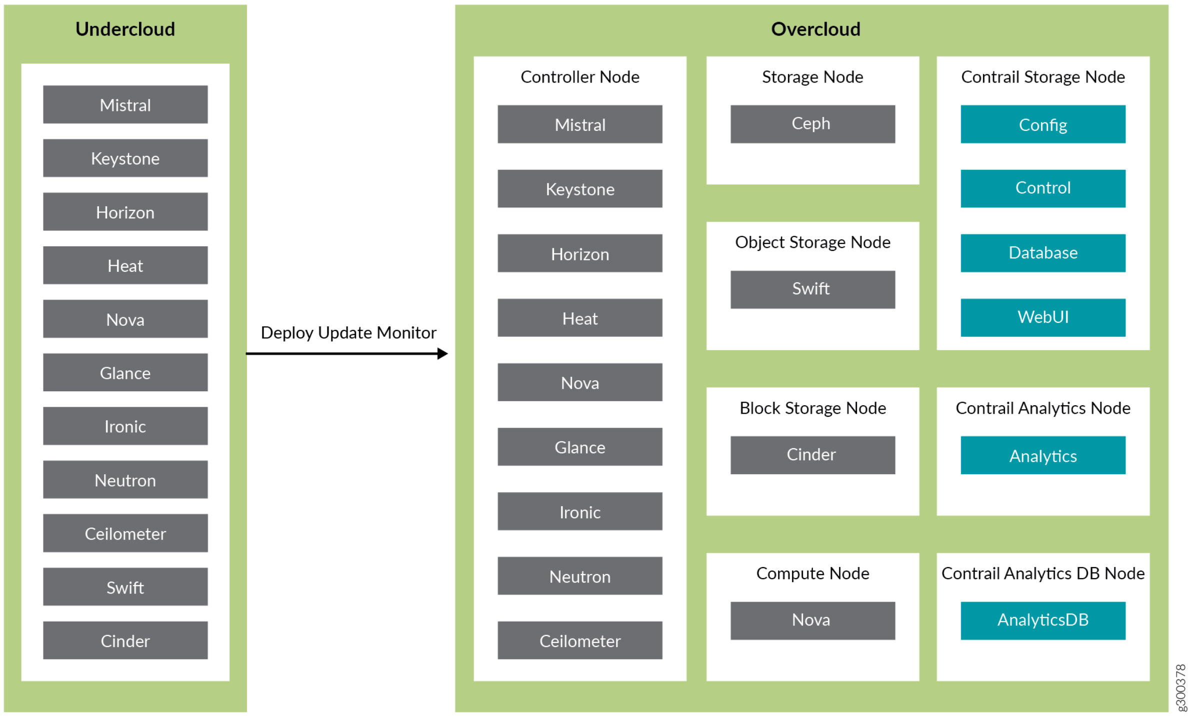
OSPd supports composable roles, which are groups of services that you define through Heat templates. Composable roles allow you to integrate Contrail into the overcloud environment.

The following are the Contrail roles used for integrating into the overcloud:

- Contrail Controller
- Contrail Analytics
- Contrail Analytics Database
- Contrail-TSN
- Contrail-DPDK

Figure 43 on page 176 shows the relationship and components of an undercloud and overcloud architecture for Contrail.

Figure 43: 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 4: 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 5 on page 180).

Table 5: 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 44 on page 180 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 44: Physical View

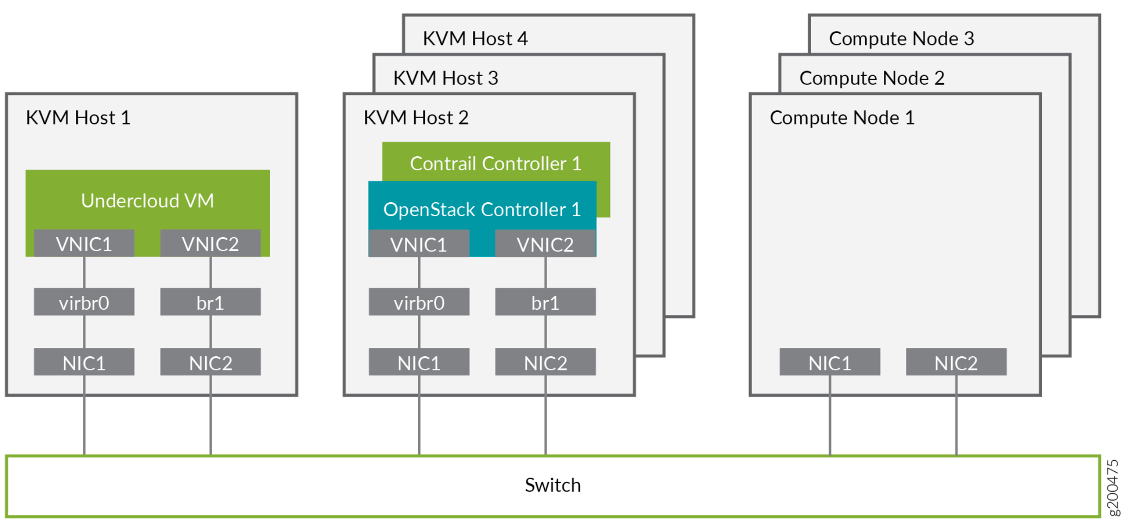
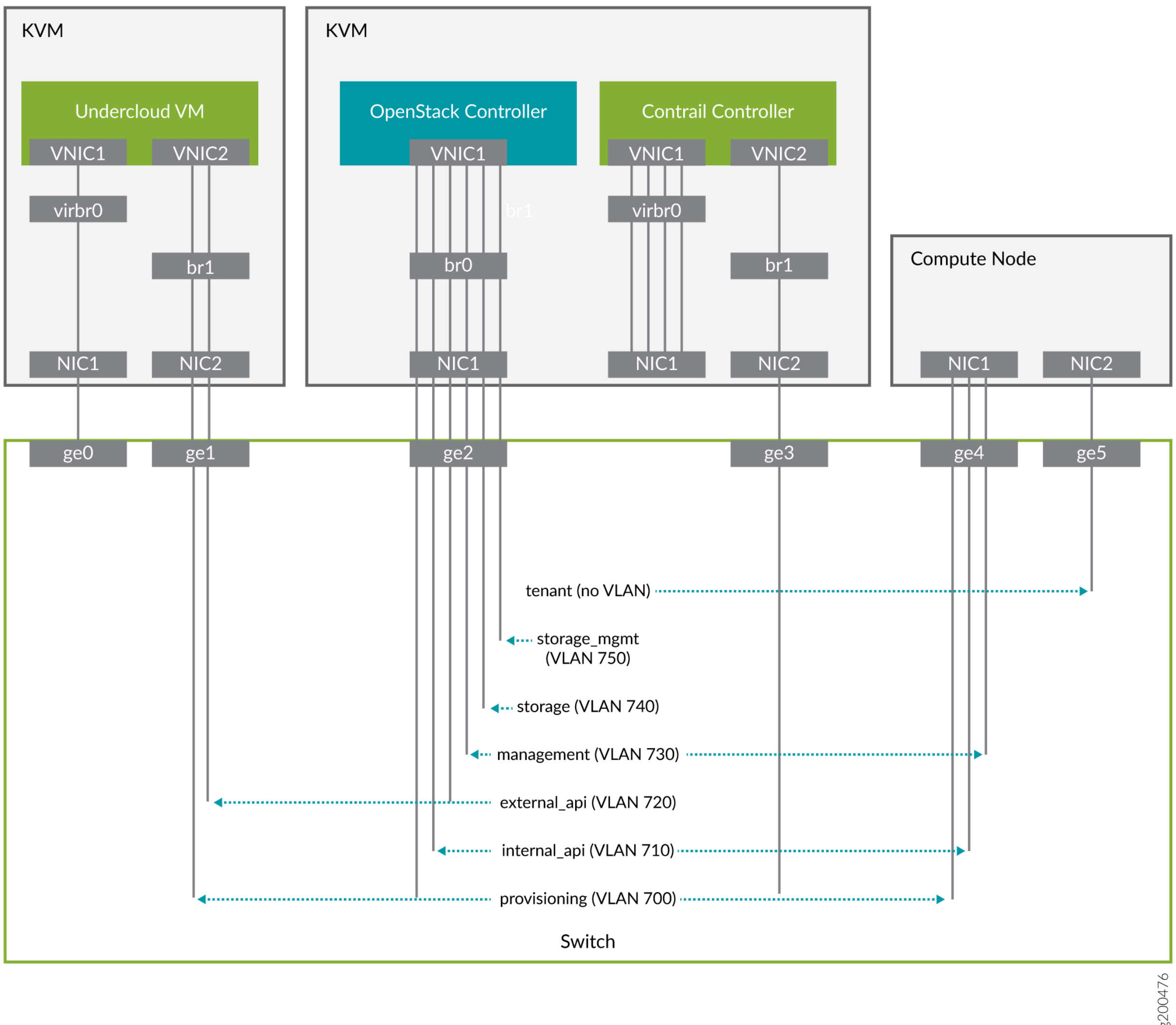


Figure 45 on page 181 shows the logical view of the connectivity where VLANs are used to provide further network separation for the different OpenStack network types.

Figure 45: 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 6: External Physical Switch Port and VLAN Configuration

Port	Trunked VLAN	Native VLAN
ge0	-	-

Table 6: 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 7: 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
```

4. Change admin password for overcloud.

```
(undercloud) [stack@queensa contrail]$ pwd
/home/stack/tripleo-heat-templates/environments/contrail
```

```
(undercloud) [stack@queensa contrail]$ vi contrail-services.yaml
parameter_defaults:
  AdminPassword: <password>
```

5. 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
```

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
  ```

- 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 8: Network Customization

Network	VLAN	overcloud Nodes
provisioning	-	All
internal_api	710	All

Table 8: Network Customization *(Continued)*

Network	VLAN	overcloud Nodes
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

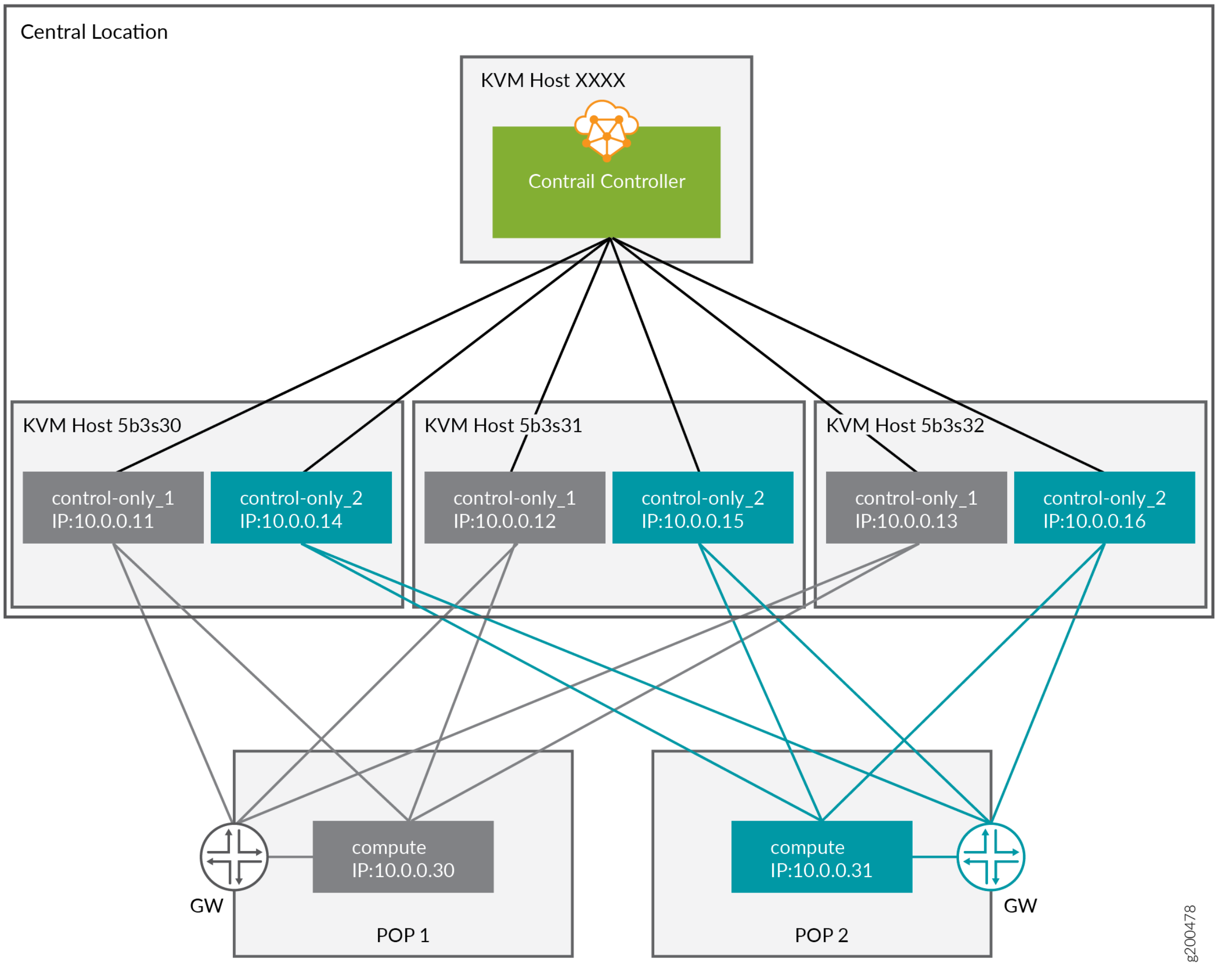
Remote Compute

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 9: 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 9: Mapping Table *(Continued)*

Nova Name	Ironic Name	UUID	KVM	IP Address	POP
overcloud -contrailcontrolonly -5	control-only-2-5b3s32	Ironic UUID: 58a803ae-a785-470e-9789-139abbfa74fb DMI UUID: AF92F485-C30C-4D0A-BDC4-C6AE97D06A66	5b3s32	10.0.0.16	POP2

ControlOnly preparation

Add ControlOnly overcloud VMs to overcloud KVM host

NOTE: This has to be done on the overcloud KVM hosts

Two ControlOnly overcloud VM definitions will be created on each of the overcloud KVM hosts.

```
ROLES=control-only:2
num=4
ipmi_user=<user>
ipmi_password=<password>
libvirt_path=/var/lib/libvirt/images
port_group=overcloud
prov_switch=br0

/bin/rm ironic_list
IFS=',' read -ra role_list <<< "${ROLES}"
for role in ${role_list[@]}; do
  role_name=`echo $role|cut -d ":" -f 1`
  role_count=`echo $role|cut -d ":" -f 2`
  for count in `seq 1 ${role_count}`; do
    echo $role_name $count
    qemu-img create -f qcow2 ${libvirt_path}/${role_name}_${count}.qcow2 99G
    virsh define /dev/stdin <<EOF
 $(virt-install --name ${role_name}_${count} \
--disk ${libvirt_path}/${role_name}_${count}.qcow2 \
```

```
--vcpus=4 \
--ram=16348 \
--network network=br0,model=virtio,portgroup=${port_group} \
--network network=br1,model=virtio \
--virt-type kvm \
--cpu host \
--import \
--os-variant rhel7 \
--serial pty \
--console pty,target_type=virtio \
--graphics vnc \
--print-xml)
EOF
    vbmc add ${role_name}_${count} --port 1623${num} --username ${ipmi_user} --password $
{ipmi_password}
    vbmc start ${role_name}_${count}
    prov_mac=`virsh domiflist ${role_name}_${count}|grep ${prov_switch}|awk '{print $5}'`
    vm_name=${role_name}-${count}-`hostname -s`
    kvm_ip=`ip route get 1  |grep src |awk '{print $7}'`
    echo ${prov_mac} ${vm_name} ${kvm_ip} ${role_name} 1623${num}>> ironic_list
    num=$(expr $num + 1)
  done
done
```

NOTE: The generated *ironic_list* will be needed on the undercloud to import the nodes to Ironic.

Get the ironic_lists from the overcloud KVM hosts and combine them.

```
cat ironic_list_control_only
52:54:00:3a:2f:ca control-only-1-5b3s30 10.87.64.31 control-only 16234
52:54:00:31:4f:63 control-only-2-5b3s30 10.87.64.31 control-only 16235
52:54:00:0c:11:74 control-only-1-5b3s31 10.87.64.32 control-only 16234
52:54:00:56:ab:55 control-only-2-5b3s31 10.87.64.32 control-only 16235
52:54:00:c1:f0:9a control-only-1-5b3s32 10.87.64.33 control-only 16234
52:54:00:f3:ce:13 control-only-2-5b3s32 10.87.64.33 control-only 16235
```

Import:

```
ipmi_password=<password>
ipmi_user=<user>

DEPLOY_KERNEL=$(openstack image show bm-deploy-kernel -f value -c id)
DEPLOY_RAMDISK=$(openstack image show bm-deploy-ramdisk -f value -c id)

num=0
while IFS= read -r line; do
  mac=`echo $line|awk '{print $1}'`
  name=`echo $line|awk '{print $2}'`
  kvm_ip=`echo $line|awk '{print $3}'`
  profile=`echo $line|awk '{print $4}'`
  ipmi_port=`echo $line|awk '{print $5}'`
  uuid=`openstack baremetal node create --driver ipmi \
                                        --property cpus=4 \
                                        --property memory_mb=16348 \
                                        --property local_gb=100 \
                                        --property cpu_arch=x86_64 \
                                        --driver-info ipmi_username=${ipmi_user}  \
                                        --driver-info ipmi_address=${kvm_ip} \
                                        --driver-info ipmi_password=${ipmi_password} \
                                        --driver-info ipmi_port=${ipmi_port} \
                                        --name=${name} \
                                        --property capabilities=boot_option:local \
                                        -c uuid -f value`
  openstack baremetal node set ${uuid} --driver-info deploy_kernel=$DEPLOY_KERNEL --driver-info
deploy_ramdisk=$DEPLOY_RAMDISK
  openstack baremetal port create --node ${uuid} ${mac}
  openstack baremetal node manage ${uuid}
  num=$(expr $num + 1)
done < <(cat ironic_list_control_only)
```

ControlOnly node introspection

```
openstack overcloud node introspect --all-manageable --provide
```

Get the ironic UUID of the ControlOnly nodes

```
openstack baremetal node list |grep control-only
| 7d758dce-2784-45fd-be09-5a41eb53e764 | control-only-1-5b3s30  | None | power off | available |
False |
| d26abdeb-d514-4a37-a7fb-2cd2511c351f | control-only-2-5b3s30  | None | power off | available |
False |
| 91dd9fa9-e8eb-4b51-8b5e-bbaffb6640e4 | control-only-1-5b3s31  | None | power off | available |
False |
| 09fa57b8-580f-42ec-bf10-a19573521ed4 | control-only-2-5b3s31  | None | power off | available |
False |
| f4766799-24c8-4e3b-af54-353f2b796ca4 | control-only-1-5b3s32  | None | power off | available |
False |
| 58a803ae-a785-470e-9789-139abbfa74fb | control-only-2-5b3s32  | None | power off | available |
False |
```

The first ControlOnly node on each of the overcloud KVM hosts will be used for POP1, the second for POP2, and so and so forth.

Get the ironic UUID of the POP compute nodes:

```
openstack baremetal node list |grep compute
| 91d6026c-b9db-49cb-a685-99a63da5d81e | compute-3-5b3s30 | None | power off | available | False
|
| 8028eb8c-e1e6-4357-8fcf-0796778bd2f7 | compute-4-5b3s30 | None | power off | available | False
|
| b795b3b9-c4e3-4a76-90af-258d9336d9fb | compute-3-5b3s31 | None | power off | available | False
|
| 2d4be83e-6fcc-4761-86f2-c2615dd15074 | compute-4-5b3s31 | None | power off | available | False
|
```

The first two compute nodes belong to POP1 the second two compute nodes belong to POP2.

Create an input YAML using the ironic UUIDs:

```
 ~/subcluster_input.yaml
---
- subcluster: subcluster1
  asn: "65413"
  control_nodes:
    - uuid: 7d758dce-2784-45fd-be09-5a41eb53e764
```

```
        ipaddress: 10.0.0.11
      - uuid: 91dd9fa9-e8eb-4b51-8b5e-bbaffb6640e4
        ipaddress: 10.0.0.12
      - uuid: f4766799-24c8-4e3b-af54-353f2b796ca4
        ipaddress: 10.0.0.13
    compute_nodes:
      - uuid: 91d6026c-b9db-49cb-a685-99a63da5d81e
        vrouter_gateway: 10.0.0.1
      - uuid: 8028eb8c-e1e6-4357-8fcf-0796778bd2f7
        vrouter_gateway: 10.0.0.1
  - subcluster: subcluster2
    asn: "65414"
    control_nodes:
      - uuid: d26abdeb-d514-4a37-a7fb-2cd2511c351f
        ipaddress: 10.0.0.14
      - uuid: 09fa57b8-580f-42ec-bf10-a19573521ed4
        ipaddress: 10.0.0.15
      - uuid: 58a803ae-a785-470e-9789-139abbfa74fb
        ipaddress: 10.0.0.16
    compute_nodes:
      - uuid: b795b3b9-c4e3-4a76-90af-258d9336d9fb
        vrouter_gateway: 10.0.0.1
      - uuid: 2d4be83e-6fcc-4761-86f2-c2615dd15074
        vrouter_gateway: 10.0.0.1
```

NOTE: Only control_nodes, compute_nodes, dpdk_nodes and sriov_nodes are supported.

Generate subcluster environment:

```
~/tripleo-heat-templates/tools/contrail/create_subcluster_environment.py -i ~/
subcluster_input.yaml \
               -o ~/tripleo-heat-templates/environments/contrail/contrail-subcluster.yaml
```

Check subcluster environment file:

```
cat ~/tripleo-heat-templates/environments/contrail/contrail-subcluster.yaml
parameter_defaults:
  NodeDataLookup:
    041D7B75-6581-41B3-886E-C06847B9C87E:
```

```
    contrail_settings:
      CONTROL_NODES: 10.0.0.14,10.0.0.15,10.0.0.16
      SUBCLUSTER: subcluster2
      VROUTER_GATEWAY: 10.0.0.1
  09BEC8CB-77E9-42A6-AFF4-6D4880FD87D0:
    contrail_settings:
      BGP_ASN: '65414'
      SUBCLUSTER: subcluster2
  14639A66-D62C-4408-82EE-FDDC4E509687:
    contrail_settings:
      BGP_ASN: '65414'
      SUBCLUSTER: subcluster2
  28AB0B57-D612-431E-B177-1C578AE0FEA4:
    contrail_settings:
      BGP_ASN: '65413'
      SUBCLUSTER: subcluster1
  3993957A-ECBF-4520-9F49-0AF6EE1667A7:
    contrail_settings:
      BGP_ASN: '65413'
      SUBCLUSTER: subcluster1
  73F8D030-E896-4A95-A9F5-E1A4FEBE322D:
    contrail_settings:
      BGP_ASN: '65413'
      SUBCLUSTER: subcluster1
  7933C2D8-E61E-4752-854E-B7B18A424971:
    contrail_settings:
      CONTROL_NODES: 10.0.0.14,10.0.0.15,10.0.0.16
      SUBCLUSTER: subcluster2
      VROUTER_GATEWAY: 10.0.0.1
  AF92F485-C30C-4D0A-BDC4-C6AE97D06A66:
    contrail_settings:
      BGP_ASN: '65414'
      SUBCLUSTER: subcluster2
  BB9E9D00-57D1-410B-8B19-17A0DA581044:
    contrail_settings:
      CONTROL_NODES: 10.0.0.11,10.0.0.12,10.0.0.13
      SUBCLUSTER: subcluster1
      VROUTER_GATEWAY: 10.0.0.1
  E1A809DE-FDB2-4EB2-A91F-1B3F75B99510:
    contrail_settings:
      CONTROL_NODES: 10.0.0.11,10.0.0.12,10.0.0.13
```

```
        SUBCLUSTER: subcluster1
        VROUTER_GATEWAY: 10.0.0.1
```

Deployment

Add contrail-subcluster.yaml, contrail-ips-from-pool-all.yaml and contrail-scheduler-hints.yaml to the OpenStack deploy command:

```
openstack overcloud deploy --templates ~/tripleo-heat-templates \
 -e ~/overcloud_images.yaml \
 -e ~/tripleo-heat-templates/environments/network-isolation.yaml \
 -e ~/tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
 -e ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
 -e ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
 -e ~/tripleo-heat-templates/environments/contrail/contrail-subcluster.yaml \
 -e ~/tripleo-heat-templates/environments/contrail/contrail-ips-from-pool-all.yaml \
 -e ~/tripleo-heat-templates/environments/contrail/contrail-scheduler-hints.yaml \
 --roles-file ~/tripleo-heat-templates/roles_data_contrail_aio.yaml
```

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

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 174
   - "Setting Up the Infrastructure" on page 179
   - "Setting Up the Undercloud" on page 188
   - "Setting Up the Overcloud" on page 191

   NOTE: Do not perform steps for "Installing Overcloud" on page 235.

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 235 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 | **174**

Setting Up the Infrastructure | **179**

Setting Up the Undercloud | **188**

Setting Up the Overcloud | **191**

CHAPTER 10

Using Contrail with Red Hat OpenShift

IN THIS CHAPTER

Installing a Standalone Red Hat OpenShift Container Platform 3.11 Cluster with Contrail Using Contrail OpenShift Deployer

You can install Contrail Networking together with a standalone Red Hat OpenShift Container Platform 3.11 cluster using Contrail OpenShift deployer. Consider the topology illustrated here.

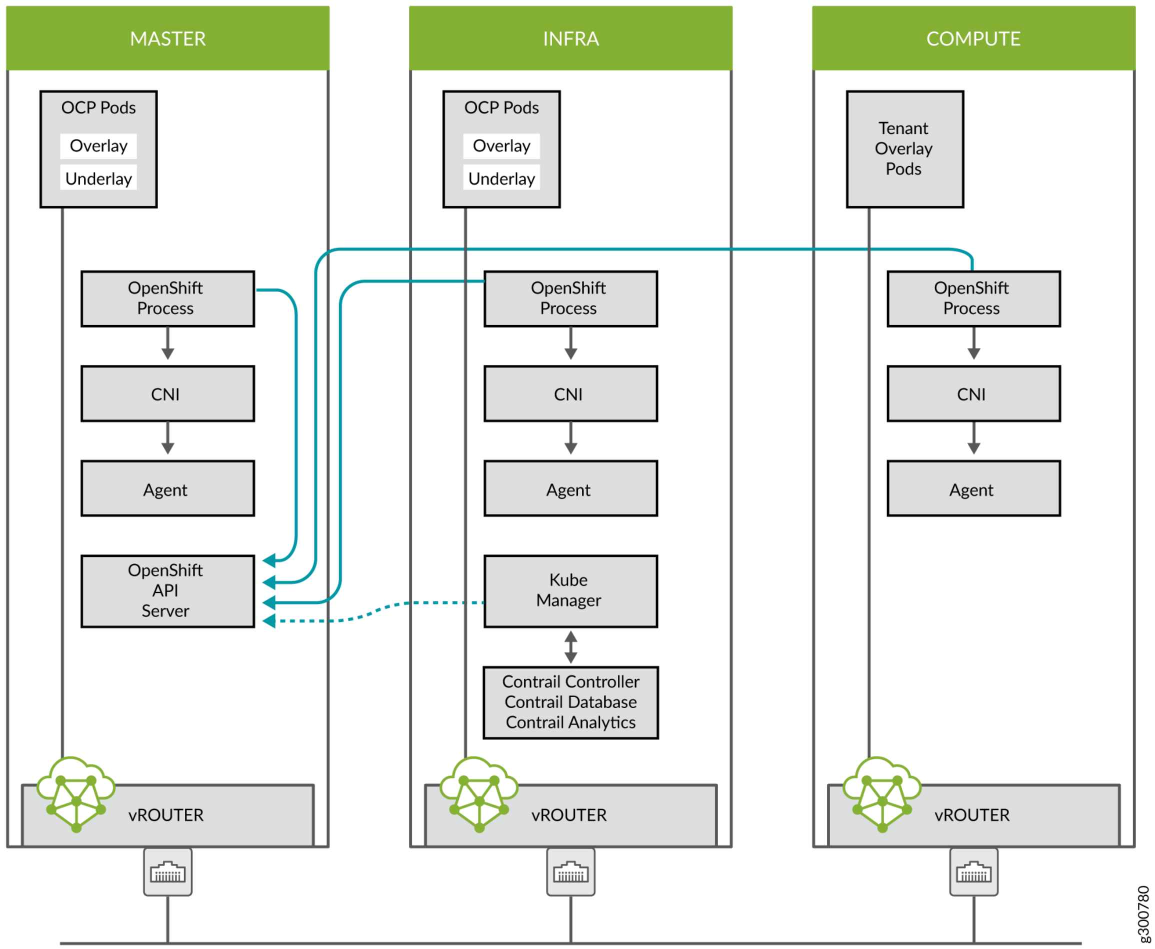
Prerequisites

The recommended system requirements are:

System Requirements	Master Node	Infrastructure Node	Compute Node
CPU/RAM	8 vCPU, 16 GB RAM	16 vCPU, 64 GB RAM	As per OpenShift recommendations.
Disk	100 GB	250 GB	

NOTE: If you use NFS mount volumes, check disk capacity and mounts. Also, openshift-logging with NFS is not recommended.

Figure 46: Sample installation topology



Perform the following steps to install a standalone OpenShift 3.11 cluster along with Contrail Networking using contrail-openshift-deployer.

1. Set up environment nodes for RHEL OpenShift enterprise installations:

   a. Subscribe to RHEL.

   ```
   (all-nodes)# subscription-manager register --username <> --password <> --force
   ```

   b. From the list of available subscriptions, find and attach the pool ID for the OpenShift Container Platform subscription.

   ```
   (all-nodes)# subscription-manager attach --pool=pool-ID
   ```

   c. Disable all yum repositories.

   ```
   (all-nodes)# subscription-manager repos --disable="*"
   ```

d. Enable only the required repositories.

```
(all-nodes)# subscription-manager repos \
   --enable="rhel-7-server-rpms" \
   --enable="rhel-7-server-extras-rpms" \
   --enable="rhel-7-server-ose-3.11-rpms" \
   --enable=rhel-7-fast-datapath-rpms \
   --enable="rhel-7-server-ansible-2.6-rpms"
```

e. Install required packages, such as python-netaddr, iptables-services, and so on.

`(all-nodes)# yum install -y tcpdump wget git net-tools bind-utils yum-utils iptables-services bridge-utils bash-completion kexec-tools sos psacct python-netaddr openshift-ansible`

NOTE: CentOS OpenShift Origin installations are not supported.

2. Get the files from the latest tar ball. Download the OpenShift Container Platform install package from Juniper software download site and modify the contents of the `openshift-ansible` inventory file.

   a. Download the Openshift Deployer (**contrail-openshift-deployer-*release-tag*.tgz**) installer from the Juniper software download site, https://www.juniper.net/support/downloads/?p=contrail#sw. See README Access for Contrail Networking Registry 19xx for appropriate release tags.

   b. Copy the install package to the node from where Ansible is deployed. Ensure that the node has password-free access to the OpenShift master and slave nodes.

   `scp contrail-openshift-deployer-`*`release-tag`*`.tgz` *`openshift-ansible-node`*`:/root/`

   c. Log in to the Ansible node and untar the `contrail-openshift-deployer-`*`release-tag`*`.tgz` package.

   `tar -xzvf contrail-openshift-deployer-`*`release-tag`*`.tgz -C /root/`

   d. Verify the contents of the **openshift-ansible** directory.

   `cd /root/openshift-ansible/`

   e. Modify the **inventory/ose-install** file to match your OpenShift environment.

   Populate the **inventory/ose-install** file with Contrail configuration parameters specific to your system. The following mandatory parameters must be set. For example:

```
contrail_version=5.1
contrail_container_tag=<>
contrail_registry="hub.juniper.net/contrail-nightly"
contrail_registry_username=<>
contrail_registry_password=<>
```

```
openshift_use_openshift_sdn=false
os_sdn_network_plugin_name='cni'
openshift_use_contrail=true
```

NOTE: The `contrail_container_tag` value for this release can be found in the README Access to Contrail Registry 19XX file.

Juniper Networks recommends that you obtain the Ansible source files from the latest release.

This procedure assumes that there is one master node, one infrastructure node, and one compute node.

```
master : server1 (1x.xx.xx.11)
infrastructure : server2 (1x.xx.xx.22)
compute : server3 (1x.xx.xx.33)
```

3. Edit **/etc/hosts** to include all the nodes information.

```
[root@server1]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
1x.xx.xx.100 puppet
1x.xx.xx.11 server1.contrail.juniper.net server1
1x.xx.xx.22 server2.contrail.juniper.net server2
1x.xx.xx.33 server3.contrail.juniper.net server3
```

4. Set up password-free SSH access to the Ansible node and all the nodes.

```
ssh-keygen -t rsa
ssh-copy-id root@1x.xx.xx.11
ssh-copy-id root@1x.xx.xx.22
ssh-copy-id root@1x.xx.xx.33
```

5. Run Ansible playbook to install OpenShift Container Platform with Contrail. Before you run Ansible playbook, ensure that you have edited **inventory/ose-install** file.

```
(ansible-node)# cd /root/openshift-ansible
(ansible-node)# ansible-playbook -i inventory/ose-install playbooks/prerequisites.yml
(ansible-node)# ansible-playbook -i inventory/ose-install playbooks/deploy_cluster.yml
```

   For a sample **inventory/ose-install** file, see "No Link Title" on page 244.

6. Create a password for the admin user to log in to the UI from the master node.

```
(master-node)# htpasswd /etc/origin/master/htpasswd admin
```

   NOTE: If you are using a load balancer, you must manually copy the htpasswd file into all your master nodes.

7. Assign cluster-admin role to admin user.

```
(master-node)# oc adm policy add-cluster-role-to-user cluster-admin admin
(master-node)# oc login -u admin
```

8. Open a Web browser and type the entire fqdn name of your master node or load balancer node, followed by `:8443/console`.

```
https://<your host name from your ose-install inventory>:8443/console
```

   Use the user name and password created in step 6 to log in to the Web console.

   Your DNS should resolve the host name for access. If the host name is not resolved, modify the /etc/hosts file to route to the above host.

NOTE: OpenShift 3.11 cluster upgrades are not supported.

Sample inventory/ose-install File

```
[OSEv3:vars]

###########################################################################
```

```
### OpenShift Basic Vars
###########################################################################
openshift_deployment_type=openshift-enterprise
deployment_type=openshift-enterprise
containerized=false
openshift_disable_check=docker_image_availability,memory_availability,package_availability,disk_a
vailability,package_version,docker_storage

# Default node selectors
openshift_hosted_infra_selector="node-role.kubernetes.io/infra=true"

oreg_auth_user=<>
oreg_auth_password=<>

###########################################################################
### OpenShift Master Vars
###########################################################################

openshift_master_api_port=8443
openshift_master_console_port=8443
openshift_master_cluster_method=native

# Set this line to enable NFS
openshift_enable_unsupported_configurations=True


###########################################################################
### OpenShift Network Vars
###########################################################################

openshift_use_openshift_sdn=false
os_sdn_network_plugin_name='cni'
openshift_use_contrail=true

###########################################################################
### OpenShift Authentication Vars
###########################################################################

# htpasswd Authentication
openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge':
'true', 'kind': 'HTPasswdPasswordIdentityProvider'}]

###########################################################################
```

```
### OpenShift Router and Registry Vars
###########################################################################

openshift_hosted_router_replicas=1
openshift_hosted_registry_replicas=1

openshift_hosted_registry_storage_kind=nfs
openshift_hosted_registry_storage_access_modes=['ReadWriteMany']
openshift_hosted_registry_storage_nfs_directory=/export
openshift_hosted_registry_storage_nfs_options='*(rw,root_squash)'
openshift_hosted_registry_storage_volume_name=registry
openshift_hosted_registry_storage_volume_size=10Gi
openshift_hosted_registry_pullthrough=true
openshift_hosted_registry_acceptschema2=true
openshift_hosted_registry_enforcequota=true
openshift_hosted_router_selector="node-role.kubernetes.io/infra=true"
openshift_hosted_registry_selector="node-role.kubernetes.io/infra=true"

###########################################################################
### OpenShift Service Catalog Vars
###########################################################################

openshift_enable_service_catalog=True

template_service_broker_install=True
openshift_template_service_broker_namespaces=['openshift']

ansible_service_broker_install=True

openshift_hosted_etcd_storage_kind=nfs
openshift_hosted_etcd_storage_nfs_options="*(rw,root_squash,sync,no_wdelay)"
openshift_hosted_etcd_storage_nfs_directory=/export
openshift_hosted_etcd_storage_labels={'storage': 'etcd-asb'}
openshift_hosted_etcd_storage_volume_name=etcd-asb
openshift_hosted_etcd_storage_access_modes=['ReadWriteOnce']
openshift_hosted_etcd_storage_volume_size=2G




###########################################################################
### OpenShift Metrics and Logging Vars
```

```
###########################################################################
# Enable cluster metrics
openshift_metrics_install_metrics=True

openshift_metrics_storage_kind=nfs
openshift_metrics_storage_access_modes=['ReadWriteOnce']
openshift_metrics_storage_nfs_directory=/export
openshift_metrics_storage_nfs_options='*(rw,root_squash)'
openshift_metrics_storage_volume_name=metrics
openshift_metrics_storage_volume_size=2Gi
openshift_metrics_storage_labels={'storage': 'metrics'}

openshift_metrics_cassandra_nodeselector={"node-role.kubernetes.io/infra":"true"}
openshift_metrics_hawkular_nodeselector={"node-role.kubernetes.io/infra":"true"}
openshift_metrics_heapster_nodeselector={"node-role.kubernetes.io/infra":"true"}

# Enable cluster logging. ((
####openshift_logging_install_logging=True
openshift_logging_install_logging=False
#openshift_logging_storage_kind=nfs
#openshift_logging_storage_access_modes=['ReadWriteOnce']
#openshift_logging_storage_nfs_directory=/export
#openshift_logging_storage_nfs_options='*(rw,root_squash)'
#openshift_logging_storage_volume_name=logging
#openshift_logging_storage_volume_size=5Gi
#openshift_logging_storage_labels={'storage': 'logging'}
#openshift_logging_es_cluster_size=1
#openshift_logging_es_nodeselector={"node-role.kubernetes.io/infra":"true"}
#openshift_logging_kibana_nodeselector={"node-role.kubernetes.io/infra":"true"}
#openshift_logging_curator_nodeselector={"node-role.kubernetes.io/infra":"true"}

###########################################################################
### OpenShift Prometheus Vars
###########################################################################

## Add Prometheus Metrics:
openshift_hosted_prometheus_deploy=True
openshift_prometheus_node_selector={"node-role.kubernetes.io/infra":"true"}
openshift_prometheus_namespace=openshift-metrics

# Prometheus
openshift_prometheus_storage_kind=nfs
openshift_prometheus_storage_access_modes=['ReadWriteOnce']
```

```
openshift_prometheus_storage_nfs_directory=/export
openshift_prometheus_storage_nfs_options='*(rw,root_squash)'
openshift_prometheus_storage_volume_name=prometheus
openshift_prometheus_storage_volume_size=1Gi
openshift_prometheus_storage_labels={'storage': 'prometheus'}
openshift_prometheus_storage_type='pvc'

# For prometheus-alertmanager
openshift_prometheus_alertmanager_storage_kind=nfs
openshift_prometheus_alertmanager_storage_access_modes=['ReadWriteOnce']
openshift_prometheus_alertmanager_storage_nfs_directory=/export
openshift_prometheus_alertmanager_storage_nfs_options='*(rw,root_squash)'
openshift_prometheus_alertmanager_storage_volume_name=prometheus-alertmanager
openshift_prometheus_alertmanager_storage_volume_size=1Gi
openshift_prometheus_alertmanager_storage_labels={'storage': 'prometheus-alertmanager'}
openshift_prometheus_alertmanager_storage_type='pvc'

# For prometheus-alertbuffer
openshift_prometheus_alertbuffer_storage_kind=nfs
openshift_prometheus_alertbuffer_storage_access_modes=['ReadWriteOnce']
openshift_prometheus_alertbuffer_storage_nfs_directory=/export
openshift_prometheus_alertbuffer_storage_nfs_options='*(rw,root_squash)'
openshift_prometheus_alertbuffer_storage_volume_name=prometheus-alertbuffer
openshift_prometheus_alertbuffer_storage_volume_size=1Gi
openshift_prometheus_alertbuffer_storage_labels={'storage': 'prometheus-alertbuffer'}
openshift_prometheus_alertbuffer_storage_type='pvc'


#########################################################################
### Openshift HA
#########################################################################

# Openshift HA
openshift_master_cluster_hostname=load-balancer-0-3eba0c20dc494dfc93d5d50d06bbde89
openshift_master_cluster_public_hostname=load-balancer-0-3eba0c20dc494dfc93d5d50d06bbde89


#########################################################################
### Contrail Variables
########################################################################

service_subnets="172.30.0.0/16"
pod_subnets="10.128.0.0/14"
```

```
# Below are Contrail variables. Comment them out if you don't want to install Contrail through
ansible-playbook
contrail_version=5.1
contrail_container_tag=<>
contrail_registry=hub.juniper.net/contrail
contrail_registry_username=<>
contrail_registry_password=<>
openshift_docker_insecure_registries=hub.juniper.net/contrail
contrail_nodes=[10.0.0.5,10.0.0.3,10.0.0.4]
vrouter_physical_interface=eth0


###########################################################################
### OpenShift Hosts
###########################################################################
[OSEv3:children]
masters
etcd
nodes
lb
nfs
openshift_ca

[masters]
kube-master-2-3eba0c20dc494dfc93d5d50d06bbde89
kube-master-1-3eba0c20dc494dfc93d5d50d06bbde89
kube-master-0-3eba0c20dc494dfc93d5d50d06bbde89

[etcd]
kube-master-2-3eba0c20dc494dfc93d5d50d06bbde89
kube-master-1-3eba0c20dc494dfc93d5d50d06bbde89
kube-master-0-3eba0c20dc494dfc93d5d50d06bbde89

[lb]
load-balancer-0-3eba0c20dc494dfc93d5d50d06bbde89

[nodes]
kube-master-2-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-master'
controller-0-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-infra'
compute-1-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-compute'
controller-2-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-infra'
kube-master-1-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-master'
```

```
kube-master-0-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-master'
compute-0-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-compute'
controller-1-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-infra'

[nfs]
load-balancer-0-3eba0c20dc494dfc93d5d50d06bbde89

[openshift_ca]
kube-master-2-3eba0c20dc494dfc93d5d50d06bbde89
kube-master-1-3eba0c20dc494dfc93d5d50d06bbde89
kube-master-0-3eba0c20dc494dfc93d5d50d06bbde89
```

NOTE: The /etc/resolv.conf must have write permissions.

Caveats and Troubleshooting Instructions

- If a Java error occurs, install the `yum install java-1.8.0-openjdk-devel.x86_64` package and rerun `deploy_cluster`.
- If the service_catalog parameter does not pass but the cluster is operational, check whether the **/etc/resolv.conf** has `cluster.local` in its search line, and the nameserver as host IP address.
- NTP is installed by OpenShift and must be synchronized by the user. This does not affect any Contrail functionality but is displayed in the `contrail-status` output.
- If the `ansible_service_broker` component of OpenShift is not up and its `ansible_service_broker_deploy` displays an error, it means that the `ansible_service_broker` pod did not come up properly. The most likely reason is that the `ansible_service_broker` pod failed its liveliness and readiness checks. Modify the liveliness and readiness checks of this pod when it's brought online to make it operational. Also, verify that the `ansible_service_broker` pod uses the correct URL from Red Hat.

RELATED DOCUMENTATION

Installing a Nested Red Hat OpenShift Container Platform 3.11 Cluster Using Contrail Ansible Deployer

Installing a Nested Red Hat OpenShift Container Platform 3.11 Cluster Using Contrail Ansible Deployer

You can install a nested Red Hat OpenShift Container Platform 3.11 cluster along with Contrail Networking using Contrail Ansible deployer.

Prerequisites

Ensure that the following prerequisites are met for a successful provisioning of a nested Contrail-OpenShift cluster.

- The recommended system requirements are:

System Requirements	Master Node	Infrastructure Node	Compute Node
CPU/RAM	8 vCPU, 16 GB RAM	16 vCPU, 64 GB RAM	As per OpenShift recommendations.
Disk	100 GB	250 GB	

- A running Red Hat OpenStack Platform Director (RHOSPD) 13 cluster with Contrail. OpenShift Contrail release must be same as RHOSPD 13 Contrail release.

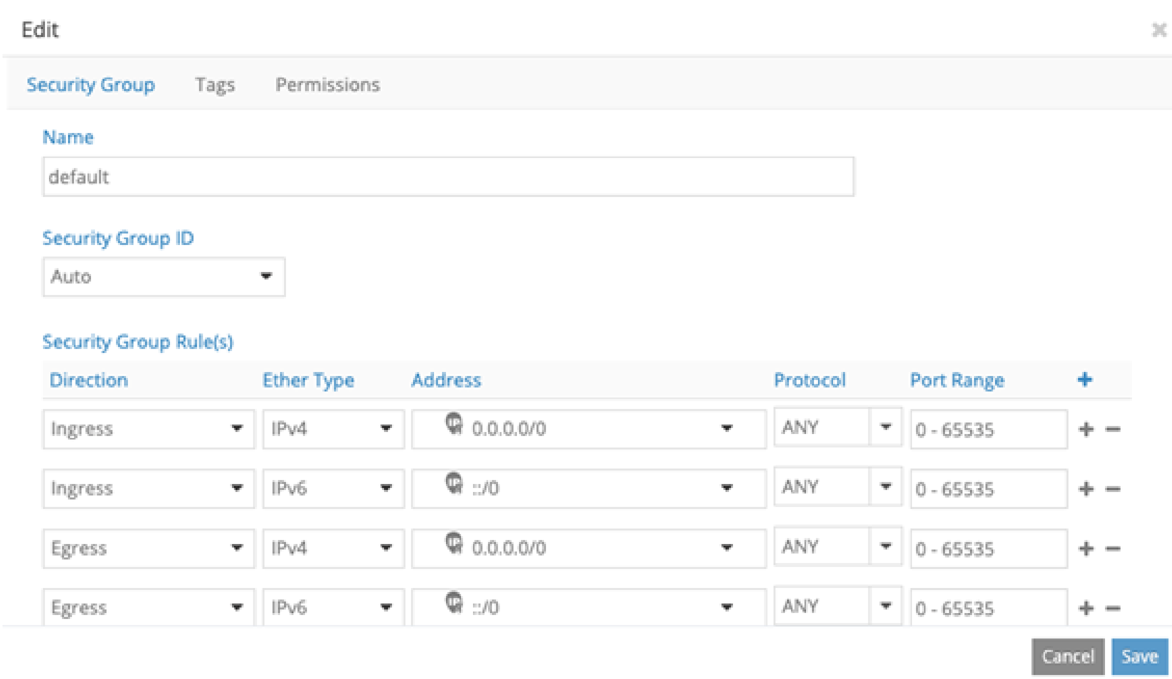
- RHOSPD environments require that the Contrail vrouter, Contrail config and OpenStack keystone are in “internal-api” network. Modify the ServiceNetMap parameters in the **contrail-services.yaml** file to configure in “internal-api” network.

```
parameter_defaults:
  ServiceNetMap:
    ContrailDatabaseNetwork: internal_api
    ContrailAnalyticsNetwork: internal_api
    ContrailAnalyticsAlarmNetwork: internal_api
    ContrailAnalyticsDatabaseNetwork: internal_api
    ContrailAnalyticsSnmpNetwork: internal_api
    ContrailConfigNetwork: internal_api
    ContrailControlNetwork: internal_api
    ContrailWebuiNetwork: internal_api
    ContrailVrouterNetwork: internal_api
    ContrailCertmongerUserNetwork: internal_api
    KeystoneAdminApiNetwork: internal_api
```

- Ensure that the vRouter gateway in the **contrail-services.yaml** file is part of “internal-api” network.

```
# Custom Contrail container configuration settings
  ContrailSettings:
    VROUTER_GATEWAY: 10.1.0.254
```

- OpenShift nodes (VMs) must have Internet connectivity.
- Default security group of the virtual-network where OpenShift nodes are launched must be modified to allow all ingress traffic to communicate with OpenShift networks provided in the OpenShift inventory file.



Provisioning Nested OpenShift Cluster

Provisioning a nested OpenShift cluster is a two-step process.

1. Create link-local services in the Contrail-OpenStack cluster.

   A nested OpenShift cluster is managed by the same Contrail controller that manages the underlying OpenStack cluster. Hence, the nested Openshift cluster needs IP reachability to the Contrail controller and OpenStack keystone service. Since the OpenShift cluster is actually an overlay on the OpenStack cluster, we use the Link Local Service feature of Contrail to provide IP reachability to and from the overlay OpenShift cluster and OpenStack cluster.

   To configure a Link Local Service, we need a Fabric IP and Service IP. Fabric IP is the node IP on which the Contrail Controller and OpenStack services are running. Service IP is a unique and unused IP in the entire OpenStack cluster and is shared with the OpenShift cluster to reach Contrail Controller and OpenStack services. Service IP (along with port number) is used by the data plane to identify the fabric IP. For each node of the OpenStack cluster, one service IP must be identified.

You must configure the following Link Local Services in Contrail.

Contrail Controller and OpenStack Process	Service IP	Service Port	Fabric IP	Fabric Port
Contrail Config	<Service IP for the running node>	8082	<Node IP of running node>	8082
Contrail Analytics	<Service IP for the running node>	8086	<Node IP of running node>	8086
Contrail Msg Queue	<Service IP for the running node>	5673	<Node IP of running node>	5673
Contrail VNC DB	<Service IP for the running node>	9161	<Node IP of running node>	9161
Keystone	<Service IP for the running node>	35357	<Node IP of running node>	35357
K8s-cni-to-agent	<Service IP for the running node>	9091	<Node IP of running node>	9091

For example, consider a sample cluster of seven nodes.

```
Contrail Config : 192.168.1.100
Contrail Analytics : 192.168.1.100, 192.168.1.101
Contrail Msg Queue : 192.168.1.100
Contrail VNC DB : 192.168.1.100, 192.168.1.101, 192.168.1.102
Keystone: 192.168.1.200
Vrouter: 192.168.1.201, 192.168.1.202, 192.168.1.203
```

Allocate seven unused IP addresses for the seven nodes.

```
192.168.1.100  --> 10.10.10.1
192.168.1.101  --> 10.10.10.2
192.168.1.102  --> 10.10.10.3
192.168.1.200  --> 10.10.10.4
192.168.1.201/192.168.1.202/192.168.1.203  --> 10.10.10.5
```

NOTE: One Service IP address can represent all vRouter nodes.

The following link-local services must be created:

Contrail controller and OpenStack process	Service IP	Service Port	Fabric IP	Fabric Port
Contrail Config	10.10.10.1	8082	192.168.1.100	8082
Contrail Analytics 1	10.10.10.1	8086	192.168.1.100	8086
Contrail Analytics 2	10.10.10.1	8086	192.168.1.101	8086
Contrail Msg Queue	10.10.10.2	5673	192.168.1.100	5673
Contrail VNC DB 1	10.10.10.1	9161	192.168.1.100	9161
Contrail VNC DB 2	10.10.10.2	9161	192.168.1.101	9161
Contrail VNC DB 3	10.10.10.2	9161	192.168.1.102	9161
Keystone	10.10.10.4	35357	192.168.1.200	35357
K8s-cni-to-agent	10.10.10.5	9091	127.0.0.1	9091

2. Install OpenShift using OpenShift Ansible deployer.

Perform the following steps to install the nested OpenShift 3.11 cluster along with Contrail Networking using OpenShift Ansible deployer.

a. Set up environment nodes for RHEL OpenShift enterprise installations:

   i. Subscribe to RHEL.

   ```
   (all-nodes)# subscription-manager register --username <> --password <> --force
   ```

   ii. From the list of available subscriptions, find and attach the pool ID for the OpenShift Container Platform subscription.

   ```
   (all-nodes)# subscription-manager attach --pool=pool-ID
   ```

   iii. Disable all yum repositories.

   ```
   (all-nodes)# subscription-manager repos --disable="*"
   ```

   iv. Enable only the required repositories.

   ```
   (all-nodes)# subscription-manager repos \
                   --enable="rhel-7-server-rpms" \
                   --enable="rhel-7-server-extras-rpms" \
                   --enable="rhel-7-server-ose-3.11-rpms" \
                   --enable=rhel-7-fast-datapath-rpms \
                   --enable="rhel-7-server-ansible-2.6-rpms"
   ```

   v. Install required packages, such as python-netaddr, iptables-services, and so on.

   ```
   (all-nodes)# yum install -y tcpdump wget git net-tools bind-utils yum-utils iptables-services
   bridge-utils bash-completion kexec-tools sos psacct python-netaddr openshift-ansible
   ```

   NOTE: CentOS OpenShift Origin installations are not supported.

b. Get the files from the latest tar ball. Download the OpenShift Container Platform install package from Juniper software download site and modify the contents of the `openshift-ansible` inventory file.

   i. Download Openshift Ansible (contrail-ansible-deployer-release-tag.tgz) installer from the Juniper software download site, https://www.juniper.net/support/downloads/?p=contrail#sw. See README Access for Contrail Networking Registry 19xx for appropriate release tags.

ii. Copy the install package to the node from where Ansible is deployed. Ensure that the node has password-free access to the OpenShift master and slave nodes.

```
scp contrail-ansible-deployer-release-tag.tgz openshift-ansible-node:/root/
```

iii. Log in to the Ansible node and untar the contrail-ansible-deployer-release-tag.tgz package.

```
tar -xzvf contrail-ansible-deployer-release-tag.tgz -C /root/
```

iv. Verify the contents of the **openshift-ansible** directory.

```
cd /root/openshift-ansible/
```

v. Modify the **inventory/ose-install** file to match your OpenShift environment.

Populate the **inventory/ose-install** file with Contrail configuration parameters specific to your system. The following mandatory parameters must be set.

```
contrail_version=1907
          contrail_container_tag=<>
          contrail_registry="hub.juniper.net/contrail"
          contrail_registry_username=<>
          contrail_registry_password=<>
          openshift_use_openshift_sdn=false
          os_sdn_network_plugin_name='cni'
          openshift_use_contrail=true
```

NOTE: The `contrail_container_tag` value for this release can be found in the README Access to Contrail Registry 19XX file.

i. Download Openshift Ansible (**contrail-ansible-deployer-*release-tag*.tgz**) installer from the Juniper software download site, https://www.juniper.net/support/downloads/?p=contrail#sw. See README Access for Contrail Networking Registry 19xx for appropriate release tags.

ii. Copy the install package to the node from where Ansible is deployed. Ensure that the node has password-free access to the OpenShift master and slave nodes.

```
scp contrail-ansible-deployer-release-tag.tgz openshift-ansible-node:/root/
```

iii. Log in to the Ansible node and untar the `contrail-ansible-deployer-release-tag.tgz` package.

```
tar -xzvf contrail-ansible-deployer-release-tag.tgz -C /root/
```

iv. Verify the contents of the **openshift-ansible** directory.

```
cd /root/openshift-ansible/
```

v. Modify the **inventory/ose-install** file to match your OpenShift environment.

Populate the **inventory/ose-install** file with Contrail configuration parameters specific to your system. The following mandatory parameters must be set.

```
contrail_version=1907
                      contrail_container_tag=<>
                      contrail_registry="hub.juniper.net/contrail"
                      contrail_registry_username=<>
                      contrail_registry_password=<>
                      openshift_use_openshift_sdn=false
                      os_sdn_network_plugin_name='cni'
                      openshift_use_contrail=true
```

NOTE: The `contrail_container_tag` value for this release can be found in the README Access to Contrail Registry 19XX file.

NOTE: Juniper Networks recommends that you obtain the Ansible source files from the latest release.

This procedure assumes that there is one master node, one infrastructure node, and one compute node.

```
master : server1 (1x.xx.xx.11)
infrastructure : server2 (1x.xx.xx.22)
compute : server3 (1x.xx.xx.33)
```

c. Edit **/etc/hosts** to include all the nodes information.

```
[root@server1]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
1x.xx.xx.100 puppet
```

```
1x.xx.xx.11 server1.contrail.juniper.net server1
1x.xx.xx.22 server2.contrail.juniper.net server2
1x.xx.xx.33 server3.contrail.juniper.net server3
```

d. Set up password-free SSH access to the Ansible node and all the nodes.

```
ssh-keygen -t rsa
ssh-copy-id root@1x.xx.xx.11
ssh-copy-id root@1x.xx.xx.22
ssh-copy-id root@1x.xx.xx.33
```

e. Run Ansible playbook to install OpenShift Container Platform with Contrail. Before you run Ansible playbook, ensure that you have edited **inventory/ose-install** file.

```
(ansible-node)# cd /root/openshift-ansible
(ansible-node)# ansible-playbook -i inventory/ose-install playbooks/prerequisites.yml
(ansible-node)# ansible-playbook -i inventory/ose-install playbooks/deploy_cluster.yml
```

For a sample **inventory/ose-install** file, see "No Link Title" on page 259.

f. Create a password for the admin user to log in to the UI from the master node.

```
(master-node)# htpasswd /etc/origin/master/htpasswd admin
```

NOTE: If you are using a load balancer, you must manually copy the htpasswd file into all your master nodes.

g. Assign cluster-admin role to admin user.

```
(master-node)# oc adm policy add-cluster-role-to-user cluster-admin admin
(master-node)# oc login -u admin
```

h. Open a Web browser and type the entire fqdn name of your master node or load balancer node, followed by :8443/console.

```
https://<your host name from your ose-install inventory>:8443/console
```

Use the user name and password created in step "2.f" on page 258 to log in to the Web console.

Your DNS should resolve the host name for access. If the host name is not resolved, modify the /etc/hosts file to route to the above host.

NOTE: OpenShift 3.11 cluster upgrades are not supported.

Sample inventory/ose-install File

```
[OSEv3:vars]


###########################################################################
### OpenShift Nested mode vars
###########################################################################
nested_mode_contrail=true
rabbitmq_node_port=5673
contrail_nested_masters_ip="1.1.1.1 2.2.2.2 3.3.3.3"          <---  ips of contrail controllers
auth_mode=keystone
keystone_auth_host=<w.x.y.z>        <--- This should be the IP where Keystone service is running.
keystone_auth_admin_tenant=admin
keystone_auth_admin_user=admin
keystone_auth_admin_password=MAYffWrX7ZpPrV2AMAa9zAUvG     <-- Keystone admin password.
keystone_auth_admin_port=35357
keystone_auth_url_version=/v3
#k8s_nested_vrouter_vip is a service IP for the running node which we configured above
k8s_nested_vrouter_vip=10.10.10.5   <-- Service IP configured for CNI to Agent communication.
(K8s-cni-to-agent in above examples)
#k8s_vip is kubernetes api server ip
k8s_vip=<W.X.Y.Z>                   <-- IP of the Openshift Master Node.
#cluster_network is the one which vm network belongs to
cluster_network="{'domain': 'default-domain', 'project': 'admin', 'name': 'net1'}" <-- FQName of
the Virtual Network where Virtual Machines are running. The VMs in which Openshift cluster is
being installed in nested mode.
#config_nodes="x.x.x.x,y.y.y.y.y"
#analytics_nodes="x.x.x.x,y.y.y.y.y"
#config_api_vip=x.x.x.x
#analytics_api_vip=x.x.x.x


###########################################################################
```

```
### OpenShift Basic Vars
###########################################################################
openshift_deployment_type=openshift-enterprise
deployment_type=openshift-enterprise
containerized=false
openshift_disable_check=docker_image_availability,memory_availability,package_availability,disk_a
vailability,package_version,docker_storage

# Default node selectors
openshift_hosted_infra_selector="node-role.kubernetes.io/infra=true"

oreg_auth_user=<>
oreg_auth_password=<>

###########################################################################
### OpenShift Master Vars
###########################################################################

openshift_master_api_port=8443
openshift_master_console_port=8443
openshift_master_cluster_method=native

# Set this line to enable NFS
openshift_enable_unsupported_configurations=True


###########################################################################
### OpenShift Network Vars
###########################################################################

openshift_use_openshift_sdn=false
os_sdn_network_plugin_name='cni'
openshift_use_contrail=true

###########################################################################
### OpenShift Authentication Vars
###########################################################################

# htpasswd Authentication
openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge':
'true', 'kind': 'HTPasswdPasswordIdentityProvider'}]

###########################################################################
```

```
### OpenShift Router and Registry Vars
###########################################################################

openshift_hosted_router_replicas=1
openshift_hosted_registry_replicas=1

openshift_hosted_registry_storage_kind=nfs
openshift_hosted_registry_storage_access_modes=['ReadWriteMany']
openshift_hosted_registry_storage_nfs_directory=/export
openshift_hosted_registry_storage_nfs_options='*(rw,root_squash)'
openshift_hosted_registry_storage_volume_name=registry
openshift_hosted_registry_storage_volume_size=10Gi
openshift_hosted_registry_pullthrough=true
openshift_hosted_registry_acceptschema2=true
openshift_hosted_registry_enforcequota=true
openshift_hosted_router_selector="node-role.kubernetes.io/infra=true"
openshift_hosted_registry_selector="node-role.kubernetes.io/infra=true"

###########################################################################
### OpenShift Service Catalog Vars
###########################################################################

openshift_enable_service_catalog=True

template_service_broker_install=True
openshift_template_service_broker_namespaces=['openshift']

ansible_service_broker_install=True

openshift_hosted_etcd_storage_kind=nfs
openshift_hosted_etcd_storage_nfs_options="*(rw,root_squash,sync,no_wdelay)"
openshift_hosted_etcd_storage_nfs_directory=/export
openshift_hosted_etcd_storage_labels={'storage': 'etcd-asb'}
openshift_hosted_etcd_storage_volume_name=etcd-asb
openshift_hosted_etcd_storage_access_modes=['ReadWriteOnce']
openshift_hosted_etcd_storage_volume_size=2G

###########################################################################
### OpenShift Metrics and Logging Vars
```

```
###########################################################################
# Enable cluster metrics
openshift_metrics_install_metrics=True

openshift_metrics_storage_kind=nfs
openshift_metrics_storage_access_modes=['ReadWriteOnce']
openshift_metrics_storage_nfs_directory=/export
openshift_metrics_storage_nfs_options='*(rw,root_squash)'
openshift_metrics_storage_volume_name=metrics
openshift_metrics_storage_volume_size=2Gi
openshift_metrics_storage_labels={'storage': 'metrics'}

openshift_metrics_cassandra_nodeselector={"node-role.kubernetes.io/infra":"true"}
openshift_metrics_hawkular_nodeselector={"node-role.kubernetes.io/infra":"true"}
openshift_metrics_heapster_nodeselector={"node-role.kubernetes.io/infra":"true"}

# Enable cluster logging. ((
####openshift_logging_install_logging=True
openshift_logging_install_logging=False
#openshift_logging_storage_kind=nfs
#openshift_logging_storage_access_modes=['ReadWriteOnce']
#openshift_logging_storage_nfs_directory=/export
#openshift_logging_storage_nfs_options='*(rw,root_squash)'
#openshift_logging_storage_volume_name=logging
#openshift_logging_storage_volume_size=5Gi
#openshift_logging_storage_labels={'storage': 'logging'}
#openshift_logging_es_cluster_size=1
#openshift_logging_es_nodeselector={"node-role.kubernetes.io/infra":"true"}
#openshift_logging_kibana_nodeselector={"node-role.kubernetes.io/infra":"true"}
#openshift_logging_curator_nodeselector={"node-role.kubernetes.io/infra":"true"}

###########################################################################
### OpenShift Prometheus Vars
###########################################################################

## Add Prometheus Metrics:
openshift_hosted_prometheus_deploy=True
openshift_prometheus_node_selector={"node-role.kubernetes.io/infra":"true"}
openshift_prometheus_namespace=openshift-metrics

# Prometheus
openshift_prometheus_storage_kind=nfs
openshift_prometheus_storage_access_modes=['ReadWriteOnce']
```

```
openshift_prometheus_storage_nfs_directory=/export
openshift_prometheus_storage_nfs_options='*(rw,root_squash)'
openshift_prometheus_storage_volume_name=prometheus
openshift_prometheus_storage_volume_size=1Gi
openshift_prometheus_storage_labels={'storage': 'prometheus'}
openshift_prometheus_storage_type='pvc'

# For prometheus-alertmanager
openshift_prometheus_alertmanager_storage_kind=nfs
openshift_prometheus_alertmanager_storage_access_modes=['ReadWriteOnce']
openshift_prometheus_alertmanager_storage_nfs_directory=/export
openshift_prometheus_alertmanager_storage_nfs_options='*(rw,root_squash)'
openshift_prometheus_alertmanager_storage_volume_name=prometheus-alertmanager
openshift_prometheus_alertmanager_storage_volume_size=1Gi
openshift_prometheus_alertmanager_storage_labels={'storage': 'prometheus-alertmanager'}
openshift_prometheus_alertmanager_storage_type='pvc'

# For prometheus-alertbuffer
openshift_prometheus_alertbuffer_storage_kind=nfs
openshift_prometheus_alertbuffer_storage_access_modes=['ReadWriteOnce']
openshift_prometheus_alertbuffer_storage_nfs_directory=/export
openshift_prometheus_alertbuffer_storage_nfs_options='*(rw,root_squash)'
openshift_prometheus_alertbuffer_storage_volume_name=prometheus-alertbuffer
openshift_prometheus_alertbuffer_storage_volume_size=1Gi
openshift_prometheus_alertbuffer_storage_labels={'storage': 'prometheus-alertbuffer'}
openshift_prometheus_alertbuffer_storage_type='pvc'


###########################################################################
### Openshift HA
###########################################################################

# Openshift HA
openshift_master_cluster_hostname=load-balancer-0-3eba0c20dc494dfc93d5d50d06bbde89
openshift_master_cluster_public_hostname=load-balancer-0-3eba0c20dc494dfc93d5d50d06bbde89


###########################################################################
### Contrail Variables
##########################################################################

service_subnets="172.30.0.0/16"
pod_subnets="10.128.0.0/14"
```

```
# Below are Contrail variables. Comment them out if you don't want to install Contrail through
ansible-playbook
contrail_version=1907
contrail_container_tag=<>
contrail_registry=hub.juniper.net/contrail
contrail_registry_username=<>
contrail_registry_password=<>
openshift_docker_insecure_registries=hub.juniper.net/contrail
contrail_nodes=[10.0.0.5,10.0.0.3,10.0.0.4]
vrouter_physical_interface=eth0


###########################################################################
### OpenShift Hosts
###########################################################################
[OSEv3:children]
masters
etcd
nodes
lb
nfs
openshift_ca

[masters]
kube-master-2-3eba0c20dc494dfc93d5d50d06bbde89
kube-master-1-3eba0c20dc494dfc93d5d50d06bbde89
kube-master-0-3eba0c20dc494dfc93d5d50d06bbde89

[etcd]
kube-master-2-3eba0c20dc494dfc93d5d50d06bbde89
kube-master-1-3eba0c20dc494dfc93d5d50d06bbde89
kube-master-0-3eba0c20dc494dfc93d5d50d06bbde89

[lb]
load-balancer-0-3eba0c20dc494dfc93d5d50d06bbde89

[nodes]
kube-master-2-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-master'
controller-0-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-infra'
compute-1-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-compute'
controller-2-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-infra'
kube-master-1-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-master'
```

```
kube-master-0-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-master'
compute-0-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-compute'
controller-1-3eba0c20dc494dfc93d5d50d06bbde89 openshift_node_group_name='node-config-infra'

[nfs]
load-balancer-0-3eba0c20dc494dfc93d5d50d06bbde89

[openshift_ca]
kube-master-2-3eba0c20dc494dfc93d5d50d06bbde89
kube-master-1-3eba0c20dc494dfc93d5d50d06bbde89
kube-master-0-3eba0c20dc494dfc93d5d50d06bbde89
```

NOTE: The /etc/resolv.conf must have write permissions.

Release History Table

Release	Description
1907	You can install a nested Red Hat OpenShift Container Platform 3.11 cluster along with Contrail Networking using Contrail Ansible deployer.

RELATED DOCUMENTATION

Installing a Standalone Red Hat OpenShift Container Platform 3.11 Cluster with Contrail Using Contrail OpenShift Deployer

CHAPTER 11

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.

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.

Deploying Contrail Charms in a Bundle

Follow these steps to deploy Contrail Charms in a bundle.

1. Deploy Contrail Charms.

   To deploy Contrail Charms in a bundle, use the `juju deploy <bundle_yaml_file>` command.

   The following example shows you how to use **bundle_yaml_file** to deploy Contrail on Amazon Web Services (AWS) Cloud.

```
series: xenial
services:
```

```
ubuntu:
  charm: cs:xenial/ubuntu
  num_units: 3
  to: [ "1", "2", "3" ]
ntp:
  charm: cs:xenial/ntp
  num_units: 0
  options:
    source: ntp.juniper.net
mysql:
  charm: cs:xenial/percona-cluster
  options:
    dataset-size: 15%
    max-connections: 10000
    root-password: password
    sst-password: password
    vip: ip-address
    vip_cidr: 24
  num_units: 3
  to: [ "lxd:1", "lxd:2", "lxd:3" ]
rabbitmq-server:
  charm: cs:xenial/rabbitmq-server
  num_units: 3
  to: [ "lxd:1", "lxd:2", "lxd:3" ]
heat:
  charm: cs:xenial/heat
  num_units: 3
  options:
    vip: ip-address
    vip_cidr: 24
  to: [ "lxd:1", "lxd:2", "lxd:3" ]
keystone:
  charm: cs:xenial/keystone
  options:
    admin-password: password
    admin-role: admin
    openstack-origin: cloud:xenial-newton
    vip: ip-address
    vip_cidr: 24
  num_units: 3
  to: [ "lxd:1", "lxd:2", "lxd:3" ]
nova-cloud-controller:
  charm: cs:xenial/nova-cloud-controller
```

```
    options:
      network-manager: Neutron
      openstack-origin: cloud:xenial-newton
      vip: ip-address
      vip_cidr: 24
    num_units: 3
    to: [ "lxd:1", "lxd:2", "lxd:3" ]
  neutron-api:
    charm: cs:xenial/neutron-api
    series: xenial
    options:
      manage-neutron-plugin-legacy-mode: false
      openstack-origin: cloud:xenial-newton
      vip: ip-address
      vip_cidr: 24
    num_units: 3
    to: [ "lxd:1", "lxd:2", "lxd:3" ]
  glance:
    charm: cs:xenial/glance
    options:
      openstack-origin: cloud:xenial-newton
      vip: ip-address
      vip_cidr: 24
    num_units: 3
    to: [ "lxd:1", "lxd:2", "lxd:3" ]
  openstack-dashboard:
    charm: cs:xenial/openstack-dashboard
    options:
      openstack-origin: cloud:xenial-newton
      vip: ip-address
      vip_cidr: 24
    num_units: 3
    to: [ "lxd:1", "lxd:2", "lxd:3" ]
  nova-compute:
    charm: cs:xenial/nova-compute
    options:
      openstack-origin: cloud:xenial-newton
    num_units: 3
    to: [ "4", "5", "6" ]
  mysql-hacluster:
    charm: cs:xenial/hacluster
    options:
      cluster_count: 3
```

```
    num_units: 0
  keystone-hacluster:
    charm: cs:xenial/hacluster
    options:
      cluster_count: 3
    num_units: 0
  ncc-hacluster:
    charm: cs:xenial/hacluster
    options:
      cluster_count: 3
    num_units: 0
  neutron-hacluster:
    charm: cs:xenial/hacluster
    options:
      cluster_count: 3
    num_units: 0
  glance-hacluster:
    charm: cs:xenial/hacluster
    options:
      cluster_count: 3
    num_units: 0
  dashboard-hacluster:
    charm: cs:xenial/hacluster
    options:
      cluster_count: 3
    num_units: 0
  heat-hacluster:
    charm: cs:xenial/hacluster
    options:
      cluster_count: 3
    num_units: 0
  contrail-openstack:
    charm: cs:~juniper-os-software/contrail-openstack
    series: xenial
    num_units: 0
  contrail-agent:
    charm: cs:~juniper-os-software/contrail-agent
    num_units: 0
    series: xenial
    options:
      log-level: "SYS_DEBUG"
  contrail-analytics:
    charm: cs:~juniper-os-software/contrail-analytics
```

```
    num_units: 3
    series: xenial
    to: [ "1", "2", "3" ]
  contrail-analyticsdb:
    charm: cs:~juniper-os-software/contrail-analyticsdb
    num_units: 3
    series: xenial
    options:
      log-level: "SYS_DEBUG"
      cassandra-minimum-diskgb: 4
      cassandra-jvm-extra-opts: "-Xms1g -Xmx2g"
    to: [ "1", "2", "3" ]
  contrail-controller:
    charm: cs:~juniper-os-software/contrail-controller
    series: xenial
    options:
      vip: ip-address
      log-level: "SYS_DEBUG"
      cassandra-minimum-diskgb: 4
      cassandra-jvm-extra-opts: "-Xms1g -Xmx2g"
    to: [ "1", "2", "3" ]
  contrail-keystone-auth:
    charm: cs:~juniper-os-software/contrail-keystone-auth
    series: xenial
    num_units: 1
    to: [ "lxd:1" ]

  contrail-keepalived:
    charm: cs:~boucherv29/keepalived-19
    series: xenial
    options:
      virtual_ip: ip-address
  contrail-haproxy:
    charm: haproxy
    series: xenial
    expose: true
    options:
      peering_mode: "active-active"
    to: [ "1", "2", "3" ]

relations:
  # openstack
  - [ "ubuntu", "ntp" ]
```

```
- [ mysql, mysql-hacluster ]
- [ "keystone", "mysql" ]
- [ keystone, keystone-hacluster ]
- [ "glance", "mysql" ]
- [ "glance", "keystone" ]
- [ glance, glance-hacluster ]
- [ "nova-cloud-controller", "mysql" ]
- [ "nova-cloud-controller", "rabbitmq-server" ]
- [ "nova-cloud-controller", "keystone" ]
- [ "nova-cloud-controller", "glance" ]
- [ nova-cloud-controller, ncc-hacluster ]
- [ "neutron-api", "mysql" ]
- [ "neutron-api", "rabbitmq-server" ]
- [ "neutron-api", "nova-cloud-controller" ]
- [ "neutron-api", "keystone" ]
- [ neutron-api, neutron-hacluster ]
- [ "nova-compute:amqp", "rabbitmq-server:amqp" ]
- [ "nova-compute", "glance" ]
- [ "nova-compute", "nova-cloud-controller" ]
- [ "nova-compute", "ntp" ]
- [ "openstack-dashboard:identity-service", "keystone" ]
- [ openstack-dashboard, dashboard-hacluster ]
- [ "heat", "mysql" ]
- [ "heat", "rabbitmq-server" ]
- [ "heat", "keystone" ]
- [ "heat", "heat-hacluster" ]

#contrail
- [ "contrail-keystone-auth", "keystone" ]
- [ "contrail-controller", "contrail-keystone-auth" ]
- [ "contrail-analytics", "contrail-analyticsdb" ]
- [ "contrail-controller", "contrail-analytics" ]
- [ "contrail-controller", "contrail-analyticsdb" ]
- [ "contrail-openstack", "nova-compute" ]
- [ "contrail-openstack", "neutron-api" ]
- [ "contrail-openstack", "heat" ]
- [ "contrail-openstack", "contrail-controller" ]
- [ "contrail-agent:juju-info", "nova-compute:juju-info" ]
- [ "contrail-agent", "contrail-controller"]

#haproxy
- [ "haproxy:juju-info", "keepalived:juju-info" ]
- [ "contrail-analytics", "haproxy" ]
```

```
  - [ "contrail-controller:http-services", "haproxy" ]
  - [ "contrail-controller:https-services", "haproxy" ]

machines:
  "1":
    series: xenial
    #constraints: mem=15G root-disk=40G
    constraints: tags=contrail-controller-vm-1
  "2":
    series: xenial
    #constraints: mem=15G root-disk=40G
    constraints: tags=contrail-controller-vm-2
  "3":
    series: xenial
    #constraints: mem=15G root-disk=40G
    constraints: tags=contrail-controller-vm-3
  "4":
    series: xenial
    #constraints: mem=4G root-disk=20G
    constraints: tags=compute-storage-1
  "5":
    series: xenial
    #constraints: mem=4G root-disk=20G
    constraints: tags=compute-storage-2
  "6":
    series: xenial
    #constraints: mem=4G root-disk=20G
    constraints: tags=compute-storage-3
```

You can create or modify the Contrail Charm deployment bundle YAML file to:

- Point to machines or instances where the Contrail Charms must be deployed.
- Include the options you need.

  Each Contrail Charm has a specific set of options. The options you choose depend on the charms you select. For more information on the options that are available, see "Options for Juju Charms" on page 281.

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 10: 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 10: 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.
no_proxy		Specify the list of destinations that must be directly accessed.

- Options for **contrail-analytics** Charms.

Table 11: Options for contrail-analytics

Option	Default option	Description
control-network		Specify the IP address and network mask of the control network.
docker-registry		Specify the URL of the docker-registry.
docker-registry-insecure	false	Specify if the docker-registry should be configured.
docker-user		Log in to the docker registry.
docker-password		Specify the docker-registry password.
image-tag		Specify the docker image tag.
log-level	SYS_NOTICE	Specify the log level for Contrail services. Options: SYS_EMERG, SYS_ALERT, SYS_CRIT, SYS_ERR, SYS_WARN, SYS_NOTICE, SYS_INFO, SYS_DEBUG
http_proxy		Specify URL.
https_proxy		Specify URL.
no_proxy		Specify the list of destinations that must be directly accessed.

- Options for **contrail-analyticsdb** Charms.

Table 12: Options for contrail-analyticsdb

Option	Default option	Description
`control-network`		Specify the IP address and network mask of the control network.
`cassandra-minimum-diskgb`	`256`	Specify the minimum disk requirement.
`cassandra-jvm-extra-opts`		Specify the memory limit.
`docker-registry`		Specify the URL of the docker-registry.
`docker-registry-insecure`	`false`	Specify if the docker-registry should be configured.
`docker-user`		Log in to the docker registry.
`docker-password`		Specify the docker-registry password.
`image-tag`		Specify the docker image tag.
`log-level`	`SYS_NOTICE`	Specify the log level for Contrail services. Options: `SYS_EMERG`, `SYS_ALERT`, `SYS_CRIT`, `SYS_ERR`, `SYS_WARN`, `SYS_NOTICE`, `SYS_INFO`, `SYS_DEBUG`
`http_proxy`		Specify URL.
`https_proxy`		Specify URL.
`no_proxy`		Specify the list of destinations that must be directly accessed.

- Options for **contrail-controller** Charms.

Table 13: 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.
`cloud-admin-role`	`admin`	Specify the role name in keystone for users who have admin-level access.
`global-read-only-role`		Specify the role name in keystone for users who have read-only access.
`vip`		Specify if the Contrail API VIP is used for configuring client-side software. If not specified, private IP of the first Contrail API VIP unit will be used.
`use-external-rabbitmq`	`false`	To enable the Charm to use the internal RabbitMQ server, set `use-external-rabbitmq` to `false`. To use an external AMQP server, set`use-external-rabbitmq` to `true`. **NOTE**: Do not change the flag after deployment.
`flow-export-rate`	`0`	Specify how many flow records are exported by vRouter agent to the Contrail Collector when a flow is created or deleted.

Table 13: Options for contrail-controller *(Continued)*

Option	Default option	Description
`docker-registry`		Specify the URL of the docker-registry.
`docker-registry-insecure`	`false`	Specify if the docker-registry should be configured.
`docker-user`		Log in to the docker registry.
`docker-password`		Specify the docker-registry password.
`image-tag`		Specify the docker image tag.
`log-level`	`SYS_NOTICE`	Specify the log level for Contrail services. Options: `SYS_EMERG`, `SYS_ALERT`, `SYS_CRIT`, `SYS_ERR`, `SYS_WARN`, `SYS_NOTICE`, `SYS_INFO`, `SYS_DEBUG`
`http_proxy`		Specify URL.
`https_proxy`		Specify URL.
`no_proxy`		Specify the list of destinations that must be directly accessed.

- Options for **contrail-keystone-auth** Charms.

Table 14: 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 15: 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 15: Options for contrail-controller *(Continued)*

Option	Default option	Description
`no_proxy`		Specify the list of destinations that must be directly accessed.

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.

```
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 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. For more information, see Understanding Juju Charms.

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/Juniper/contrail-charms/blob/R5/contrail-agent/README.md.

   - `contrail-analytics`

     For more information, see https://github.com/Juniper/contrail-charms/blob/R5/contrail-analytics/README.md.

   - `contrail-analyticsdb`

For more information, see https://github.com/Juniper/contrail-charms/blob/R5/contrail-analyticsdb/README.md.

- `contrail-controller`

  For more information, see https://github.com/Juniper/contrail-charms/blob/R5/contrail-controller/README.md.

- `contrail-kubernetes-master`

  For more information, see https://github.com/Juniper/contrail-charms/blob/R5/contrail-kubernetes-master/README.md.

- `contrail-kubernetes-node`

  For more information, see https://github.com/Juniper/contrail-charms/blob/R5/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 --constraints mem=8G cores=2 root-disk=32G --series=xenial  #for Kubernetes
worker machine
juju add-machine --constraints mem=18G cores=2 root-disk=32G --series=xenial #for Kubernetes
master machine
juju add-machine --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"
```

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.

Release History Table

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 | 288
Installing Contrail with OpenStack by Using Juju Charms | 266

CHAPTER 12

Using Contrail and AppFormix with Kolla/Ocata OpenStack

IN THIS CHAPTER

Contrail, AppFormix, and OpenStack Kolla/Ocata Deployment Requirements

IN THIS SECTION

Starting with Contrail Release 5.0.1, the combined installation of Contrail and AppFormix allows Contrail monitoring by AppFormix. The following topics are referenced for the deployment.

- "Installing Contrail with OpenStack and Kolla Ansible " on page 58
- ."Installing AppFormix for OpenStack" on page 317

- "Install AppFormix for OpenStack in HA" on page 322

The following software and hardware requirements apply to the combined Contrail, AppFormix, 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.
- AppFormix Targets: Refer to “Software Requirements” in *Contrail Insights General Requirements*.
- Targets running both Contrail and AppFormix: CentOS 7.5 Ansible 2.4.2 for the installer.
- AppFormix 2.18.x and later.

Hardware Requirements

- It is strongly recommended that the AppFormix Controller and Contrail services be installed on separate targets.
- See "Installing Contrail Cluster using Contrail Command and instances.yml" on page 42 and "Installing AppFormix for OpenStack" on page 317 for specifics about requirements for installation.

Preparing for the Installation

IN THIS SECTION

In Contrail Release 5.1, nodes on which Contrail, 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, AppFormix, 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 AppFormix Controller or AppFormix Agent is going to be installed. See the instructions in "Installing AppFormix for OpenStack" on page 317.

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 " on page 58.

2. Set-up the base host. See “Set Up the Base Host” in "Installing Contrail with OpenStack and Kolla Ansible " on page 58. 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 AppFormix 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 " on page 58.

   - The AppFormix inventory section of the `instances.yaml` file is configured according to guidelines in "Installing AppFormix for OpenStack" on page 317.

TCP/IP Port Conflicts Between Contrail and AppFormix

It is strongly recommended that AppFormix Controller and Contrail services be installed on separate target nodes. However, if AppFormix Controller and Contrail services are installed on the same target, the following configuration is required to resolve port conflicts.

The following AppFormix ports must be reconfigured in the AppFormix `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 AppFormix Deployment

Enable the following plugins by including them in the AppFormix `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 AppFormix

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 AppFormix in the `instances.yaml` file. A list of IP addresses with a *vrouter* role in the `instances.yaml` file.

Configuring Openstack_Controller Hosts for AppFormix

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 AppFormix 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**

AppFormix License

You must have an appropriate license that supports the combined deployment of Contrail with AppFormix for OpenStack. To obtain a license, send an email to “AppFormix-Key-Request@juniper.net. Also, the following `group_vars` 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 | 58
Installing Contrail Cluster using Contrail Command and instances.yml | 42

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 " on page 58 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 AppFormix-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 AppFormix in the `instances.yaml` file. Refer to `Compute monitoring: Listing IP addresses to monitor` in the `compute`section of AppFormix in the `instances.yaml` file.

Refer to "Installing AppFormix for OpenStack" on page 317 and validate target configuration requirements and inventory parameters for AppFormix Controller and Agent. In place of `-i inventory/`use `-i /absolute-file-path/instances.yaml`.

Following is an example listing the AppFormix playbook invocation from the AppFormix-2.18.x directory where `appformix_openstack.yml` is located:

- Install AppFormix:

```
ansible-playbook -i /path/instances.yaml appformix_openstack.yml (-vvv for debug)
```

RELATED DOCUMENTATION

Installing Contrail with OpenStack and Kolla Ansible | 58
Installing Contrail Cluster using Contrail Command and instances.yml | 42
Installing AppFormix for OpenStack | 317

Accessing Contrail in AppFormix Management Infrastructure in UI

AppFormix 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 AppFormix User Guide.

Open the Dashboard in a Web browser and log in.

`http://`*<controller-IP-address>*`:9000`

RELATED DOCUMENTATION

AppFormix User Guide

Notes and Caveats

- Versions of AppFormix-2.17 and earlier are not supported with Ansible-2.4.2. The combined Contrail and AppFormix installation is not validated on these earlier releases.

- The installation was validated with AppFormix-2.18 Agent.
- To view and monitor Contrail in the AppFormix 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 AppFormix OpenStack HA installation steps, see "Install AppFormix for OpenStack in HA" on page 322.

RELATED DOCUMENTATION

Install AppFormix for OpenStack in HA | 322

Example Instances.yml for Contrail and AppFormix OpenStack Deployment

See "Installing Contrail with OpenStack and Kolla Ansible " on page 58 and "Installing AppFormix for OpenStack" on page 317 for specific inventory file details:

The following items are part of the `all` section in the `instances.yaml` file for AppFormix:

```
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 AppFormix:

```
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 AppFormix 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>

  # Appformix 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/appformix-platform-images-<version>.tar.gz
      - /opt/software/appformix/appformix-dependencies-images-<version>.tar.gz
      - /opt/software/appformix/appformix-network_device-images-<version>.tar.gz
      - /opt/software/appformix/appformix-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 19XX.

RELATED DOCUMENTATION

Installing AppFormix for OpenStack

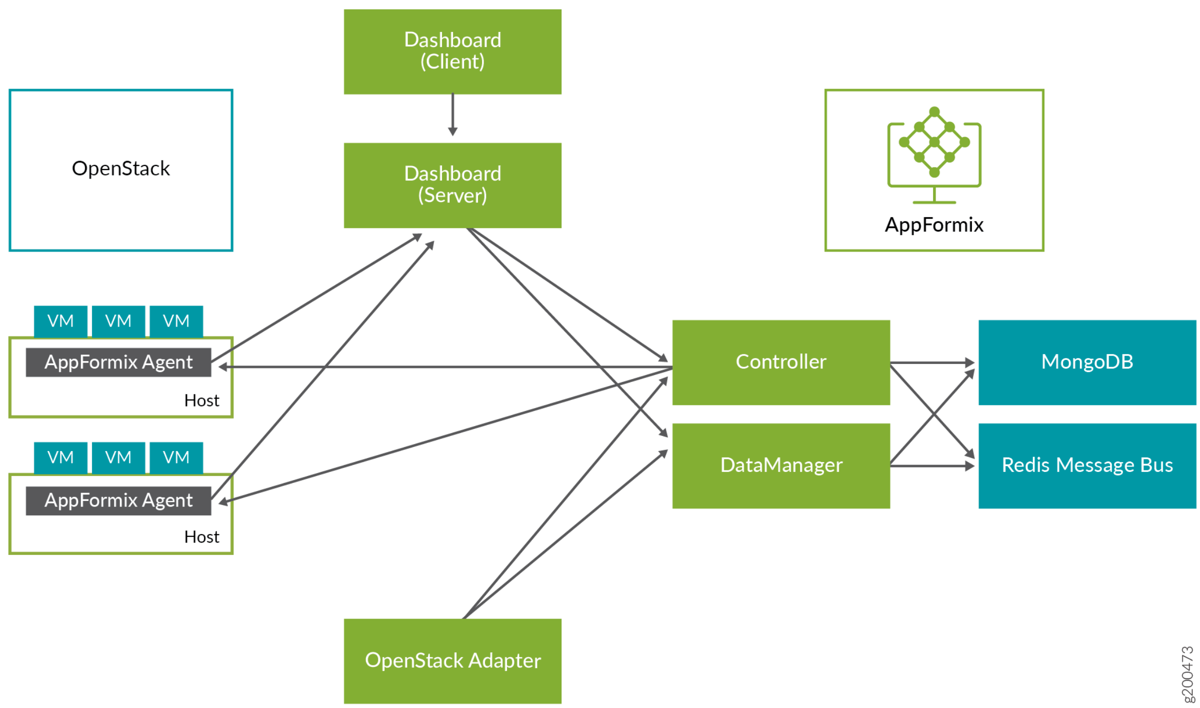
IN THIS SECTION

AppFormix provides resource control and visibility for hosts and virtual machines in an OpenStack environment. This topic explains how to install AppFormix for OpenStack. See the *Contrail Insights General Requirements* before reading this section.

Architecture

AppFormix provides resource control and visibility for hosts, containers, and virtual machines in your cloud infrastructure. Figure 47 on page 318 shows the AppFormix architecture with OpenStack.

Figure 47: AppFormix Architecture with OpenStack



- Agent monitors resource usage on the compute nodes.
- Controller offers REST APIs to configure the system.
- DataManager stores data from multiple Agents.
- Dashboard provides a Web-based user interface.
- An adapter discovers platform-specific resources and configures the AppFormix Controller.
- Adapters exist for OpenStack, Kubernetes, and Amazon EC2.

Installing AppFormix

To install AppFormix:

1. Install Ansible on the AppFormix Controller node. Ansible will install docker and docker-py on the controller.

```
apt-get install python-pip python-dev      #Installs Pip

pip install ansible==2.3                   #Installs Ansible 2.3
```

```
sudo apt-get install build-essential libssl-dev libffi-dev    #Dependencies

pip install markupsafe httplib2           #Dependencies
```

2. On the vRouter compute nodes where AppFormix Agent runs verify that python `virtualenv` is installed.

```
apt-get install -y python-pip

pip install virtualenv
```

3. Enable passwordless login to facilitate AppFormix Controller node with Ansible to install agents on the nodes. Run the same command on the AppFormix Controller node also.

```
ssh-keygen -t rsa   #Creates Keys

ssh-copy-id -i ~/.ssh/id_rsa.pub <target_host>    #Copies key from the node to other hosts
```

4. 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]
203.0.113.5
 203.0.113.17

 # AppFormix controller host
 #
 # Host variables can be defined to control AppFormix 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]
203.0.113.119
```

5. Verify that all the hosts listed in the inventory file are reachable from the AppFormix Controller.

```
 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
```

6. At the top-level of the distribution, create a directory named `group_vars`.

```
mkdir group_vars
```

7. Every installation requires an authorized license file and Docker images. In `group_vars` directory, create a file named all. Add the following:

```
openstack_platform_enabled: true

appformix_version: <version>
appformix_manager_version: <version>
appformix_license: path/to/appformix-license-file.sig          # Location of License Provided

appformix_docker_images:
    - /path/to/appformix-platform-images-<version>.tar.gz
    - /path/to/appformix-dependencies-images-<version>.tar.gz
    - /path/to/appformix-openstack-images-<version>.tar.gz
```

8. Source the `openrc` file from the OpenStack controller node (`/etc/contrail/openstackrc`) to the AppFormix Controller to authenticate the adapter to access admin privileges over the controller services.

```
export OS_USERNAME=<admin user>
export OS_PASSWORD=<password>
export OS_AUTH_URL=http://<openstack-auth-URL>/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
```

9. Run Ansible with the created inventory file.

```
ansible-playbook -i inventory appformix_openstack.yml
```

Removing a Node from AppFormix

To remove a node from AppFormix:

1. Edit the inventory file and add `appformix_state=absent` to each node that you want to remove from AppFormix.

```
 # Example naming schemes are as below:
 #   hostname ansible_ssh_user='username' ansible_sudo_pass='password'

 # List all Compute Nodes
 [compute]
 203.0.113.5 appformix_state=absent
203.0.113.17
```

2. Run Ansible with the edited inventory file. This will remove the node and all its resources from AppFormix.

```
ansible-playbook -i inventory appformix_openstack.yml
```

RELATED DOCUMENTATION

Contrail Insights General Requirements
AppFormix Installation for OpenStack Cluster
Install AppFormix for OpenStack in HA | 322
Contrail Insights Agent Requirements
Platform Dependencies

Install AppFormix for OpenStack in HA

IN THIS SECTION

HA Design Overview

AppFormix 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.

AppFormix Platform can be deployed in a 3-node, 5-node, or 7-node configuration for high availability.

Requirements

- For each host, on which AppFormix Platform is installed, see *Contrail Insights General Requirements* for hardware and software requirements. For a list of AppFormix Agent supported platforms, see *Contrail Insights Agent Requirements*.

- You need an AppFormix license 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: Kubernetes
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 AppFormix for High Availability

To install AppFormix to multiple hosts for high availability:

1. Download the AppFormix installation packages from software downloads to the AppFormix Platform node. Get the following files:

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

2. Install Ansible on the installer node. Ansible will install docker and docker-py 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 AppFormix 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 AppFormix
[compute]
203.0.113.5
203.0.113.17
```

```
# AppFormix 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/appformix-platform-images-<version>.tar.gz
- /path/to/appformix-dependencies-images-<version>.tar.gz
- /path/to/appformix-openstack-images-<version>.tar.gz
```

NOTE: In AppFormix version 3.2.0, support for monitoring Openstack Octavia LoadBalancer services has been added. To enable this service monitoring, 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
```

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
```

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
```

NOTE: If step 8. is being done with offline installation and failed due to step 8. not being done, then the appformix *.tar.gz need to be removed from the `/tmp/` folder on the appformix_controller node. This is the workaround required as of version 2.11.1.

RELATED DOCUMENTATION

Contrail Insights General Requirements

AppFormix Installation for OpenStack Cluster

Contrail Insights Agent Requirements

Platform Dependencies

CHAPTER 13

Upgrading Contrail Software

IN THIS CHAPTER

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 Contrail Command container.

You can resolve the issue by:

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

   Run the following command on the Contrail Command node to backup the DB.

   ```
   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.

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

3. Modify the **yaml formatted db dump** by adding or removing the fields as per the new database schema.
4. Restore the modified **yaml formatted db dump** to the SQL database.

```
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.

Upgrading Contrail Networking using Contrail Command UI

Take snapshots of your current configurations before you proceed with the upgrade process.

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

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

You must upgrade Contrail Command before you proceed with the following procedure. For details, refer to "Upgrading Contrail Command using Backup Restore Procedure" on page 327.

NOTE: This procedure is not applicable for upgrading Contrail Networking from Release 1909 to Release 1910.

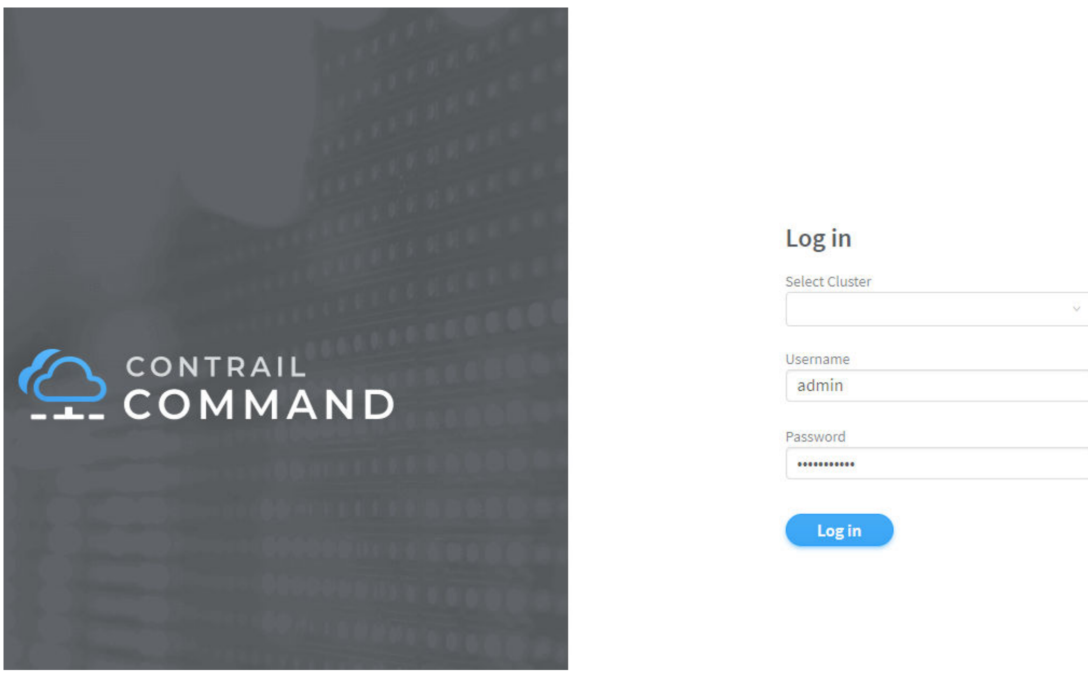
Refer to "Upgrading Contrail Networking using contrail-ansible Deployer" on page 331 to upgrade Contrail Networking from Release 1909 to Release 1910.

1. Log in to Contrail Command UI by navigating to https://Contrail-Command-Server-IP-Address:9091. The default username is **admin** and the default password is **contrail123**.

   NOTE: We strongly recommend creating a unique username and password combination. For information on setting username and password credentials, see "Installing Contrail Command" on page 18.

   Enter the credentials and click **Log in**

   NOTE: You must not **Select Cluster**.

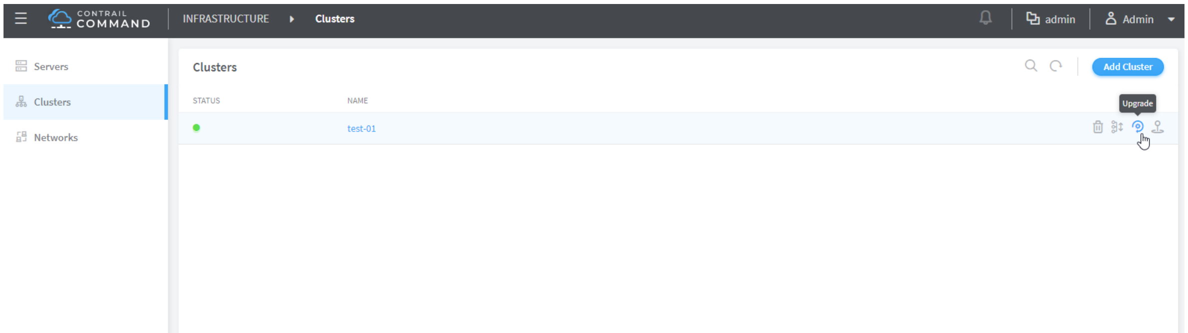
   

2. Click on **Clusters**.

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

3. Select the cluster you want to upgrade.

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



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

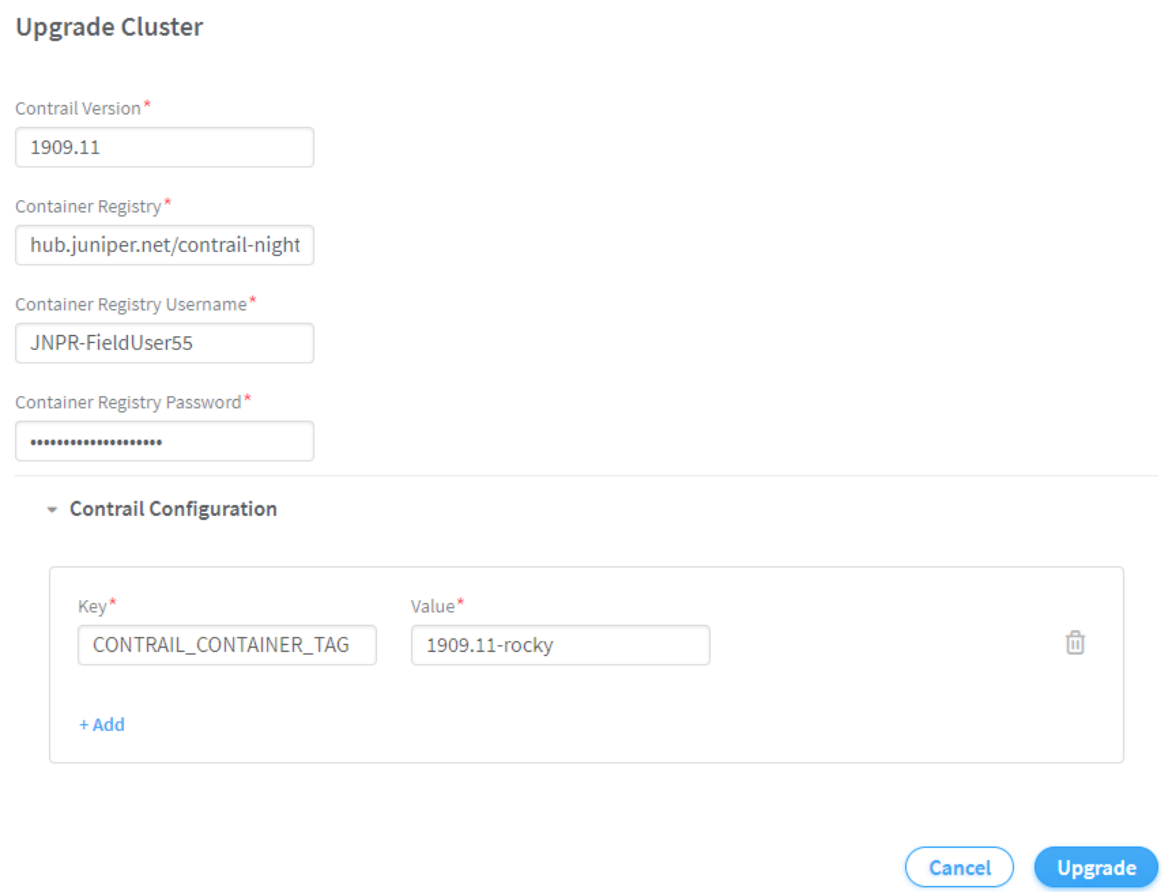
   Contrail Version depicts the current installed Contrail version. You must update the value to the desired version number.

   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* located at README Access to Contrail Registry 19XX.
5. If you have Appformix and Appformix Flows installed in the cluster, you must provide appropriate versions of Appformix and Appformix Flows packages in **/opt/software/appformix** and **/opt/software/xflow** directories on the Contrail Command server. For more details, refer to "Contrail and AppFormix Deployment Requirements" on page 76 and "Installing AppFormix and AppFormix Flows using Contrail Command" on page 77.
6. Click on **Upgrade**.



RELATED DOCUMENTATION

Installing Contrail Command | 18

Upgrading Contrail Networking using contrail-ansible Deployer

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 386.

Use the following procedure to upgrade Contrail Networking using contrail-ansible deployer.

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

1. Navigate to the directory where the **contrail-ansible-deployer-19<xx>.<NN>.tgz** was untarred.

   See "Sample instances.yml File" on page 45.

```
cd contrail-ansible-deployer-19<xx>.<NN>/contrail-ansible-deployer/config/
```

```
vi contrail-ansible-deployer-19<xx>.<NN>/contrail-ansible-deployer/config/instances.yaml
```

Sample **instances.yaml** files for various other deployments are available at the same directory.

2. Update *CONTRAIL_VERSION* and *CONTRAIL_CONTAINER_TAG* to the desired version tag in this **instances.yml** file.

   Access *CONTRAIL_CONTAINER_TAG* located at README Access to Contrail Registry 19XX.

   For example:

```
CONTRAIL_VERSION = 1907.55
CONTRAIL_CONTAINER_TAG = 1907.55-queens
```

3. Run the following commands from **contrail-ansible-deployer** directory.

   - For Contrail with OpenStack deployment:

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

   - For Contrail with Kubernetes deployment:

```
cd contrail-ansible-deployer
ansible-playbook -e orchestrator=kubernetes -i inventory/ playbooks/install_k8s.yml -v
ansible-playbook -e orchestrator=kubernetes -i inventory/ playbooks/install_contrail.yml -v
```

The ansible playbook logs are available on the terminal during execution. You can also access it at `/var/log/ansible.log`.

RELATED DOCUMENTATION

Installing Contrail Cluster using Contrail Command and instances.yml | 42
Upgrading Contrail Networking using Contrail Command UI | 328

Upgrading Contrail Networking using In-Place Upgrade Procedure

This document provides steps to upgrade Contrail Networking using in-place upgrade procedure.

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

BEST PRACTICE: You must take snapshots of your current system before proceeding with the upgrade process.

For a list of supported platforms, see https://www.juniper.net/documentation/en_US/release-independent/contrail/topics/reference/contrail-supported-platforms.pdf.

1. Update kernel version on all the compute nodes.

   ```
   yum -y update kernel*
   ```

   NOTE: You must not update kernel version if you are upgrading from Contrail Networking Release 1910 to Release 1911.

2. Update *CONTRAIL_VERSION* and *CONTRAIL_CONTAINER_TAG* to the desired version tag in this **instances.yml** file.

   Access *CONTRAIL_CONTAINER_TAG* located at README Access to Contrail Registry 19XX.

3. Run the following commands from **contrail-ansible-deployer** directory.

   For Contrail with OpenStack deployment:

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

4. Reboot the compute node.
5. Check the status of Contrail service on the compute node.

   All services must be *active* .

   ```
   sudo contrail-status
   ```

The ansible playbook logs are available on the terminal during execution. You can also access it at `/var/log/ansible.log`.

RELATED DOCUMENTATION

Upgrading Contrail Networking using Contrail Command UI | 328
Upgrading Contrail Networking using contrail-ansible Deployer | 331
Upgrading Contrail Networking with Red Hat Openstack 13 using ISSU | 341

Updating Contrail Networking using the Zero Impact Upgrade Process in an Environment using Red Hat Openstack

IN THIS SECTION

This document provides the steps needed to update a Contrail Networking deployment that is using Red Hat Openstack as it's orchestration platform. The procedure 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 Red Hat Openstack version 13 (RHOSP13) as the orchestration platform is already operational.
- The overcloud nodes in the RHOSP13 environment have an enabled Red Hat Enterprise Linux (RHEL) subscription.
- Your environment is running Contrail Release 1912 and upgrading to Contrail Release 1912.L1 or to Contrail Release 2003 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 if needed for the 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" on page 386.

- 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.

- If you are updating Red Hat Openstack simultaneously with Contrail Networking, update Red Hat Openstack to the latest minor release version and ensure that the new overcloud images are prepared for the upgrade. See the Upgrading the Undercloud section of the Keeping Red Hat OpenStack Platform Updated guide from Red Hat for additional information.

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

Updating Contrail Networking in an Environment using Red Hat Openstack

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

1. Prepare your docker registry. The registry is often included in the undercloud, but it can also be a separate node.

   Docker registry setup is environment independent. See Docker Registry from Docker for additional information on Docker registry setup.
2. Backup the Contrail TripleO Heat Templates. See Using the Contrail Heat Template.
3. Get the Contrail TripleO Heat Templates (Stable/Queens branch) from https://github.com/Juniper/contrail-tripleo-heat-templates.
4. (Optional) Update the Contrail TripleO Puppet module to the latest version and prepare Swift Artifacts, as applicable.

   Below are sample commands entered in the undercloud:

```
[stack@queensa ~]$ mkdir -p ~/usr/share/openstack-puppet/modules/tripleo
[stack@queensa ~]$ git clone -b stable/queens https://github.com/Juniper/contrail-tripleo-
puppet usr/share/openstack-puppet/modules/tripleo
[stack@queensa ~]$ tar czvf puppet-modules.tgz usr/
[stack@queensa ~]$ upload-swift-artifacts -c contrail-artifacts -f puppet-modules.tgz
```

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 20XX.

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:

```
openstack overcloud update prepare
--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-registrationresource-
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
```

7. 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 `docker pull` command for Docker containers and the `openstack overcloud container image upload` command for Openstack containers, or running the `contrail-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 the *ContrailImageTag* parameter for a specific image from the README Access to Contrail Registry 20XX.

- `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
```

8. 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 --nodes 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 --nodes 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.

```
docker ps -a
```

- Update the remaining Contrail Controller nodes:

  Example:

  ```
  openstack overcloud update run --nodes overcloud-contrailcontroller-1
  openstack overcloud update run --nodes overcloud-contrailcontroller-2
  openstack overcloud update run --nodes overcloud-contrailcontroller-3
  ...
  ```

9. Update the Openstack Controllers using the `openstack overcloud update run` commands:

   Example:

   ```
   openstack overcloud update run --nodes overcloud-controller-0
   openstack overcloud update run --nodes overcloud-controller-1
   openstack overcloud update run --nodes overcloud-controller-2
   ...
   ```

10. 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 --nodes overcloud-novacompute-1
    openstack overcloud update run --nodes overcloud-novacompute-2
    openstack overcloud update run --nodes 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
```

11. 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 6.

```
openstack overcloud update converge
--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-registrationresource-
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
```

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 | 266

Upgrading Contrail Networking with Red Hat Openstack 13 using ISSU

IN THIS SECTION

This document provides steps to upgrade Contrail Networking with an in-service software upgrade (ISSU) in an environment using Red Hat Openstack Platform 13 (RHOSP13).

When to Use This Procedure

Use this procedure to upgrade Contrail Networking when it is running in environments using RHOSP13.

This procedure has been validated for the following Contrail Networking upgrades:

Table 16: Contrail Networking with RHOSP13 Validated Upgrade Scenarios

Starting Contrail Networking Release	Target Contrail Networking Upgrade Release
5.0 or 5.1	1907
1907	1908
1908	1909
1909	1910
1910	1911
1911	1912

Starting in Contrail Networking Releases 1912.L0 and 2003, use the Zero Impact Upgrade (ZIU) procedure to upgrade Contrail Networking in environments using Red Hat Openstack orchestration. See "Updating Contrail Networking using the Zero Impact Upgrade Process in an Environment using Red Hat Openstack" on page 334.

If you want to use this procedure to upgrade your Contrail Networking release to other releases, you must engage Juniper Networks professional services. Contact your Juniper representative for information on working with professional services.

Before you begin

- Obtain the *ContrailImageTag* value for your Contrail Networking release. You can obtain this value from the readme files at the following locations:
  - Contrail Networking Release 20: README Access to Contrail Networking Registry 20xx
  - Contrail Networking Release 19: README Access to Contrail Registry 19XX

- Enable RHEL subscription for the overcloud nodes.
- Enable SSH migration for the Compute nodes if you do not have CEPH or alike storage.

  Upgrading the compute nodes requires workload migrations and CEPH or alike storage allows VM migration.

  - Modify *MigrationSshKey* value at **~/tripleo-heat-templates/environments/contrail/contrail-services.yaml** file.

    The *MigrationSshKey* parameter with SSH keys for migration is typically provided during the overcloud deployment. The parameter is used to pass SSH keys between computes nodes to allow a VM to migrate from one compute node to another. The *MigrationSshKey* parameter is an optional parameter that can be added to the **contrail-services.yaml** file. The parameter is not included in the **contrail-services.yaml** file by default.

    Run the following commands to find out the SSH keys:

    ```
    (undercloud) [stack@queensa ~]$ cat .ssh/id_rsa
    ```

    ```
    (undercloud) [stack@queensa ~]$ cat .ssh/id_rsa.pub
    ```

- Backup the Contrail configuration database.

  See "How to Backup and Restore Contrail Databases in JSON Format" on page 386.

Procedure

1. Get Contrail TripleO Heat Templates (Stable/Queens branch) from https://github.com/Juniper/contrail-tripleo-heat-templates.

   Take a back up of the existing directory if you are copying the latest directory, *contrail-tripleo-heat-templates*. You need to restore the configuration in *contrail-net.yaml*, *contrail-services.yaml*, *compute-nic-config.yaml* (for compute node running kernel mode), and *contrail-dpdk-nic-config.yaml* (for compute node running dpdk mode) files.
2. Update Contrail TripleO Puppet module to the latest version and prepare Swift Artifacts, as applicable.

   ```
   (undercloud) [stack@queensa ~]$ mkdir -p ~/usr/share/openstack-puppet/modules/tripleo
   (undercloud) [stack@queensa ~]$ git clone -b stable/queens https://github.com/Juniper/
   contrail-tripleo-puppet usr/share/openstack-puppet/modules/tripleo
   (undercloud) [stack@queensa ~]$ tar czvf puppet-modules.tgz usr/
   (undercloud) [stack@queensa ~]$ upload-swift-artifacts -c contrail-artifacts -f puppet-
   modules.tgz
   ```
3. Prepare docker registry with Contrail Networking images. It can be undercloud or a separate node.

4. Update the version of Red Hat running in the undercloud.

> **NOTE**: This procedure updates the version of Red Hat running in the undercloud before deploying the Contrail Controller In-Service Software Upgrade (ISSU) node in 5.
> You can deploy the Contrail Controller In-Service Software Upgrade (ISSU) node before performing this step if there is a reason to change the sequence in your environment.

Before you begin the upgrade process:

- Identify the Red Hat software to run with your version of Contrail Networking. See Contrail Networking Supported Platforms.

- If you have updated the undercloud using a copy of the heat templates, copy the heat templates from **/usr/share/openstack-tripleo-heat-templates** to **/home/stack/tripleo-heat-templates**.

- Add the new server nodes as bare metal nodes, and run introspection on the nodes to make them ready for deployment. These steps are summarized in "Setting Up the Overcloud" on page 191.

For details about performing this upgrade process, refer to RedHat Chapter 3. Upgrading the undercloud.

If you come across an issue during the update, see "Failed upgrade run command for any overcloud node" on page 353.

5. Deploy the Contrail Controller In-Service Software Upgrade (ISSU) node.

   a. Prepare new server node and create flavor *contrail-controller-issu* for the ISSU node.The hardware requirements for ISSU node is the same as for the Contrail Controller Node.

   b. Prepare the parameters in the yaml file, **~/tripleo-heat-templates/environments/contrail/contrail-issu.yaml**:

      - *ContrailIssuSshKey*—Generate and set the ssh keys. You require SSH access between ISSU and Contrail Controller nodes.

        ContrailIssuSshKey is same as *MigrationSshKey* .

      - *ContrailIssuImageTag*—Set the new docker images tag for the upgrade procedure.

      - *ContrailControllerIssuCount*—Set the required number of ISSU nodes. The value can be *1 or 3* and is dependent on various cluster requirements including cluster size, expected upgrade duration, etc.

   c. Update *ServiceNetMap* parameter in the *~/tripleo-heat-templates/environments/contrail/contrail-services.yaml* file.

ContrailIssuControlNetwork—Set the same value as *ContrailControlNetwork*. The default value is **tenant**.

d. Run `deploy` command with all the parameters used for deployment and the new environment file.

```
openstack overcloud deploy ...\
-e ~/tripleo-heat-templates/environments/contrail/contrail-issu.yaml
```

e. Check the status of Contrail service on the ISSU node.

All services must be *active* .

```
sudo contrail-status
```

6. Prepare for the upgrade procedure.

a. Update the parameter *ContrailImageTag* to the new version.

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

b. Download the new OpenStack container and use the new **overcloud_images.yaml** environment file which has the new containers.

```
openstack overcloud container image prepare \
--push-destination=192.x.x.1:8787  \
--tag-from-label {version}-{release} \
--output-images-file ~/local_registry_images.yaml  \
--namespace=registry.access.redhat.com/rhosp13  \
--prefix=openstack-  \
--tag-from-label {version}-{release}  \
--output-env-file ~/overcloud_images.yaml
```

Upload the OpenStack containers.

```
openstack overcloud container image upload --config-file ~/local_registry_images.yaml
```

c. Run the `openstack overcloud upgrade prepare --stack overcloud --templates ~/tripleo-heat-templates` command with all the options from deploy and the ISSU node to update the heat templates.

The files that are updated in this step vary by deployment. In the following example, the *overcloud_images.yaml*, *network-isolation.yaml*, *contrail-plugins.yaml*, *contrail-services.yaml*, *contrail-net.yaml*, *contrail-issu.yaml*, and *roles_data.yam* are prepared for the overcloud update.

```
openstack overcloud upgrade prepare --stack overcloud --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-issu.yaml \
 --roles-file ~/tripleo-heat-templates/roles_data.yaml
```

7. Run In-Service Software Upgrade (ISSU) sync.

   a. Make SSH connection to the ISSU node.

   NOTE: If you have 3 ISSU nodes deployed, you must perform SSH operations and run scripts on the same node for the entire procedure.

   b. Locate ISSU directory.

   `cd /etc/contrail/issu`

   c. Pair ISSU node with the old cluster.

   `./issu_node_pair.sh`

   d. Check the status of Contrail service on the ISSU node.

```
sudo contrail-status
```

   The *config_devicemgr*, *config_schema*, and *config_svcmonitor* containers should all be in the *inactive* state. All other containers should be in the *active* state.

   e. Run the ISSU `sync` container.

   `./issu_node_sync.sh`

   f. Check ISSU container logs.

```
sudo docker logs issu-run-sync
Config Sync initiated...
```

```
Config Sync done...
Started runtime sync...
Start Compute upgrade…
```

```
sudo docker exec issu-run-sync cat /var/log/contrail/issu_contrail_run_sync.log
…
2019-02-21 17:03:56,769 SYS_DEBUG Control on node 192.168.206.115 has CID 427885c366a5
2019-02-21 17:03:56,875 SYS_INFO Signal sent to process. exit_code = 0, stdout =
"[u'427885c366a5\n']", stderr="[]"
2019-02-21 17:03:56,878 SYS_INFO Start Compute upgrade...
```

g. Restart *contrail_control_control* container on all the ISSU nodes.

```
openstack server list --name issu -c Networks -f value | cut -d'=' -f2 | xargs -i ssh
heat-admin@{} sudo docker restart contrail_control_control
```

NOTE: The *issu_node_sync* script is run in step 7.e.

ISSU nodes are not rebooted during this upgrade procedure when these instructions can be precisely followed. ISSU node reboots, however, are sometimes required in specialized circumstances.

If an ISSU node is rebooted after step 7.e, rerun the *issu_node_sync* script:

```
./issu_node_sync
```

This script starts the *issu_node_sync* container and stops the *config_devicemgr*, *config_schema*, and *config_svcmonitor* containers.

After running the *issu_node_sync* script, you can verify that the *issu-run-sync* container is active and running:

```
docker ps -a | grep issu-run-sync
```

You must also restart the *contrail_control_control* container on all the ISSU nodes after the *issu_node_sync* script is run:

```
openstack server list --name issu -c Networks -f value | cut -d'=' -f2 | xargs -i ssh
heat-admin@{} sudo docker restart contrail_control_control
```

8. Upgrade the Compute nodes.

   Perform these steps on all the Compute Nodes.

   a. Select the Compute node for upgrade and migrate workload from it.

   ```
   openstack server migrate  --wait instance_<name>
   openstack server resize --confirm instance_<name>
   ```

   b. Verify the migrated instance has *active* state.

   ```
   openstack server show instance_<name>
   ```

   c. Upgrade the selected Compute Nodes.

      You can use comma-separated list for the various Compute nodes.

      Run the following commands on the undercloud node:

   ```
   nodes=overcloud-novacompute-0;openstack overcloud upgrade run --nodes $nodes --playbook
   upgrade_steps_playbook.yaml
   ```

      Run the following commands on the undercloud node:

   ```
   openstack overcloud upgrade run --nodes $nodes --playbook deploy_steps_playbook.yaml
   ```

   d. If the compute nodes use a new kernel or new system-level components after step 8.c, perform the following steps:

      i. Reboot the selected nodes.

      ii. For kernel-mode compute nodes:

          Make SSH connection to the upgrades nodes.

      ```
      sudo docker stop contrail_vrouter_agent
      sudo ifdown vhost0
      sudo docker start contrail-vrouter-kernel-init
      sudo ifup vhost0
      sudo docker start contrail_vrouter_agent
      ```

e. If reboot is not required after step 8.c, re-initialize *vhost0* interfaces on all the DPDK mode compute nodes.

   Make SSH connection to the upgraded Compute nodes and run the following commands:

```
ifdown vhost0
ifup vhost0
```

f. Check the status of Contrail service on the upgraded Compute nodes.

   `sudo contrail-status`

   The status must be *active*.

9. Upgrade Contrail Plugins including *Neutron, Heat,* etc. on OpenStack controllers and connect them to the ISSU node.

   Example for environment with a single OpenStack controller:

```
nodes=overcloud-controller-0
openstack overcloud upgrade run --nodes $nodes  --playbook upgrade_steps_playbook.yaml
openstack overcloud upgrade run --nodes $nodes  --playbook deploy_steps_playbook.yaml
```

   Example for environment with multiple Openstack controllers (3 controllers shown):

```
nodes=overcloud-controller-0,overcloud-controller-1,overcloud-controller-2
openstack overcloud upgrade run --nodes $nodes --playbook upgrade_steps_playbook.yaml
openstack overcloud upgrade run --nodes $nodes --playbook deploy_steps_playbook.yaml
```

10. Disconnect the ISSU node from the Contrail control plane.

   a. Make SSH connection to ISSU node.

   b. Run the following commands:

```
cd /etc/contrail/issu/
./issu_node_sync_post.sh
./issu_node_pair.sh del
```

   c. Check the status of Contrail service on the ISSU node.

   `sudo contrail-status`

   The status must be *active* or *backup*.

11. Upgrade the Contrail control plane node.

a. Run the following commands:

```
nodes=overcloud-contrailcontroller-0,overcloud-contrailcontroller-1,overcloud-
contrailcontroller-2 openstack overcloud upgrade run --nodes $nodes  --playbook
upgrade_steps_playbook.yaml
openstack overcloud upgrade run --nodes $nodes  --playbook deploy_steps_playbook.yaml
openstack overcloud upgrade run --nodes $nodes --playbook
post_upgrade_steps_playbook.yaml
```

b. Check the status of Contrail service on the Contrail control plane node.

`sudo contrail-status`

The status must be *active* or *backup*.

12. Upgrade Contrail Analytics and Contrail AnalyticsDB nodes:

Example for an environment with three Contrail Analytics and three Contrail AnalyticsDB nodes:

```
nodes=contrailanalytics-0,contrailanalytics-1,contrailanalytics-2,contrailanalyticsdatabase-
0,contrailanalyticsdatabase-1,contrailanalyticsdatabase-2
openstack overcloud upgrade run --nodes $nodes --playbook upgrade_steps_playbook.yaml
openstack overcloud upgrade run --nodes $nodes --playbook deploy_steps_playbook.yaml
```

13. Connect the ISSU node to the upgraded Contrail control plane node.

a. Make SSH connection to the ISSU node.

b. Pair the ISSU node with upgraded Contrail control plane.

`cd /etc/contrail/issu ./issu_node_pair.sh add pair_with_new`

c. Sync data with new Contrail control plane.

`issu_config=issu_revert.conf ./issu_node_sync.sh`

d. Restart *control* container on the upgraded nodes.

Run the following command from the Director.

```
openstack server list --name "overcloud-contrailcontroller-" -c Networks -f value | cut -d'=' -f2 | xargs
-i ssh heat-admin@{} sudo docker restart contrail_control_control
```

14. Run the post upgrade task on the compute nodes and the Openstack controllers.

```
nodes=overcloud-novacompute-0,overcloud-novacompute-1 openstack overcloud upgrade run --
nodes $nodes --playbook post_upgrade_steps_playbook.yaml
```

```
nodes=overcloud-controller-0 openstack overcloud upgrade run --nodes $nodes --playbook
post_upgrade_steps_playbook.yaml
```

15. Disconnect ISSU and upgraded Contrail control plane.

    a. Make SSH connection to ISSU node.

    b. Un-pair ISSU node with the old Contrail cluster.

    ```
    cd /etc/contrail/issu/
    issu_config=issu_revert.conf ./issu_node_sync_post.sh
    ./issu_node_pair.sh del pair_with_new
    ```

16. Reconnect the OpenStack nodes and Compute nodes to the upgraded control plane.

    Run the command with all the parameters from `deploy`.

    ```
    openstack overcloud upgrade converge  \
    --stack overcloud \
    ...
    -e ~/tripleo-heat-templates/environments/contrail/contrail-issu.yaml
    ```

17. If the nodes use new kernel or new system level components, reboot the OpenStack controller and Contrail controller nodes.

    - Reboot OpenStack controllers as mentioned in section 5.1 of RedHat Rebooting the Overcloud chapter.

    - Reboot Contrail controllers one by one.

      Make SSH connection to each controller and perform sudo reboot.
      You must wait till the node is rebooted and Contrail services are up.

      `sudo contrail-status`

18. Check the status of Contrail service on all the upgrades nodes.

    `sudo contrail-status`

    The status must be *active*.

19. Remove the ISSU node from the cluster.

    `set ContrailControllerIssuCount: 0`

    Run stack deploy command with all the parameters.

    `openstack overcloud deploy \ … -e ~/tripleo-heat-templates/environments/contrail/contrail-issu.yaml`

Troubleshoot

IN THIS SECTION

Following are the known issues:

Failed upgrade run command for OpenStack controller

IN THIS SECTION

Problem

Description

You see the following error:

```
nodes=overcloud-controller-0
openstack overcloud upgrade run --nodes $nodes --playbook upgrade_steps_playbook.yaml
...
TASK [Enable the cinder_volume cluster resource] *******************************
Thursday 25 July 2019  11:38:57 -0400 (0:00:00.887)       0:03:16.905 *********
FAILED - RETRYING: Enable the cinder_volume cluster resource (5 retries left).
FAILED - RETRYING: Enable the cinder_volume cluster resource (4 retries left).
FAILED - RETRYING: Enable the cinder_volume cluster resource (3 retries left).
FAILED - RETRYING: Enable the cinder_volume cluster resource (2 retries left).
FAILED - RETRYING: Enable the cinder_volume cluster resource (1 retries left).

fatal: [overcloud-controller-0]: FAILED! => {"attempts": 5, "changed": false, "error": "Error:
resource 'openstack-cinder-volume' is not running on any node\n", "msg": "Failed, to set the
```

```
resource openstack-cinder-volume to the state enable", "output": "", "rc": 1}

PLAY RECAP **********************************************************************
overcloud-controller-0     : ok=149  changed=68   unreachable=0    failed=1

Thursday 25 July 2019  11:39:31 -0400 (0:00:34.195)       0:03:51.101 *********
```

For details, refer to https://access.redhat.com/solutions/4122571.

Solution

- Make SSH connection to the OpenStack controller node.
- Run the following command:
  `sudo docker rm cinder_volume_init_bundle`
- Check if the cinder volume is in failed resources list.

  `sudo pcs status`
- Check if the cinder volume is not in failed resource list.

  `sudo pcs resource cleanup`
- Re-run the upgrade `run` command.

Failed upgrade run command for any overcloud node

IN THIS SECTION

Problem

Description

You see the following error:

```
*******************************************************
TASK [include_tasks] ************************************************************
Wednesday 02 October 2019 09:21:02 -0400 (0:00:00.448) 0:00:29.101 *****
fatal: [overcloud-novacompute-1]: FAILED! => {"msg": "No variable found with this name:
Compute_pre_deployments"}NO MORE HOSTS LEFT
********************************************************
```

Solution

This is a broken default behavior if a variable is missing.

Edit the **tripleo-heat-templates/common/deploy-steps.j2** to apply the following change:

```
content_copyzoom_out_map
(undercloud) [stack@queensa common]$ diff -U 3 deploy-steps.j2.org deploy-steps.j2
--- deploy-steps.j2.org 2019-10-04 09:09:57.414000000 -0400
+++ deploy-steps.j2     2019-10-04 09:13:51.120000000 -0400
@@ -433,7 +433,7 @@
                 - include_tasks: deployments.yaml
                   vars:
                     force: false
-                  with_items: "{{ '{{' }} lookup('vars', tripleo_role_name +
'_pre_deployments')|default([]) {{ '}}' }}"
+                  with_items: "{{ '{{' }} hostvars[inventory_hostname][tripleo_role_name ~
'_pre_deployments']|default([]) {{ '}}' }}"
               tags:
                 - overcloud
                 - pre_deploy_steps
@@ -521,7 +521,7 @@
                 - include_tasks: deployments.yaml
                   vars:
                     force: false
-                  with_items: "{{ '{{' }} lookup('vars', tripleo_role_name +
'_post_deployments')|default([]) {{ '}}' }}"
+                  with_items: "{{ '{{' }} hostvars[inventory_hostname][tripleo_role_name ~
```

```
'_post_deployments']|default([]) {{ '}}' }}"
                tags:
                  - overcloud
                  - post_deploy_steps
```

After editing the **deploy-steps.j2**, run the `prepare` command as given in step 5.6.c.
Once it is completed, continue the upgrade procedure where you left off.

RELATED DOCUMENTATION

Upgrading Contrail Networking Release 5.x or Release 190x with RHOSP13 to Contrail Networking Release till 1909 with RHOSP13

Understanding Red Hat OpenStack Platform Director | 174

How to Upgrade From Contrail Networking Release 4.1.4 with RHOSP10 to Contrail Networking Release 1912.L1 or 1912.L2 with RHOSP13

IN THIS SECTION

The goal of this topic is to provide a combined procedure to upgrade Red Hat OpenStack Platform (RHOSP) from RHOSP 10 to RHOSP 13 by leveraging Red Hat Fast Forward Upgrade (FFU) procedure while simultaneously upgrading Contrail Networking from Release 4.1.4 to Release 1912.L1 or 1912.L2. The procedure leverages the In Service Software Upgrade (ISSU) procedure from Contrail to minimize the downtime.

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.

Refer to Red Hat OpenStack Platform 13 Fast Forward Upgrades for details on RHOSP 10 to RHOSP 13 FFU procedure of OpenStack Platform environment from one long life version to the next long life version.

When to Use This Procedure

We recommend engaging Juniper Networking professional services for assistance with performing this procedure.

This procedure is used when you are running Contrail Networking Release 4.1.4 in an environment using Red Hat Openstack Platform 10 (RHOSP 10) for orchestration and want to upgrade your environment to a Contrail Networking 19 release using RHOSP13. The procedure leverages the Red Hat Fast Forward Upgrade (FFU) process for the RHOSP upgrade and the In Service Software Upgrade (ISSU) procedure for the Contrail Networking upgrade.

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

Table 17: Validated Upgrade Scenarios

Starting Contrail Networking Release	Target Upgraded Contrail Networking Release
4.1.4	1912.L1
4.1.4	1912.L2

If you want to use this procedure to upgrade your Contrail Networking release to other releases, you must engage Juniper Networks professional services. Contact your Juniper representative for additional information.

If you want to upgrade from an environment using Contrail Networking Release 3 and RHOSP10 to an environment running Contrail Networking Release 19 and RHOSP13, see "How to Upgrade From Contrail Networking Release 3.x or 4.x with RHOSP10 to Contrail Networking Release 1907 with RHOSP13" on page 365. You can also use this procedure to upgrade from a Contrail Networking Release 4 environment to Contrail Networking Release 1907.

If you are using RHOSP13 for orchestration and would like to upgrade to a Contrail Networking Release 20 environment without upgrading RHOSP, see "Updating Contrail Networking using the Zero Impact Upgrade Process in an Environment using Red Hat Openstack" on page 334

If you are using RHOSP13 for orchestration and would like to upgrade to a Contrail Networking Release 19 environment without upgrading RHOSP, see "Upgrading Contrail Networking with Red Hat Openstack 13 using ISSU" on page 341.

Before You Begin

Before you begin:

- Obtain the *ContrailImageTag* from the README Access to Contrail Registry 19XX.

- Enable FFU RedHat subscription for overcloud nodes as the upgrade involves a yum update which needs RPM repositories enabled. The subscription must include access to both OSP10 and OSP13 repositories (*rhel-7-server-openstack-13-rpms*). Additionally, the subscription must have access to the *repo rhel-server-rhscl-7-rpms* repository. ISSU node uses *python27-python-pip* from this repository.

- Ensure ISSU nodes have internet access during the installation to install *docker-compose* via *pip*.

  If the ISSU nodes do not have internet access, you must download required packages manually using a node with internet access after the ISSU node installation. You must then transfer packages from this node to the ISSU nodes and install them.

- Per Red Hat Openstack support guidelines, do not change IP addresses during this upgrade.

Upgrading Contrail Networking Release 4.1.4 with RHOSP10 to Contrail Networking Release 1912.L1 or 1912.L2 with RHOSP13

To perform the upgrade:

1. Follow *chapter 2* from step 2.1 through step 2.8 of Red Hat OpenStack Platform 13 Fast Forward Upgrades to prepare for an Openstack Platform Upgrade.

   > **NOTE**: Do not reboot the compute nodes as mentioned in step 2.9.

2. Prepare to deploy Contrail In-Service Software Upgrade (ISSU) node.

   a. Update Contrail TripleO Heat Templates (Stable/Newton branch) from https://github.com/Juniper/contrail-tripleo-heat-templates.

   b. Update Contrail TripleO Puppet module to the latest version and prepare Swift Artifacts.

   ```
   mkdir -p ~/usr/share/openstack-puppet/modules/tripleo
   git clone -b stable/newton https://github.com/Juniper/contrail-tripleo-puppet
   usr/share/openstack-puppet/modules/tripleo
   tar czvf puppet-modules.tgz usr/
   upload-swift-artifacts -c contrail-artifacts -f puppet-modules.tgz
   ```

   c. Prepare docker registry with Contrail Networking Release 1912.L1 or 1912.L2 images. It can be undercloud or a separate node.

      In addition to Contrail Networking images, the required ansible deployer image is *contrail-kolla-ansible-deployer* which is available at https://hub.docker.com/r/opencontrailnightly/contrail-kolla-ansible-deployer/tags.

      You must use the same tag as of other Contrail Networking containers.

To prepare images:

```
contrail-tripleo-heat-templates/tools/contrail/import_contrail_container_docker.sh
```

For details on building container, refer to https://github.com/Juniper/contrail-deployers-containers.

3. Deploy Contrail In-Service Software Upgrade (ISSU) node.

   a. Prepare new bare-metal server node and create flavor *contrail-controller-issu* for ISSU node. The hardware requirements for ISSU node is the same as for the Contrail Controller Node.

   b. Prepare parameters in the yaml file.

```
tripleo-heat-templates/environments/contrail/contrail-issu-ffu.yaml:
ContrailIssuSshKey - generate and put ssh keys (for ISSU there is SSH access required
between ISSU and Contrail Controller nodes)
Provide docker registry info: ContrailRegistryCertUrl, ContrailRegistry,
ContrailRegistryInsecure, ContrailImageTag
```

   c. Run deploy command with all parameters used for deployment and new environment file.

```
'-e tripleo-heat-templates/environments/contrail/contrail-issu-ffu.yam
```

   d. If you do not have internet access, after ISSU node installation, you must download required packages manually an a node with internet access, transfer packages on all the ISSU nodes and install them.

      To do so:

      i. Prepare required modules for docker-compose.

         Run the following commands on the a RHEL7 node which has internet access:

```
yum install -y --enablerepo=rhel-server-rhscl-7-rpms python27-python-pip
source scl_source enable python27
mkdir pip-pkg docker-compose-pkg
pip download pip -d ./pip-pkg
pip download pip -d ./docker-compose-pkg
tar cvfz docker-compose.tar.gz docker-compose-pkg pip-pkg
```

      ii. Upload the archive to all the ISSU nodes.

iii. Install packages on all the ISSU nodes.

```
source scl_source enable python27
tar xvfz docker-compose.tar.gz
pushd pip-pkg
pip install * -f ./ --no-index
popd
pushd docker-compose-pkg
pip install * -f ./ --no-index
popd
```

4. Run In-Service Software Upgrade (ISSU).

   a. Make SSH connection to ISSU node.

   > **NOTE**: If you have 3 ISSU nodes deployed, you must perform SSH operations and run scripts on the same node for the entire procedure.

   b. Deploy ISSU node.

   `cd /etc/contrail/issu ./issu_node_deploy.sh`

   c. Check status of Contrail Networking service.

   `sudo contrail-status`

   The status must be *active*.

   d. Pair ISSU node with the old cluster.

   `./issu_node_pair.sh`

   e. Check status of Contrail service on ISSU node.

   `sudo contrail-status`

   > **NOTE**: All services must be *active* except *config_devicemgr, config_schema and config_svcmonitor.*

   f. Run ISSU `sync` container.

   `./issu_node_sync.sh`

g. Check ISSU container logs.

```
sudo docker logs issu-run-sync
Config Sync initiated...
Config Sync done...
Started runtime sync...
Start Compute upgrade…
```

```
sudo docker exec issu-run-sync cat /var/log/contrail/issu_contrail_run_sync.log
2019-02-21 17:03:56,769 SYS_DEBUG Control on node 192.168.206.115 has CID 427885c366a5
2019-02-21 17:03:56,875 SYS_INFO Signal sent to process. exit_code = 0, stdout =
"[u'427885c366a5\n']", stderr="[]"
2019-02-21 17:03:56,878 SYS_INFO Start Compute upgrade..
```

h. Restart the *contrail_control_control* container on all ISSU nodes:

```
sudo docker restart contrail_control_control
```

5. Upgrade vRouter on Compute nodes.

   Perform these steps on all the Compute Nodes.

   a. Select the Compute node for upgrade and migrate workload from it.

   b. Make SSH connection to Compute node.

   c. Deploy new vRouter

      `cd /etc/contrail/issu/ ./issu_compute_deploy.sh`

   d. Verify the **network-functions-vrouter-ffu-env-pre** file.

```
cat /etc/sysconfig/network-scripts/network-functions-vrouter-ffu-env-pre
TYPE=kernel
BIND_INT=ens3
CONTRAIL_VROUTER_AGENT_CONTAINER_NAME=contrail_vrouter_agent
```

   You required this file because *os-net-config* is outdated in Newton and doesn't provide the required variables for Contrail Networking Release 19 software.

For kernel mode and DPDK, the file is created automatically by the script mentioned in the step 5.d . Verify the file is created as per your deployment.

```
DRIVER=uio_pci_generic
CPU_LIST=0x1f
BOND_MODE=4
BOND_POLICY=layer2+3
VLAN_ID=101
```

e. Reboot the node.

   `sudo reboot`

f. After node has rebooted, make SSH connection to the compute node and verify node status.

   `modinfo vrouter (for kernel mode) sudo contrail-status`

g. Repeat this step to upgrade the vRouter on each compute node.

6. Upgrade undercloud to OpenStack Plaform (OSP) 13.

   Follow *chapter 3* till *chapter 5* of Red Hat OpenStack Platform 13 Fast Forward Upgrades to configure a container image source and prepare for the overcloud upgrade.

7. Copy the TripleO (OOO, OpenStack on OpenStack) Heat Templates and prepare united Contrail parameters for OSP 10 with new parameters for OSP 13.

   a. Backup old TripleO Heat Templates.

```
mv tripleo-heat-templates tripleo-heat-templates-osp10
cp -r /usr/share/openstack-tripleo-heat-templates/ ~/tripleo-heat-templates
git clone -b stable/queens https://github.com/Juniper/contrail-tripleo-heat-templates
cp -r ~/contrail-tripleo-heat-templates/* ~/tripleo-heat-templates
```

   b. Set united for OSP 10 and OSP 13 parameters.

```
tripleo-heat-templates/environments/contrail/contrail-services.yaml
tripleo-heat-templates/environments/contrail/contrail-net.yaml
```

   c. Define NIC templates.

```
tripleo-heat-templates/network/config/contrail/compute-nic-config.yaml
tripleo-heat-templates/network/config/contrail/contrail-controller-nic-config.yaml
tripleo-heat-templates/network/config/contrail/controller-nic-config.yaml
```

d. Define role parameters for Computer Node, DPDK, TSN, etc.

```
ComputeParameters:
    ContrailSettings:
      VROUTER_GATEWAY: 192.168.206.2
      MAINTANENCE_MODE: true
      BGP_ASN: 64512
      BGP_AUTO_MESH: true
```

e. Remove *contrail-artifacts*, if any to avoid rewriting contrail Puppets.

```
swift delete contrail-artifacts
rm -f .tripleo/environments/deployment-artifacts.yaml
```

f. Make SSH connection to ISSU node.

g. Stop the ISSU service and un-pair ISSU node with the old Contrail Control plane.

`cd /etc/contrail/issu/ ./issu_node_sync_post.sh ./issu_node_pair.sh del`

8. Upgrade overcloud to OpenStack Platform (OSP) 13.

> **NOTE**: Ensure the *contrail-services.yaml* file has an empty string for the *ContrailVrouterHugepages1GB:* parameter before the compute nodes are updated in this step.
>
> ```
> parameter_defaults:
>   …
>   ContrailVrouterHugepages1GB: ""
> ```
>
> The compute node updates occur in Step 6.2 of the Red Hat OpenStack Platform 13 Fast Forward Upgrades procedure.

Follow *chapter 6* of Red Hat OpenStack Platform 13 Fast Forward Upgrades.

a. You can use this role file—**tripleo-heat-templates/roles_data_contrail_ffu.yaml** or update the role file with ISSU role and ISSU services.

b. Add the following command environment files:

- **tripleo-heat-templates/environments/contrail/contrail-plugins.yaml**
- **tripleo-heat-templates/environments/contrail/contrail-services.yaml**

For example:

```
openstack overcloud ffwd-upgrade prepare  \
   --templates tripleo-heat-templates/ \
  --roles-file tripleo-heat-templates/roles_data_contrail_ffu.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-services.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-issu-ffu.yaml
```

c. Perform the following steps after upgrading all the controller nodes as stated in step 6.2 of Red Hat OpenStack Platform 13 Fast Forward Upgrades:

> **NOTE**: If the *contrail api* and *nodemgr* statuses are `Inactive` with a *failed to connect to keystone* error after upgrading the controller nodes, perform the following steps:
>
> ```
> docker cp config_api_1:/entrypoint.sh ./
> vi entrypoint.sh
> #!/bin/bash
> export KEYSTONE_AUTH_URL_VERSION=/v3
> docker cp entrypoint.sh config_api_1:/entrypoint.sh
> docker restart config_api_1
> ```
>
> You may have to repeat these exact steps on the config node manager.

i. Upgrade Contrail control plane nodes.

```
openstack overcloud upgrade run --nodes
ContrailController,ContrailAnalytics,ContrailAnalyticsDatabase --skip-tags validation
```

ii. Check status of Contrail Networking service.

```
sudo contrail-status
```

The status must be *active*.

iii. Make SSH connection to ISSU node.

iv. Pair ISSU node with upgraded Contrail control plane.

```
cd /etc/contrail/issu ./issu_node_pair.sh add pair_with_new
```

v. Check status of Contrail Networking service.

```
sudo contrail-status
```

On ISSU node, the status for all the services must be *active*.

On Contrail control nodes, the status for all the services except *config_device_manager, config_schema* and *config_svc_monitor* must be *active*. The status for *config_device_manager, config_schema* and *config_svc_monitor* must be *inactive*.

vi. Sync data with new Contrail control plane.

```
issu_config=issu_revert.conf ./issu_node_sync.sh
```

vii. Check ISSU Logs.

```
sudo docker logs issu-run-sync sudo docker exec issu-run-sync cat /var/log/contrail/
issu_contrail_run_sync.log
```

d. Follow *Step 6.3* of Red Hat OpenStack Platform 13 Fast Forward Upgrades to upgrade compute nodes.

e. Perform the following steps after upgrading all the compute nodes as stated in step 6.4 of Red Hat OpenStack Platform 13 Fast Forward Upgrades:

i. Check status of Contrail Networking service on compute nodes.

```
sudo contrail-status
```

The status must be *active*.

ii. Make SSH connection to ISSU node.

iii. Un-pair ISSU node with the old Contrail cluster.

```
issu_config=issu_revert.conf ./issu_node_sync_post.sh ./issu_node_pair.sh del pair_with_new
```

iv. Check status of Contrail Networking service on control nodes.

```
sudo contrail-status
```

All services must be *active*.

9. Follow steps 6.5 till 6.10 of Red Hat OpenStack Platform 13 Fast Forward Upgrades to upgrade CEPH storage node, converged nodes, etc.
10. Finalize ISSU upgrade.

    Remove ISSU node from the cluster.

    ```
    set ContrailControllerIssuCount: 0
    ```

    Run stack deploy command.
11. Follow chapter 7 of Red Hat OpenStack Platform 13 Fast Forward Upgrades to execute post upgrade steps.

SEE ALSO

Setting Up the Infrastructure | 179

RELATED DOCUMENTATION

Understanding Contrail Networking Components | 4
How to Upgrade From Contrail Networking Release 3.x or 4.x with RHOSP10 to Contrail Networking Release 1907 with RHOSP13 | 365
Updating Contrail Networking using the Zero Impact Upgrade Process in an Environment using Red Hat Openstack | 334
Upgrading Contrail Networking with Red Hat Openstack 13 using ISSU | 341

How to Upgrade From Contrail Networking Release 3.x or 4.x with RHOSP10 to Contrail Networking Release 1907 with RHOSP13

IN THIS SECTION

The goal of this topic is to provide a combined procedure to upgrade Red Hat OpenStack Platform (RHOSP) from RHOSP 10 to RHOSP 13 by leveraging Red Hat Fast Forward Upgrade (FFU) procedure while simultaneously upgrading Contrail Networking from Release 4.1.x to Release 1907. The procedure leverages the In Service Software Upgrade (ISSU) procedure from Contrail to minimize the downtime.

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.

Refer to Red Hat OpenStack Platform 13 Fast Forward Upgrades for details on RHOSP 10 to RHOSP 13 Fast Forward Upgrade (FFU) procedure of OpenStack Platform environment from one long life version to the next long life version.

NOTE: This procedure is also applicable for upgrading Contrail Networking Release 3.x or 4.x with RHOSP10 to Contrail Networking Release *5.1.x* with RHOSP13.

When to Use This Procedure

We recommend engaging Juniper Networking professional services for assistance with performing this procedure.

This procedure is used when you are running Contrail Networking Release 3 or Contrail Networking Release 4 in an environment using Red Hat Openstack Platform 10 (RHOSP 10) for orchestration and want to upgrade your environment to a Contrail Networking 19 release using RHOSP13. The procedure leverages the Red Hat Fast Forward Upgrade (FFU) procedure for the RHOSP upgrade and the In Service Software Upgrade (ISSU) procedure for the Contrail Networking upgrade.

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

Table 18: Validated Upgrade Scenarios

Starting Contrail Networking Release	Target Upgraded Contrail Networking Release
X	X
X	X
X	X

If you want to use this procedure to upgrade your Contrail Networking release to other releases, you must engage Juniper Networks professional services. Contact your Juniper representative for additional information.

Before You Begin

Access *ContrailImageTag* located at README Access to Contrail Registry 19XX.

NOTE: You must enable FFU RedHat subscription for overcloud nodes as the upgrade involves yum update which needs RPM repositories enabled. The subscription must include access to both OSP10 and OSP13 repositories (*rhel-7-server-openstack-13-rpms*). Additionally, the subscription must have access to the *repo rhel-server-rhscl-7-rpms* repository. ISSU node uses *python27-python-pip* from this repository.

ISSU nodes must have internet access during installation to install *docker-compose* via *pip*.

If you do not have internet access, after ISSU node installation, you must download required packages manually an a node with internet access, transfer packages on all the ISSU nodes and install them.

Upgrading Contrail Networking Release 3.x or 4.x with RHOSP10 to Contrail Networking Release 1907 with RHOSP13

1. Follow *chapter 2* from step 2.1 through step 2.8 of Red Hat OpenStack Platform 13 Fast Forward Upgrades to prepare for an Openstack Platform Upgrade.

   NOTE: Do not reboot the compute nodes as mentioned in step 2.9.

2. Prepare to deploy Contrail In-Service Software Upgrade (ISSU) node.

   a. Update Contrail TripleO Heat Templates (Stable/Newton branch) from https://github.com/Juniper/contrail-tripleo-heat-templates.

   b. Update Contrail TripleO Puppet module to the latest version and prepare Swift Artifacts.

   ```
   mkdir -p ~/usr/share/openstack-puppet/modules/tripleo
   git clone -b stable/newton https://github.com/Juniper/contrail-tripleo-puppet
   usr/share/openstack-puppet/modules/tripleo
   tar czvf puppet-modules.tgz usr/
   upload-swift-artifacts -c contrail-artifacts -f puppet-modules.tgz
   ```

   c. Prepare docker registry with Contrail Networking R1907 images. It can be undercloud or a separate node.

   In addition to Contrail Networking images, the required ansible deployer image is *contrail-kolla-ansible-deployer* which is available at https://hub.docker.com/r/opencontrailnightly/contrail-kolla-ansible-deployer/tags.

   You must use the same tag as of other Contrail Networking containers.

   To prepare images:

   ```
   contrail-tripleo-heat-templates/tools/contrail/import_contrail_container_docker.sh
   ```

For details on building container, refer to https://github.com/Juniper/contrail-deployers-containers.

3. Deploy Contrail In-Service Software Upgrade (ISSU) node.

   a. Prepare new bare-metal server node and create flavor *contrail-controller-issu* for ISSU node.The hardware requirements for ISSU node is the same as for the Contrail Controller Node.

   b. Prepare parameters in the yaml file.

```
tripleo-heat-templates/environments/contrail/contrail-issu-ffu.yaml:
ContrailIssuSshKey - generate and put ssh keys (for ISSU there is SSH access required
between ISSU and Contrail Controller nodes)
Provide docker registry info: ContrailRegistryCertUrl, ContrailRegistry,
ContrailRegistryInsecure, ContrailImageTag
```

   c. Run deploy command with all parameters used for deployment and new environment file.

```
'-e tripleo-heat-templates/environments/contrail/contrail-issu-ffu.yam
```

   d. If you do not have internet access, after ISSU node installation, you must download required packages manually an a node with internet access, transfer packages on all the ISSU nodes and install them.

      To do so:

      i. Prepare required modules for docker-compose.

         Run the following commands on the a RHEL7 node which has internet access:

```
yum install -y --enablerepo=rhel-server-rhscl-7-rpms python27-python-pip
source scl_source enable python27
mkdir pip-pkg docker-compose-pkg
pip download pip -d ./pip-pkg
pip download pip -d ./docker-compose-pkg
tar cvfz docker-compose.tar.gz docker-compose-pkg pip-pkg
```

      ii. Upload the archive to all the ISSU nodes.

      iii. Install packages on all the ISSU nodes.

```
source scl_source enable python27
tar xvfz docker-compose.tar.gz
```

```
pushd pip-pkg
pip install * -f ./ --no-index
popd
pushd docker-compose-pkg
pip install * -f ./ --no-index
popd
```

4. Run In-Service Software Upgrade (ISSU).

a. Make SSH connection to ISSU node.

> **NOTE**: If you have 3 ISSU nodes deployed, you must perform SSH operations and run scripts on the same node for the entire procedure.

b. Deploy ISSU node.

`cd /etc/contrail/issu ./issu_node_deploy.sh`

c. Check status of Contrail Networking service.

`sudo contrail-status`

The status must be *active*.

d. Pair ISSU node with the old cluster.

`./issu_node_pair.sh`

e. Check status of Contrail service on ISSU node.

`sudo contrail-status`

> **NOTE**: All services must be *active* except *config_devicemgr, config_schema and config_svcmonitor*.

f. Run ISSU `sync` container.

`./issu_node_sync.sh`

g. Check ISSU container logs.

```
sudo docker logs issu-run-sync
Config Sync initiated...
Config Sync done...
```

```
Started runtime sync...
Start Compute upgrade…
```

```
sudo docker exec issu-run-sync cat /var/log/contrail/issu_contrail_run_sync.log
2019-02-21 17:03:56,769 SYS_DEBUG Control on node 192.168.206.115 has CID 427885c366a5
2019-02-21 17:03:56,875 SYS_INFO Signal sent to process. exit_code = 0, stdout =
"[u'427885c366a5\n']", stderr="[]"
2019-02-21 17:03:56,878 SYS_INFO Start Compute upgrade..
```

5. Upgrade vRouter on Compute nodes.

   Perform these steps on all the Compute Nodes.

   a. Select the Compute node for upgrade and migrate workload from it.

   b. Make SSH connection to Compute node.

   c. Deploy new vRouter

      `cd /etc/contrail/issu/ ./issu_compute_deploy.sh`

   d. Verify the **network-functions-vrouter-ffu-env-pre** file.

```
cat /etc/sysconfig/network-scripts/network-functions-vrouter-ffu-env-pre
TYPE=kernel
BIND_INT=ens3
CONTRAIL_VROUTER_AGENT_CONTAINER_NAME=contrail_vrouter_agent
```

      You required this file because *os-net-config* is outdated in Newton and doesn't provide the required variables for Contrail Networking Release 5.x or Release 1907 and later.

      For kernel mode and DPDK, the file is created automatically by the script mentioned in the step 5.d . Verify the file is created as per your deployment.

```
DRIVER=uio_pci_generic
CPU_LIST=0x1f
BOND_MODE=4
BOND_POLICY=layer2+3
VLAN_ID=101
```

   e. Reboot the node.

      `sudo reboot`

f. Make SSH connection to the compute node and verify node status.

`modinfo vrouter (for kernel mode) sudo contrail-status`

6. Upgrade undercloud to OpenStack Plaform (OSP) 13.

   Follow *chapter 3* till *chapter 5* of Red Hat OpenStack Platform 13 Fast Forward Upgrades to configure a container image source and prepare for the overcloud upgrade.

7. Copy the TripleO (OOO, OpenStack on OpenStack) Heat Templates and prepare united Contrail parameters for OSP 10 with new parameters for OSP 13.

   a. Backup old TripleO Heat Templates.

```
mv tripleo-heat-templates tripleo-heat-templates-osp10
cp -r /usr/share/openstack-tripleo-heat-templates/ ~/tripleo-heat-templates
git clone -b stable/queens https://github.com/Juniper/contrail-tripleo-heat-templates
cp -r ~/contrail-tripleo-heat-templates/* ~/tripleo-heat-templates
```

   b. Set united for OSP 10 and OSP 13 parameters.

```
tripleo-heat-templates/environments/contrail/contrail-services.yaml
tripleo-heat-templates/environments/contrail/contrail-net.yaml
```

   c. Define NIC templates.

```
tripleo-heat-templates/network/config/contrail/compute-nic-config.yaml
tripleo-heat-templates/network/config/contrail/contrail-controller-nic-config.yaml
tripleo-heat-templates/network/config/contrail/controller-nic-config.yaml
```

   d. Define role parameters for Computer Node, DPDK, TSN, etc.

```
ComputeParameters:
    ContrailSettings:
      VROUTER_GATEWAY: 192.168.206.2
      MAINTANENCE_MODE: true
      BGP_ASN: 64512
      BGP_AUTO_MESH: true
```

e. Remove *contrail-artifacts*, if any to avoid rewriting contrail Puppets.

```
swift delete contrail-artifacts
rm -f .tripleo/environments/deployment-artifacts.yaml
```

f. Make SSH connection to ISSU node.

g. Stop the ISSU service and un-pair ISSU node with the old Contrail Control plane.

`cd /etc/contrail/issu/ sudo docker rm --force issu-run-sync ./issu_node_pair.sh del`

8. Upgrade overcloud to OpenStack Plaform (OSP) 13.

Follow *chapter 6* of Red Hat OpenStack Platform 13 Fast Forward Upgrades.

a. You can use this role file—**tripleo-heat-templates/roles_data_contrail_ffu.yaml** or update the role file with ISSU role and ISSU services.

b. Add the following command environment files:

- **tripleo-heat-templates/environments/contrail/contrail-plugins.yaml**
- **tripleo-heat-templates/environments/contrail/contrail-services.yaml**

For example:

```
openstack overcloud ffwd-upgrade prepare  \
   --templates tripleo-heat-templates/ \
  --roles-file tripleo-heat-templates/roles_data_contrail_ffu.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-services.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-issu-ffu.yaml
```

c. Perform the following steps after upgrading all the controller nodes as stated in step 6.2 of Red Hat OpenStack Platform 13 Fast Forward Upgrades:

i. Upgrade Contrail control plane nodes.

```
openstack overcloud upgrade run --nodes
ContrailController,ContrailAnalytics,ContrailAnalyticsDatabase,ContrailControllerIssu --skip-tags
validation
```

ii. Check status of Contrail Networking service.

```
sudo contrail-status
```

The status must be *active*.

iii. Make SSH connection to ISSU node.

iv. Pair ISSU node with upgraded Contrail control plane.

```
cd /etc/contrail/issu ./issu_node_pair.sh add pair_with_new
```

v. Check status of Contrail Networking service.

```
sudo contrail-status
```

On ISSU node, the status for all the services must be *active*.

On Contrail control nodes, the status for all the services except *config_device_manager, config_schema* and *config_svc_monitor* must be *active*. The status for *config_device_manager, config_schema* and *config_svc_monitor* must be *inactive*.

vi. Sync data with new Contrail control plane.

```
issu_config=issu_revert.conf ./issu_node_sync.sh
```

vii. Check ISSU Logs.

```
sudo docker logs issu-run-sync sudo docker exec issu-run-sync cat /var/log/contrail/
issu_contrail_run_sync.log
```

d. Follow *Step 6.3* of Red Hat OpenStack Platform 13 Fast Forward Upgrades to upgrade compute nodes.

e. Perform the following steps after upgrading all the compute nodes as stated in step 6.4 of Red Hat OpenStack Platform 13 Fast Forward Upgrades:

i. Check status of Contrail Networking service on compute nodes.

```
sudo contrail-status
```

The status must be *active*.

ii. Make SSH connection to ISSU node.

iii. Disable ISSU node.

```
sudo docker rm --force issu-run-sync
```

iv. Un-pair ISSU node with the old Contrail cluster.

```
./issu_node_pair.sh del pair_with_new
```

v. Check status of Contrail Networking service on control nodes.

```
sudo contrail-status
```

The status must be *active*.

9. Follow steps 6.5 till 6.10 of Red Hat OpenStack Platform 13 Fast Forward Upgrades to upgrade CEPH storage node, converged nodes, etc.
10. Finalize ISSU upgrade.

    Remove ISSU node from the cluster.

    ```
    set ContrailControllerIssuCount: 0
    ```

    Run stack deploy command.
11. Follow chapter 7 of Red Hat OpenStack Platform 13 Fast Forward Upgrades to execute post upgrade steps.

SEE ALSO

Setting Up the Infrastructure | 179

Upgrading Contrail Networking using the Ansible Deployer In-Service Software Upgrade Procedure in OpenStack Environments

IN THIS SECTION

When to Use This Procedure?

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, 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 19: 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
```

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

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

The configuration appears as follows and can be stored locally:

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

1. Detect the config-api image ID.

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

2. Run the pre-synchronization.

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

3. Run the run-synchronization.

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

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

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

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

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

Transferring the Compute Nodes into the New Cluster

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

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

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. 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
```

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", "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
```

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. 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.

10. 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.

CHAPTER 14

Backup and Restore Contrail Software

IN THIS CHAPTER

How to Backup and Restore Contrail Databases in JSON Format

IN THIS SECTION

This document shows how to backup and restore the Contrail databases—Cassandra and Zookeeper—in JSON format.

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 config nodes. Create the **/tmp/db-dump** directory on any of the config node hosts.

```
mkdir /tmp/db-dump
```

2. On the same config 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 configuration node includes the complete Cassandra database for all configuration nodes in the cluster. Steps 1 and 2, therefore, only need to be performed on one configuration node.

3. Stop the following docker configuration services on all of the Contrail configuration 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 config node in the cluster.

4. Return to the config 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 config 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 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 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 388.

```
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
```

NOTE: In case of RHOSP deployment with RedHat IDM (IPA) (only if CA certificate is different), mount the following file too:

`-v /etc/ipa/ca.crt:/etc/ipa/ca.crt`

6. From the docker container created on the config node in Step "5" on page 388, 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 config 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 configuration node includes the complete Cassandra database for all configuration nodes in the cluster. You, therefore, only need to perform step 4 through 6 from one of the configuration nodes.

7. (Optional. Recommended) From the same config node, enter the `cat db-dump.json | python -m json.tool | less` command to view a more readable version of the file transfer.

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

8. From the same config 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 configuration 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 config node.

10. On each config 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 config 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 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 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 387.

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 397.

```
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 387.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_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_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_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 TSL 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 contrail_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 contrail_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 contrail_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 387.

```
## 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 TSL 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 15

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
```

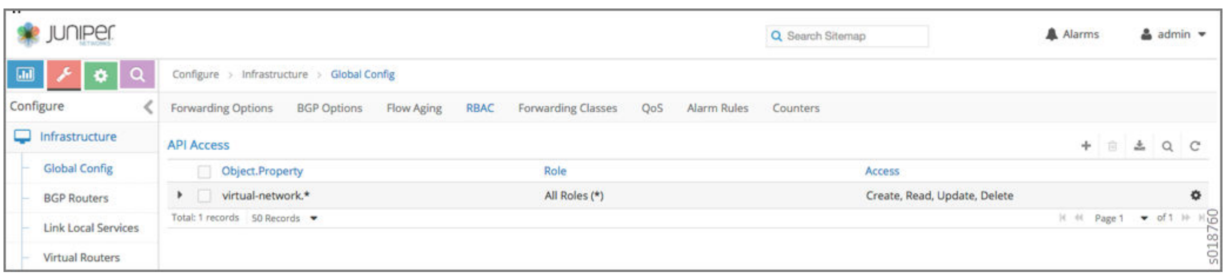
3. Restart services by restarting the container.

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 48 on page 415.

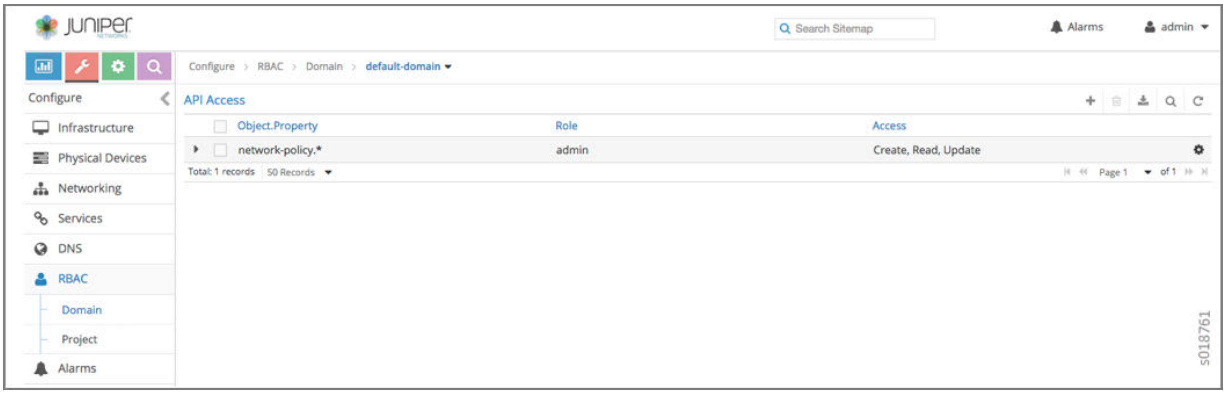
Figure 48: RBAC Global Level



Configuring RBAC at the Domain Level

To configure RBAC at the domain level, navigate to **Configure > RBAC > Domain**, see Figure 49 on page 416.

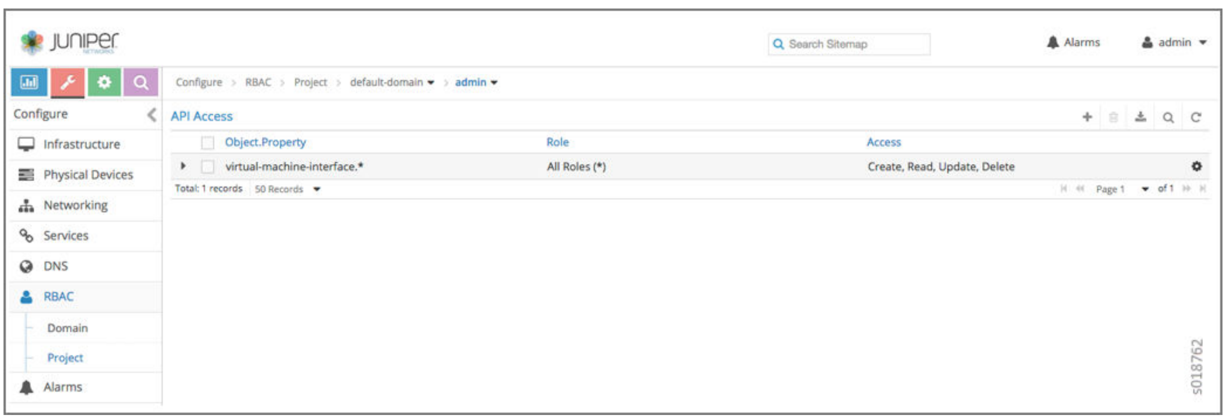
Figure 49: RBAC Domain Level



Configuring RBAC at the Project Level

To configure RBAC at the project level, navigate to **Configure > RBAC > Project**, see Figure 50 on page 416.

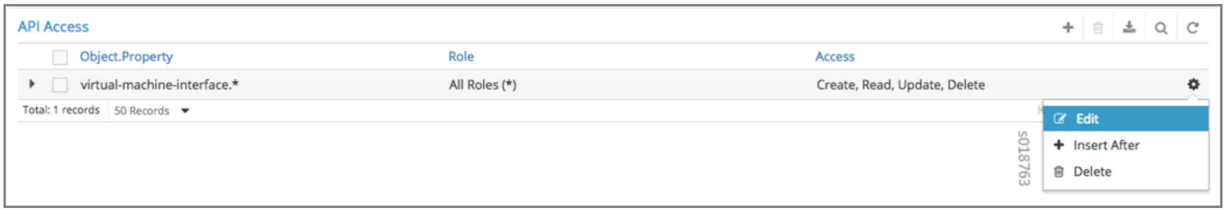
Figure 50: RBAC Project Level



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 51 on page 417.

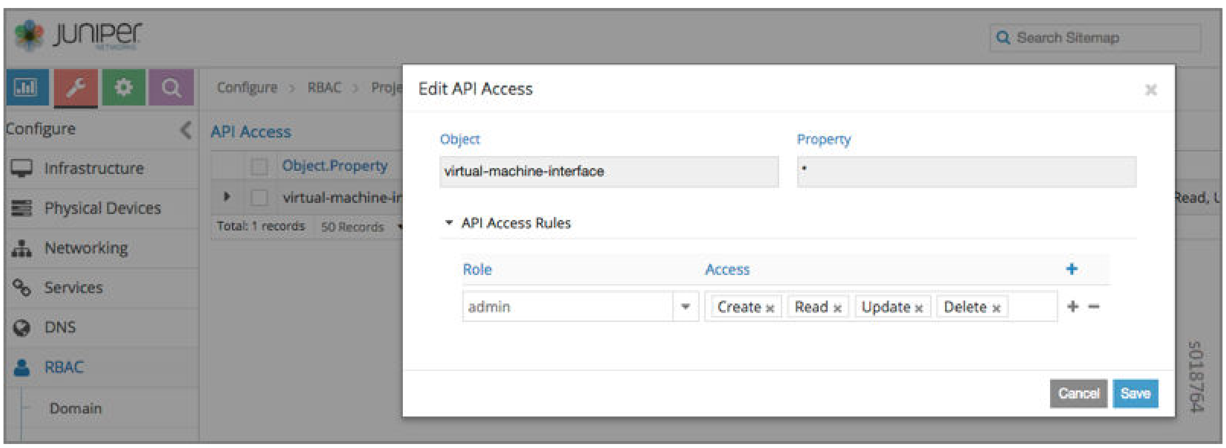
Figure 51: RBAC Details API Access



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 52 on page 417.

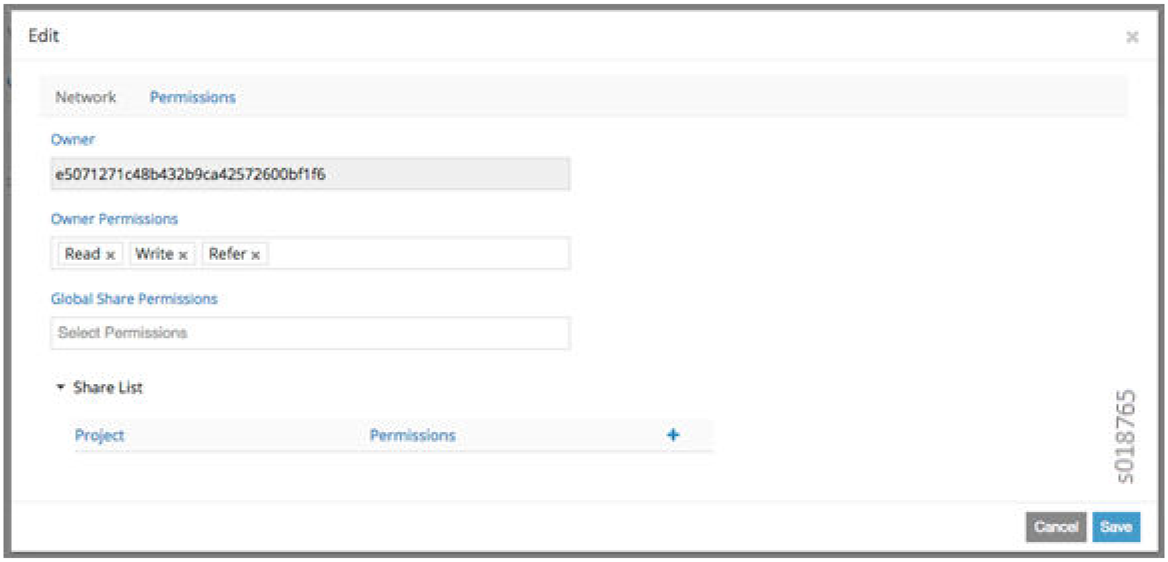
Figure 52: Edit API Access



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 53 on page 418.

Figure 53: 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.

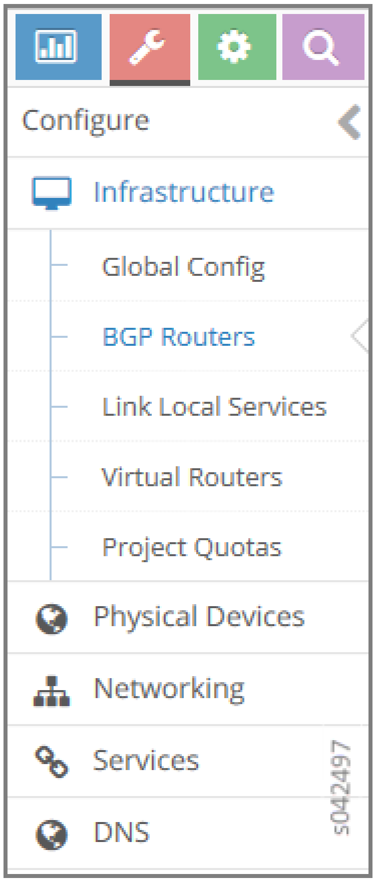
These topics provide instructions to configure the Control Node with BGP.

Configuring the Control Node from Contrail Web UI

To configure BGP peering in the control node:

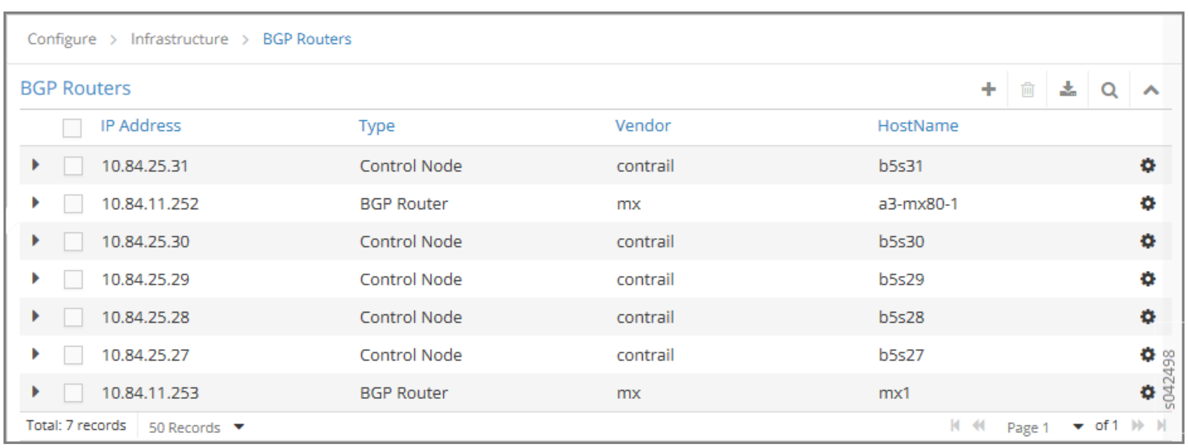
1. From the Contrail Controller module control node (**http://*nn.nn.nn.nn*:8143**), select **Configure > Infrastructure > BGP Routers**; see Figure 54 on page 421.

Figure 54: Configure> Infrastructure > BGP Routers



A summary screen of the control nodes and BGP routers is displayed; see Figure 55 on page 421.

Figure 55: BGP Routers Summary

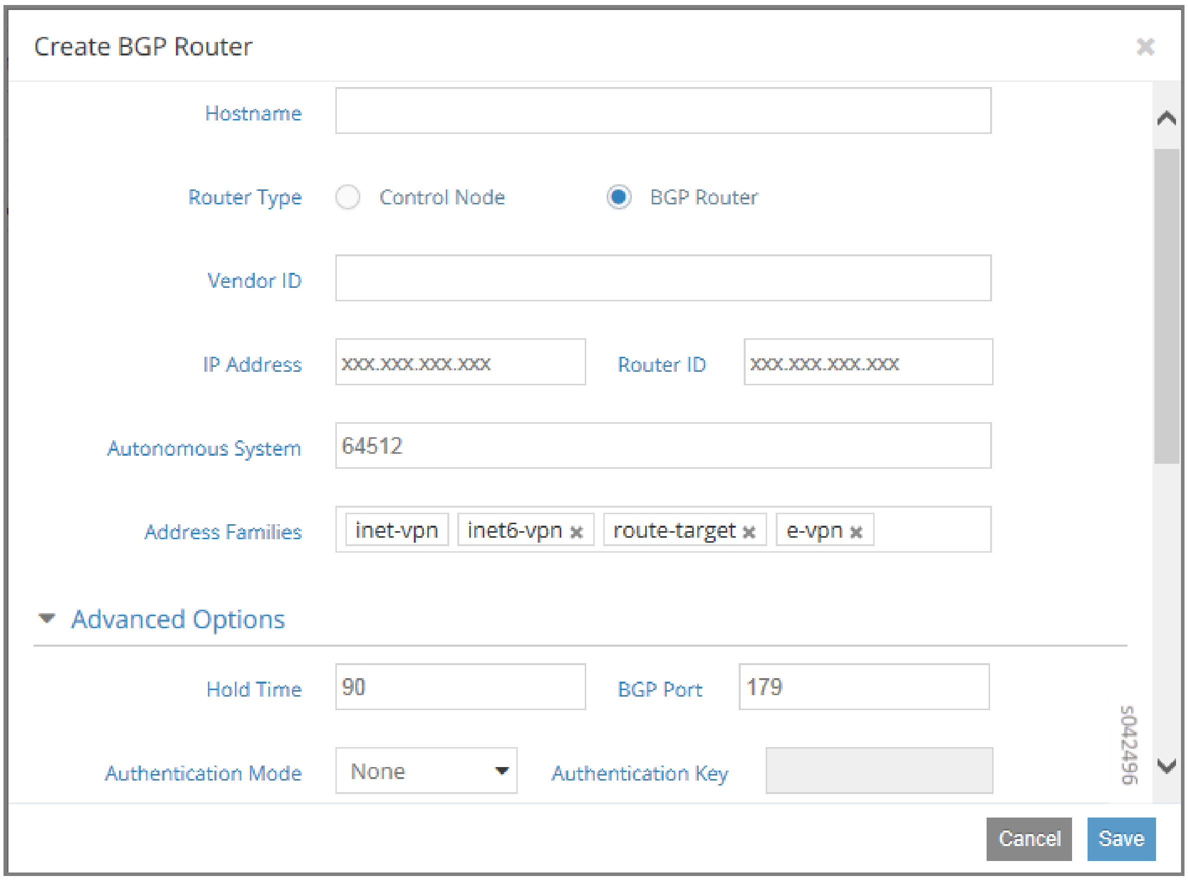


2. (Optional) The global AS number is 64512 by default. To change the AS number, on the **BGP Router** summary screen click the gear wheel and select **Edit**. In the Edit BGP Router window enter the new number.
3. To create control nodes and BGP routers, on the **BGP Routers** summary screen, click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

   icon. The **Create BGP Router** window is displayed; see Figure 56 on page 422.

Figure 56: 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 20 on page 423.

Table 20: Create BGP Router Fields

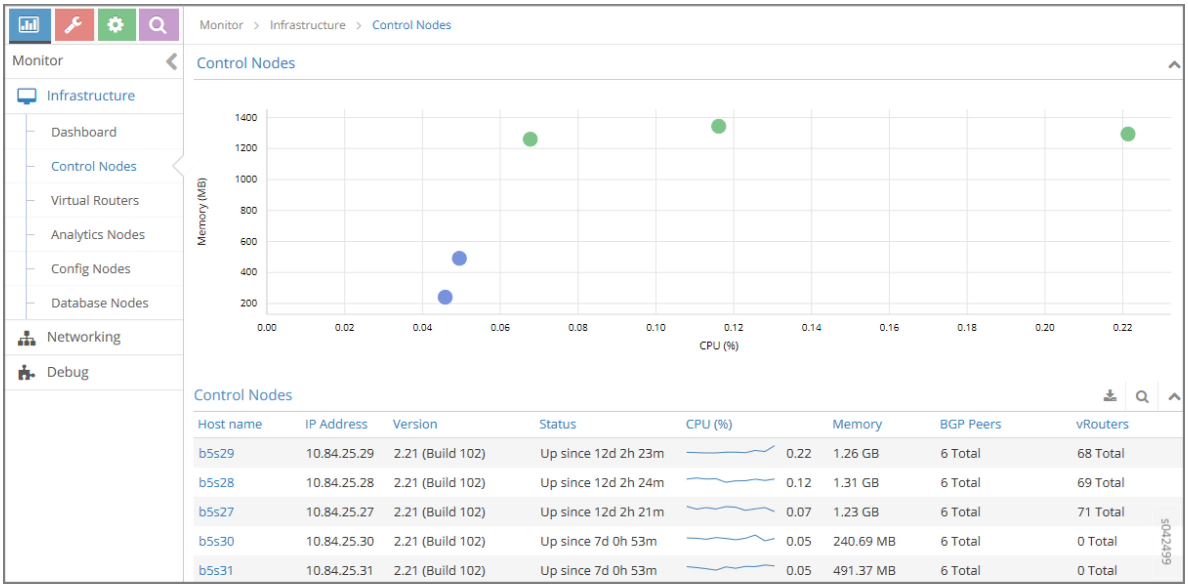
Field	Description
Hostname	Enter a name for the node being added.
Vendor ID	Required for external peers. Populate with a text identifier, for example, “MX-0”. (BGP peer only)
IP Address	The IP address of the node.
Router ID	Enter the router ID.
Autonomous System	Enter the AS number in the range 1-65535 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.

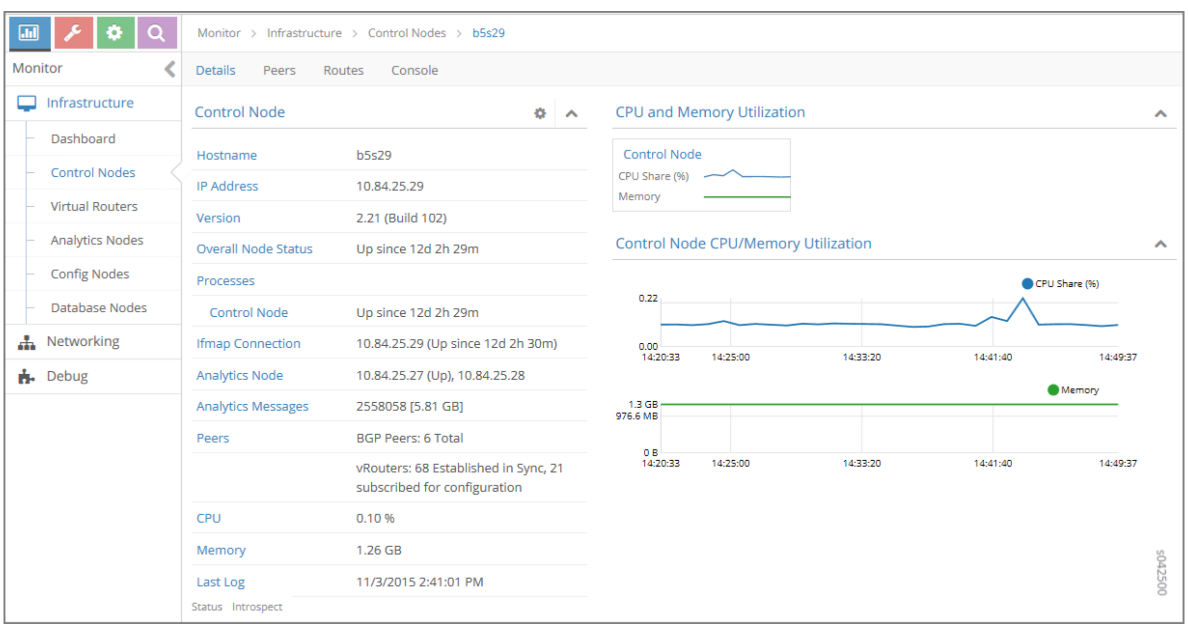
7. You can check for peers by selecting **Monitor > Infrastructure > Control Nodes**; see Figure 57 on page 424.

Figure 57: Control Nodes



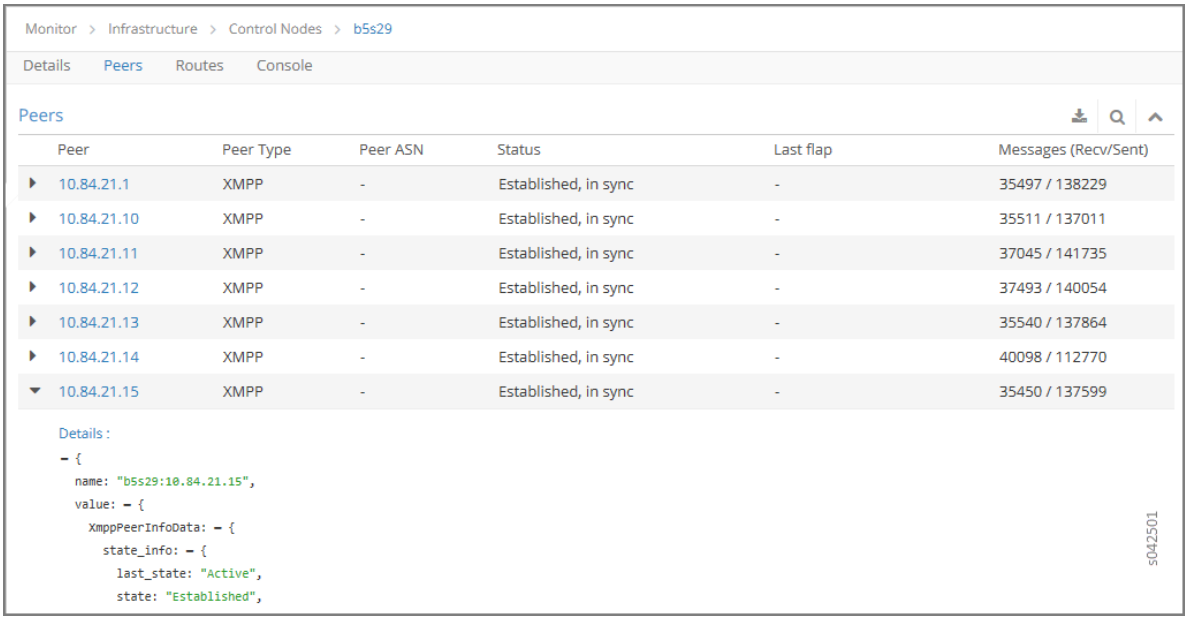
In the **Control Nodes** window, click any hostname in the memory map to view its details; see Figure 58 on page 424.

Figure 58: Control Node Details



8. Click the **Peers** tab to view the peers of a control node; see Figure 59 on page 425.

Figure 59: Control Node Peers Tab



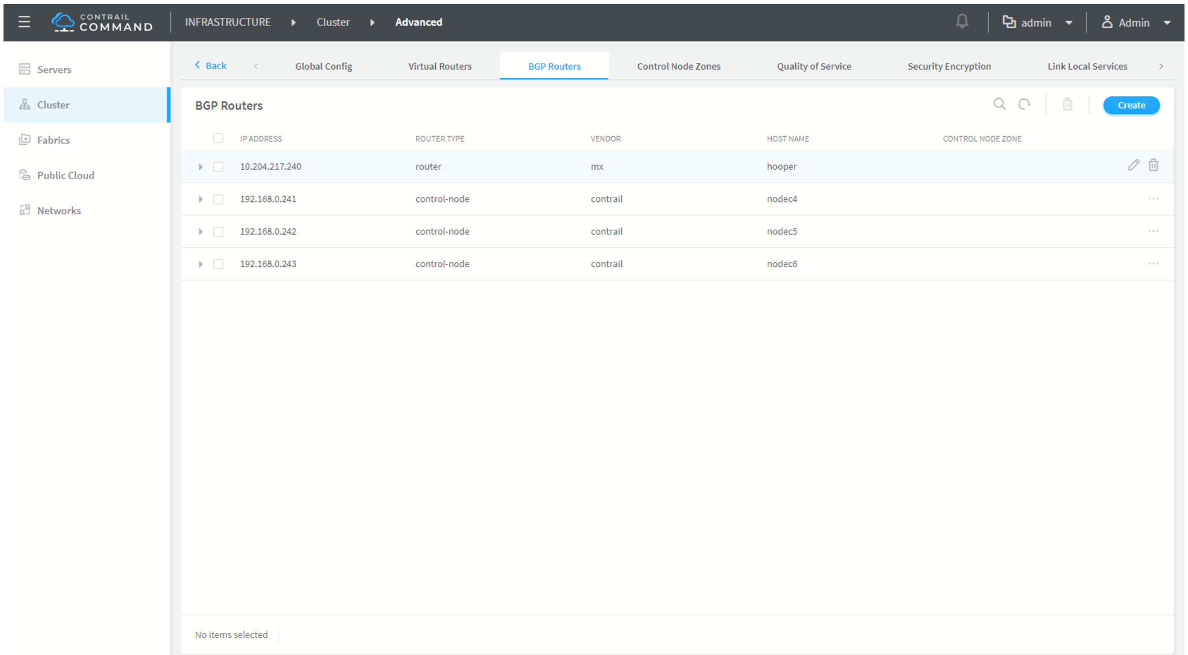
Configuring the Control Node with BGP from Contrail Command

To configure BGP peering in the control node:

1. From Contrail Command UI select **Infrastructure > Cluster > Advanced** page.

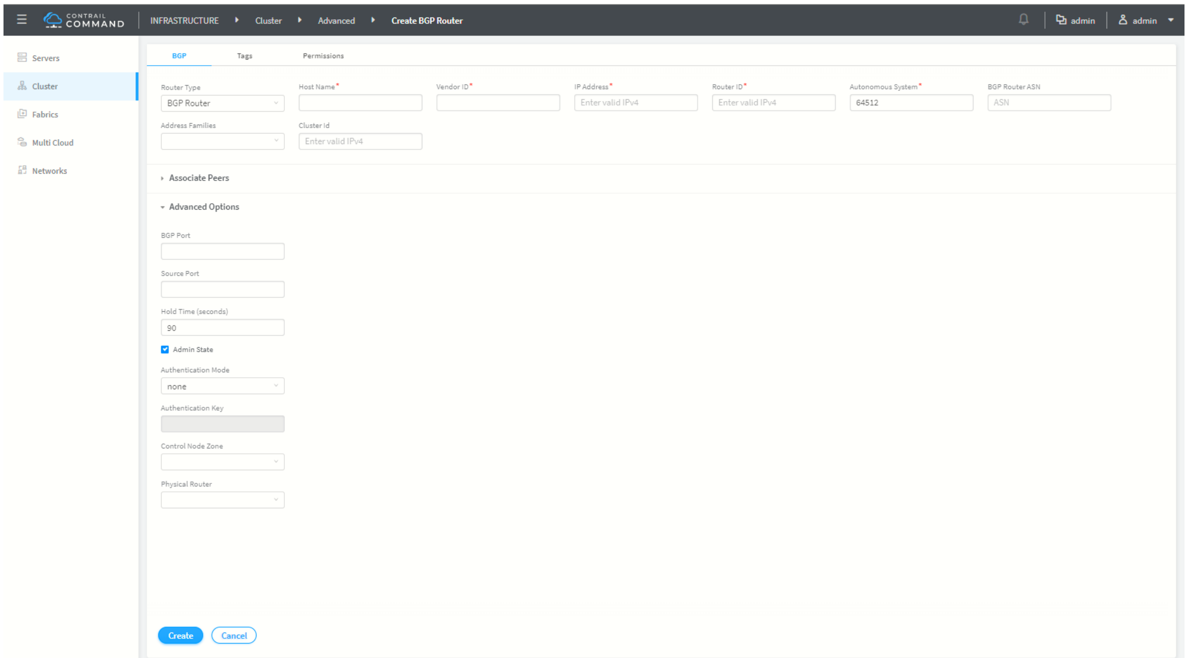
   Click the **BGP Routers** tab. A list of control nodes and BGP routers is displayed. See Figure 60 on page 426.

Figure 60: 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 61 on page 427.

Figure 61: Create BGP Router



4. In the **Create BGP Router** page, populate the fields with values to create your system. See Table 21 on page 427.

Table 21: 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)
IP Address	The IP address of the node.
Router ID	Enter the router ID.

Table 21: Create BGP Router *(Continued)*

Fields	Description
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.
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	

Table 21: Create BGP Router *(Continued)*

Fields	Description
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

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:

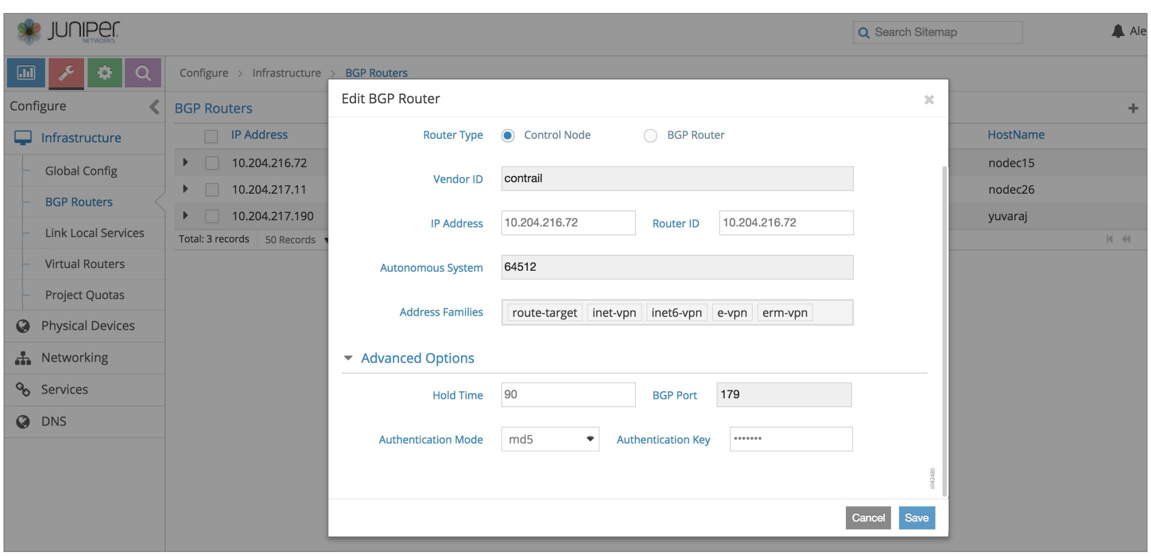
```
contrail-controller/src/config/utils/provision_control.py

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 62 on page 430, a list of BGP peers is displayed.

Figure 62: 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

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`
`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.

The graceful restart and long-lived graceful restart features can be enabled only for BGP peers in Contrail 3.2.

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 supports for long-lived graceful restart (LLGR) with XMPP helper mode. Graceful restart and long lived graceful restart can be enabled using 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.

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, which typically results in traffic loss for several seconds for new and /existing flows.
- 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 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 63 on page 439.

Figure 63: Configuring Graceful Restart