

Contrail™

Contrail Installation and Upgrade Guide

Published
2020-11-20

Release
5.1

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

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

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

Contrail™ Contrail Installation and Upgrade Guide

5.1

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

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

YEAR 2000 NOTICE

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

END USER LICENSE AGREEMENT

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

Table of Contents

About the Documentation | xii

Documentation and Release Notes | xii

Documentation Conventions | xiii

Documentation Feedback | xv

Requesting Technical Support | xvi

Self-Help Online Tools and Resources | xvi

Creating a Service Request with JTAC | xvii

1

Installing and Upgrading Contrail

Understanding Contrail | 3

Understanding Contrail Networking | 3

Understanding Contrail Networking Components | 5

Understanding Contrail Containers | 6

Contrail Containers | 7

Understanding Contrail Microservices Architecture | 7

What is Contrail Microservices Architecture? | 7

Installing Contrail with Microservices Architecture | 8

Understanding contrail-ansible-deployer used in Contrail Command | 8

What is the contrail-ansible-deployer? | 9

playbooks/provision_instances.yml | 9

playbooks/configure_instances.yml | 9

playbooks/install_contrail.yml | 10

Preparing to Install with Contrail Command | 10

Prerequisites | 10

Supported Providers | 10

Configure a Yaml File for Your Environment | 10

Provider Configuration | 11

Global Services Configuration | 14

Contrail Services Configuration | 14

Kolla Services Configuration | 15

Instances Configuration | 15

Installing a Contrail System | 16

Supported Platforms and Server Requirements | 17

Server Requirements and Supported Platforms | 17

Contrail Command | 18

Installing Contrail Command | 18

Installing Contrail Cluster using the Contrail Command UI | 26

Installing Contrail Cluster using Contrail Command and instances.yml | 39

Importing Contrail Cluster Data using Contrail Command | 44

Adding a New Compute Node to Existing Containerized Contrail Cluster Using Contrail Command | 49

Installing AppFormix using Contrail Command | 54

Enable AppFormix Plugins | 60

Enable LLDP and Analytics To Collect | 62

Deploying Contrail Command and Contrail All-In-One Cluster | 64

Installing Contrail | 68

Installing Contrail with OpenStack and Kolla Ansible | 68

Set Up the Base Host | 69

Multiple Interface Configuration Sample for Multinode OpenStack HA and Contrail | 72

Configuration Sample—Multiple Interface | 72

Single Interface Configuration Sample for Multinode OpenStack HA and Contrail | 74

Configuration Sample—Single Interface | 74

Frequently Asked Questions | 77

Using Host-Specific Parameters | 77

Containers from Private Registry Not Accessible | 77

Error: Failed to insert vrouter kernel module | 77

Fatal Error When Vrouter Doesn't Specify OpenStack | 78

Need for HAProxy and Virtual IP on a Single OpenStack Cluster | 79

Using the kolla_toolbox Container to Run OpenStack Commands | 79

Adding a New Compute Node to Existing Containerized Contrail Cluster | 81

Using Contrail with Kubernetes | 85

Installing and Managing Contrail Microservices Architecture Using Helm Charts | 85

- Understanding Helm Charts | 86

- Contrail Helm Deployer Charts | 86

- Contrail Kubernetes Resource implementation | 87

- Example: Contrail Pods Deployment Options | 87

- Installing Contrail Using Helm Charts | 88

Provisioning of Kubernetes Clusters | 89

- Provisioning of a Standalone Kubernetes Cluster | 89

- Provisioning of Nested Contrail Kubernetes Clusters | 90

 - Configure network connectivity to Contrail configuration and data plane functions. | 91

 - Generate a single yaml file to create a Contrail-k8s cluster | 92

 - Instantiate the Contrail-k8s cluster | 93

- Provisioning of Non-Nested Contrail Kubernetes Clusters | 94

Installing Standalone Kubernetes Contrail Cluster using the Contrail Command UI | 96

Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability | 105

- System Specifications | 105

- Preparing to Install | 106

- Installation of OpenStack Helm Charts | 109

- Installation of Contrail Helm Charts | 111

- Basic Testing OpenStack Helm Contrail Cluster | 116

- Accessing the Contrail OpenStack Helm Cluster | 116

Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata | 117

- System Specifications | 117

- Installation Steps | 118

- Accessing the Contrail OpenStack Helm Cluster | 120

Accessing a Contrail OpenStack Helm Cluster | 120

- Overview | 121

- Installing the OpenStack Client | 121

- Create openstackrc File and Test OpenStack Client | 121

- Accessing the Contrail Web UI | 122

- Accessing OpenStack Horizon | 122

- Accessing the Virtual Machine Console from Horizon | 123

OpenStack References | **123**

Frequently Asked Questions About Contrail and Helm Charts | **123**

How do I set up the vhost0 interface for the vrouter on the non-management interface of the compute node? | **124**

How do I configure the Contrail control BGP server to listen on a different port? | **125**

How can I pass additional parameters to services in Contrail by using the configuration file in INI format? | **125**

How do I configure services for the vrouter agent? | **125**

What are the Contrail services that can be configured? | **126**

How can I pass additional parameters to the Contrail Web UI services with configuration file in JS format? | **126**

How can I verify all pods of Contrail are up and running? | **127**

How can I see the logs of each of the containers? | **127**

How can I enter into a pod? | **127**

Installing Contrail Networking for Kubernetes using Helm | **128**

Verifying Configuration for CNI for Kubernetes | **134**

View Pod Name and IP Address | **134**

Verify Reachability of Pods | **135**

Verify If Isolated Namespace-Pods Are Not Reachable | **135**

Verify If Non-Isolated Namespace-Pods Are Reachable | **136**

Verify If a Namespace is Isolated | **137**

Using Contrail with Mesos | 138

Understanding Contrail with Mesos Architecture | **138**

Contrail with Mesos Architecture Diagram | **139**

Setup information | **139**

Components | **140**

Contrail Controller | **140**

Mesos Manager | **141**

Contrail Container Network Interface (CNI) | **142**

Installing Contrail with Mesos | **143**

Using VMware vCenter with Containerized Contrail | 146

Integrating vCenter for Contrail | 146

- Prerequisites | 146**
- ESX Agent Manager | 147**
- Set Up vCenter Server | 147**
- Configure Contrail Parameters | 152**
- Install Contrail | 152**
- Monitor and Manage ContrailVM from ESX Agent Manager | 152**

Configuring Underlay Network for ContrailVM | 155

- Standard Switch Setup | 155**
- Distributed Switch Setup | 157**
- PCI Pass-Through Setup | 159**
- SR-IOV Setup | 162**

Installing and Provisioning Contrail VMware vRealize Orchestrator Plugin | 166

- Accessing vRO Control Center | 167**
- Installing vRO Plugin | 170**
- Accessing vRO Desktop Client | 172**
- Connecting to vRO using the Desktop Client | 172**
- Connecting to Contrail Controller | 173**
- Deploying Contrail vRO Plugin | 176**

Using Contrail with Red Hat OpenStack | 177

Understanding Red Hat OpenStack Platform Director | 177

- Red Hat OpenStack Platform Director | 177**
- Contrail Roles | 178**
- Undercloud Requirements | 179**
- Overcloud Requirements | 179**
- Networking Requirements | 180**
- Compatibility Matrix | 181**
- Installation Summary | 181**

Setting Up the Infrastructure | 182

- Target Configuration (Example) | 182**
- Configure the External Physical Switch | 184**
- Configure KVM Hosts | 185**

Create the Overcloud VM Definitions on the Overcloud KVM Hosts | 187

Create the Undercloud VM Definition on the Undercloud KVM Host | 189

Setting Up the Undercloud | 191

Install the Undercloud | 191

Perform Post-Install Configuration | 193

Setting Up the Overcloud | 194

Configuring the Overcloud | 194

Customizing the Contrail Service with Templates (contrail-services.yaml) | 200

Customizing the Contrail Network with Templates | 201

Overview | 201

Roles Configuration (roles_data_contrail_aio.yaml) | 202

Network Parameter Configuration (contrail-net.yaml) | 205

Network Interface Configuration (*-NIC-*.yaml) | 206

Advanced vRouter Kernel Mode Configuration | 217

Advanced vRouter DPDK Mode Configuration | 219

Advanced vRouter SRIOV + Kernel Mode Configuration | 222

Advanced vRouter SRIOV + DPDK Mode Configuration | 225

Advanced Scenarios | 228

Installing Overcloud | 236

Using Contrail with Red Hat OpenShift | 238

Installing a Standalone Red Hat OpenShift Container Platform 3.11 Cluster with Contrail Using Contrail OpenShift Deployer | 238

Using Contrail and AppFormix with Kolla/Ocata OpenStack | 249

Contrail and AppFormix Deployment Requirements | 249

Software Requirements | 249

Hardware Requirements | 250

Preparing for the Installation | 250

Preparing the Targets | 250

Preparing the Base Host using Ansible Installer | 250

TCP/IP Port Conflicts Between Contrail and AppFormix | 251

Plugins to Enable for Contrail and AppFormix Deployment | 251

Configuring Contrail Monitoring in AppFormix | 251

Compute Monitoring: Listing IP Addresses to Monitor | 252

- Configuring Openstack_Controller Hosts for AppFormix | 252
- Other AppFormix group_vars That Must be Enabled in instances.yml | 252
- AppFormix License | 253

Run the Playbooks | 253

Accessing Contrail in AppFormix Management Infrastructure in UI | 254

Notes and Caveats | 255

Example Instances.yml for Contrail and AppFormix OpenStack Deployment | 255

Installing AppFormix for OpenStack | 259

- Architecture | 259
- Installing AppFormix | 260
- Removing a Node from AppFormix | 263

Installing AppFormix for OpenStack in HA | 264

- HA Design Overview | 264
- Requirements | 265
 - Hardware Requirements | 265
 - Software Requirements | 265
 - Connectivity | 265
 - AppFormix Agent Supported Platforms | 265
- Installing AppFormix for High Availability | 266

Using Contrail with Juju Charms | 269

Installing Contrail by Using Juju Charms | 269

- Preparing to Deploy Contrail by Using Juju Charms | 270
- Deploying Contrail Charms | 272
 - Deploying Contrail Charms in a Bundle | 272
 - Deploying Juju Charms Manually | 279
- Options for Juju Charms | 285

Upgrading Contrail Software | 292

Upgrading Contrail In-Service Software from Releases 3.2 and 4.1 to 5.0.x using Ansible Deployer | 292

- Contrail In-Service Software Upgrade (ISSU) Overview | 292
- Prerequisites | 293
- Preparing the Contrail System for the Ansible Deployer ISSU Procedure | 294
- Provisioning Control Nodes and Performing Synchronization Steps | 295

Transferring the Compute Nodes into the New Cluster | 298

Finalizing the Contrail Ansible Deployer ISSU Process | 301

Upgrading Contrail In-Service Software from Releases 3.2 and 4.1 to 5.0.x using Helm Deployer | 303

Contrail In-Service Software Upgrade (ISSU) Overview | 303

Prerequisites | 304

Preparing the Contrail System for the Helm Deployer ISSU Procedure | 304

Provisioning Control Nodes and Performing Synchronization Steps | 305

Transferring the Compute Nodes into the New Cluster | 308

Finalizing the Contrail Helm Deployer ISSU Process | 312

Upgrading Contrail Command using Backup Restore Procedure | 313

Backup and Restore Contrail Software | 315

Backing up Contrail Databases in JSON Format | 315

Preliminary Caution | 315

Simple Database Backup in JSON Format | 316

Restore Database from the Backup | 316

Example Backup and Restore in JSON | 318

Example: Perform Simple Database Backup in JSON Format | 318

Example: Restore Database from the Backup | 319

Post Installation Tasks | 324

Configuring Role and Resource-Based Access Control | 324

Contrail Role and Resource-Based Access (RBAC) Overview | 324

API-Level Access Control | 325

Rule Sets and ACL Objects | 326

Object Level Access Control | 326

Configuration | 327

Parameter: aaa-mode | 327

Parameter: cloud_admin_role | 327

Global Read-Only Role | 328

Parameter Changes in /etc/neutron/api-paste.ini | 329

Upgrading from Previous Releases | 329

Configuring RBAC Using the Contrail User Interface | 329

Configuring RBAC at the Global Level | 329

Configuring RBAC at the Domain Level | 330

Configuring RBAC at the Project Level | 330

Configuring RBAC Details | 331

RBAC Resources | 332

Configuring Role-Based Access Control for Analytics | 332

Configuring the Control Node with BGP | 333

Configuring the Control Node from Contrail Web UI | 335

Configuring the Control Node with BGP from Contrail Command | 341

Configuring MD5 Authentication for BGP Sessions | 345

Configuring Transport Layer Security-Based XMPP in Contrail | 347

Overview: TLS-Based XMPP | 347

TLS XMPP in Contrail | 347

Configuring XMPP Client and Server in Contrail | 347

Configuring Control Node for XMPP Server | 347

Configuring DNS Server for XMPP Server | 348

Configuring Control Node for XMPP Client | 348

Configuring Graceful Restart and Long-lived Graceful Restart | 349

Application of Graceful Restart and Long-lived Graceful Restart | 349

BGP Graceful Restart Helper Mode | 350

Feature Highlights | 350

XMPP Helper Mode | 350

Configuration Parameters | 351

Cautions for Graceful Restart | 352

Configuring Graceful Restart with the Contrail User Interface | 353

About the Documentation

IN THIS SECTION

- Documentation and Release Notes | xii
- Documentation Conventions | xiii
- Documentation Feedback | xv
- Requesting Technical Support | xvi

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

- Contrail Command.
- 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.

Documentation and Release Notes

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

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

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

Documentation Conventions

Table 1 on page xiii defines notice icons used in this guide.

Table 1: Notice Icons







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

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

Table 2: Text and Syntax Conventions

Convention	Description	Examples
Bold text like this	Represents text that you type.	To enter configuration mode, type the configure command: user@host> configure
Fixed-width text like this	Represents output that appears on the terminal screen.	user@host> show chassis alarms No alarms currently active

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

Convention	Description	Examples
<i>Italic text like this</i>	<ul style="list-style-type: none"> Introduces or emphasizes important new terms. Identifies guide names. Identifies RFC and Internet draft titles. 	<ul style="list-style-type: none"> A policy <i>term</i> is a named structure that defines match conditions and actions. <i>Junos OS CLI User Guide</i> RFC 1997, <i>BGP Communities Attribute</i>
<i>Italic text like this</i>	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name: <pre>[edit] root@# set system domain-name domain-name</pre>
Text like this	Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components.	<ul style="list-style-type: none"> To configure a stub area, include the stub statement at the [edit protocols ospf area area-id] hierarchy level. The console port is labeled CONSOLE.
< > (angle brackets)	Encloses optional keywords or variables.	stub <default-metric <i>metric</i> >;
(pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	broadcast multicast <i>(string1 string2 string3)</i>
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	rsvp { # Required for dynamic MPLS only
[] (square brackets)	Encloses a variable for which you can substitute one or more values.	community name members [community-ids]

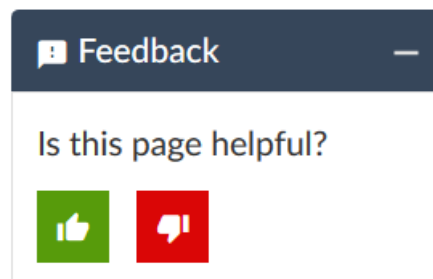
Table 2: Text and Syntax Conventions (*continued*)

Convention	Description	Examples
Indentation and braces ({ })	Identifies a level in the configuration hierarchy.	<pre>[edit] routing-options { static { route default { nexthop <i>address</i>; retain; } } }</pre>
; (semicolon)	Identifies a leaf statement at a configuration hierarchy level.	
GUI Conventions		
Bold text like this	Represents graphical user interface (GUI) items you click or select.	<ul style="list-style-type: none">• In the Logical Interfaces box, select All Interfaces.• To cancel the configuration, click Cancel.
> (bold right angle bracket)	Separates levels in a hierarchy of menu selections.	In the configuration editor hierarchy, select Protocols>Ospf .

Documentation Feedback

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

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



- Click the thumbs-up icon if the information on the page was helpful to you.
- Click the thumbs-down icon if the information on the page was not helpful to you or if you have suggestions for improvement, and use the pop-up form to provide feedback.

- E-mail—Send your comments to techpubs-comments@juniper.net. Include the document or topic name, URL or page number, and software version (if applicable).

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active Juniper Care or Partner Support Services support contract, or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

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

Self-Help Online Tools and Resources

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

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

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

Creating a Service Request with JTAC

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

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

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

1

PART

Installing and Upgrading Contrail

Understanding Contrail | **3**

Supported Platforms and Server Requirements | **17**

Contrail Command | **18**

Installing Contrail | **68**

Using Contrail with Kubernetes | **85**

Using Contrail with Mesos | **138**

Using VMware vCenter with Containerized Contrail | **146**

Using Contrail with Red Hat OpenStack | **177**

Using Contrail with Red Hat OpenShift | **238**

Using Contrail and AppFormix with Kolla/Ocata OpenStack | **249**

Using Contrail with Juju Charms | **269**

Upgrading Contrail Software | **292**

Backup and Restore Contrail Software | **315**

Post Installation Tasks | **324**

Understanding Contrail

IN THIS CHAPTER

- [Understanding Contrail Networking | 3](#)
- [Understanding Contrail Networking Components | 5](#)
- [Understanding Contrail Containers | 6](#)
- [Understanding Contrail Microservices Architecture | 7](#)
- [Understanding contrail-ansible-deployer used in Contrail Command | 8](#)

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.

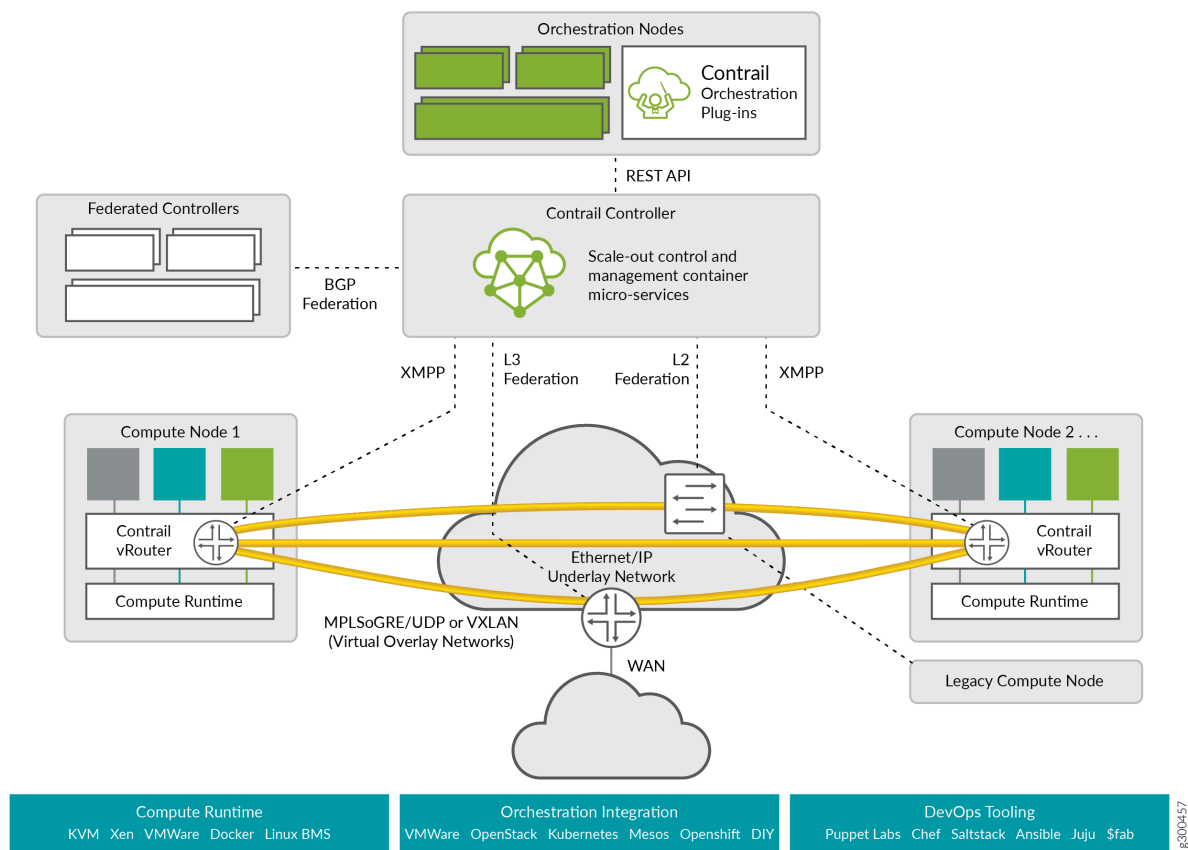
For service providers, Contrail Networking automates network resource provisioning and orchestration to dynamically create highly scalable virtual networks and to chain a rich set of Juniper Networks or third-party virtualized network functions (VNFs) and physical network functions (PNFs) to form differentiated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

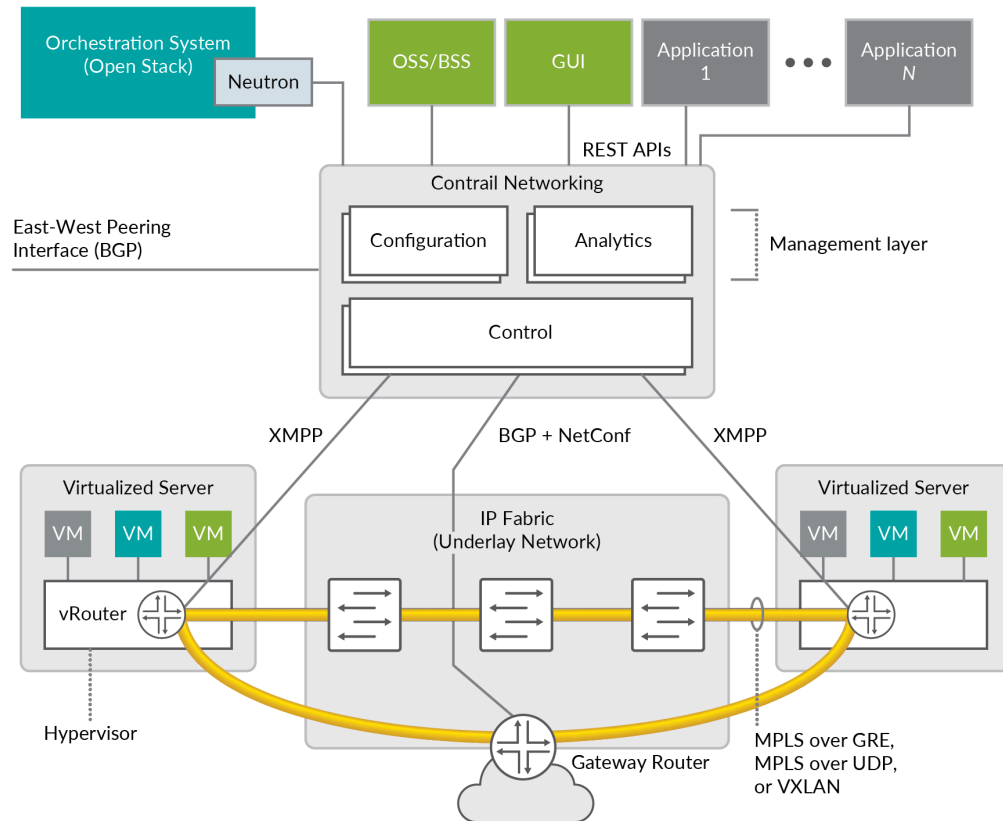
[Understanding Contrail Containers](#) | 6

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 | 3](#)

Understanding Contrail Containers

IN THIS SECTION

- [Contrail Containers | 7](#)

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? | 7](#)
- [Installing Contrail with Microservices Architecture | 8](#)

What is Contrail Microservices Architecture?

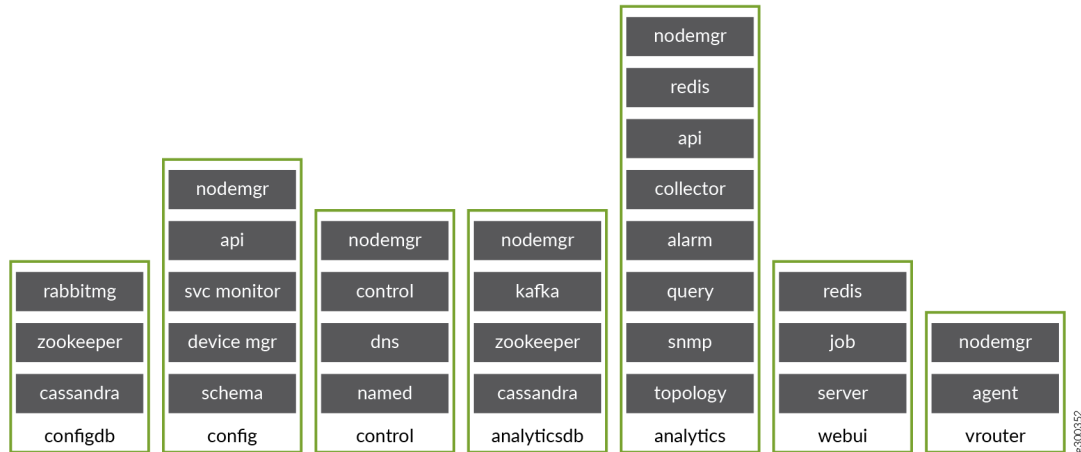
Employing microservices provides a number of benefits which includes:

- 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 8 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 8](#)
- [Installing and Managing Contrail Microservices Architecture Using Helm Charts on page 85](#)

Understanding contrail-ansible-deployer used in Contrail Command

IN THIS SECTION

- What is the contrail-ansible-deployer? | 9
- Preparing to Install with Contrail Command | 10
- Supported Providers | 10

- [Configure a Yaml File for Your Environment | 10](#)
- [Installing a Contrail System | 16](#)

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*. 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 39](#).

What is the contrail-ansible-deployer?

IN THIS SECTION

- [playbooks/provision_instances.yml | 9](#)
- [playbooks/configure_instances.yml | 9](#)
- [playbooks/install_contrail.yml | 10](#)

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

IN THIS SECTION

- [Provider Configuration | 11](#)
- [Global Services Configuration | 14](#)

- [Contrail Services Configuration | 14](#)
- [Kolla Services Configuration | 15](#)
- [Instances Configuration | 15](#)

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.

```
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.

```

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:
    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: nl-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*.

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 39.

RELATED DOCUMENTATION

| [Installing Contrail Cluster using Contrail Command and instances.yml](#) | 39

Supported Platforms and Server Requirements

IN THIS CHAPTER

- [Server Requirements and Supported Platforms](#) | 17

Server Requirements and Supported Platforms

The minimum requirement for a proof-of-concept (POC) system 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.

For a list of supported platforms, see [Supported Platforms Contrail 5.1](#).

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.

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.

Access *Container Tags* located at [README Access to Contrail Registry](#).

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

Contrail Command

IN THIS CHAPTER

- Installing Contrail Command | 18
- Installing Contrail Cluster using the Contrail Command UI | 26
- Installing Contrail Cluster using Contrail Command and instances.yml | 39
- Importing Contrail Cluster Data using Contrail Command | 44
- Adding a New Compute Node to Existing Containerized Contrail Cluster Using Contrail Command | 49
- Installing AppFormix using Contrail Command | 54
- Deploying Contrail Command and Contrail All-In-One Cluster | 64

Installing Contrail Command

Contrail Networking supports Contrail Command user interface (UI). Contrail Command is an intuitive, wizard-based UI which provides automated work flows such as the following:

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

Requirements

The system requirements to install the Contrail Command server are:

- A VM or physical server with:
 - 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 [Supported Platforms Contrail 5.1](#).

NOTE: Email contrail-registry@juniper.net for Contrail container registry credentials.

Configuration

Prerequisite

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
```

Step-by-Step Procedure

Perform the following steps to configure and install Contrail Command.

1. Install Docker to pull *contrail-command-deployer* container. This package is required 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-18.06.0.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.

Access *container_tag* for *contrail-command-deployer* located at [README Access to Contrail 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. Create the input configuration *command_servers.yml* file.

Use the [Minimal command_servers.yml file on page 20](#) to create the minimal input configuration file. For an exhaustive list of supported parameters, use [Complete command_servers.yml File on page 21](#).

4. Start the contrail-command-deployer container to deploy the Contrail Command UI.

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

<ABSOLUTE_PATH_TO_COMMAND_SERVERS_FILE>—absolute path to the **command_servers.yml** file that you created in step 3.

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

```
docker run -td --net host -v /root/command_servers.yml:/command_servers.yml --privileged --name
contrail_command_deployer hub.juniper.net/contrail/contrail-command-deployer:5.1.0-0.38
```

The **contrail_command** and **contrail_psql** containers are deployed.

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

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

NOTE: Code changes that involve schema modifications require updating the database container as well. Step 5 is recommended only if the UI application requires an update.

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

```
docker logs -f contrail_command_deployer
```

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

Sample command_servers.yml Files

Minimal command_servers.yml file

The following sample file has minimum configurations that you need when you install Contrail Command. You can also use this file for releases prior to Release 5.1.

```

---
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>

    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 need when you install Contrail Command.

```

---
# 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 Contrail Cluster using the Contrail Command UI | 26](#)

[Installing Contrail Cluster using Contrail Command and instances.yml | 39](#)

[Importing Contrail Cluster Data using Contrail Command | 44](#)

Installing Contrail Cluster using the Contrail Command UI

IN THIS SECTION

- [Requirements | 26](#)
- [Overview | 27](#)
- [Configuration | 28](#)

This topic describes the installation workflow of Contrail Cluster with Openstack orchestration using Contrail UI.

Requirements

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

For a list of supported platforms, see [Supported Platforms Contrail 5.1](#).

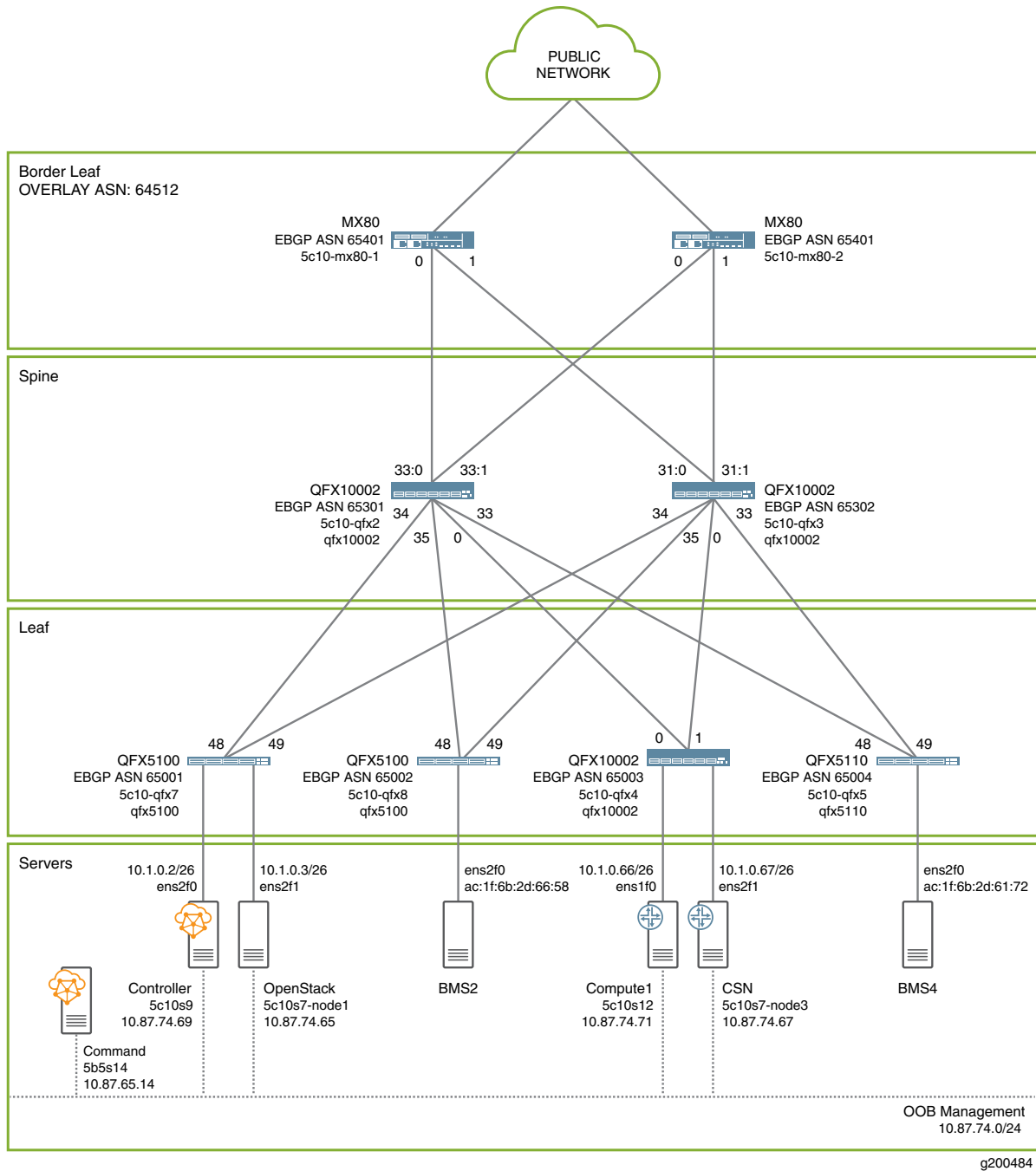
Overview

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

Topology

Consider a sample cluster topology, with a non-HA environment of one Contrail Controller and one OpenStack Controller, one compute node and one CSN, as displayed in [Figure 4 on page 28](#).

Figure 4: Sample Contrail Cluster Topology



Configuration

Deploying a Contrail Cluster

Step-by-Step Procedure

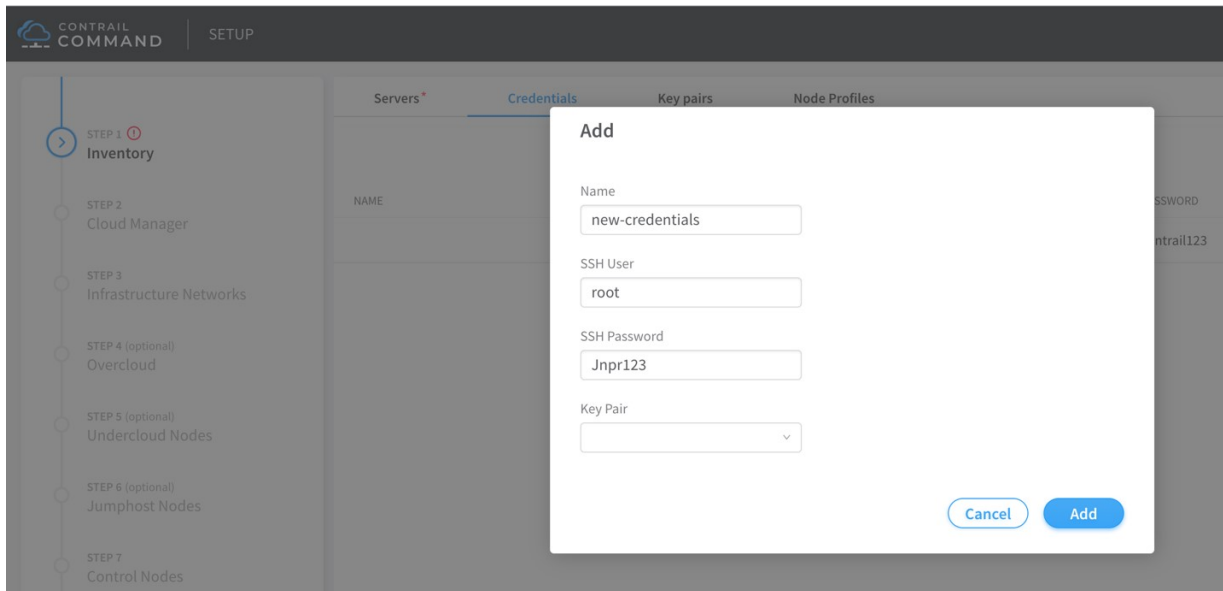
To deploy a Contrail Cluster using Contrail Command, perform the following steps.

1. Login to Contrail Command UI.

The default username is **root** and the default password is **contrail123**.

2. The default username for server is **root** and the default password is **cOntrail123**.

You can create new server credentials



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

- One by one
- CSV file bulk import
- **One by one:** You can add each server one by one using the default credentials or any desired credentials from the drop down list. To add servers one by one, you must enter the following mandatory parameters:
 - Hostname
 - Management IP address
 - Management Interface
- **CSV file bulk import:** You can upload a CSV file which contains the details of each server. This CSV file must conform to the Contrail Command format.

You can download the CSV template from Contrail Command and reuse the template as per your requirements.

To download a sample CSV file, navigate to **Infrastructure > Servers > Add Servers** in the Contrail Command UI.

Use the Bulk Import option for large deployments. Bulk Import option requires a CSV file input.

Click **Bulk Import (csv)**, to download the template.

A sample CSV file is shown here:

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

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

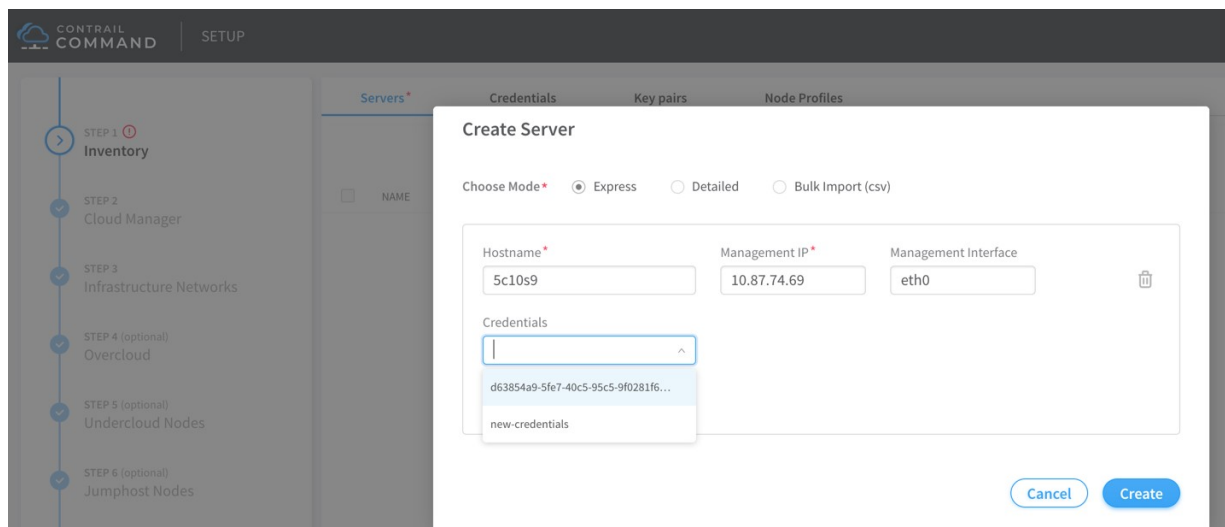
physical,5c10s7-node1,10.87.74.65,,eno1,,,,,,,,,2,eno1,,10.87.74.65,,ens2f1,,10.1.0.3,

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

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

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

Figure 5: Add Server



4. Create a cluster.

Click **Provisioning Manager > Create Cluster**

If **Container registry** = hub.juniper.net/contrail . This registry is secure. Unselect the **Insecure** box.

Also, **Contrail version** = *contrail_container_tag* for your release of Contrail as listed in [README Access for Contrail](#).

Default vRouter Gateway = Default gateway for the compute nodes. If any one of the compute nodes has a different default gateway than the one provided here, enter that gateway in step 7 and step 8 for service nodes.

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

VXLAN, MPLSoUDP, MPLSoGRE

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

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

Figure 6: Create Cluster

STEP 1
Inventory

STEP 2
Cloud Manager

STEP 3
Infrastructure Networks

STEP 4 (optional)
Overcloud

STEP 5 (optional)
Undercloud Nodes

STEP 6 (optional)
Jumphost Nodes

STEP 7
Control Nodes

STEP 8
Orchestrator Nodes

STEP 9 (optional)
Compute Nodes

STEP 10 (optional)
Contrail Service Nodes

STEP 11 (optional)
Appformix Nodes

STEP 12
Summary

Choose Provisioning Manager*

☐ RHOSP Manager

☒ Contrail Cloud Manager

Cluster Name*

contrail-cluster

Container Registry*

hub.juniper.net/contrail

☐ Insecure

Container Registry Username*

username

Container Registry Password*

password

Contrail Version *

5.1.0-0.38

Provisioner Type

Ansible

Domain Suffix

local

NTP Server

10.84.5.100

Default Vrouter Gateway

10.1.0.254

Encapsulation Priority

VXLAN,MPLSoUDP,MPL...

☐ Enable ZTP ⓘ

Contrail Configuration

Key

Value

CONTROLLER_NODES

10.87.74.69

Key

Value

CONTROL_NODES

10.1.0.2

Key

Value

TSN_NODES

10.1.0.67

Key

Value

CONTRAIL_CONTAINER_TAG

5.1.0-0.38-queens

Previous

Next

5. Select the contrail-control node.

Figure 7: Select Control Nodes

STEP 1
Inventory

STEP 2
Cloud Manager

STEP 3 (optional)
Infrastructure Networks

STEP 4 (optional)
Overcloud

STEP 5 (optional)
Undercloud Nodes

STEP 6 (optional)
Jumphost Nodes

STEP 7
Control Nodes

☐ High availability mode

Available servers

Search servers

Add all

HOSTNAME	IP ADDRESS	DISK PARTITION
Sc10s7-node1	10.87.74.65	>
Sc10s7-node3	10.87.74.67	>
Sc10s12	10.87.74.71	>

Assigned Control nodes

Search servers

Remove all

HOSTNAME	IP ADDRESS	DISK PARTITION
Sc10s9	10.87.74.69	>

Roles*

contrail_config_node x

contrail_config_database_node x

contrail_analytics_node x

contrail_analytics_alarm_node x

contrail_analytics_snmp_node x

contrail_analytics_database_node x

contrail_control_node x

contrail_webui_node x

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

6. Select the orchestration type and nodes.

The supported orchestrators are OpenStack and Kubernetes.

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

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

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

Add the following under custom configuration for VM based setup:

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

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

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

Set the following in Kolla Globals:

```
enable_ironic = no
enable_swift = yes
swift_disk_partition_size = 10GB
```

NOTE: The default value of *swift_disk_partition_size* is 5 GB. If you have 2 or more images, you must have at least 10 GB allocated to *swift_disk_partition_size* for hitless image upload procedure.

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

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

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

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

Figure 8: Select Orchestrator Nodes

CONTRAIL COMMAND | SETUP

STEP 1 Inventory

STEP 2 Cloud Manager

STEP 3 (optional) Infrastructure Networks

STEP 4 (optional) Overcloud

STEP 5 (optional) Undercloud Nodes

STEP 6 (optional) Jumphost Nodes

STEP 7 Control Nodes

STEP 8 Orchestrator Nodes

STEP 9 (optional) Compute Nodes

STEP 10 (optional) Contrail Service Nodes

STEP 11 (optional) Appformix Nodes

STEP 12 Summary

Orchestrator type ^{*}

Openstack ☐ Show Advanced

Openstack Registry

default

Openstack Release

queens

Control & Data Network Virtual IP address

Enter valid IPv4

Management Network Virtual IP address

Enter valid IPv4

Customize configuration ⓘ

Place customized configuration...

Kolla Globals

Key	Value
enable_ironic	yes
enable_swift	yes

+ Add

Kolla Passwords

Key	Value
keystone_admin_password	Jnpr123

Previous Next

Select the Openstack (orchestration) node.

CONTRAIL COMMAND | SETUP

STEP 1 Inventory

STEP 2 Cloud Manager

STEP 3 (optional) Infrastructure Networks

STEP 4 (optional) Overcloud

STEP 5 (optional) Undercloud Nodes

STEP 6 (optional) Jumphost Nodes

STEP 7 Control Nodes

STEP 8 Orchestrator Nodes

STEP 9 (optional) Compute Nodes

STEP 10 (optional) Contrail Service Nodes

STEP 11 (optional) Appformix Nodes

STEP 12 Summary

Orchestrator type ^{*}

Openstack ☐ Show Advanced

Openstack Registry

default

Openstack Release

queens

Control & Data Network Virtual IP address

Enter valid IPv4

Management Network Virtual IP address

Enter valid IPv4

Customize configuration ⓘ

Place customized configuration...

Kolla Globals

Key	Value
enable_ironic	yes
enable_swift	yes

+ Add

Kolla Passwords

Key	Value
keystone_admin_password	Jnpr123

+ Add

Available servers

HOSTNAME	IP ADDRESS	DISK PARTITION
5c10s9	10.87.74.69	>
5c10s7-node3	10.87.74.67	>
5c10s12	10.87.74.71	>

Search servers Add all

Assigned Openstack nodes

HOSTNAME	IP ADDRESS	DISK PARTITION
5c10s7-node1	10.87.74.65	>

Search servers Remove all

Roles*

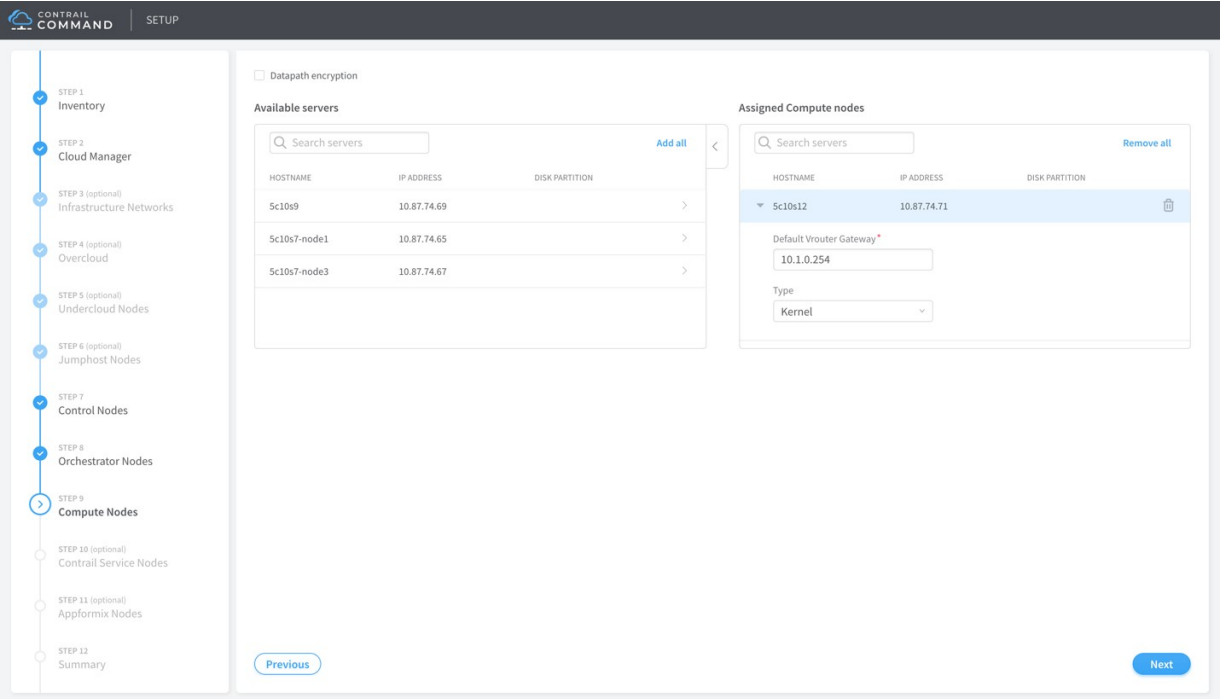
openstack_control_node openstack_network_node
openstack_storage_node openstack_monitoring_node

Previous Next

7. Select the Compute Nodes. For each compute node, enter the gateway, if it is different from what was added in the global parameters in step 4. Then set the mode.

As of now, only *Kernel* mode is supported

Figure 9: Select Compute Nodes



8. Select the Contrail Service Node.

Figure 10: Select Contrail Service Nodes

CONTRAIL COMMAND | SETUP

STEP 1 Inventory

STEP 2 Cloud Manager

STEP 3 (optional) Infrastructure Networks

STEP 4 (optional) Overcloud

STEP 5 (optional) Undercloud Nodes

STEP 6 (optional) Jumphost Nodes

STEP 7 Control Nodes

STEP 8 Orchestrator Nodes

STEP 9 Compute Nodes

STEP 10 (optional) **Contrail Service Nodes**

STEP 11 (optional) Appformix Nodes

STEP 12 Summary

Available servers

Search servers [Add all](#)

HOSTNAME	IP ADDRESS	DISK PARTITION
5c10s9	10.87.74.69	>
5c10s7-node1	10.87.74.65	>
5c10s12	10.87.74.71	>

Assigned Service nodes

Search servers [Remove all](#)

HOSTNAME	IP ADDRESS	DISK PARTITION
5c10s7-node3	10.87.74.67	>

Default Vrouter Gateway*

10.1.0.254

[Previous](#) [Next](#)

9. (Optional) You can add a new server to install AppFormix.

For details, refer to [“Installing AppFormix using Contrail Command”](#) on page 54.

CONTRAIL
COMMAND

SETUP

STEP 1
Inventory

STEP 2
Cloud Manager

STEP 3 (optional)
Infrastructure Networks

STEP 4 (optional)
Overcloud

STEP 5 (optional)
Undercloud Nodes

STEP 6 (optional)
Jumphost Nodes

STEP 7
Control Nodes

STEP 8
Orchestrator Nodes

STEP 9
Compute Nodes

STEP 10 (optional)
Contrail Service Nodes

STEP 11 (optional)
Appformix Nodes

STEP 12
Summary

☐ Show Advanced

Available servers

Search servers

Add all

HOSTNAME	IP ADDRESS	DISK PARTITION
5c10s9	10.87.74.69	>
5c10s7-node1	10.87.74.65	>
5c10s7-node3	10.87.74.67	>
5c10s12	10.87.74.71	>

Assigned Appformix Nodes

Search servers

Remove all

HOSTNAME	IP ADDRESS	DISK PARTITION
Assign one or more servers		

Previous

Next

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

Figure 11: Verify Summary

CONTRAIL COMMAND | SETUP

STEP 12 Summary

Cluster overview

Display name	contrail-cluster
Container registry	hub.juniper.net/contrail
Container registry username	username
Container registry password	password
Contrail version	5.1.0-0.38
Provisioner type	ansible
Domain Suffix	local
NTP server	10.84.5.100
Default Vrouter Gateway	10.1.0.254
Encapsulation priority	VXLAN,MPLSoUDP,MPLSoGRE
Enable ZTP	false
Contrail configuration	
CONTROLLER_NODES	10.87.74.69
CONTROL_NODES	10.1.0.2
TSN_NODES	10.1.0.67
CONTRAIL_CONTAINER_TAG	5.1.0-0.38-queens
High availability mode	false
Orchestrator	openstack
Openstack release	queens
Openstack internal virtual IP	-
Openstack external virtual IP	-
Openstack registry	default
Kolla globals	
enable_ironic	yes
enable_swift	yes
Kolla passwords	
keystone_admin_password	Jnpr123

[Previous](#) [Provision](#)

CONTRAIL COMMAND | SETUP

STEP 12 Summary

Container registry username	username
Container registry password	password
Contrail version	5.1.0-0.38
Provisioner type	ansible
Domain Suffix	local
NTP server	10.84.5.100
Default Vrouter Gateway	10.1.0.254
Encapsulation priority	VXLAN,MPLSoUDP,MPLSoGRE
Enable ZTP	false
Contrail configuration	
High availability mode	false
Orchestrator	openstack
Openstack release	queens
Openstack internal virtual IP	-
Openstack external virtual IP	-
Openstack registry	default
Kolla globals	
Kolla passwords	

Nodes overview

All cluster nodes		Control nodes	Compute nodes	Openstack nodes	Service nodes
NAME	TYPE	IP ADDRESS	NODE PROFILE	ROLES	
Sc10s12	physical/virtual node	10.87.74.71		Compute node	
Sc10s7-node1	physical/virtual node	10.87.74.65		Openstack node	
Sc10s7-node3	physical/virtual node	10.87.74.67		Service node	
Sc10s9	physical/virtual node	10.87.74.69		Control node	

[Previous](#) [Provision](#)

RELATED DOCUMENTATION

[Installing Contrail Command | 18](#)

[Installing Contrail Cluster using Contrail Command and instances.yml | 39](#)

[Importing Contrail Cluster Data using Contrail Command | 44](#)

Installing Contrail Cluster using Contrail Command and instances.yml

Contrail Networking supports deploying Contrail cluster using Contrail Command and the **instances.yml** file.

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 [Supported Platforms Contrail 5.1](#).

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](https://hub.docker.com/r/juniper/contrail-command-deployer). To download these containers and for access to [hub.juniper.net](https://hub.docker.com/r/juniper/contrail-command-deployer), 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 [Sample instances.yml File on page 41](#) for a sample **instances.yml** file.
4. Start the *contrail_command_deployer* container to deploy the Contrail Command (UI) server and provision Contrail Cluster using the **instances.yml** file provided.

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

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

5. (Optional) Track the progress of 4.

docker logs -f contrail_command_deployer

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

NOTE: 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 Contrail Cluster using the Contrail Command UI | 26](#)

[Importing Contrail Cluster Data using Contrail Command | 44](#)

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.

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 [Supported Platforms Contrail 5.1](#).

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 import Contrail Cluster data.

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

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

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

yum install -y docker-ce-18.03.1.ce

systemctl start docker
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- For VMware vCenter, use **vcenter**.
- For Mesos, use **mesos**.
- Import Contrail-Cluster provisioned using OSPDirector/TripleO Life Cycle Manager for RedHat OpenStack Orchestration.

Prerequisites:

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

```
docker run -td --net host -e orchestrator=tripleo -e action=import_cluster -e  
undercloud=<IP_ADDRESS_OF_UNDERCLOUD_NODE> -e  
undercloud_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 Contrail Cluster using the Contrail Command UI | 26](#)

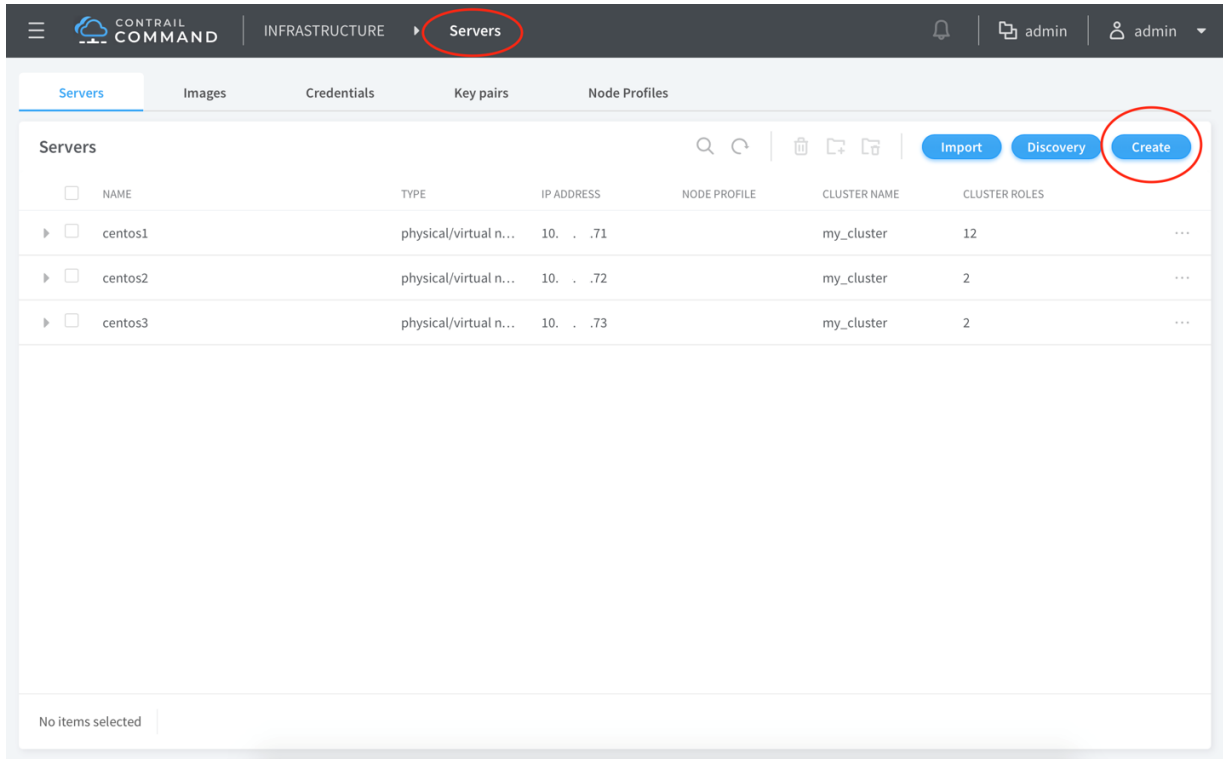
[Installing Contrail Cluster using Contrail Command and instances.yml | 39](#)

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

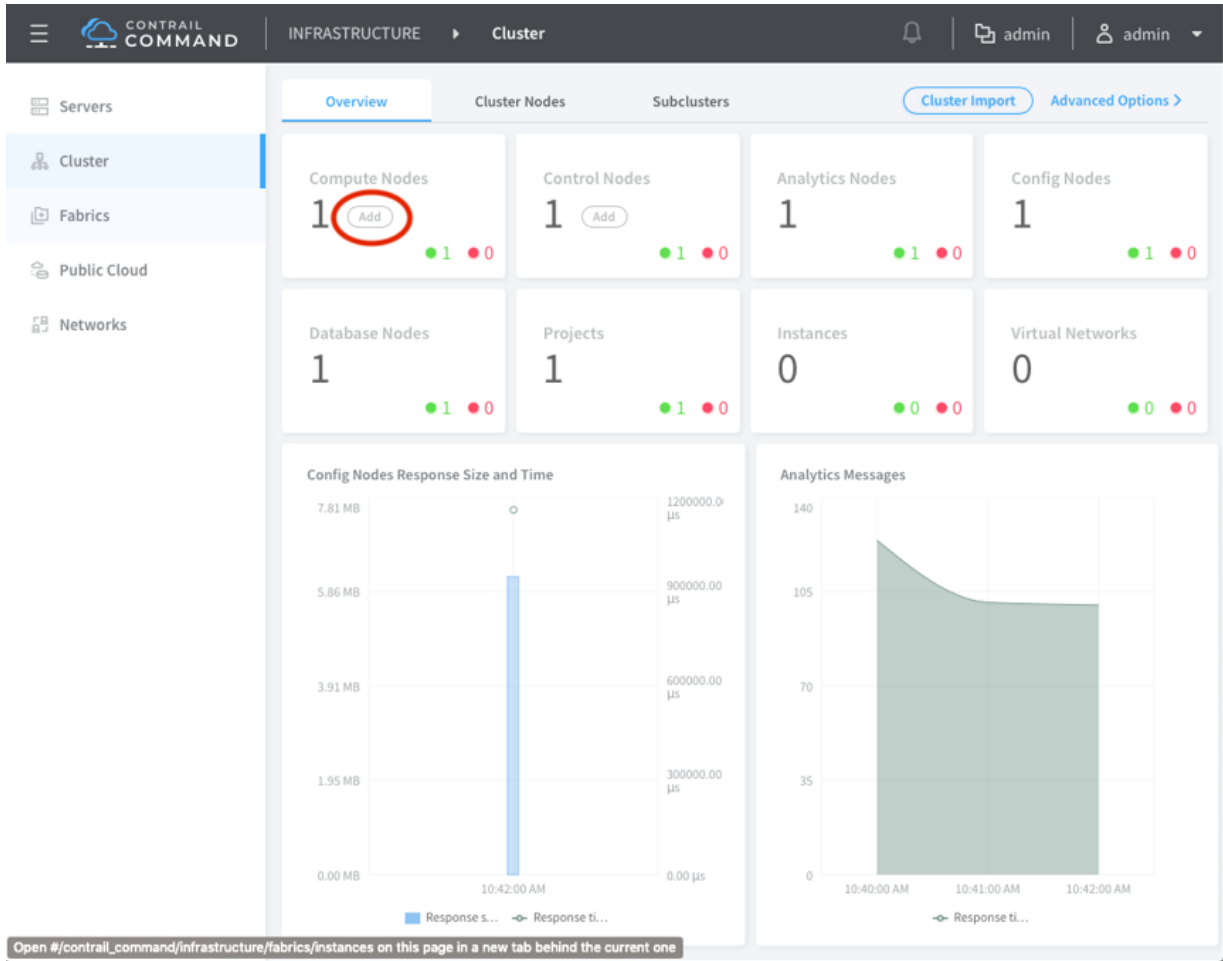
You can add or remove a new node from an existing containerized Contrail cluster.

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

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

CONTRAIL COMMAND | INFRASTRUCTURE | Cluster

Overview | Cluster Nodes | Subclusters | Cluster Import | Advanced Options >

Assign Compute Nodes

Available servers

Search servers [Add all](#) <

HOSTNAME	IP ADDRESS	DISK PARTITION
centos3	10. . .73	>

Assigned Compute nodes

Search servers Remove all

Assign one or more servers

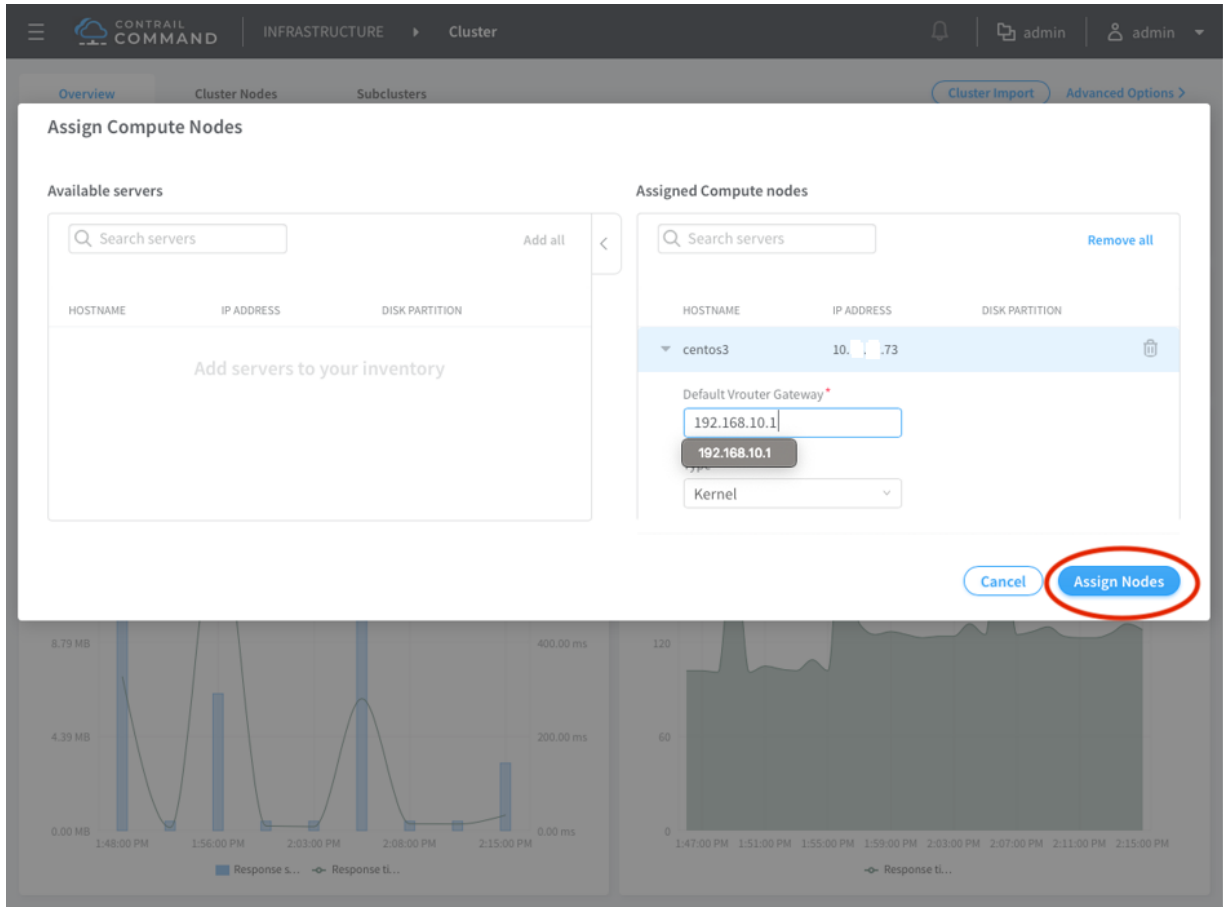
[Cancel](#) [Assign Nodes](#)

8.79 MB 4.39 MB 0.00 MB 1:48:00 PM 1:56:00 PM 2:03:00 PM 2:08:00 PM 2:15:00 PM Response s... Response ti...

400.00 ms 200.00 ms 0.00 ms

120 60 0 1:47:00 PM 1:51:00 PM 1:55:00 PM 1:59:00 PM 2:03:00 PM 2:07:00 PM 2:11:00 PM 2:15:00 PM Response ti...

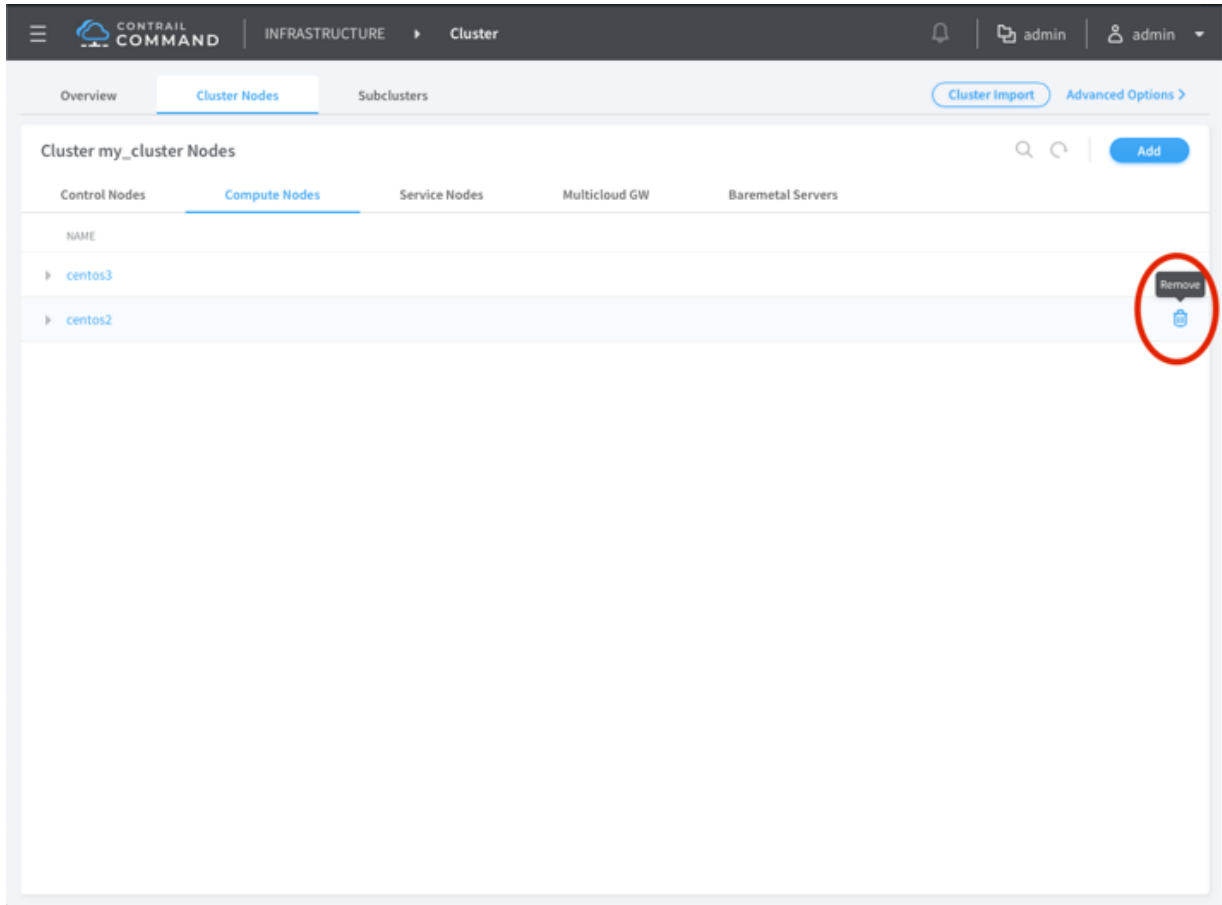
c. Click **Assign Nodes**.



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

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

1. Login to Contrail Command UI as a super user using credentials *admin* for username and *contrail123* for password.
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 Containerized Contrail Cluster”](#) on page 81.

Installing AppFormix using Contrail Command

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

3-Node setup includes:

Node 1—Contrail Command

Node 2—OpenStack and Contrail

Node 3—AppFormix

- Install Centos 7.5 on all three nodes. Install **libvirt** on the compute nodes using the following command:

```
yum -y install libvirt
```

- Copy the SSH key from the **contrail_command** Docker container to all servers.

The workflow for installing AppFormix and receiving statistics is:

1. Install AppFormix using Contrail Command.
2. Configure AppFormix plugins using Ansible.
3. Enable LLDP for each device and analytics you want to collect.

To install AppFormix using Contrail Command:

1. Download the following AppFormix packages from

<https://support.juniper.net/support/downloads/> and copy the **tar.gz** files to **/opt/software/appformix/** on the Contrail Command host.

NOTE: Contrail version 5.1 supports AppFormix version 2.19.10.

```
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
```

2. Verify the **command_servers.yml** file was added before installing Contrail Command as specified in [“Installing Contrail Command” on page 18](#).

Copy the images and license files for AppFormix inside the **contrail_command** container in a directory named **/opt/software/appformix** to make the AppFormix images available in the container. Directory structure needs to be created inside Docker. To do this, add **user_command_volumes:** as shown in the following example to the **command_servers.yml** referenced in [“Installing Contrail Command” on page 18](#).

NOTE: Using the following statements requires that the images and license file reside in **/opt/software/appformix** on the All-In-One (AIO) Contrail Cluster server. When adding 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
command_servers:
server1:
  ip: 192.168.100.129

```

3. On the Appformix Nodes tab, select **Show Advanced** and enter the values for each of the following parameters:

Figure 12: Setup > AppFormix Nodes

Table 3: Advanced Settings Descriptions and Example Corresponding Values from the instances.yml File

Field	Description
AppFormix License (Required)	Path to License <code>/opt/software/appformix/<appformix-license>-<version>.sig</code>

Table 3: Advanced Settings Descriptions and Example Corresponding Values from the instances.yml File (continued)

Field	Description
AppFormix Image Dir (Required)	<p>Path to AppFormix image directory. The path can include platform-images, network-device-images, openstack-images, or dependencies-images depending on your installation.</p> <pre> /opt/software/appformix/appformix-platform-images-2.19.10.tar.gz /opt/software/appformix/appformix-dependencies-images-2.19.10.tar.gz /opt/software/appformix/appformix-network_device-images-2.19.10.tar.gz /opt/software/appformix/appformix-openstack-images-2.19.10.tar.gz </pre>
AppFormix Version (Required)	<p>Current version to be installed.</p> <pre>appformix_version: 2.19.10</pre>

4. Select **OpenStack Platform Enabled** and enter the values for:

Table 4: OpenStack Platform Settings Descriptions and Examples Corresponding Values from the instances.yml File

Field	Description
Contrail Analytics URL (Required)	<p>URL for the Contrail analytics API. This field should reference the All-In-One (AIO) Contrail Cluster server.</p> <pre>contrail_analytics_url: 'http://192.168.0.28:8081'</pre>
Contrail Config URL (Required)	<p>URL for the Contrail configuration API. port on the All-In-One (AIO) Contrail Cluster server.</p> <pre>contrail_config_url: 'http://192.168.0.28:8082'</pre>
Contrail Cluster Name	<p>Name by which the Contrail instance will be displayed in the Dashboard.</p> <pre>contrail_cluster_name: 'ContrailCluster1'</pre>

5. Select the following options. Most of these are enabled by default.

Table 5: Discovery and Monitor Settings Descriptions and Examples Corresponding Values from the instances.yml File

Field	Description
AppFormix kvm Instance Discovery	<p>Identifies instances on Linux kernel-based virtual machines (KVM.)</p> <pre>appformix_kvm_instance_discovery: True</pre>

Table 5: Discovery and Monitor Settings Descriptions and Examples Corresponding Values from the instances.yml File (continued)

Field	Description
Network Device Monitoring Enabled	Monitors metrics from network devices, including the topology connections between devices. <code>appformix_network_device_monitoring_enabled: True</code>
JTI Network Device Monitoring Enabled	Enables JTI streaming telemetry for streaming metrics to AppFormix Agent. This can be enabled for devices running a supported version of Junos. <code>appformix_jti_network_device_monitoring_enabled: True</code>
Remote Host Monitoring Enabled	Monitor host remotely without installing AppFormix Agent on the host. Metric collection is supported using SNMP (version 2c or 3) and Intelligent Platform Management Interface (IPMI). <code>appformix_remote_host_monitoring_enabled: True</code>
Network Device Discovery Enabled (Required)	Network devices of a monitored network topology discovered by AppFormix. <code>network_device_discovery_enabled: True</code>

6. Enter the values for the AppFormix Configuration parameters:

- **appformix_haproxy_datamanager_port_http: 8200**
- **appformix_haproxy_datamanager_port_https: 8201**

You can also edit or remove roles from the AppFormix Configuration pane.

Figure 13: AppFormix Configuration Parameters

CONTRAIL COMMAND | SETUP

STEP 1 Inventory

STEP 2 Cloud Manager

STEP 3 (optional) Infrastructure Networks

STEP 4 (optional) Overcloud

STEP 5 (optional) Undercloud Nodes

STEP 6 (optional) JumpHost Nodes

STEP 7 Control Nodes

STEP 8 Orchestrator Nodes

STEP 9 Compute Nodes

STEP 10 (optional) Contrail Service Nodes

STEP 11 (optional) **AppFormix Nodes**

STEP 12 Summary

☒ Show Advanced

Appformix License: /opt/software/appf | Appformix Image Dir: /opt/software/appf | Appformix Version: 2.19.10

☒ Openstack Platform Enabled

Contrail Analytics URL: http://192.168.1.2:8080 | Contrail Config URL: http://192.168.1.2:8080 | Contrail Cluster Name: slp

☒ Appformix kvm Instance Discovery | ☒ Network Device Monitoring Enabled

☒ JTI Network Device Monitoring Enabled | ☒ Remote Host Monitoring Enabled | ☒ Network Device Discovery Enabled

AppFormix Configuration

Key	Value
appformix_haproxy_dataman	8200
appformix_haproxy_dataman	8201

[+ Add](#)

Available servers

Search servers: Add all

HOSTNAME	IP ADDRESS	DISK PARTITION
Add servers to your inventory		

[Previous](#)

Assigned AppFormix Nodes

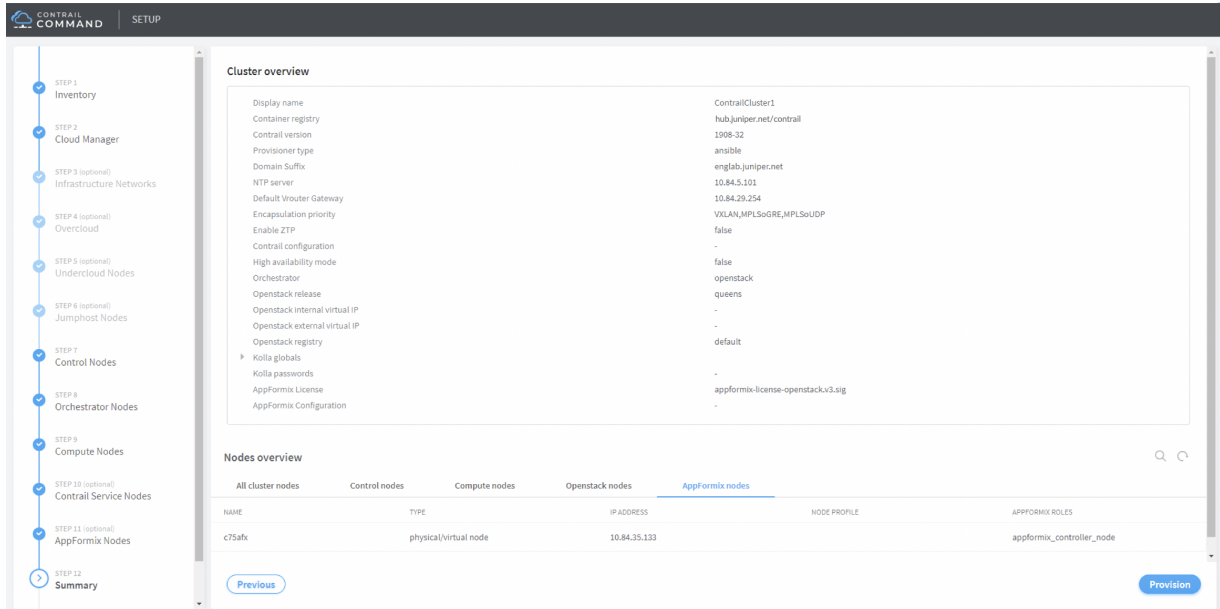
Search servers: Remove all

HOSTNAME	IP ADDRESS	DISK PARTITION
pod1-app	192.168.1.3	
pod1-alo-ztp	192.168.1.2	

[Next](#)

- Add the AppFormix server from the drop-down list in the Available servers section.
- Select a node in the Assigned AppFormix Nodes section. The roles selected are for the server added in 7. The following roles are for AppFormix:
 - appformix_controller
 - appformix_bare_host
 - appformix_openstack_controller
 - appformix_compute
- Click **Next** to continue to Summary.
- Verify the summary of your configuration and click **Provision**.

Figure 14: Setup Configuration Summary



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

Enable AppFormix Plugins

After the Contrail Command UI- based installation is complete for Openstack, Contrail, and AppFormix, follow these steps to enable AppFormix plugins and receive statistics.

Run the following commands on the **contrail-command** node.

To enable the AppFormix plugins:

1. Activate the virtual environment:

```
# docker exec -it contrail_command bash

# cd /usr/share/contrail/appformix-ansible-deployer/appformix

# source venv/bin/activate

# cd /opt/software/appformix/inventory/
```

2. Create a host file with entries from the **group_vars/instances.yml**.

```
# vi group_vars/instances.yml hosts
# cat hosts
[appformix_controller]
192.168.0.46          ansible_user=root  ansible_ssh_pass=<password>

[compute]
192.168.0.28          ansible_user=root  ansible_ssh_pass=<password>

[openstack_controller]
192.168.0.28          ansible_user=root  ansible_ssh_pass=<password>
```

3. Add the following to the **inventory/group_vars/all** file.

```
appformix_openstack_factory_plugins:
- { plugin_info: 'certified_plugins/cinder_api_logparser.json',
  log_file_path: '/var/log/cinder/cinder-api.log' }
- { plugin_info: 'certified_plugins/glance_logparser.json',
  log_file_path:
'/var/lib/docker/volumes/kolla_logs/_data/glance/glance-api.log' }
- { plugin_info: 'certified_plugins/heavy_hitters.json' }
- { plugin_info: 'certified_plugins/keystone_logparser.json',
  log_file_path:
'/var/lib/docker/volumes/kolla_logs/_data/keystone/keystone.log' }
- { plugin_info: 'certified_plugins/neutron_logparser.json',
  log_file_path:
'/var/lib/docker/volumes/kolla_logs/_data/neutron/neutron-server.log' }
- { plugin_info: 'certified_plugins/nova_logparser.json',
  log_file_path: '/var/lib/docker/volumes/kolla_logs/_data/nova/nova-api.log' }

appformix_plugins: '{{ appformix_openstack_factory_plugins }} + {{
appformix_application_factory_plugins }} + {{ appformix_contrail_factory_plugins
}} + {{ appformix_remote_host_factory_plugins }} + {{
appformix_network_device_factory_plugins }}'
```

4. Call the AppFormix playbook and the AppFormix plugins are added.

```
# cd /opt/software/appformix/; ansible-playbook -i inventory
--skip-tags=install_docker appformix-2.19.10/appformix_openstack.yml
```

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 **Enabled** in Device Info for LLDP.
 - 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 15: Enable LLDP and Add Management IP for Network Device

Configure Network Device		Device Info	
SNMP	+	LLDP:	Contrail ▼
JTI	+	Chassis Type:	Coreswitch ▼
GRPC	+	Management IP:	10.102.70.213
Exit		Next	

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

Figure 16: Add Selected Resource for Network Device MIB Configurations

Configure Network Device

SNMP Configurations

SNMP Version: 2c

SNMP Community: public

MIB Configurations

Resource: Select Resource

+ Add

Selected MIBs

IF-MIB::ifTable

Back

Submit

5. Click **Submit** to complete.

You can set alarms on different aspects of each network device. These alarms can be created in either the AppFormix UI or Contrail Command.

RELATED DOCUMENTATION

AppFormix General Requirements
Configuring Instances in AppFormix
Configuring AppFormix Alarms using Contrail Command
Viewing Cluster Node Details and Metric Values
Metrics Collected by AppFormix

Deploying Contrail Command and Contrail All-In-One Cluster

Contrail Release 5.1 (or higher) supports deploying Contrail Command and All-In-One (AIO) Contrail Cluster using a single docker command without providing any configuration files.

You can provide the required deployment parameters with the docker command.

Contrail Command and Contrail Cluster are deployed on the same *Deployer* node.

NOTE: Contrail Release 5.1 does not support AppFormix deployment with single line deployer.

Assumptions

- The default SSH username is **root** and the default SSH password is **cOntrail123**.
- The password is the same across all Contrail services (UI Database, Keystone, Client and OpenStack Keystone). The default password is **contrail123**.
- OpenStack is deployed with the default SKU, that is, *queens*.

System Requirements

- A VM or physical server with:
 - 16 vCPUs
 - 64 GB RAM
 - 300 GB storage out of which 256 GB is allocated to */root* directory
- Internet access to and from the VM or physical server, which is the *Deployer* node.
- (Recommended) x86 server with CentOS 7.6 as the base OS.

For a list of supported platforms, see [Supported Platforms Contrail 5.1](#).

NOTE:

- Email **contrail-registry@juniper.net** for Contrail container registry credentials.
- Access *Release Build* tags for contrail-command-deployer located at [README Access to Contrail Registry](#).

Run the following commands to deploy Contrail Command and All-In-One (AIO) Contrail Cluster:

1. Install docker on the Deployer node.

```
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
```

For details, refer to <https://docs.docker.com/install/linux/docker-ce/centos/>.

2. Login to docker private secure registry.

NOTE: Email contrail-registry@juniper.net for Contrail container registry credentials.

```
docker login hub.juniper.net
```

Login using your docker registry credentials.

3. Provide deployer container image.

Access *Release Build* tags for contrail-command-deployer located at [README Access to Contrail Registry](#).

```
export TAG=<BUILD_NO>
export CCD_IMAGE=hub.juniper.net/contrail/contrail-command-deployer:$TAG
```

For example:

```
export TAG=5.1.0-0.38
```

4. Deploy Contrail Command and Contrail Cluster.

Use any of the two methods:

Contrail Cluster deployment uses the default management IPv4 interface if the *Deployer* node has multiple network interfaces.

- Method 1—Provide all the required parameters during the **docker run** command.

```
docker run -td --net host --privileged --name contrail_command_deployer
$CCD_IMAGE deploy_cem container_registry=hub.juniper.net/contrail
container_tag=$TAG contrail_container_tag=$TAG-queens ntpserver=<NTP_IP>
vrouter_gateway=<VROUTER_GATEWAY_IP>
```

- Method 2—Provide all the required parameters in a **configuration file** as key values.

NOTE: The file must be mounted during the **docker run** command.

- Create a configuration file.

```
ntpserver: <NTP_IP>
vrouter_gateway: <VROUTER_GATEWAY_IP>
container_tag: <COMMAND_BUILD_TAG>
contrail_container_tag: <CONTRAIL_BUILD_TAG>
container_registry: hub.juniper.net/contrail
```

```
export MIN_CONFIG_FILE=<ABSOLUTE_PATH_OF_MIN_CONFIG_FILE>
docker run -td --net host -v $MIN_CONFIG_FILE:/cluster_config.yml
--privileged --name contrail_command_deployer $CCD_IMAGE
```

You can track the progress of Contrail Command installation by running :

```
docker logs -f contrail_command_deployer
```

After Contrail Command is deployed, run the following command to check the Contrail Cluster provision logs.

```
docker exec -it contrail_command tailf /var/log/ansible.log
```

You can access the Contrail Command UI from any browser at https://<COMMAND_IP>:8079.

NOTE: Flush the *iptables* rules on the *Deployer* node if firewall is blocking port access.

```
iptables -F
```

Supported parameters

You can explicitly specify the nodes (physical server or VM) to deploy Contrail Command and Contrail Cluster.

- Mandatory parameters:

```
# NTP server IP address
ntpserver: <NTP_IP>
# vRouter gateway IP address of contrail cluster
vrouter_gateway: <VROUTER_GATEWAY_IP>
```

```
# contrail container registry credentials
# Note: These are Optional if both contrail command and contrail cluster are
# installed on Deployer node
registry_username: <REGISTRY_USERNAME>
registry_password: <REGISTRY_PASSWORD>
```

- Optional parameters:

```
# contrail command IP address
# Default: Management IP address of deployer node
command_ip: <COMMAND_IP>
# contrail cluster (All-In-One) IP address
# Default: Management IP address of deployer node
cluster_ip: <ALL_IN_ONE_IP>
# contrail command login password
# Default: "contrail123"
service_password: <COMMAND_PASSWORD>
# contrail command container tag
# Default: "latest"
container_tag: <COMMAND_BUILD_TAG>
# contrail cluster container tag
# Default: "latest"
contrail_container_tag: <CONTRAIL_BUILD_TAG>
# contrail container registry info
# Default: "hub.juniper.net/contrail"
container_registry: <CONTAINER_REGISTRY>
```

You can add more parameters based on your requirements.

Installing Contrail

IN THIS CHAPTER

- Installing Contrail with OpenStack and Kolla Ansible | 68
- Adding a New Compute Node to Existing Containerized Contrail Cluster | 81

Installing Contrail with OpenStack and Kolla Ansible

IN THIS SECTION

- Set Up the Base Host | 69
- Multiple Interface Configuration Sample for Multinode OpenStack HA and Contrail | 72
- Single Interface Configuration Sample for Multinode OpenStack HA and Contrail | 74
- Frequently Asked Questions | 77

This topic provides the steps needed to install Contrail Release 5.1.X. with OpenStack, using Kolla Ansible playbook **contrail-kolla-ansible**.

Kolla is an OpenStack project that provides Docker containers and Ansible playbooks to provide production-ready containers and deployment tools for operating OpenStack clouds.

The **contrail-kolla-ansible** playbook works in conjunction with **contrail-ansible-deployer** to install OpenStack and Contrail Release 5.1.x. containers.

To deploy a Contrail Cluster using Contrail Command, see [“Installing Contrail Cluster using Contrail Command and instances.yml” on page 39](#).

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

For a list of supported platforms, see [Supported Platforms Contrail 5.1](#).

Set Up the Base Host

This procedure assumes you are installing with CentOS 7.6 kernel 3.10.0-957.11.6.el7.x86_64. The vRouter has a [dependency](#) with the host kernel. Install this kernel version on the target nodes before provisioning.

To set up the base host:

1. Download *Ansible Deployer* installer package from the [Contrail Downloads](#) page.

2. Install Ansible.

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

```
yum -y install git ansible-2.5.2.0
```

3. Untar the tgz. file.

```
- tar xvf contrail-ansible-deployer-5.1.0-0.38.tgz
```

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

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

- The provider configuration (**provider_config**) section refers to the cloud provider where the Contrail cluster will be hosted, and contains all parameters relevant to the provider. For bare metal servers, the provider is **bms**.
- The **kolla_globals** section refers to OpenStack services. For more information about all possible **kolla_globals**, see <https://github.com/Juniper/contrail-kolla-ansible/.../globals.yml>.
- Additional Kolla configurations (**contrail-kolla-ansible**) are possible as **contrail_additions**. For more information about all possible **contrail_additions** to Kolla, see <https://github.com/Juniper/contrail-kolla-ansible/.../all.yml>.
- The **contrail_configuration** section contains parameters for Contrail services.

- **CONTAINER_REGISTRY** specifies the registry from which to pull Contrail containers. It can be set to your local Docker registry if you are building your own containers. If a registry is not specified, it will try to pull the containers from the Docker hub.

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

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

```

config_database:
  config:
  control:
  analytics_database:
  analytics:
  analytics_alarm:
  analytics_snmp:
  webui:
  vrouter:
  openstack:
  openstack_compute:

```

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

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

```

provider_config:
  bms:
    ssh_pwd: <password>
    ssh_user: root
    ntpserver: <IP NTP server>
    domainsuffix: local
  instances:
    bms1:
      provider: bms
      ip: <IP BMS>1
  contrail_configuration:
    RABBITMQ_NODE_PORT: 5673
    AUTH_MODE: keystone
    KEYSTONE_AUTH_URL_VERSION: /v3
  kolla_config:
    kolla_globals:

```

```

    enable_haproxy: no
kolla_passwords:
    keystone_admin_password: <Keystone admin password>

```

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
    CONTROL_DATA_NET_LIST: 198.168.10.0/24
    KEYSTONE_AUTH_URL_VERSION: /v3
kolla_config:
    kolla_globals:
        kolla_internal_vip_address: <Internal VIP>

```



```

    contrail_api_interface_address: <Contrail API Addr>
    enable_haproxy: no
kolla_passwords:
    keystone_admin_password: <Keystone Admin Password>

```

5. Run the following commands 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**

6. 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 4

Multiple Interface Configuration Sample for Multinode OpenStack HA and Contrail

This is a configuration sample for a multiple interface, multiple node deployment of high availability OpenStack and Contrail Release 5.1.x. Use this sample to configure parameters specific to your system.

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

Configuration Sample—Multiple Interface

```

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
    CONTROL_DATA_NET_LIST: <Control Data Subnet CIDR>
    KEYSTONE_AUTH_URL_VERSION: /v3
    IPFABRIC_SERVICE_HOST: <Service Host IP>
    # Add following line for TSN Compute Node
    TSN_NODES: <TSN NODE IP List>
    # For EVPN VXLAN TSN
    ENCAP_PRIORITY: "VXLAN,MPLSoUDP,MPLSoGRE"
    PHYSICAL_INTERFACE: <Interface name>
kolla_config:
    kolla_globals:
        kolla_internal_vip_address: <Internal VIP>
        kolla_external_vip_address: <External VIP>
        contrail_api_interface_address: <Contrail API IP>
    kolla_passwords:
        keystone_admin_password: <Keystone Admin Password>

```

Single Interface Configuration Sample for Multinode OpenStack HA and Contrail

This is a configuration sample for a multiple node, single interface deployment of high availability OpenStack and Contrail Release 5.1.x. Use this sample to configure parameters specific to your system.

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

Configuration Sample—Single Interface

```

provider_config:
    bms:
        ssh_pwd: <password>

```

```

    ssh_user: root
    ntpserver: xx.xx.x.xx
    domainsuffix: local
instances:
  centos1:
    provider: bms
    ip: ip-address
    roles:
      openstack:
  centos2:
    provider: bms
    ip: ip-address
    roles:
      openstack:
  centos3:
    provider: bms
    ip: ip-address
    roles:
      openstack:
  centos4:
    provider: bms
    ip: ip-address
    roles:
      config_database:
      config:
      control:
      analytics_database:
      analytics:
      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](#).

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-ealdaa253b6c	cirros2	active

RELATED DOCUMENTATION

[Installing Contrail Cluster using Contrail Command and instances.yml](#) | 39

Adding a New Compute Node to Existing Containerized Contrail Cluster

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.

```
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, *svr5*.

```

provider_config:

  bms:
    ssh_pwd: c0ntrail123
    ssh_user: root
    ntpserver: x.x.x.x
    domainsuffix: local
  instances:
    svr1:
      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.yml** 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 Containerized Contrail Cluster Using Contrail Command”](#) on page 49.

Using Contrail with Kubernetes

IN THIS CHAPTER

- Installing and Managing Contrail Microservices Architecture Using Helm Charts | 85
- Provisioning of Kubernetes Clusters | 89
- Installing Standalone Kubernetes Contrail Cluster using the Contrail Command UI | 96
- Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability | 105
- Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata | 117
- Accessing a Contrail OpenStack Helm Cluster | 120
- Frequently Asked Questions About Contrail and Helm Charts | 123
- Installing Contrail Networking for Kubernetes using Helm | 128
- Verifying Configuration for CNI for Kubernetes | 134

Installing and Managing Contrail Microservices Architecture Using Helm Charts

IN THIS SECTION

- Understanding Helm Charts | 86
- Contrail Helm Deployer Charts | 86
- Contrail Kubernetes Resource implementation | 87
- Example: Contrail Pods Deployment Options | 87
- Installing Contrail Using Helm Charts | 88

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*.

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-collector
    - 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 105](#)
- [Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata on page 117](#)

RELATED DOCUMENTATION

[Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability | 105](#)

[Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata | 117](#)

[Accessing a Contrail OpenStack Helm Cluster | 120](#)

[Frequently Asked Questions About Contrail and Helm Charts | 123](#)

Provisioning of Kubernetes Clusters

IN THIS SECTION

- [Provisioning of a Standalone Kubernetes Cluster | 89](#)
- [Provisioning of Nested Contrail Kubernetes Clusters | 90](#)
- [Provisioning of Non-Nested Contrail Kubernetes Clusters | 94](#)

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. Re-image the node to CentOS 7.4 using Linux kernel version 3.10.0-862.3.2.
2. Install the necessary tools.

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

3. Download the **contrail-ansible-deployer-5.1.0-0.38.tgz** Ansible Deployer application tool package onto your provisioning host from [Contrail Downloads](#) page and extract the package.

```
- tar xvf contrail-ansible-deployer-5.1.0-0.38.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 8](#) 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

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 Release 5.1.x.
2. Ensure that you have an operational Kubernetes v1.9.2 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 92](#), 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.

1. [Configure network connectivity to Contrail configuration and data plane functions. | 91](#)
2. [Generate a single yaml file to create a Contrail-k8s cluster | 92](#)
3. [Instantiate the Contrail-k8s cluster | 93](#)

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 be 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 components 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	<i>Service-IP for the active node</i>	9091	127.0.0.1	9091

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

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

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

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

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-5.1.0-0.38.tgz** Ansible Deployer application tool package onto your provisioning host from [Juniper Networks](#) and extract the package.

- **tar xvf contrail-ansible-deployer-5.1.0-0.38.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 your 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 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.yaml
```

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

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.

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 Release 5.1.x 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/controller=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
```

Provisioning a Contrail Kubernetes Cluster

Follow these steps to provision Contrail Kubernetes cluster.

1. Download the **contrail-ansible-deployer-5.1.0-0.38.tgz** Ansible Deployer application tool package onto your provisioning host from [Juniper Networks](#) and extract the package.


```
- tar xvf contrail-ansible-deployer-5.1.0-0.38.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.yaml
```

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.yaml
```

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-gl5m	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

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

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 **kubectI** CLI on the Kubernetes master node.

Configuration

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 17: Create Server

The screenshot shows the 'Create Server' modal in the Contrail Command interface. On the left, a sidebar lists steps from 'Inventory' to 'Provisioning'. The modal has a 'Choose Mode' section with three radio buttons: 'Express' (selected), 'Detailed', and 'Bulk Import (csv)'. Below this are three input fields: 'Hostname' with the value 'test_host', 'Management IP' with the value '10.87.84.65', and 'Management Interface' with the value 'eth0'. There is a 'Credentials' dropdown menu currently showing 'contrail_creds'. At the bottom of the modal are three buttons: '+ Add', 'Cancel', and 'Create'. The background shows the 'Servers' tab with a table of servers and a 'Next' button at the bottom right.

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 for Contrail](#).

Default vRouter Gateway = Default gateway for the compute nodes. If any one of the compute nodes has a different default gateway than the one provided here, enter that gateway in **5** and **6** 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 18: Contrail Cluster

The screenshot shows the 'Contrail Cluster' setup page in the Contrail Command interface. The left sidebar lists steps from 'Inventory' to 'Provisioning', with 'Contrail Cluster' being the current step. The main form contains the following fields and options:

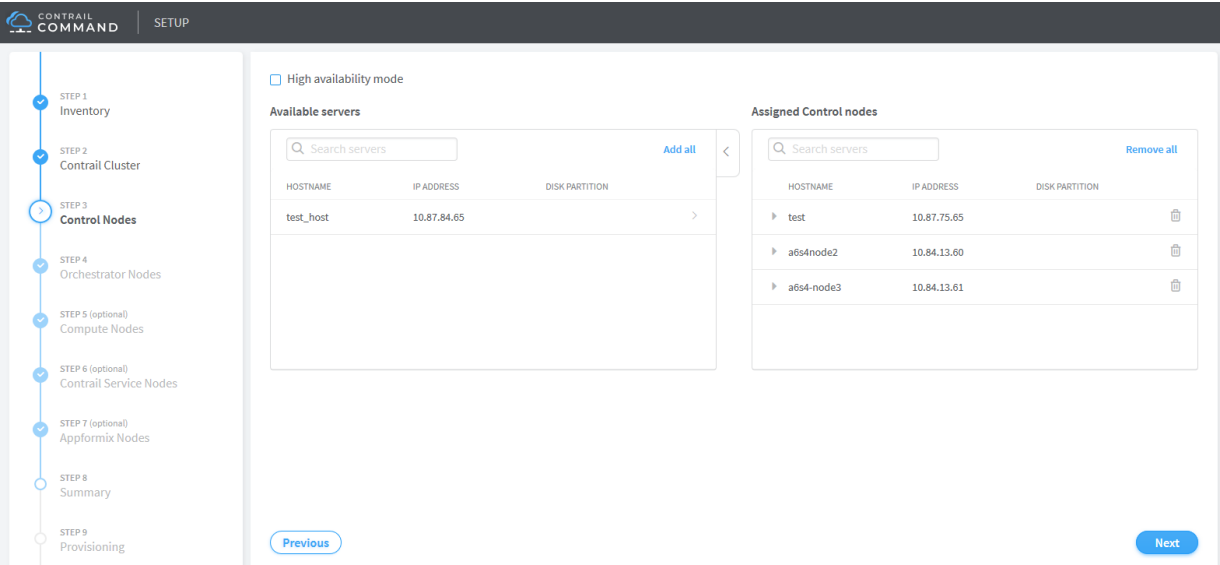
- Cluster Name***: Text input with value 'Test'.
- Container Registry***: Text input with value 'opencontrainightly'.
- Insecure**: Checkbox, currently unchecked.
- Container Registry Username***: Text input with value 'admin'.
- Container Registry Password***: Text input with value 'contrail123'.
- Contrail Version***: Text input with value 'latest'.
- Provisioner Type**: Dropdown menu with value 'Ansible'.
- Domain Suffix**: Text input with value 'local'.
- NTP Server**: Text input, currently empty.
- Default Vrouter Gateway**: Text input, currently empty.
- Encapsulation Priority**: Dropdown menu with value 'MPLSoGRE,MPLSoUDP,...'.
- Enable ZTP**: Checkbox with an information icon, currently unchecked.
- Show Advanced Options**: Checkbox, currently unchecked.

At the bottom of the form are 'Previous' and 'Next' buttons. The 'Next' button is highlighted in blue.

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

3. Select the Contrail control nodes.

Figure 19: Control Nodes



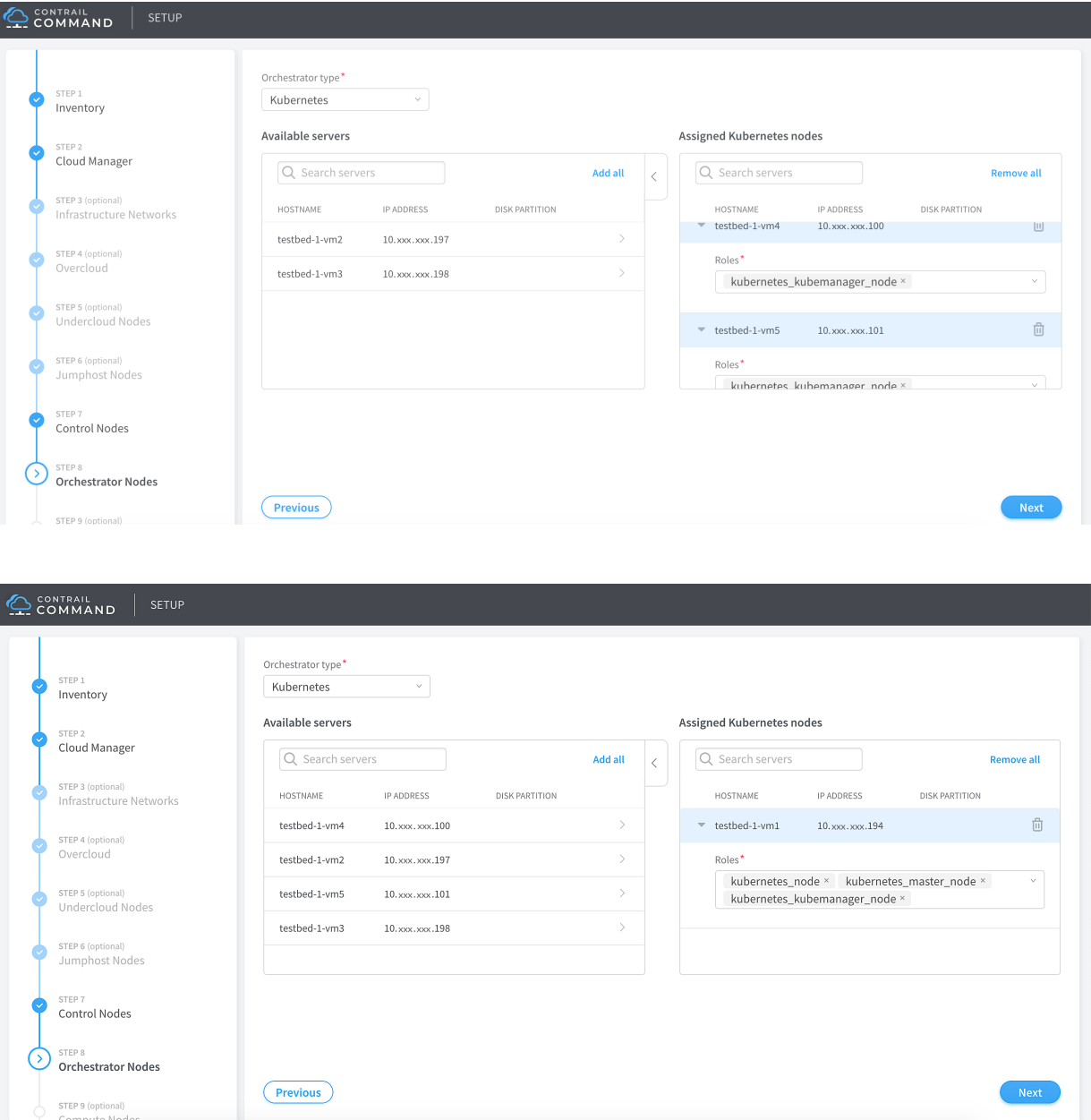
Click **Next**. The **Orchestrator Nodes** page appears.

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.

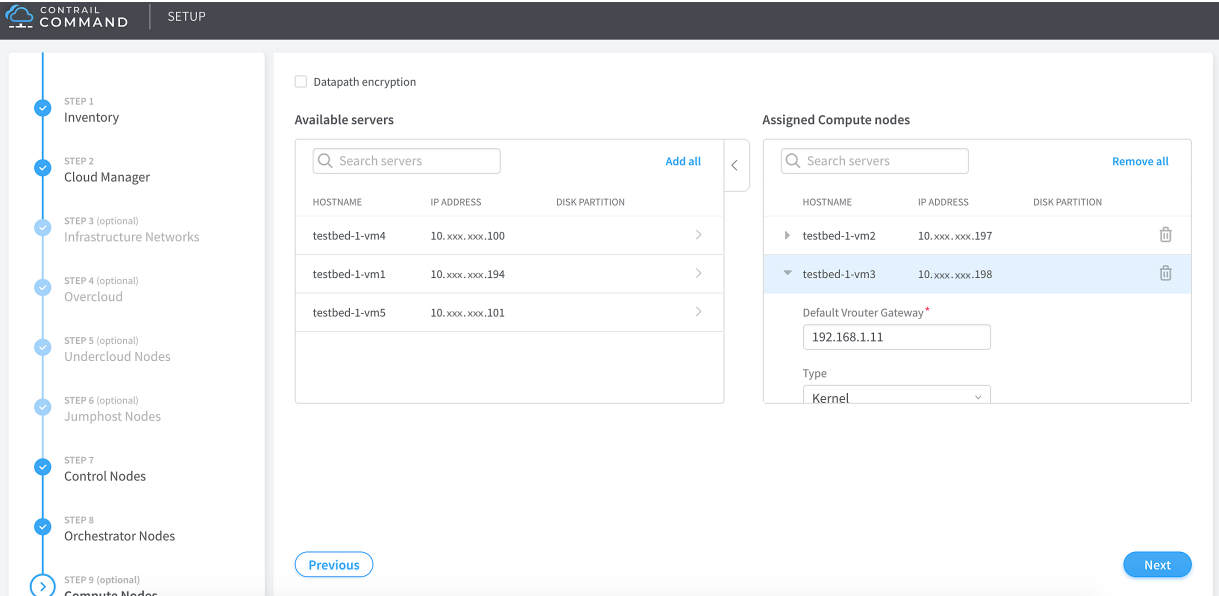
Figure 20: Orchestrator Nodes



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

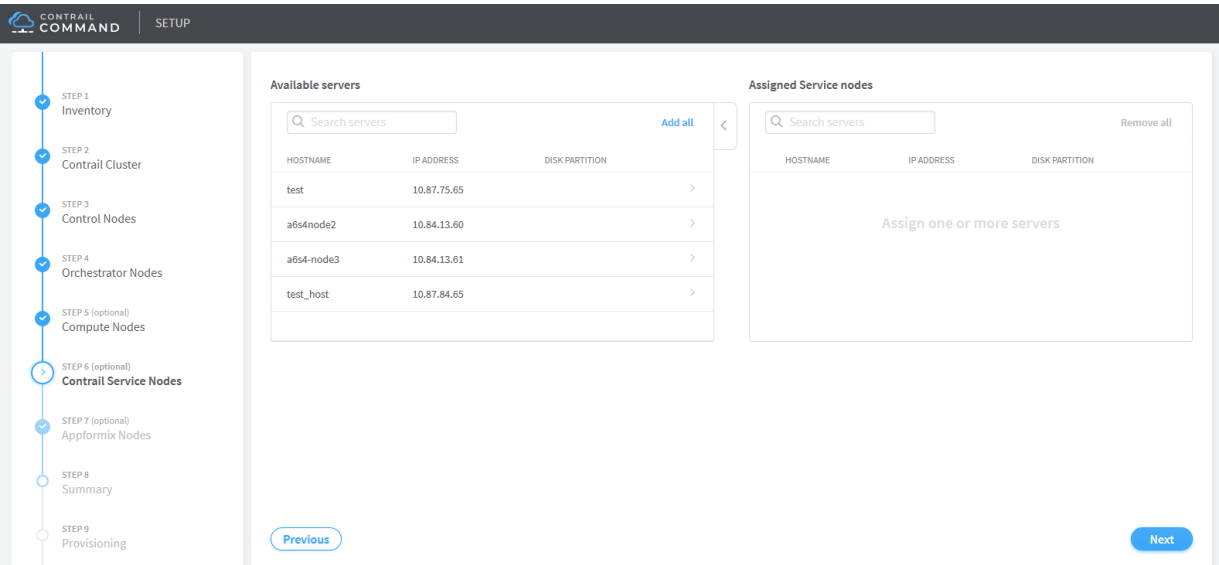
5. Select the compute node associated with the `kubernetes_node` role from the list of available servers,

Figure 21: Compute Nodes



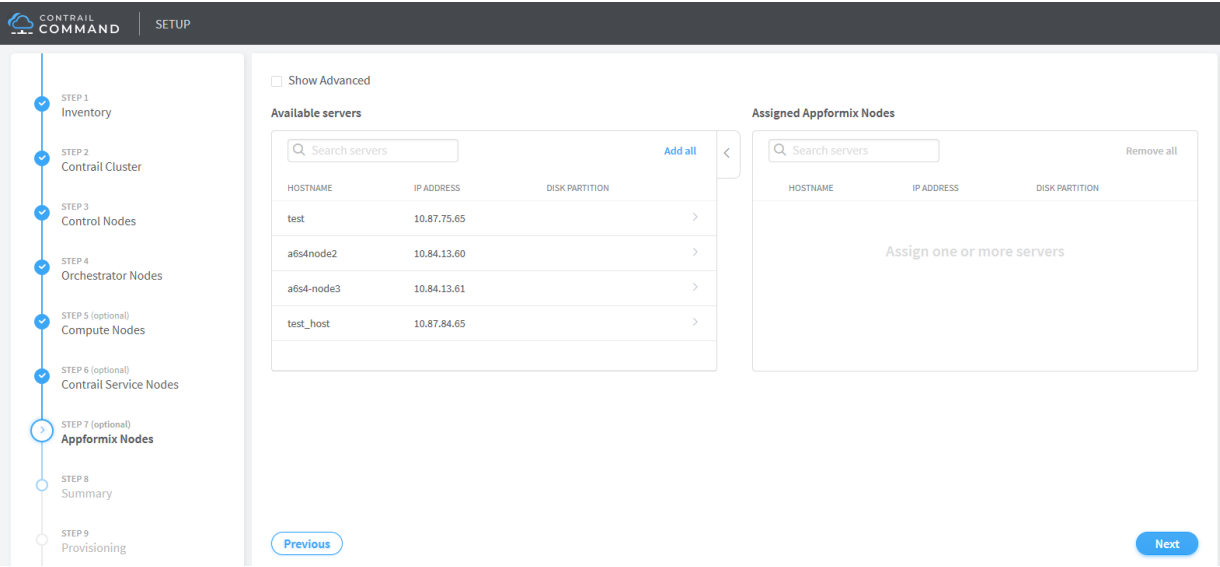
- Click **Next**. The **Contrail Service Nodes** page appears.
6. (Optional) Select the Contrail service nodes from the list of available servers.

Figure 22: Contrail Service Nodes



- Click **Next**. The **Appformix Nodes** page appears.
7. (Optional) Select the AppFormix nodes from the list of available nodes.

Figure 23: 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 24: Summary - Cluster Overview

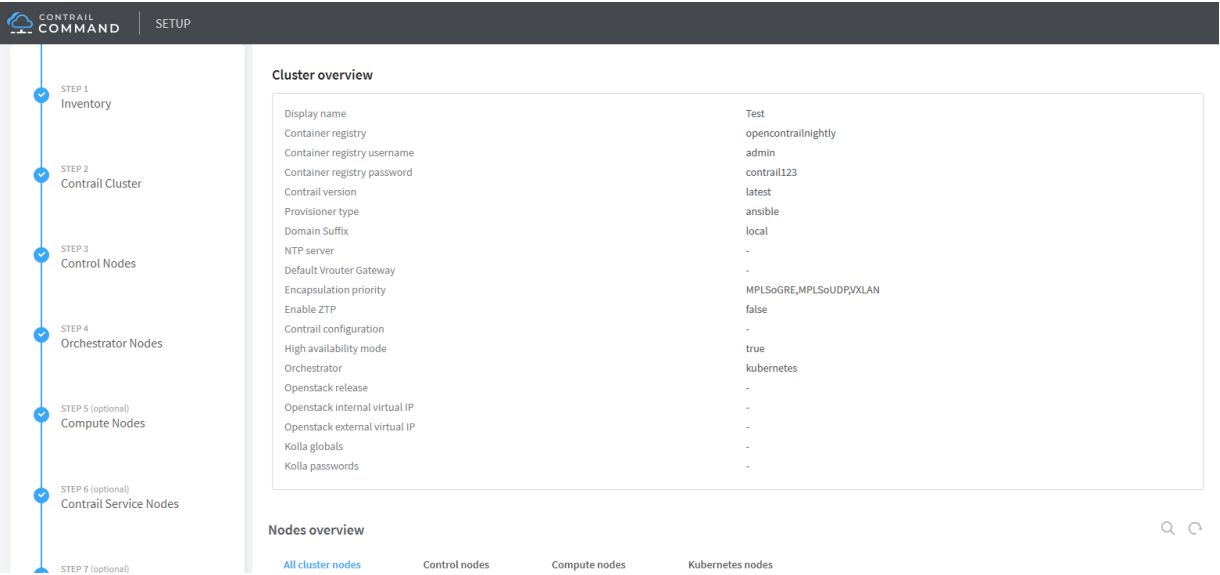


Figure 25: Summary - Nodes Overview

CONTRAIL
COMMAND

SETUP

STEP 3
Control Nodes

STEP 4
Orchestrator Nodes

STEP 5 (optional)
Compute Nodes

STEP 6 (optional)
Contrail Service Nodes

STEP 7 (optional)
Appformix Nodes

STEP 8
Summary

STEP 9
Provisioning

Provisioner type
Domain Suffix
NTP server
Default Vrouter Gateway
Encapsulation priority
Enable ZTP
Contrail configuration
High availability mode
Orchestrator
Openstack release
Openstack internal virtual IP
Openstack external virtual IP
Kolla globals
Kolla passwords

ansible
local
-
-
MPLSoGRE,MPLSoUDP VXLAN
false
-
true
kubernetes
-
-
-
-
-

Nodes overview

All cluster nodes

Control nodes

Compute nodes

Kubernetes nodes

NAME	TYPE	IP ADDRESS
test	physical/virtual node	10.87.75.65

Previous

Provision

Sample `command_servers.yml` File

You can use the following sample `command_servers.yml` file to install Contrail Command.

```
command_servers:
  server1:
    ip: 10.204.216.31
    connection: ssh
    ssh_user: root
    ssh_pass: c0ntrail123
    sudo_pass: c0ntrail123
    ntpserver: 10.204.217.158
    registry_insecure: true
    container_registry: 10.204.217.152:5000
    container_name: contrail-command
    container_tag: master-659
    #container_registry_username: <Registry_Username>
    #container_registry_password: <Registry_Password>
    config_dir: /etc/contrail
    contrail_config:
      database:
        type: postgres
        dialect: postgres
        password: contrail123
        user: root
      keystone:
        assignment:
          data:
            users:
              admin:
                password: contrail123
      insecure: true
      no_auth: true
      auth_type: basic-auth
      client:
        password: contrail123
```

NOTE: You must set the `auth_type` to **basic-auth** for Kubernetes and vCenter deployment before importing or provisioning the cluster.

RELATED DOCUMENTATION

[Installing Contrail Command | 18](#)

[Installing Contrail Cluster using Contrail Command and instances.yml | 39](#)

[Importing Contrail Cluster Data using Contrail Command | 44](#)

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

IN THIS SECTION

- [System Specifications | 105](#)
- [Preparing to Install | 106](#)
- [Installation of OpenStack Helm Charts | 109](#)
- [Installation of Contrail Helm Charts | 111](#)
- [Basic Testing OpenStack Helm Contrail Cluster | 116](#)
- [Accessing the Contrail OpenStack Helm Cluster | 116](#)

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 [Supported Platforms Contrail 5.1](#).

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.

```
#!/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-nfs-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 120](#)

RELATED DOCUMENTATION

[Installing and Managing Contrail Microservices Architecture Using Helm Charts | 85](#)

[Frequently Asked Questions About Contrail and Helm Charts | 123](#)

[Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata | 117](#)

[Accessing a Contrail OpenStack Helm Cluster | 120](#)

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

IN THIS SECTION

- [System Specifications | 117](#)
- [Installation Steps | 118](#)
- [Accessing the Contrail OpenStack Helm Cluster | 120](#)

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-deployer
Export 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 120](#)

RELATED DOCUMENTATION

[Installing and Managing Contrail Microservices Architecture Using Helm Charts | 85](#)

[Using Helm Charts to Provision Multinode Contrail OpenStack Ocata with High Availability | 105](#)

[Accessing a Contrail OpenStack Helm Cluster | 120](#)

[Frequently Asked Questions About Contrail and Helm Charts | 123](#)

Accessing a Contrail OpenStack Helm Cluster

IN THIS SECTION

- [Overview | 121](#)
- [Installing the OpenStack Client | 121](#)
- [Create openstackrc File and Test OpenStack Client | 121](#)
- [Accessing the Contrail Web UI | 122](#)
- [Accessing OpenStack Horizon | 122](#)
- [Accessing the Virtual Machine Console from Horizon | 123](#)
- [OpenStack References | 123](#)

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 105](#)
- [Using Helm Charts to Provision All-in-One Contrail with OpenStack Ocata on page 117](#)
- [Frequently Asked Questions About Contrail and Helm Charts on page 123](#)

Installing the OpenStack Client

Use this procedure to install the OpenStack CLI tool.

1. Install the OpenStack client CLI tool on the master 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

- How do I set up the vhost0 interface for the vrouter on the non-management interface of the compute node? | [124](#)
- How do I configure the Contrail control BGP server to listen on a different port? | [125](#)
- How can I pass additional parameters to services in Contrail by using the configuration file in INI format? | [125](#)
- How do I configure services for the vrouter agent? | [125](#)

- What are the Contrail services that can be configured? | 126
- How can I pass additional parameters to the Contrail Web UI services a with configuration file in JS format? | 126
- How can I verify all pods of Contrail are up and running? | 127
- How can I see the logs of each of the containers? | 127
- How can I enter into a pod? | 127

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

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 -zxvf 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-entypoint: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

- [View Pod Name and IP Address | 134](#)
- [Verify Reachability of Pods | 135](#)
- [Verify If Isolated Namespace-Pods Are Not Reachable | 135](#)
- [Verify If Non-Isolated Namespace-Pods Are Reachable | 136](#)
- [Verify If a Namespace is Isolated | 137](#)

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-isolated 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 ~]#  
kubect1 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

Installing and Provisioning Containerized Contrail Controller for Kubernetes

Using Contrail with Mesos

IN THIS CHAPTER

- Understanding Contrail with Mesos Architecture | 138
- Installing Contrail with Mesos | 143

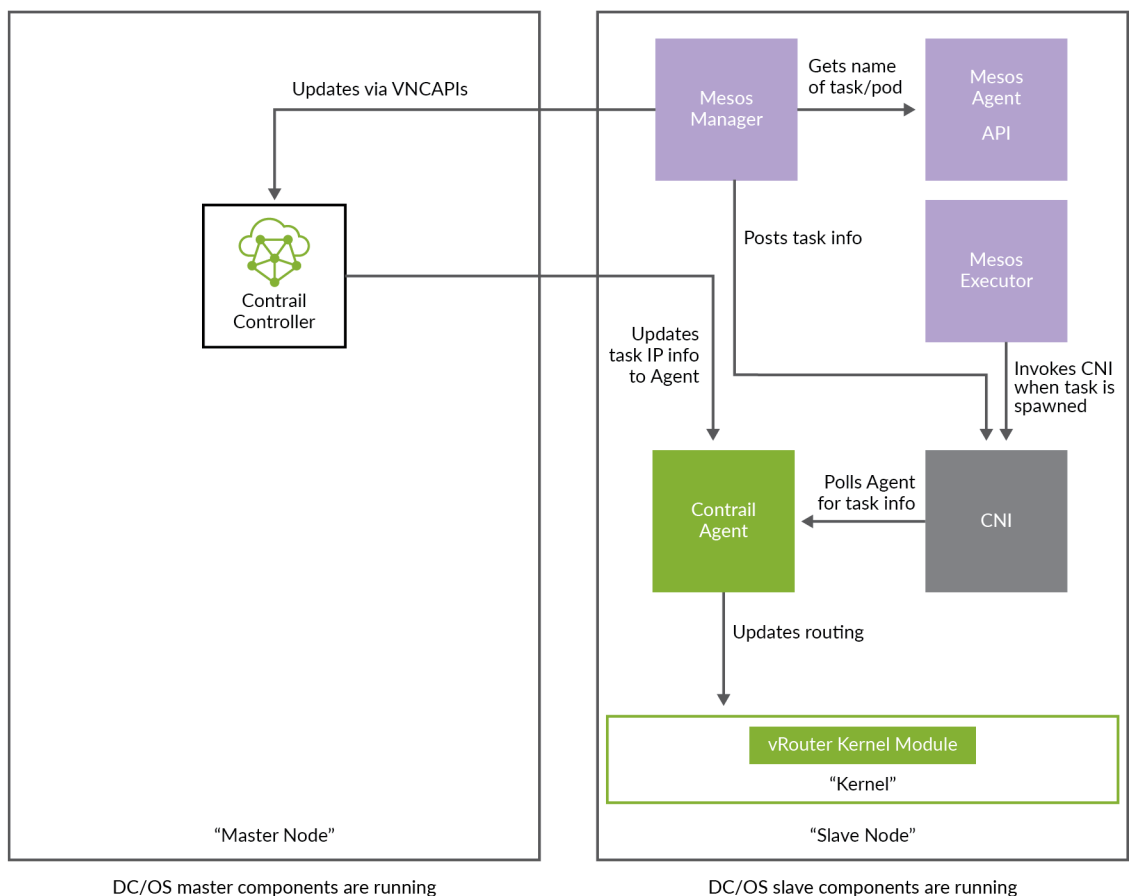
Understanding Contrail with Mesos Architecture

IN THIS SECTION

- Contrail with Mesos Architecture Diagram | 139
- Setup information | 139
- Components | 140

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 primary components including Contrail Controller, Analytics, Config, and UI.

Backup/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

IN THIS SECTION

- [Contrail Controller | 140](#)
- [Mesos Manager | 141](#)
- [Contrail Container Network Interface \(CNI\) | 142](#)

The following components are a part of the architecture:

Contrail Controller

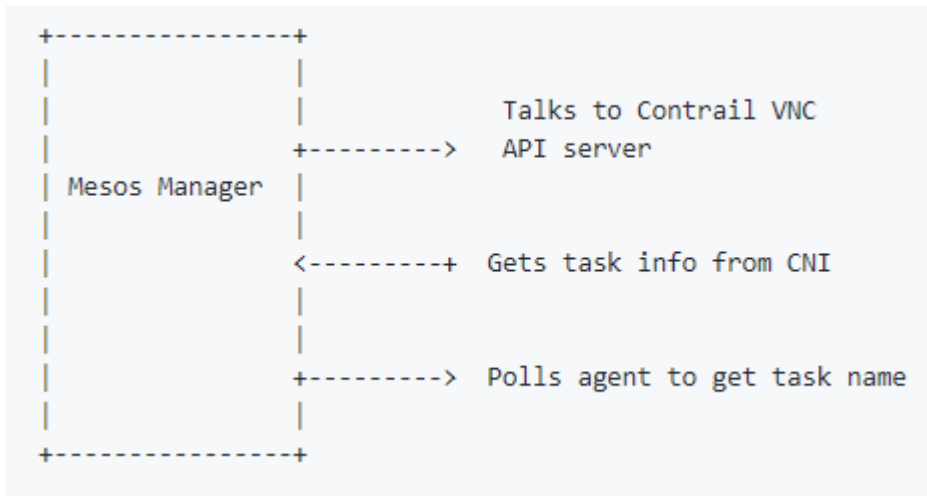
Contrail controller performs all the decision making. It includes config management, analytics, UI and control plane components for network virtualization. For further details, refer to <https://github.com/Juniper/contrail-controller>.

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.



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.

Supported DC/OS version—1.11.0

Contrail Setup

Run the following commands to set up Contrail.

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*.

1. Install ansible and clone ansible deployer.

For more details, refer to <https://github.com/Juniper/contrail-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>
```

2. Sample config yaml file.

For more details, refer to

https://github.com/Juniper/contrail-ansible-deployer/blob/master/examples/mesos_bms.md.

```
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

```

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 44.

RELATED DOCUMENTATION

[Understanding Contrail with Mesos Architecture](#) | 138

[Importing Contrail Cluster Data using Contrail Command](#) | 44

Using VMware vCenter with Containerized Contrail

IN THIS CHAPTER

- Integrating vCenter for Contrail | 146
- Configuring Underlay Network for ContrailVM | 155
- Installing and Provisioning Contrail VMware vRealize Orchestrator Plugin | 166

Integrating vCenter for Contrail

IN THIS SECTION

- Prerequisites | 146
- ESX Agent Manager | 147
- Set Up vCenter Server | 147
- Configure Contrail Parameters | 152
- Install Contrail | 152
- Monitor and Manage ContrailVM from ESX Agent Manager | 152

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 17.

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.

```
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.

```
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/contrail
  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 is

Username: *root*

Password: *cOntrail123*

```

---
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.yaml
```

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

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 26: vCenter Server Extensions

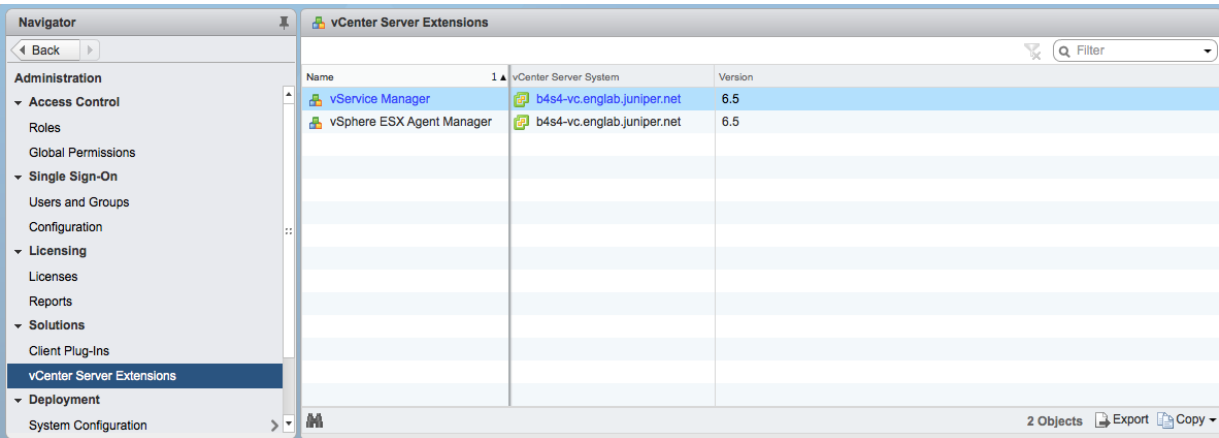
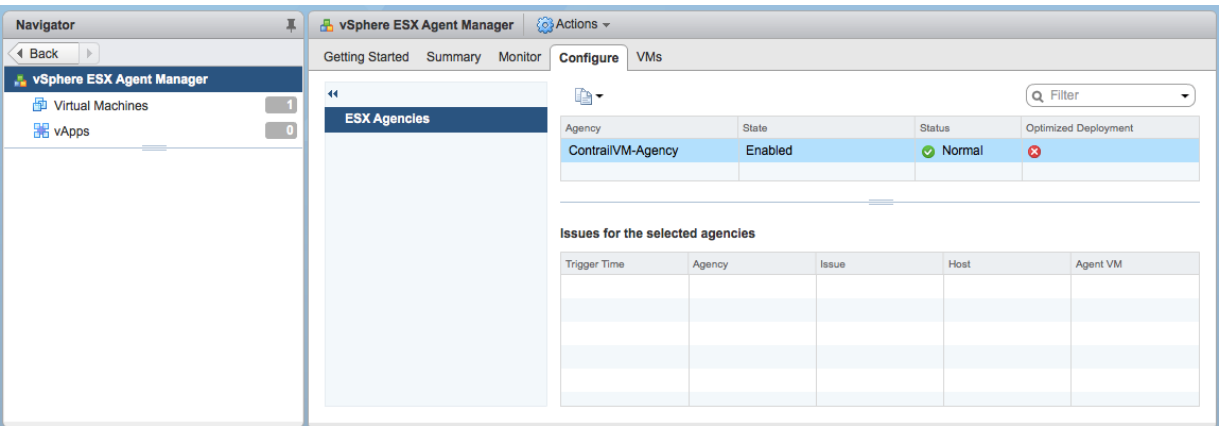
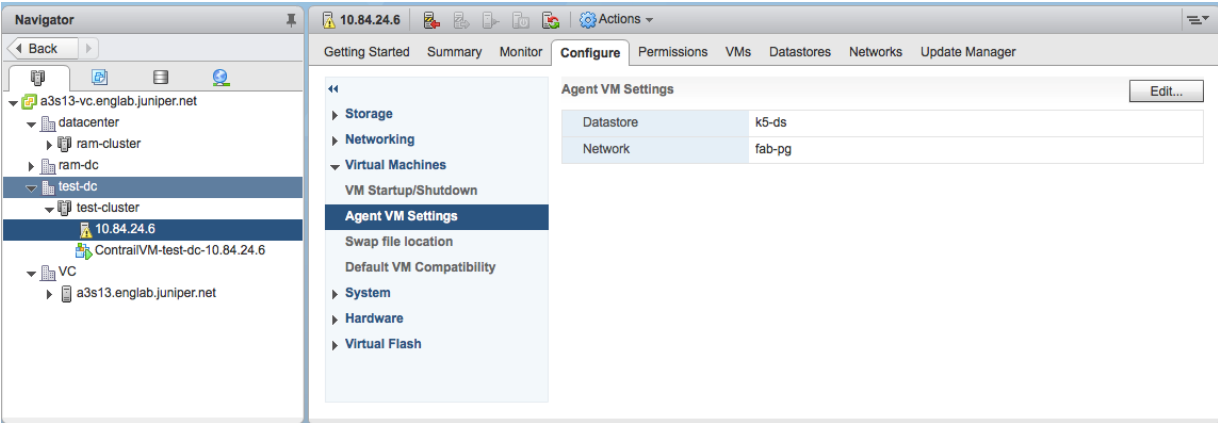


Figure 27: ESX Agencies



- 2. Add host.
 - a. Add ESXi host to the cluster.
 - b. Configure **Agent VM Settings** for the ESXI host.

Figure 28: 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 152](#).
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 155
Managing Networks From Contrail Command and VMware vCenter User Interfaces

Configuring Underlay Network for ContrailVM

IN THIS SECTION

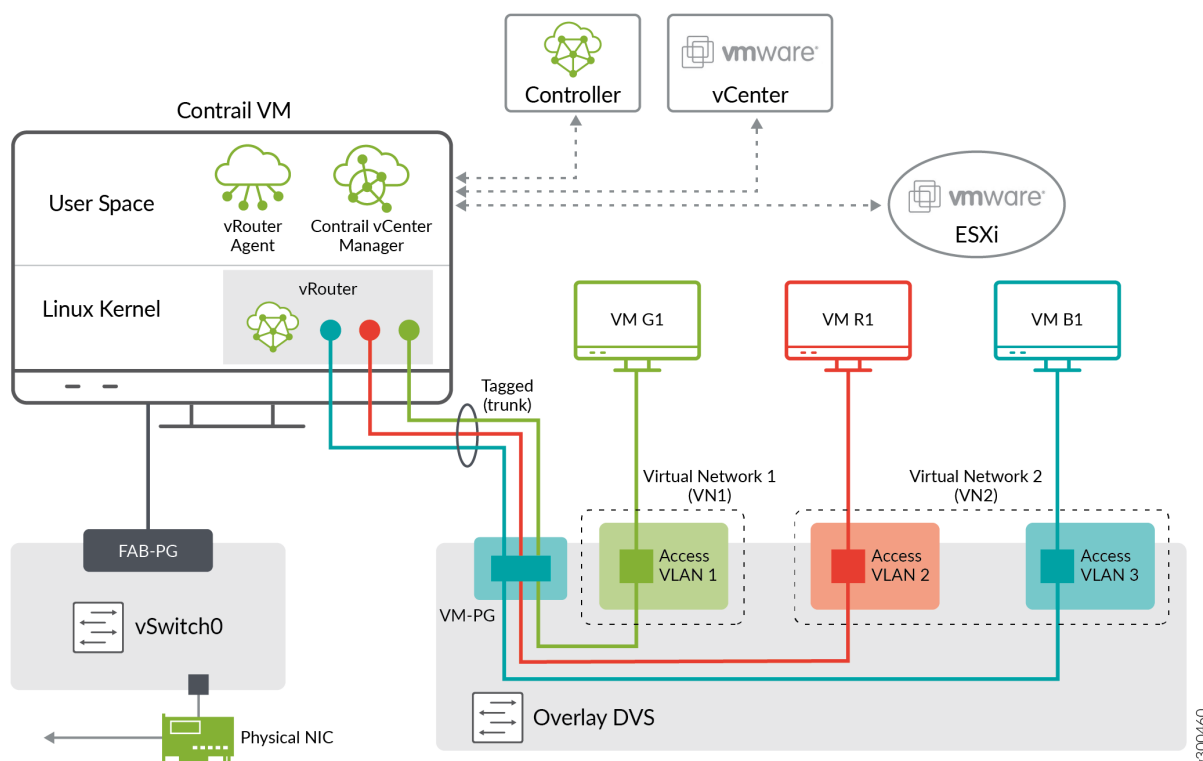
- [Standard Switch Setup | 155](#)
- [Distributed Switch Setup | 157](#)
- [PCI Pass-Through Setup | 159](#)
- [SR-IOV Setup | 162](#)

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 29 on page 156](#).

Figure 29: 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 30 on page 158](#).

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.


```

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'

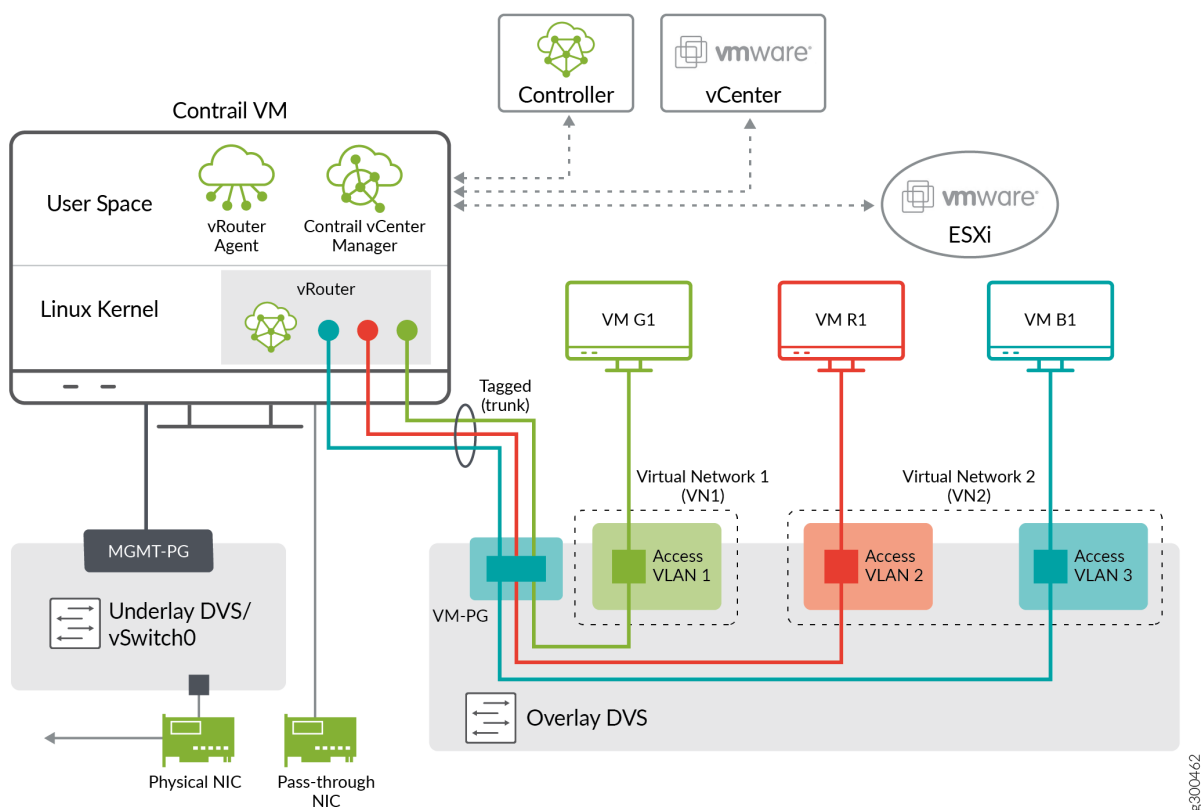
```

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 31 on page 160](#) shows a PCI pass-through setup with a single **control_data** interface.

Figure 31: PCI Pass-Through with Single Control Data Interface



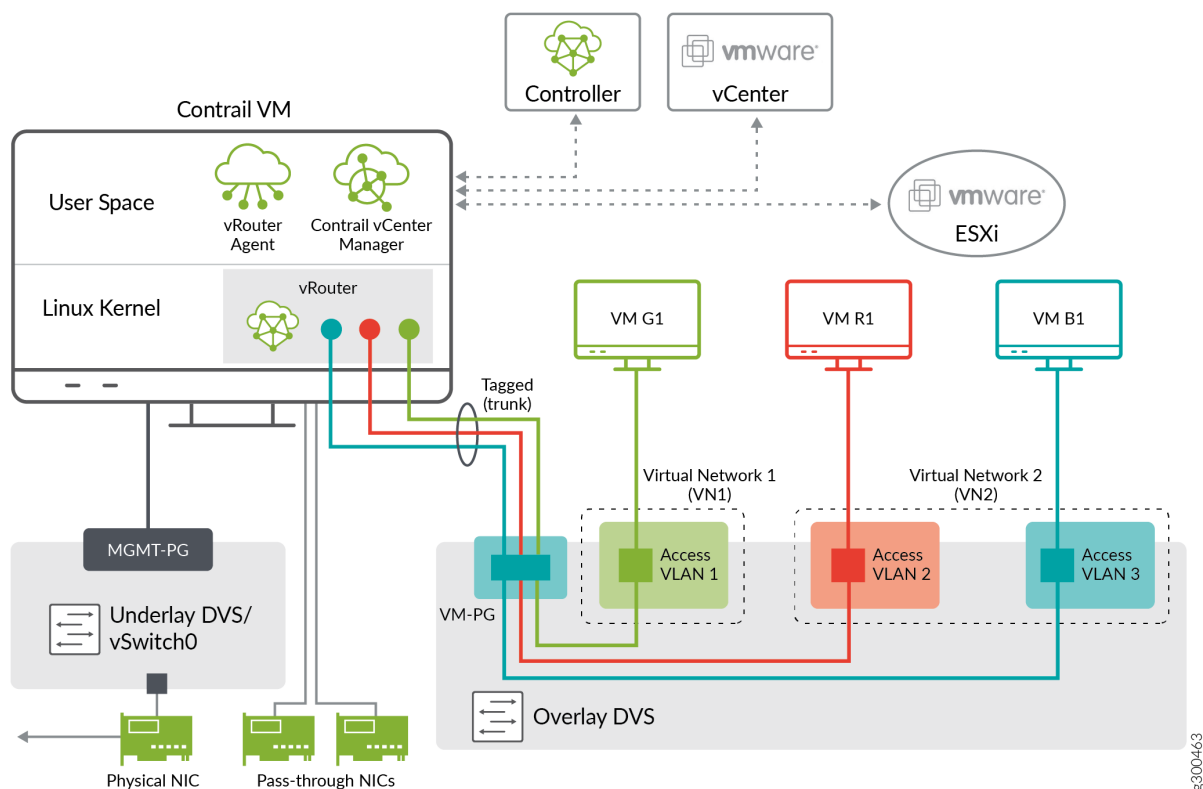
When setting up the ContrailVM with pass-through interfaces, upon provisioning ESXi hosts in the installation process, the PCI pass-through interfaces are exposed as Ethernet interfaces in the ContrailVM, and are identified in the **control_data** device field.

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 32 on page 161 shows a PCI pass-through setup with a bond_control data interface, which has multiple pass-through NICs.

Figure 32: 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:
```

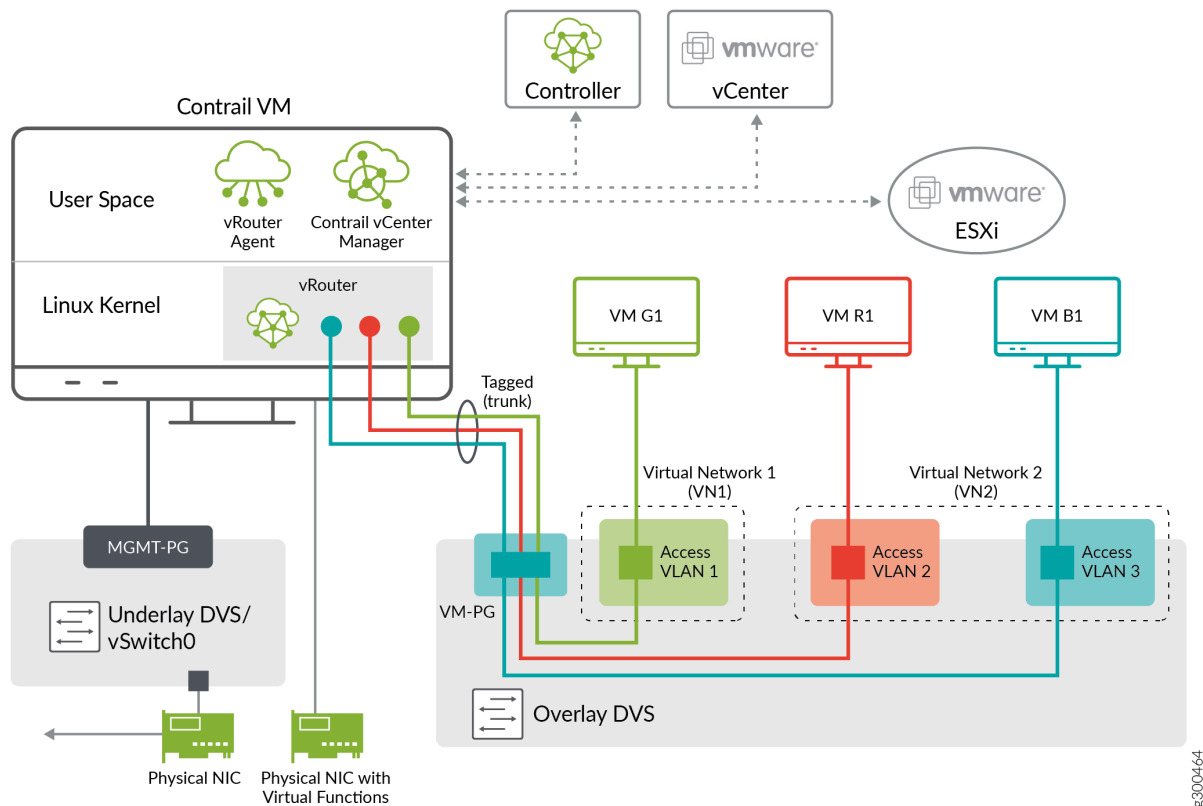
- '0000:04:00.0'
- '0000:04:00.1'

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 33 on page 162](#).

Figure 33: 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.

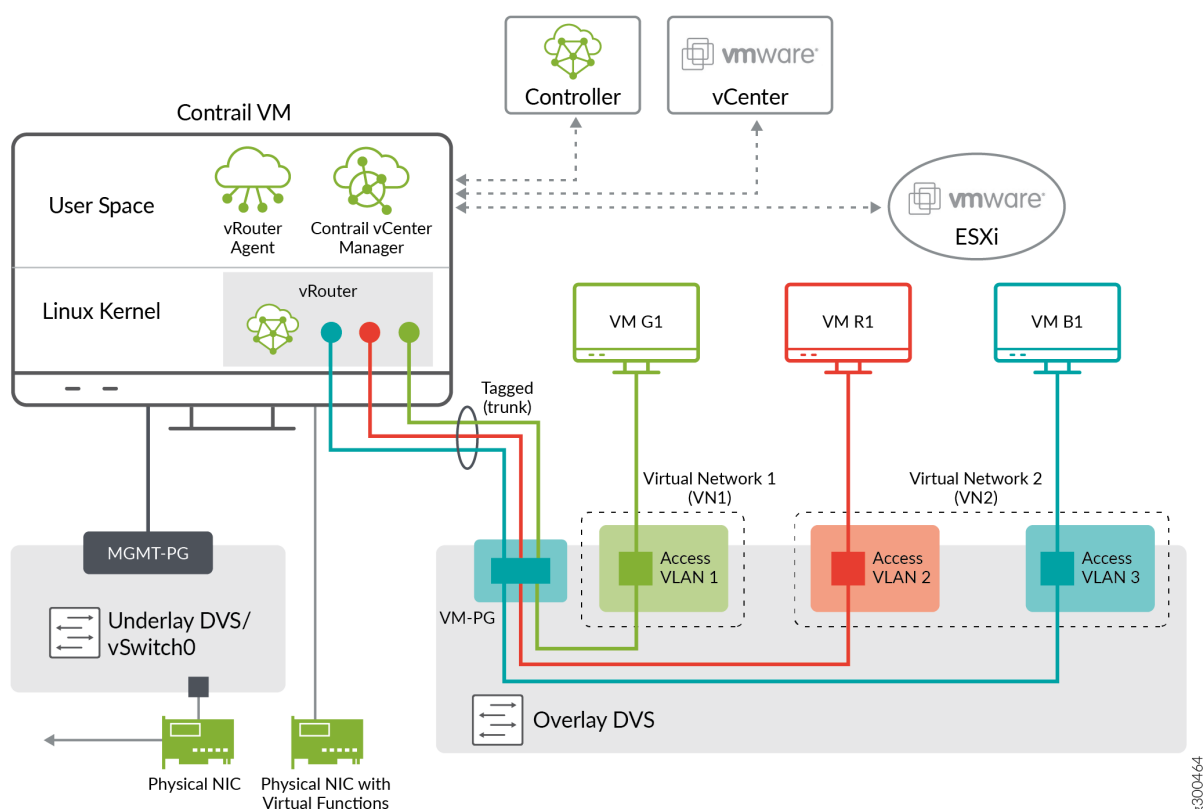
To set up the ContrailVM with SR-IOV interfaces, all configurations used for the standard switch setup are also used for the pass-through setup, providing management connectivity to the ContrailVM.

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 34 on page 163 shows a SR-IOV setup with a single **control_data** interface.

Figure 34: 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:

```

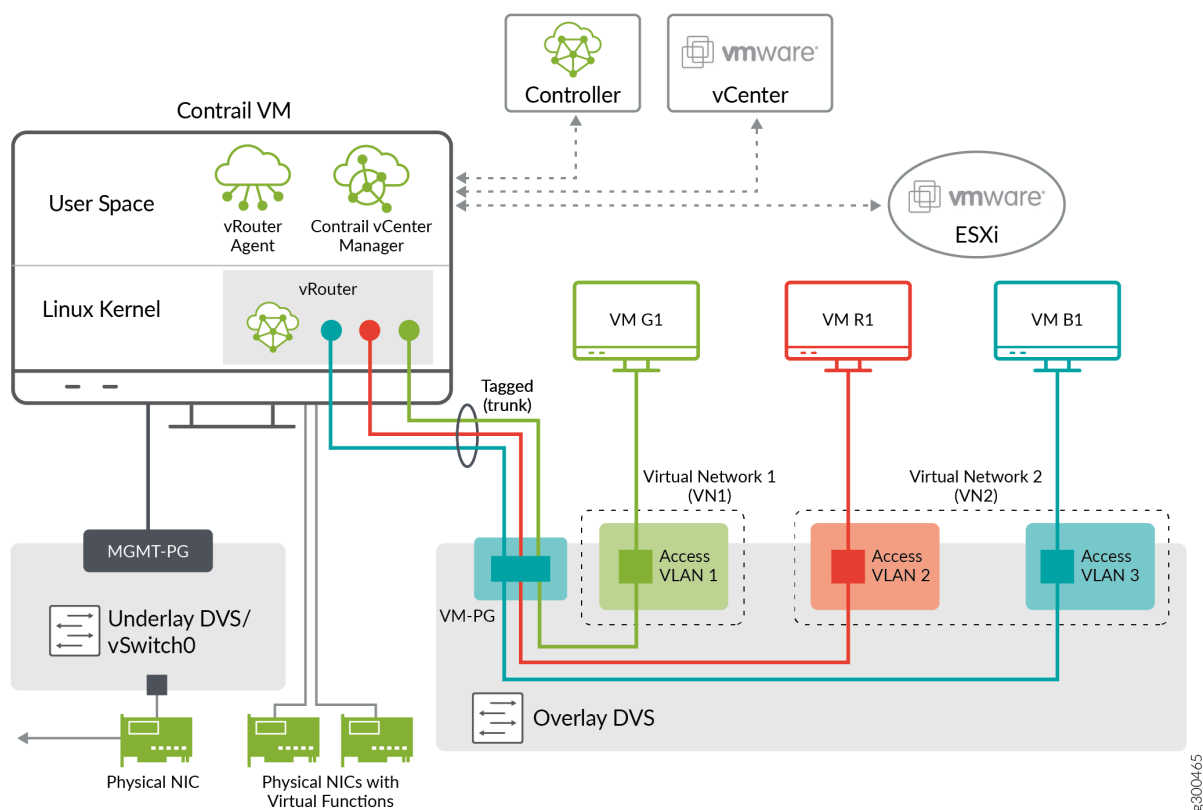
esxihosts:
- name: <esxi_host>
  username: <username>
  password: <password>
  datastore: <datastore>
  vcenter_server: <server>
  datacenter: <datacenter>
  cluster: <cluster>
  contrail_vm:

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

```

[Figure 35 on page 165](#) shows an SR-IOV configuration with a bond **control_data** interface, which has multiple SR-IOV NICs.

Figure 35: 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

- [Accessing vRO Control Center | 167](#)
- [Installing vRO Plugin | 170](#)
- [Accessing vRO Desktop Client | 172](#)
- [Connecting to vRO using the Desktop Client | 172](#)

- Connecting to Contrail Controller | 173
- Deploying Contrail vRO Plugin | 176

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:

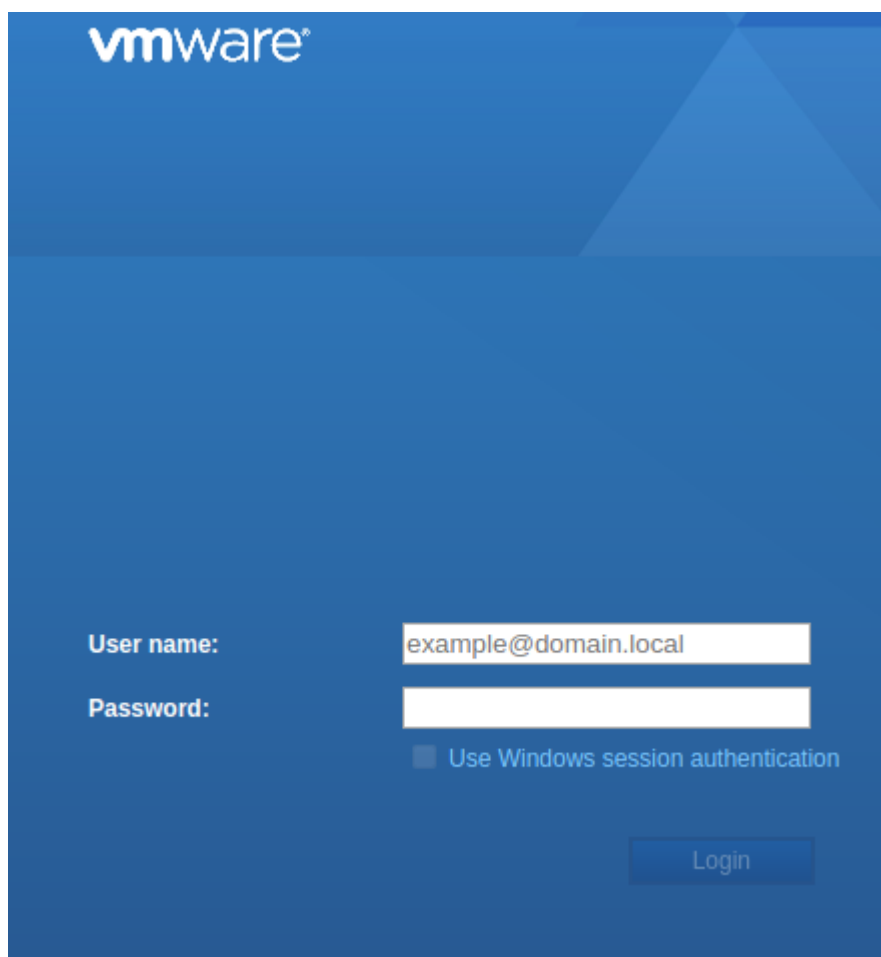
1. To access vRO Control Center through a Web browser, navigate to the <https://{vRO}:8283/vco-controlcenter> URL.

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 36: vCenter SSO service page

The image shows the vCenter SSO service page login interface. It has a blue background with the VMware logo in the top left corner. Below the logo, there are two input fields: 'User name:' with the text 'example@domain.local' and 'Password:' which is empty. Below the password field is a checkbox labeled 'Use Windows session authentication'. At the bottom right, there is a 'Login' button.

vmware®

User name: example@domain.local

Password:

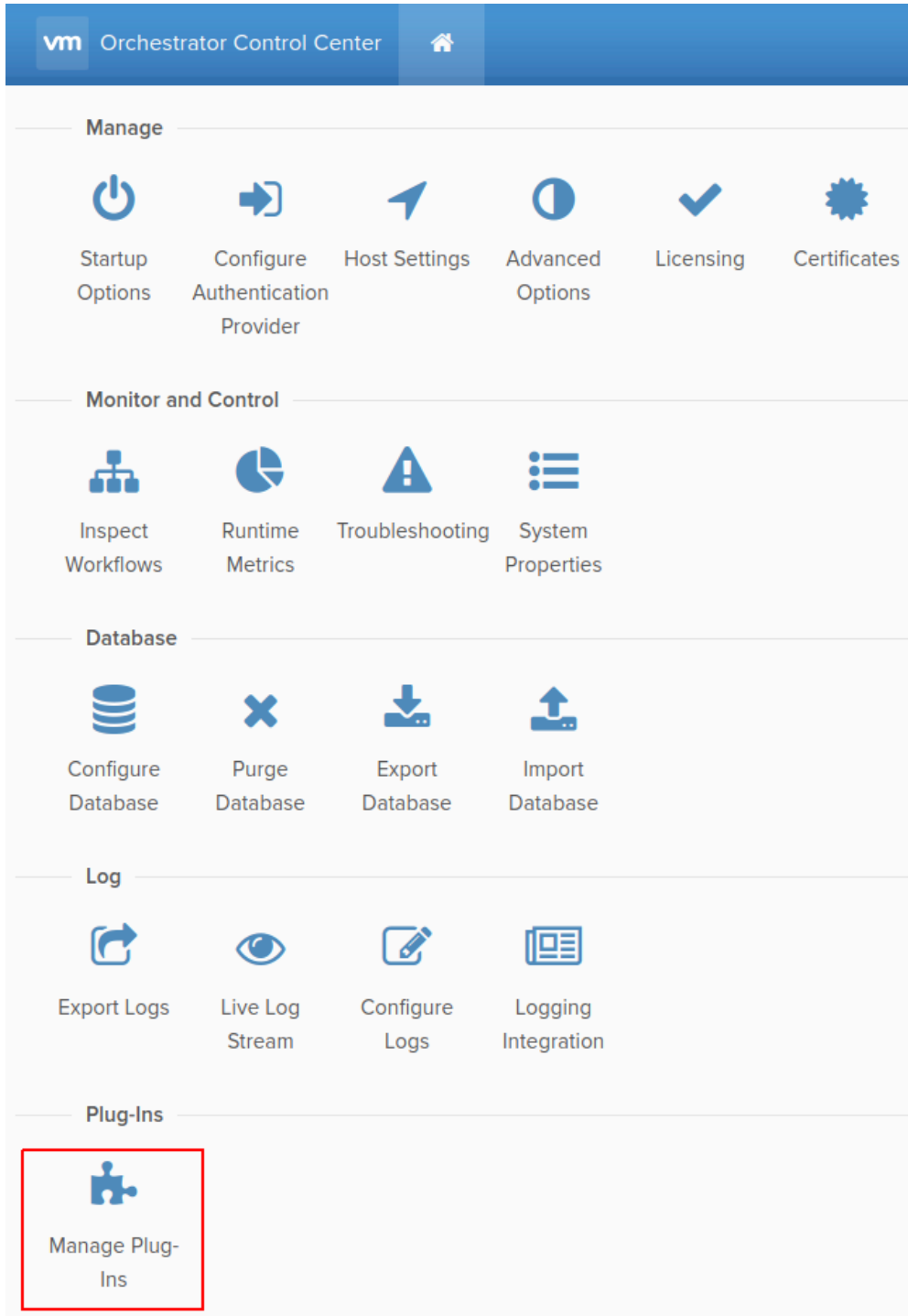
☐ Use Windows session authentication

Login

2. On the vCenter SSO service page, enter the **User name** and **Password** in the respective fields and click **Login**. See [Figure 36 on page 168](#).

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

Figure 37: Orchestrator Control Center



Installing vRO Plugin

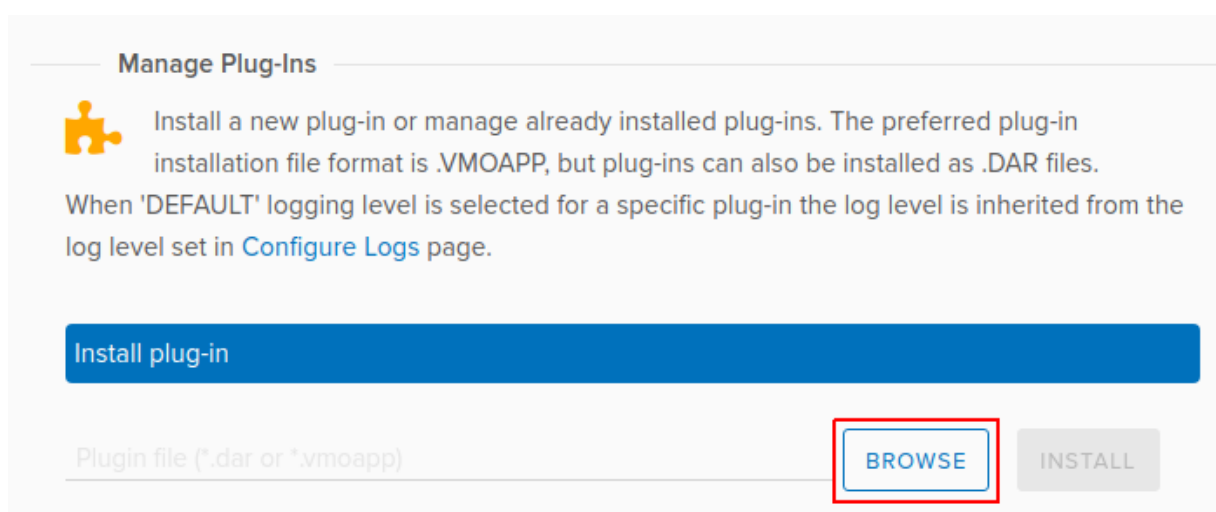
Perform the following steps to install the vRO plugin:

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 38: Manage Plug-Ins page



Manage Plug-Ins

 Install a new plug-in or manage already installed plug-ins. The preferred plug-in installation file format is .VMOAPP, but plug-ins can also be installed as .DAR files. When 'DEFAULT' logging level is selected for a specific plug-in the log level is inherited from the log level set in [Configure Logs](#) page.

Install plug-in

Plugin file (*.dar or *.vmoapp) BROWSE INSTALL

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 39: EULA page

Contrail 1.0.192
Contrail plug-in for vRealize Orchestrator

EULA:

PUT YOUR LICENSE HERE

Accept EULA ☒

CANCEL INSTALL

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 40: Getting Started with vRealize Orchestrator



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

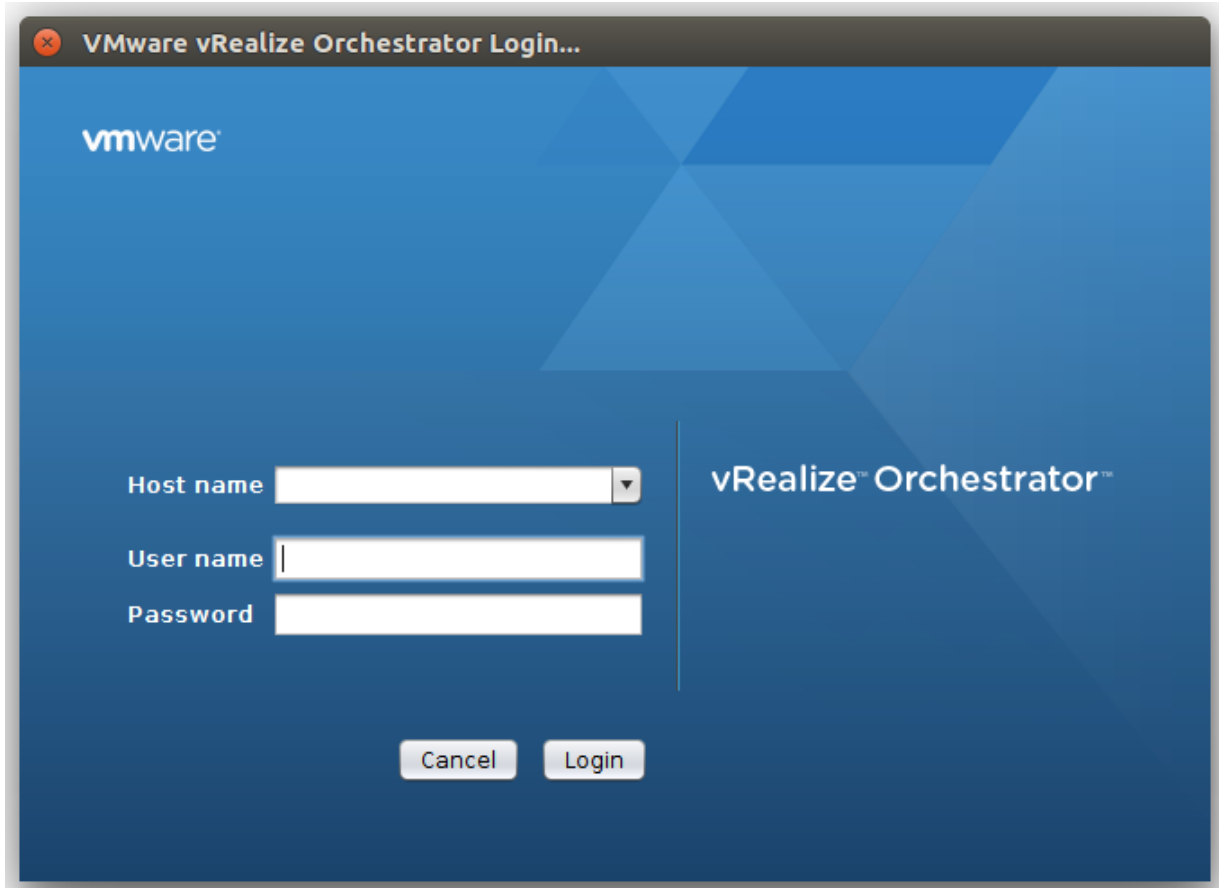
Connecting to vRO using the Desktop Client

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 41: 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.

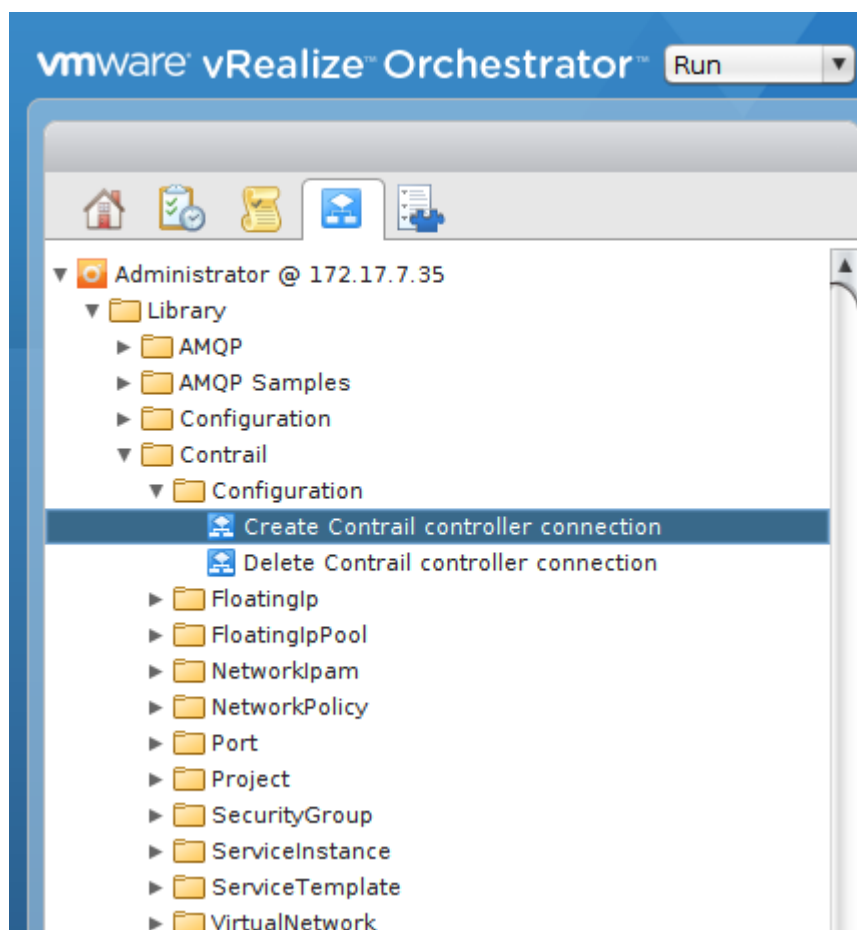
3. Click **Login** to connect to the vRO server. See [Figure 41 on page 173](#).

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 42 on page 174](#).
2. Select **Create Contrail controller connection**.

Figure 42: 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 43: Controller Tab

1 Controller
2 Credentials
3 Tenant

* Connection name
Controller

* Controller host

* Controller port
8082

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 44: Credentials Tab

✓ 1 Controller
✓ 2 Credentials
3 Tenant

User name

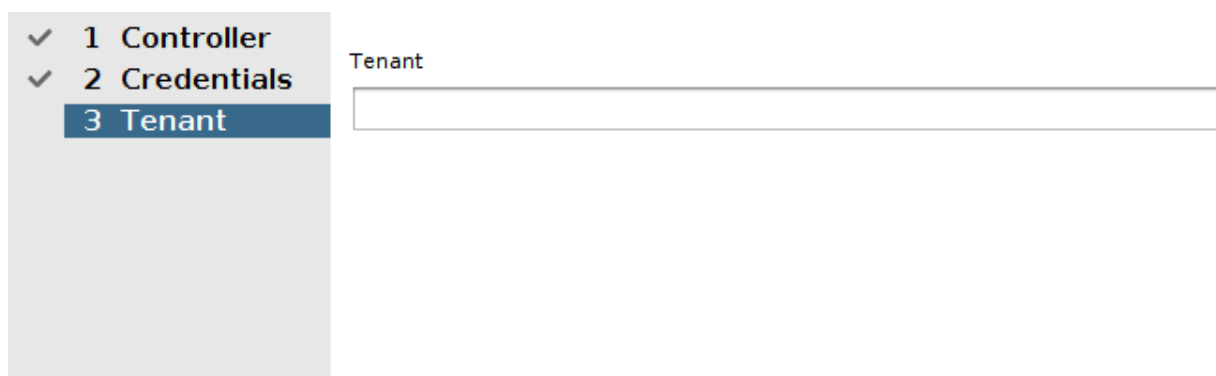
User password

Authentication server

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

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

Figure 45: Tenant Tab



✓ 1 Controller

✓ 2 Credentials

3 Tenant

Tenant

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*

Using Contrail with Red Hat OpenStack

IN THIS CHAPTER

- Understanding Red Hat OpenStack Platform Director | 177
- Setting Up the Infrastructure | 182
- Setting Up the Undercloud | 191
- Setting Up the Overcloud | 194

Understanding Red Hat OpenStack Platform Director

IN THIS SECTION

- Red Hat OpenStack Platform Director | 177
- Contrail Roles | 178
- Undercloud Requirements | 179
- Overcloud Requirements | 179
- Networking Requirements | 180
- Compatibility Matrix | 181
- Installation Summary | 181

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, Ironi, Neutron, Heat, Glance, and Ceilometer to deploy OpenStack on bare metal hardware.

- Nova and Ironi are used in the undercloud to manage the bare metal instances that comprise the infrastructure for the overcloud.
- Neutron is used to provide a networking environment in which to deploy the overcloud.
- Glance stores machine images.
- Ceilometer collects metrics about the overcloud.

For more information about OSPd architecture, see [OSPd documentation](#).

Contrail Roles

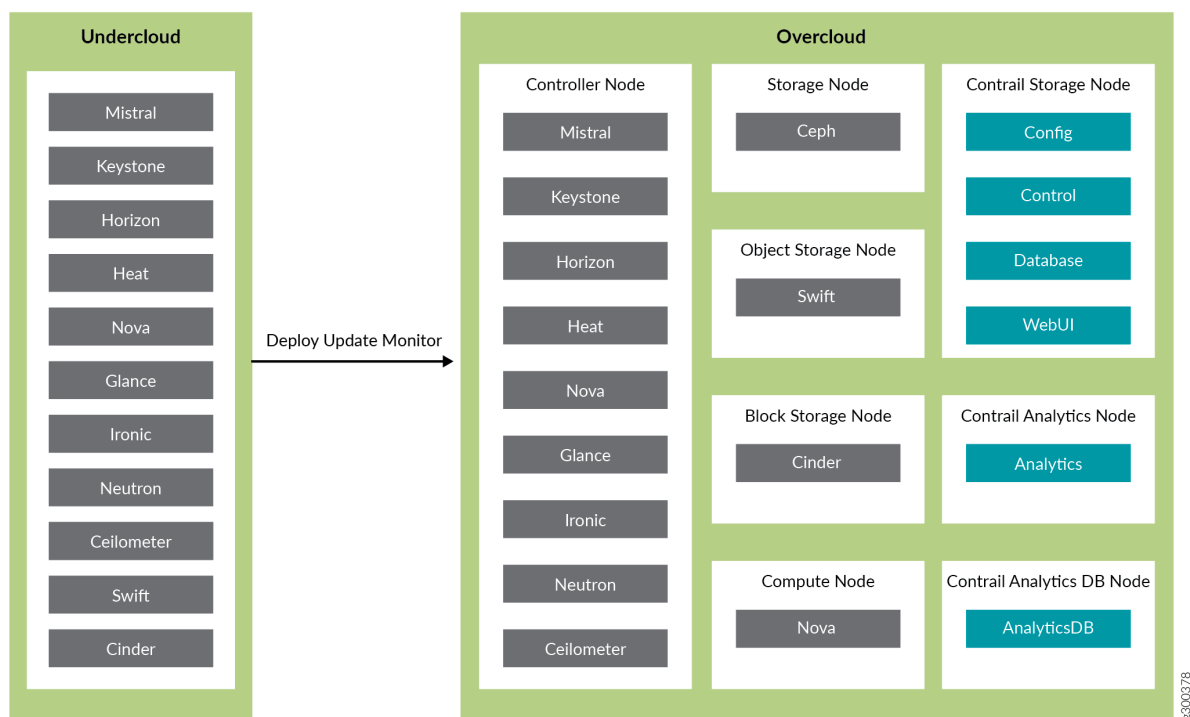
OSPd supports composable roles, which are groups of services that you define through Heat templates. Composable roles allow you to integrate Contrail into the overcloud environment.

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

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

[Figure 46 on page 179](#) shows the relationship and components of an undercloud and overcloud architecture for Contrail.

Figure 46: 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 6: 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 \(Example\) | 182](#)
- [Configure the External Physical Switch | 184](#)
- [Configure KVM Hosts | 185](#)
- [Create the Overcloud VM Definitions on the Overcloud KVM Hosts | 187](#)
- [Create the Undercloud VM Definition on the Undercloud KVM Host | 189](#)

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 7 on page 182](#)).

Table 7: Control Plane Infrastructure

KVM Host	Virtual Machines
KVM1	undercloud
KVM2	OpenStack Controller 1, Contrail Contoller 1
KVM3	OpenStack Controller 2, Contrail Contoller 2

Table 7: Control Plane Infrastructure (continued)

KVM Host	Virtual Machines
KVM4	OpenStack Controller 3, Contrail Controller 3

Figure 47 on page 183 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 47: Physical View

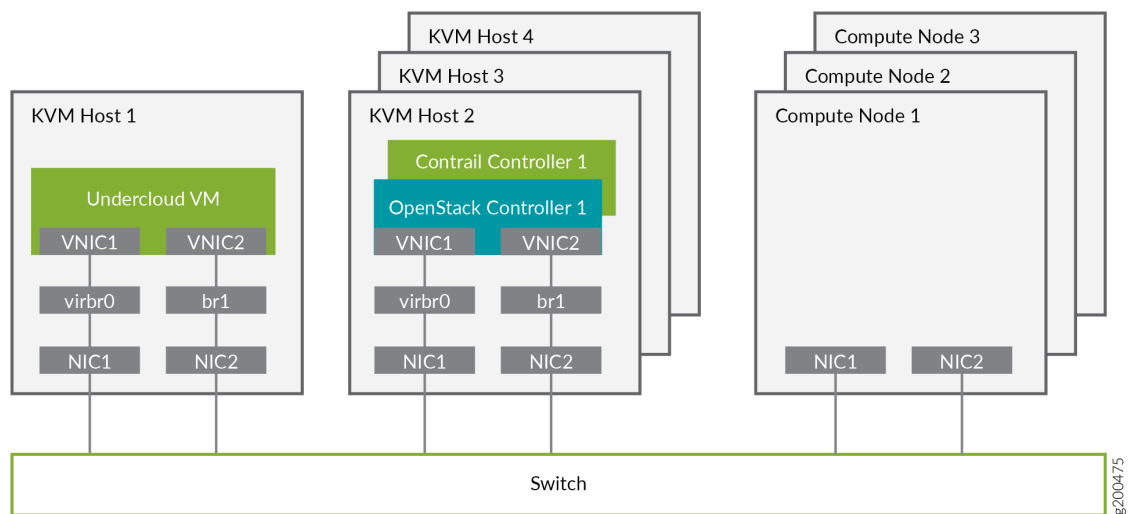
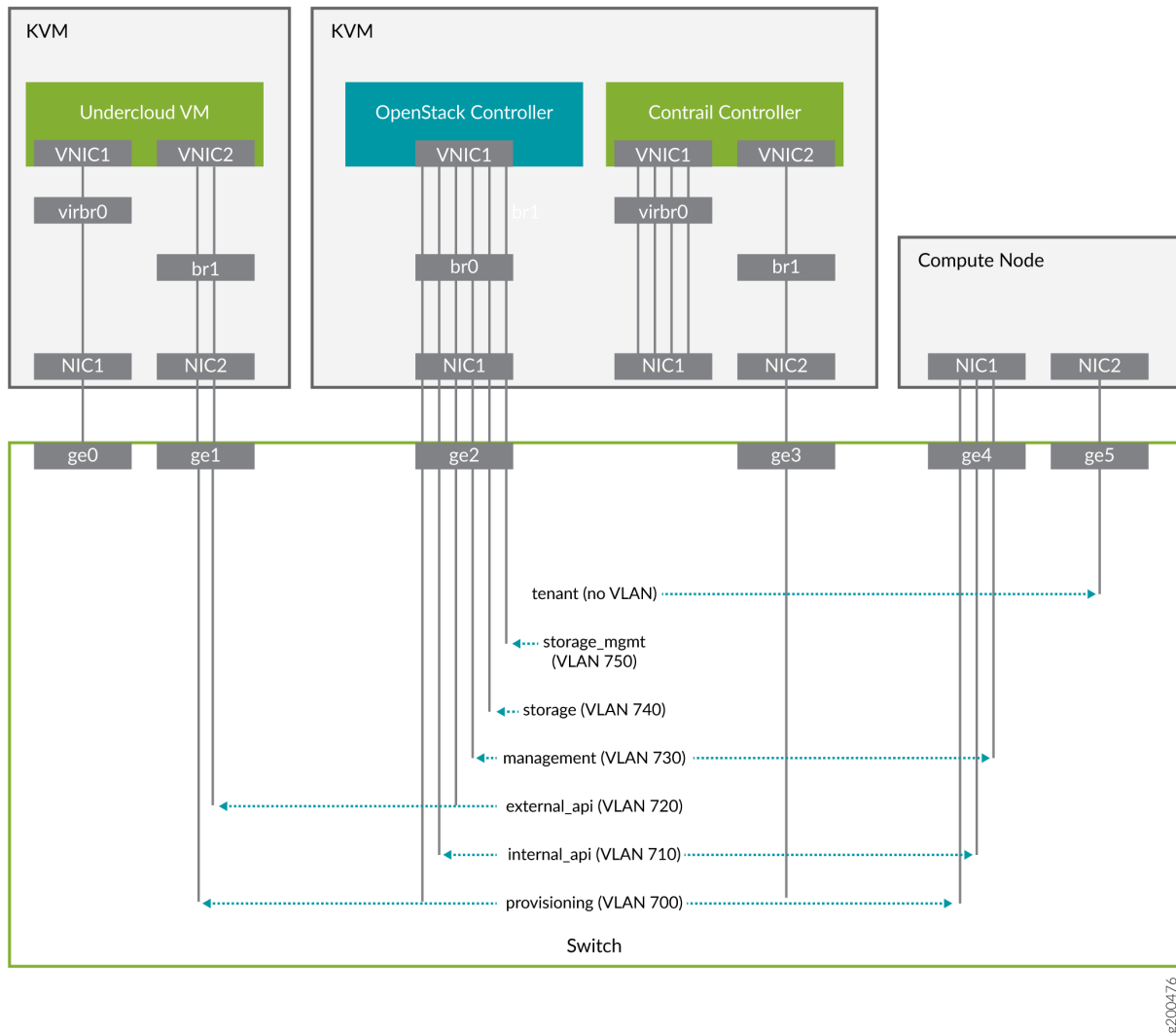


Figure 48 on page 184 shows the logical view of the connectivity where VLANs are used to provide further network separation for the different OpenStack network types.

Figure 48: 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 8: External Physical Switch Port and VLAN Configuration

Port	Trunked VLAN	Native VLAN
ge0	-	-
ge1	700, 720	-

Table 8: External Physical Switch Port and VLAN Configuration (*continued*)

Port	Trunked VLAN	Native VLAN
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 9: vSwitch Configuration

Bridge	Trunked VLAN	Native VLAN
br0	710, 720, 730 740, 750	700

Table 9: vSwitch Configuration (*continued*)

Bridge	Trunked VLAN	Native VLAN
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 | 191](#)
- [Perform Post-Install Configuration | 193](#)

Install the Undercloud

Use this example procedure to install the undercloud.

1. Log in to the undercloud VM from the undercloud KVM host.

```
ssh ${undercloud_ip}
```

2. Configure the hostname.


```
undercloud_name=`hostname -s`
undercloud_suffix=`hostname -d`
hostnamectl set-hostname ${undercloud_name}.${undercloud_suffix}
hostnamectl set-hostname --transient ${undercloud_name}.${undercloud_suffix}
```

3. Add the hostname to the `/etc/hosts` file. The following example assumes the management interface is `eth0`.

```
undercloud_ip=`ip addr sh dev eth0 | grep "inet " | awk '{print $2}' | awk -F"/" '{print $1}'`
echo ${undercloud_ip} ${undercloud_name}.${undercloud_suffix} ${undercloud_name}
>> /etc/hosts
```

4. Set up the repositories.

- CentOS

```
tripleo_repos=`python -c 'import requests;r =
requests.get("https://trunk.rdoproject.org/centos7-queens/current"); print r.text
' | grep python2-tripleo-repos|awk -F"href=\"\" " '{print $2}' | awk -F"\" \" " '{print
$1}'`
yum install -y
https://trunk.rdoproject.org/centos7-queens/current/${tripleo_repos}
tripleo-repos -b queens current
```

- RHEL

```
#Register with Satellite (can be done with CDN as well)
satellite_fqdn=device.example.net
act_key=xxx
org=example
yum localinstall -y
http://${satellite_fqdn}/pub/katello-ca-consumer-latest.noarch.rpm
subscription-manager register --activationkey=${act_key} --org=${org}
```

5. Install the Tripleo client.

```
yum install -y python-tripleoclient tmux
```

6. Copy the undercloud configuration file sample and modify the configuration as required. See [Red Hat documentation](#) for information on how to modify that file.

```
su - stack
cp /usr/share/instack-undercloud/undercloud.conf.sample ~/undercloud.conf
vi ~/undercloud.conf
```

7. Install the undercloud.

```
openstack undercloud install
source stackrc
```

Perform Post-Install Configuration

1. Configure a forwarding path between the provisioning network and the external network:

```
sudo iptables -A FORWARD -i br-ctlplane -o eth0 -j ACCEPT
sudo iptables -A FORWARD -i eth0 -o br-ctlplane -m state --state
RELATED,ESTABLISHED -j ACCEPT
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

2. Add the external API interface:

```
sudo ip link add name vlan720 link br-ctlplane type vlan id 720
sudo ip addr add 10.2.0.254/24 dev vlan720
sudo ip link set dev vlan720 up
```

3. Add the **stack** user to the docker group:

```
newgrp docker
exit
su - stack
source stackrc
```

Setting Up the Overcloud

IN THIS SECTION

- [Configuring the Overcloud | 194](#)
- [Customizing the Contrail Service with Templates \(contrail-services.yaml\) | 200](#)
- [Customizing the Contrail Network with Templates | 201](#)
- [Installing Overcloud | 236](#)

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


```

--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 -xzf 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 | 201](#)
- [Roles Configuration \(roles_data_contrail_aio.yaml\) | 202](#)
- [Network Parameter Configuration \(contrail-net.yaml\) | 205](#)
- [Network Interface Configuration \(*-NIC-*.yaml\) | 206](#)
- [Advanced vRouter Kernel Mode Configuration | 217](#)
- [Advanced vRouter DPDK Mode Configuration | 219](#)
- [Advanced vRouter SRIOV + Kernel Mode Configuration | 222](#)
- [Advanced vRouter SRIOV + DPDK Mode Configuration | 225](#)
- [Advanced Scenarios | 228](#)

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.


```

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

- [OpenStack Controller | 206](#)
- [Contrail Controller | 210](#)
- [Compute Node | 213](#)

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

- [VLAN | 217](#)
- [Bond | 218](#)
- [Bond + VLAN | 218](#)

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

- [Standard | 220](#)
- [VLAN | 220](#)
- [Bond | 221](#)
- [Bond + VLAN | 221](#)

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

- [VLAN | 223](#)
- [Bond | 223](#)
- [Bond + VLAN | 224](#)

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

- [Standard | 226](#)
- [VLAN | 226](#)
- [Bond | 227](#)
- [Bond + VLAN | 227](#)

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

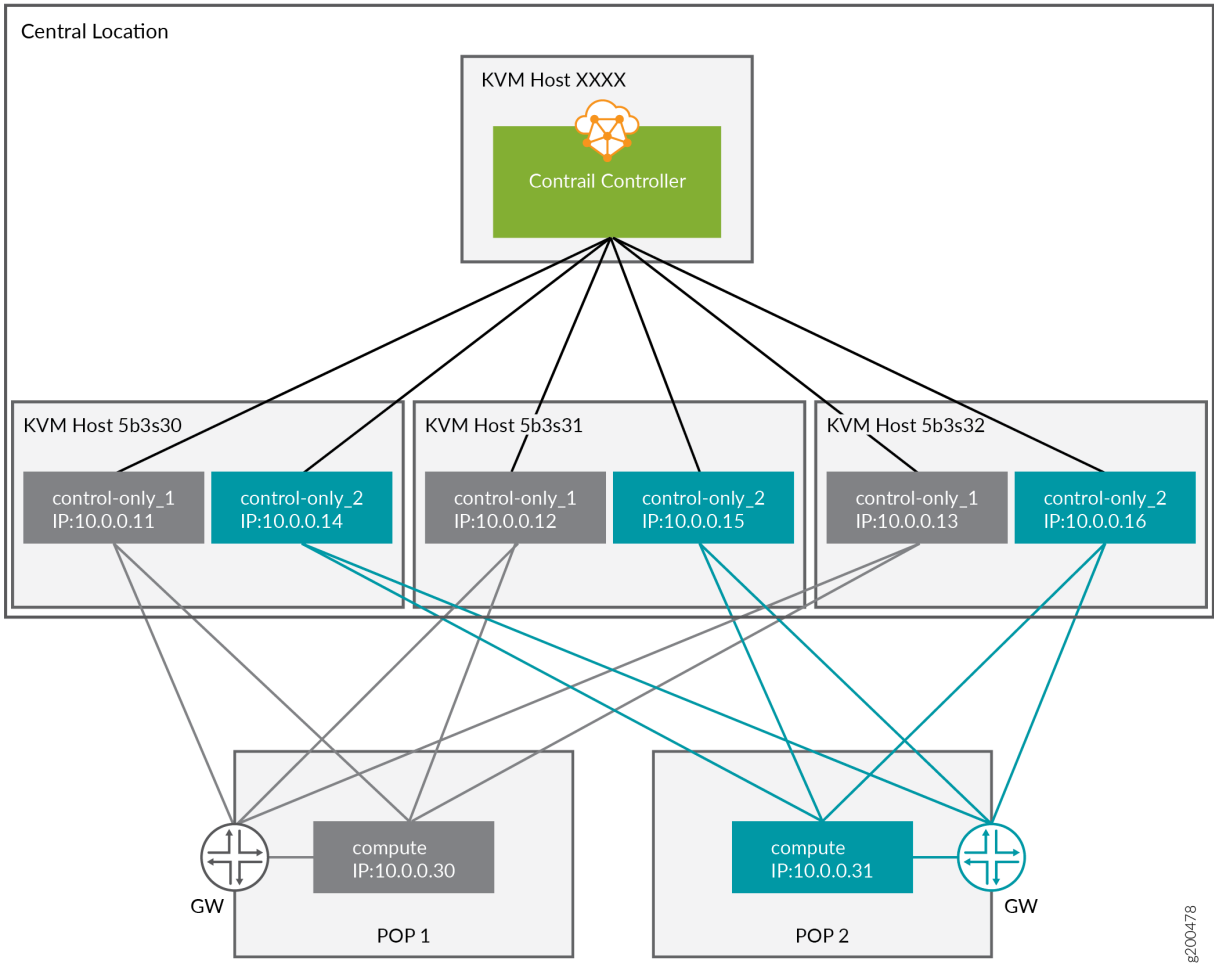
Remote Compute

Remote Compute extends the data plane to remote locations (POP) whilst keeping the control plane central. Each POP will have its own set of Contrail control services, which are running in the central location. The difficulty is to ensure that the compute nodes of a given POP connect to the Control nodes assigned to that POP. The Control nodes must have predictable IP addresses and the compute nodes have to know these IP addresses. In order to achieve that the following methods are used:

- Custom Roles
- Static IP assignment
- Precise Node placement
- Per Node hieradata

Each overcloud node has a unique DMI UUID. This UUID is known on the undercloud node as well as on the overcloud node. Hence, this UUID can be used for mapping node specific information. For each POP, a Control role and a Compute role has to be created.

Overview



Mapping Table

Table 11: 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

Table 11: Mapping Table (continued)

Nova Name	Ironic Name	UUID	KVM	IP Address	POP
overcloud -contrailcontrolonly -1	control-only-2- 5b3s30	Ironic UUID: d26abdeb-d514-4a37-a7fb-2cd2511c351f DMI UUID: 14639A66-D62C-4408-82EE-FDDC4E509687	5b3s30	10.0.0.14	POP2
overcloud -contrailcontrolonly -2	control-only-1- 5b3s31	Ironic UUID: 91dd9fa9-e8eb-4b51-8b5e-bbaffb6640e4 DMI UUID: 28AB0B57-D612-431E-B177-1C578AE0FEA4	5b3s31	10.0.0.12	POP1
overcloud -contrailcontrolonly -3	control-only-2- 5b3s31	Ironic UUID: 09fa57b8-580f-42ec-bf10-a19573521ed4 DMI UUID: 09BEC8CB-77E9-42A6-AFF4-6D4880FD87D0	5b3s31	10.0.0.15	POP2
overcloud -contrailcontrolonly -4	control-only-1- 5b3s32	Ironic UUID: 4766799-24c8-4e3b-af54-353f2b796ca4 DMI UUID: 3993957A-ECBF-4520-9F49-0AF6EE1667A7	5b3s32	10.0.0.13	POP1
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

Provisioning Red Hat OpenShift Container Platform Clusters Using Ansible Deployer

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 | 238](#)

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

Contrail Release 5.1 can be installed together with a standalone Red Hat OpenShift Container Platform 3.11 cluster using Contrail OpenShift deployer.

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.

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-5.1.X.tgz**) installer from the Juniper software download site: <https://www.juniper.net/support/downloads/?p=contrail#sw>
- b. Copy the install package to the node from where Ansible is deployed. Ensure that the node has password-free access to the OpenShift master and slave nodes.


```
scp contrail-openshift-deployer-5.1.X.tgz openshift-ansible-node:/root/
```

- c. Log in to the Ansible node and untar the **contrail-openshift-deployer-5.1.X.tgz** package.

```
tar -xzvf contrail-openshift-deployer-5.1.X.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.

```
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](#) 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)
```

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 [Sample inventory/ose-install File on page 242](#).

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=memory_availability,node_kernel_version,diskspace_availability,ceph_availability,nfs_availability,python_version,os_version,storage_availability

# 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_aceptschema2=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 liveness and readiness checks. Modify the liveness 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

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

Using Contrail and AppFormix with Kolla/Ocata OpenStack

IN THIS CHAPTER

- [Contrail and AppFormix Deployment Requirements | 249](#)
- [Preparing for the Installation | 250](#)
- [Run the Playbooks | 253](#)
- [Accessing Contrail in AppFormix Management Infrastructure in UI | 254](#)
- [Notes and Caveats | 255](#)
- [Example Instances.yml for Contrail and AppFormix OpenStack Deployment | 255](#)
- [Installing AppFormix for OpenStack | 259](#)
- [Installing AppFormix for OpenStack in HA | 264](#)

Contrail and AppFormix Deployment Requirements

Contrail supports 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 68](#)
- [Installing AppFormix for OpenStack on page 259](#)
- [Installing AppFormix for OpenStack in HA on page 264](#)

The following software and hardware requirements apply to the combined Contrail and AppFormix deployment.

Software Requirements

- Contrail Release 5.1.x Targets: Centos 7.5 with kernel 3.10.0-862.3.2.el7.x86_64.
- AppFormix Targets: Refer to “Software requirements” in [“Installing AppFormix for OpenStack” on page 259](#).

- 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 39](#) and [“Installing AppFormix for OpenStack” on page 259](#) for specifics about requirements for installation.

RELATED DOCUMENTATION

[Installing Contrail Cluster using Contrail Command and instances.yml | 39](#)

[Installing AppFormix for OpenStack | 259](#)

Preparing for the Installation

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 259](#).

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 68](#).

2. Set-up the base host. See “Set Up the Base Host” in [“Installing Contrail with OpenStack and Kolla Ansible” on page 68](#). 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 68](#).
 - The AppFormix inventory section of the **instances.yaml** file is configured according to guidelines in [“Installing AppFormix for OpenStack” on page 259](#).

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.yml` 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 | 68](#)

[Installing Contrail Cluster using Contrail Command and `instances.yml` | 39](#)

[Installing AppFormix for OpenStack | 259](#)

[Example `Instances.yml` for Contrail and AppFormix OpenStack Deployment | 255](#)

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 68 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.yml -e
orchestrator=openstack playbooks/configure_instances.yml (-vvv for debug)
```

- Install OpenStack:

```
ansible-playbook -i inventory/ -e config_file=/path/instances.yml
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 259](#) 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 | 68](#)

[Installing Contrail Cluster using Contrail Command and instances.yaml | 39](#)

[Installing AppFormix for OpenStack | 259](#)

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 [“Installing AppFormix for OpenStack in HA” on page 264](#).

RELATED DOCUMENTATION

| [Installing AppFormix for OpenStack in HA | 264](#)

Example Instances.yml for Contrail and AppFormix OpenStack Deployment

See [“Installing Contrail with OpenStack and Kolla Ansible” on page 68](#) and [“Installing AppFormix for OpenStack” on page 259](#) for specific inventory file details:

The following items are part of the **all** section in the **instances.yml** 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](#).

RELATED DOCUMENTATION

[Installing Contrail with OpenStack and Kolla Ansible | 68](#)

[Installing Contrail Cluster using Contrail Command and instances.yml | 39](#)

[Installing AppFormix for OpenStack | 259](#)

Installing AppFormix for OpenStack

IN THIS SECTION

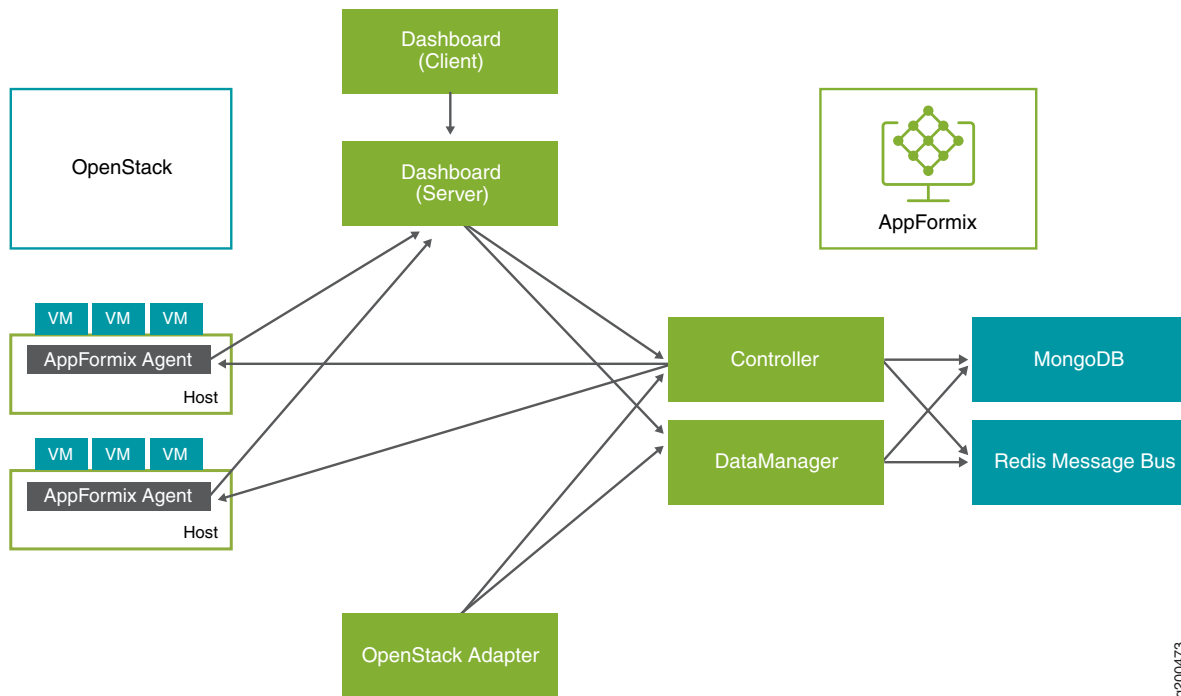
- [Architecture | 259](#)
- [Installing AppFormix | 260](#)
- [Removing a Node from AppFormix | 263](#)

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 *AppFormix General Requirements* before reading this section.

Architecture

AppFormix provides resource control and visibility for hosts, containers, and virtual machines in your cloud infrastructure. [Figure 49 on page 260](#) shows the AppFormix architecture with OpenStack.

Figure 49: AppFormix Architecture with OpenStack



g200473

- 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 httpplib2          #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. Add the username and password for credentials-based login.

```
Constraints for creating Username:
- Should not be more than 30 characters
- Can have anything mentioned below:
  1. alphanumeric character
  2. '_' or '.'
```

```
Constraints for creating Password:
- 8 characters length or more
- 1 digit or more
- 1 uppercase letter or more
- 1 lowercase letter or more
```

```
export APPFORMIX_USERNAME=<username>
export APPFORMIX_PASSWORD=<password>
```

10. 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

AppFormix General Requirements

Installing AppFormix for an OpenStack Cluster

[Installing AppFormix for OpenStack in HA | 264](#)

AppFormix Agent Requirements

Platform Dependencies

Installing AppFormix for OpenStack in HA

IN THIS SECTION

- [HA Design Overview | 264](#)
- [Requirements | 265](#)
- [Installing AppFormix for High Availability | 266](#)

HA Design Overview

AppFormix Platform can be deployed to multiple hosts for high availability. 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

Each host, on which AppFormix Platform is installed, has the following requirements.

Hardware Requirements

- CPU: 8 cores (virtual or physical)
- Memory: 16 GB
- Storage: 100 GB (recommended)

Software Requirements

- Docker 17.03.1-ce, installed on the Platform Host(s).
- Python "docker" package 3.7.1, installed on the Platform Host(s).
- Ansible 2.3.0 - 2.7.6, installed on a host that has SSH access to Platform Hosts and compute hosts to which AppFormix will be deployed.
- httpLib2 must be installed on the host where Ansible is executed.

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.
- The installer node needs to download the following packages from <https://www.juniper.net/support/downloads/?p=appformix#sw>.
 - appformix-openstack-images-<version>.tar.gz
 - appformix-platform-images-<version>.tar.gz
 - appformix-dependencies-images-<version>.tar.gz

AppFormix Agent Supported Platforms

AppFormix Agent runs on a host to monitor resource consumption of the host itself and the virtual machines and containers executing on that host.

- Ubuntu 14.04
- Red Hat Enterprise Linux 7.1
- Red Hat Enterprise Linux 6.5, 6.6

- CentOS 7.1
- CentOS 6.5, 6.6

Installing AppFormix for High Availability

To install AppFormix to multiple hosts for high availability:

1. 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 httpplib2
```

For Ansible 2.3:

```
# sudo pip install ansible==2.3 markupsafe httpplib2 cryptography==1.5
```

2. 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
```

3. Install python pip package on the hosts where AppFormix Agents run.

```
# apt-get install -y python-pip
```

4. 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
```

5. 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`.

6. 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
```

7. 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
```

8. Run Ansible with the created inventory file.

```
ansible-playbook -i inventory appformix_openstack.yml
```

9. 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

AppFormix General Requirements

Installing AppFormix for an OpenStack Cluster

AppFormix Agent Requirements

Platform Dependencies

Using Contrail with Juju Charms

IN THIS CHAPTER

- [Installing Contrail by Using Juju Charms | 269](#)

Installing Contrail by Using Juju Charms

IN THIS SECTION

- [Preparing to Deploy Contrail by Using Juju Charms | 270](#)
- [Deploying Contrail Charms | 272](#)
- [Options for Juju Charms | 285](#)

You can deploy Contrail by using Juju Charms. Juju helps you deploy, configure, and efficiently manage applications on private clouds and public clouds. Juju accesses the cloud with the help of a Juju controller. A Charm is a module containing a collection of scripts and metadata and is used with Juju to deploy Contrail.

Contrail supports the following charms:

- `contrail-agent`
- `contrail-analytics`
- `contrail-analyticsdb`
- `contrail-controller`
- `contrail-keystone-auth`
- `contrail-openstack`

These topics describe how to deploy Contrail by using Juju Charms.

Preparing to Deploy Contrail by Using Juju Charms

Follow these steps to prepare for deployment:

1. Install Juju.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install juju
```

2. Configure Juju.

You can add a cloud to Juju, identify clouds supported by Juju, and also manage clouds already added to Juju.

- **Adding a cloud**—Juju recognizes a wide range of cloud types. You can use any one of the following methods to add a cloud to Juju:
 - **Adding a Cloud by Using Interactive Command**

Example: Adding an MAAS cloud to Juju

```
juju add-cloud
```

```
Cloud Types
```

```
maas
manual
openstack
oracle
vsphere
```

```
Select cloud type: maas
```

```
Enter a name for your maas cloud: maas-cloud
```

```
Enter the API endpoint url: http://<ip-address>:<node>/MAAS
```

```
Cloud "maas-cloud" successfully added
```

```
You may bootstrap with 'juju bootstrap maas-cloud'
```

NOTE: Juju 2.x is compatible with MAAS series 1.x and 2.x.

- **Adding a Cloud Manually**

You use a YAML configuration file to add a cloud manually. Enter the following command:

```
juju add-cloud <cloud-name>
juju add-credential <cloud name>
```

For an example, to add the cloud *junmaas*, assuming that the name of the configuration file in the directory is **maas-clouds.yaml**, you run the following command:

```
juju add-cloud junmaas maas-clouds.yaml
```

The following is the format of the YAML configuration file:

```
clouds:
  <cloud_name>:
    type: <type_of_cloud>
    auth-types: [<authentication_types>]
    regions:
      <region-name>:
        endpoint: <http://<ip-address>:<node>/MAAS>
```

NOTE: The **auth-types** for a MAAS cloud type is **oauth1**.

- **Identifying a supported cloud**

Juju recognizes the cloud types given below. You use the **juju clouds** command to list cloud types that are supported by Juju.

```
$ juju clouds
```

Cloud	Regions	Default	Type	Description
aws	15	us-east-1	ec2	Amazon Web Services
aws-china	1	cn-north-1	ec2	Amazon China
aws-gov	1	us-gov-west-1	ec2	Amazon (USA Government)
azure	26	centralus	azure	Microsoft Azure
azure-china	2	chinaeast	azure	Microsoft Azure China
cloudsigma	5	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

- [Deploying Contrail Charms in a Bundle | 272](#)
- [Deploying Juju Charms Manually | 279](#)

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 285](#).

2. (Optional) Check the status of deployment.

You can check the status of the deployment by using the **juju status** command.

3. Enable configuration statements.

Based on your deployment requirements, you can enable the following configuration statements:

- **contrail-agent**

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-agent/>.

- **contrail-analytics**

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-analytics>.

- **contrail-analyticsdb**

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-analyticsdb>.

- **contrail-controller**

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-controller>.

- **contrail-keystone-auth**

For more information, see <https://jaas.ai/u/juniper-os-software/contrail-keystone-auth>.

- **contrail-openstack**

For more information see, <https://jaas.ai/u/juniper-os-software/contrail-openstack>.

Deploying Juju Charms Manually

Before you begin deployment, ensure that you have:

- Installed and configured Juju
- Created a Juju controller
- Ubuntu 16.04 or Ubuntu 18.04 installed

Follow these steps to deploy Juju Charms manually:

1. Create machine instances for OpenStack, compute, and Contrail.

```
juju add-machine --constraints mem=8G cores=2 root-disk=40G --series=xenial
#for openstack machine(s) 0
```

```
juju add-machine --constraints mem=7G cores=4 root-disk=40G --series=xenial
#for compute machine(s) 1,(3)
```

```
juju add-machine --constraints mem=15G cores=2 root-disk=300G --series=xenial
#for contrail machine 2
```

2. Deploy OpenStack services.

You can deploy OpenStack services by using any one of the following methods:

- **By specifying the OpenStack parameters in a YAML file**

The following is an example of a YAML-formatted (**nova-compute-config.yaml**) file.

```
nova-compute:
  openstack-origin: cloud:xenial-ocata
  virt-type: qemu
  enable-resize: True
```



```
enable-live-migration: True
migration-auth-type: ssh
```

Use this command to deploy OpenStack services by using a YAML-formatted file:

```
juju deploy cs:xenial/nova-compute --config ./nova-compute-config.yaml
```

- **By using CLI**

To deploy OpenStack services through the CLI:

```
juju deploy cs:xenial/nova-cloud-controller --config
console-access-protocol=novnc --config openstack-origin=cloud:xenial-ocata
```

- **By using a combination of YAML-formatted file and CLI**

To deploy OpenStack services by using a combination of YAML-formatted file and CLI:

NOTE: Use the `--to <machine number>` command to point to a machine or container where you want the application to be deployed.

```
juju deploy cs:xenial/ntp
juju deploy cs:xenial/rabbitmq-server --to lxd:0
juju deploy cs:xenial/percona-cluster mysql --config
root-password=<root-password> --config max-connections=1500 --to lxd:0
juju deploy cs:xenial/openstack-dashboard --config
openstack-origin=cloud:xenial-ocata --to lxd:0
juju deploy cs:xenial/nova-cloud-controller --config
console-access-protocol=novnc --config openstack-origin=cloud:xenial-ocata
--config network-manager=Neutron --to lxd:0
juju deploy cs:xenial/neutron-api --config
manage-neutron-plugin-legacy-mode=false --config
openstack-origin=cloud:xenial-ocata --config neutron-security-groups=true --to
lxd:0
juju deploy cs:xenial/glance --config openstack-origin=cloud:xenial-ocata --to
lxd:0
juju deploy cs:xenial/keystone --config admin-password=<admin-password> --config
admin-role=admin --config openstack-origin=cloud:xenial-ocata --to lxd:0
```

NOTE: You set OpenStack services on different machines or on different containers to prevent HAProxy conflicts from applications.

3. Deploy and configure nova-compute.

```
juju deploy cs:xenial/nova-compute --config ./nova-compute-config.yaml --to 1
```

NOTE: You can deploy nova-compute to more than one compute machine.

(Optional) To add additional computes:

```
juju add-unit nova-compute --to 3 # Add one more unit
```

4. Deploy and configure Contrail services.

```
juju deploy --series=xenial
$CHARMS_DIRECTORY/contrail-charms/contrail-keystone-auth --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-controller
--config auth-mode=rbac --config cassandra-minimum-diskgb=4 --config
cassandra-jvm-extra-opts="-Xms1g -Xmx2g" --to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-analyticsdb
cassandra-minimum-diskgb=4 --config cassandra-jvm-extra-opts="-Xms1g -Xmx2g"
--to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-analytics
--to 2
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-openstack
juju deploy --series=xenial $CHARMS_DIRECTORY/contrail-charms/contrail-agent
```

5. Enable applications to be available to external traffic:

```
juju expose openstack-dashboard
juju expose nova-cloud-controller
juju expose neutron-api
juju expose glance
juju expose keystone
```

6. Enable contrail-controller and contrail-analytics services to be available to external traffic if you do not use HAProxy.

```
juju expose contrail-controller
juju expose contrail-analytics
```

7. Apply SSL.

You can apply SSL if needed. To use SSL with Contrail services, deploy easy-rsa service and **add-relation** command to create relations to contrail-controller service and contrail-agent services.

```
juju deploy cs:~containers/xenial/easyrsa --to 0
juju add-relation easyrsa contrail-controller
juju add-relation easyrsa contrail-agent
```

8. (Optional) HA configuration.

If you use more than one controller, follow the HA solution given below:

- a. Deploy HAProxy and Keepalived services.

HAProxy charm is deployed on machines with Contrail controllers. HAProxy charm must have **peering_mode** set to **active-active**. If **peering_mode** is set to **active-passive**, HAProxy creates additional listeners on the same ports as other Contrail services. This leads to port conflicts.

Keepalived charm does not require **to** option.

```
juju deploy cs:xenial/haproxy --to <first contrail-controller machine> --config
  peering_mode=active-active
juju add-unit haproxy --to <another contrail-controller machine>
juju deploy cs:~boucherv29/keepalived-19 --config virtual_ip=<vip>
```

- b. Enable HAProxy to be available to external traffic.

```
juju expose haproxy
```

NOTE: If you enable HAProxy to be available to external traffic, do not follow step 6.

- c. Add HAProxy and Keepalived relations.

```
juju add-relation haproxy:juju-info keepalived:juju-info
juju add-relation contrail-analytics:http-services haproxy
juju add-relation contrail-controller:http-services haproxy
juju add-relation contrail-controller:https-services haproxy
```

d. Configure contrail-controller service with VIP.

```
juju set contrail-controller vip=<vip>
```

9. Add other necessary relations.

```
juju add-relation keystone:shared-db mysql:shared-db
juju add-relation glance:shared-db mysql:shared-db
juju add-relation keystone:identity-service glance:identity-service
juju add-relation nova-cloud-controller:image-service glance:image-service
juju add-relation nova-cloud-controller:identity-service keystone:identity-service
juju add-relation nova-cloud-controller:cloud-compute nova-compute:cloud-compute
juju add-relation nova-compute:image-service glance:image-service
juju add-relation nova-compute:amqp rabbitmq-server:amqp
juju add-relation nova-cloud-controller:shared-db mysql:shared-db
juju add-relation nova-cloud-controller:amqp rabbitmq-server:amqp
juju add-relation openstack-dashboard:identity-service keystone

juju add-relation neutron-api:shared-db mysql:shared-db
juju add-relation neutron-api:neutron-api nova-cloud-controller:neutron-api
juju add-relation neutron-api:identity-service keystone:identity-service
juju add-relation neutron-api:amqp rabbitmq-server:amqp

juju add-relation contrail-controller ntp
juju add-relation nova-compute:juju info ntp:juju info

juju add-relation contrail-controller contrail-keystone-auth
juju add-relation contrail-keystone-auth keystone
juju add-relation contrail-controller contrail-analytics
juju add-relation contrail-controller contrail-analyticsdb
juju add-relation contrail-analytics contrail-analyticsdb

juju add-relation contrail-openstack neutron-api
juju add-relation contrail-openstack nova-compute
juju add-relation contrail-openstack contrail-controller
```

```
juju add-relation contrail-agent:juju info nova-compute:juju info  
juju add-relation contrail-agent contrail-controller
```

Options for Juju Charms

Each Contrail Charm has a specific set of options. The options you choose depend on the charms you select. The following tables list the various options you can choose:

- Options for **contrail-agent** Charms.

Table 12: 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.
dppdk	false	Specify DPDK vRouter.
dppdk-driver	uio_pci_generic	Specify DPDK driver for the physical interface.
dppdk-hugepages	70%	Specify the percentage of huge pages reserved for DPDK vRouter and OpenStack instances.
dppdk-coremask	1	Specify the vRouter CPU affinity mask to determine on which CPU the DPDK vRouter will run.
dppdk-main-mempool-size		Specify the main packet pool size.
dppdk-pmd-txd-size		Specify the DPDK PMD Tx Descriptor size.
dppdk-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.

Table 12: Options for contrail-agent (*continued*)

Option	Default option	Description
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 13: 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 14: 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 15: Options for contrail-controller

Option	Default option	Description
control-network		Specify the IP address and network mask of the control network.

Table 15: Options for contrail-controller (*continued*)

Option	Default option	Description
auth-mode	rbac	Specify the authentication mode. Options: rbac , cloud-admin , no-auth . For more information, see https://github.com/Juniper/contrail-controller/wiki/RBAC .
cassandra-minimum-diskgb	20	Specify the minimum disk requirement.
cassandra-jvm-extra-opts		Specify the memory limit.
cloud-admin-role	admin	Specify the role name in keystone for users who have admin-level access.
global-read-only-role		Specify the role name in keystone for users who have read-only access.
vip		Specify if the Contrail API VIP is used for configuring client-side software. If not specified, private IP of the first Contrail API VIP unit will be used.
use-external-rabbitmq	false	To enable the Charm to use the internal RabbitMQ server, set use-external-rabbitmq to false . To use an external AMQP server, set use-external-rabbitmq to true . NOTE: Do not change the flag after deployment.
flow-export-rate	0	Specify how many flow records are exported by vRouter agent to the Contrail Collector when a flow is created or deleted.
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.

Table 15: Options for contrail-controller (*continued*)

Option	Default option	Description
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 16: 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 17: 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.

Table 17: 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.

Upgrading Contrail Software

IN THIS CHAPTER

- [Upgrading Contrail In-Service Software from Releases 3.2 and 4.1 to 5.0.x using Ansible Deployer | 292](#)
- [Upgrading Contrail In-Service Software from Releases 3.2 and 4.1 to 5.0.x using Helm Deployer | 303](#)
- [Upgrading Contrail Command using Backup Restore Procedure | 313](#)

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

IN THIS SECTION

- [Contrail In-Service Software Upgrade \(ISSU\) Overview | 292](#)
- [Prerequisites | 293](#)
- [Preparing the Contrail System for the Ansible Deployer ISSU Procedure | 294](#)
- [Provisioning Control Nodes and Performing Synchronization Steps | 295](#)
- [Transferring the Compute Nodes into the New Cluster | 298](#)
- [Finalizing the Contrail Ansible Deployer ISSU Process | 301](#)

Contrail In-Service Software Upgrade (ISSU) Overview

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

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

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

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

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

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

Prerequisites

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

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

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

Preparing the Contrail System for the Ansible Deployer ISSU Procedure

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

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

NOTE: You must provide keystone authorization information for setup.

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

The detailed steps for deploying the new 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, remove Contrail from the node(s) as follows:
 - Stop the agent container.
 - Restore the physical interface.
 5. Add the required node(s) to **instances.yml** with the roles **vrouter** and **openstack_legacy_compute**.
 6. Run the Contrail ansible deployer to deploy the new vrouter and to configure the old compute service.
 7. All new compute nodes will have:
 - The collector setting pointed to the new Contrail cluster
 - The Control/DNS nodes pointed to the new Contrail cluster
 - The config-api setting in **vnc_api_lib.ini** pointed to the new Contrail cluster
 8. (Optional) Run a test workload on transferred nodes to ensure the new vrouter-agent works correctly.

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

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

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

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

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

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

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

2. Run the ansible playbooks.

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

```
config_file=/root/contrail-ansible-deployer/instances.yaml
playbooks/install_contrail.yaml
```

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

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

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

```
docker restart control_control_1
```

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

Finalizing the Contrail Ansible Deployer ISSU Process

Finalize the Contrail ansible deployer ISSU as follows:

1. Stop the issu-run-sync container.

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

2. Run the post synchronization commands.

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

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

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

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

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

5. Run the following commands from any controller node.

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

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

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

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

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

IN THIS SECTION

- [Contrail In-Service Software Upgrade \(ISSU\) Overview | 303](#)
- [Prerequisites | 304](#)
- [Preparing the Contrail System for the Helm Deployer ISSU Procedure | 304](#)
- [Provisioning Control Nodes and Performing Synchronization Steps | 305](#)
- [Transferring the Compute Nodes into the New Cluster | 308](#)
- [Finalizing the Contrail Helm Deployer ISSU Process | 312](#)

Contrail In-Service Software Upgrade (ISSU) Overview

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

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

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

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

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

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

Prerequisites

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

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

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

Preparing the Contrail System for the Helm Deployer ISSU Procedure

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

1. Deploy the 5.0.x version of Contrail using the Contrail Helm deployer, but make sure to include only the following Contrail controller services:
 - Config
 - Control
 - Analytics

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

NOTE: You must provide keystone authorization information for setup.

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

Detailed instructions for deploying the new cloud using Helm are provided in [“Installing Contrail Networking for Kubernetes using Helm” on page 128](#).

Provisioning Control Nodes and Performing Synchronization Steps

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

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

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

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

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

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

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

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

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

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

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

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

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

The configuration appears as follows and can be stored locally:

```
[DEFAULTS]
# details about oldrabbit
old_rabbit_user = contrail
old_rabbit_password = ab86245f4f3640a29b700def9e194f72
old_rabbit_q_name = vnc-config.issu-queue
old_rabbit_vhost = contrail
old_rabbit_port = 5672
old_rabbit_address_list = ip-address
```

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

1. Detect the config-api image ID:

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

2. Run the pre-synchronization.

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

3. Run the run-synchronization.

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

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

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

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

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

Transferring the Compute Nodes into the New Cluster

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

1. Select the compute node(s) for transferring into the new cluster.
2. Move all workloads from the node(s) to other compute nodes. You also have the option to terminate workloads as appropriate.
3. For Contrail Release 3.x, remove Contrail from the node(s) as follows:
 - Stop the vrouter-agent service.
 - Remove the **vhost0** interface.
 - Switch the physical interface down, then up.
 - Remove the vrouter.ko module from the kernel.
4. For Contrail Release 4.x, remove Contrail from the node(s) as follows:
 - Stop the agent container.
 - Restore the physical interface.
5. Add the required node(s) to **instances.yml** with the roles **vrouter** and **openstack_legacy_compute**.
6. Run the Contrail Helm deployer to deploy the new vrouter and to configure the old compute service.
7. All new compute nodes will have:
 - The collector setting pointed to the new Contrail cluster
 - The Control/DNS nodes pointed to the new Contrail cluster
 - The config-api setting in **vnc_api_lib.ini** pointed to the new Contrail cluster
8. (Optional) Run a test workload on transferred nodes to ensure the new vrouter-agent works correctly.

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

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

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

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

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

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

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

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

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

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

1. Add the new instance to `/opt/openstack-helm-infra/tools/gate/devel/multinode-inventory.yaml`, in the nodes section.

2. Prepare the new compute nodes for Contrail deployment:

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

3. Verify the new node names by using the command `kubectl get nodes`.

4. Label the new nodes as follows:

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

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

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

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

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

7. Add vrouter as follows:

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

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

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

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

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

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

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

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

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

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

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

Finalizing the Contrail Helm Deployer ISSU Process

Finalize the Contrail Helm deployer ISSU as follows:

1. Stop the issu-run-sync container.

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

2. Run the post synchronization commands.

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

3. Start all previously stopped containers.

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

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

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

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

```
[DEFAULTS]
db_host_info={"ip-address": "node-ip-address", "ip-address": "node-ip-address",
```

```

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

```

6. Run the following commands from any controller node:

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

```

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

```

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

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 in the deployer input file(**command_servers.yml**) and deploy playbook.

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

```
export
CCD_IMAGE=ci-repo.englab.juniper.net:5010/contrail-command-deployer:5.1-<BUILD_NO>
docker pull $CCD_IMAGE
export COMMAND_SERVERS_FILE=<ABSOLUTE_PATH_OF_COMMAND_SERVERS_FILE>
docker run -td --net host -v $COMMAND_SERVERS_FILE:/command_servers.yml
--privileged --name contrail_command_deployer_<BUILD_NO> $CCD_IMAGE
```

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

If you are upgrading from Contrail release 5.0.x to Contrail release 5.1, you can skip this step.

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.

Backup and Restore Contrail Software

IN THIS CHAPTER

- [Backing up Contrail Databases in JSON Format | 315](#)

Backing up Contrail Databases in JSON Format

IN THIS SECTION

- [Preliminary Caution | 315](#)
- [Simple Database Backup in JSON Format | 316](#)
- [Restore Database from the Backup | 316](#)
- [Example Backup and Restore in JSON | 318](#)

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

Preliminary Caution



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

Simple Database Backup in JSON Format

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

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

The controller node for containerized Contrail is a controller container.

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

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

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

```
cd /usr/lib/python2.7/site-packages/cfgm_common  
python db_json_exim.py --export-to db-dump.json
```

3. See a cleaner version of the dump.

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

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

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

Restore Database from the Backup

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

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

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

2. Stop config services on all the controllers.

```
docker stop config_svcmonitor_1  
docker stop config_devicemgr_1  
docker stop config_schema_1  
docker stop config_api_1
```

```
docker stop config_nodemgr_1
```

```
docker stop config_database_nodemgr_1
```

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

```
docker stop config_database_cassandra_1
```

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

```
docker stop config_database_zookeeper_1
```

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

```
docker stop analytics_database_kafka_1
```

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

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

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

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

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

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

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

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

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

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

10. Start Zookeeper on all the controllers.

```
docker start config_database_zookeeper_1
```

11. Start Cassandra on all the controllers.

```
docker start config_database_cassandra_1
```

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

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

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

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

14. Restore the data in new running docker.

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

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

```
docker start analytics_database_kafka_1
```

16. Start config services on all the controllers.

```
docker start config_svcmonitor_1
docker start config_devicemgr_1
docker start config_schema_1
docker start config_api_1
docker start config_nodemgr
docker start config_database_nodemgr
```

Example Backup and Restore in JSON

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

- nodec53
- nodec54
- nodec55

Example: Perform Simple Database Backup in JSON Format

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

```

    "objfq_name_table": {
        "access_control_list": {
<snip>

```

Example: Restore Database from the Backup

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

```

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

```

2. Stop config services on all the controllers.

```

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

```

```

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

```

```

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

```

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

```

[root@nodec53 ~]# docker stop config_database_cassandra_1

```



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

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

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

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

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

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

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

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

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

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

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

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

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

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

7. Delete config Cassandra.

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

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

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

8. Delete config Zookeeper.

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

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

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

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

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

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

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

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

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

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

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

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

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

12. Start config services on all the controllers.

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

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

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

Post Installation Tasks

IN THIS CHAPTER

- [Configuring Role and Resource-Based Access Control | 324](#)
- [Configuring Role-Based Access Control for Analytics | 332](#)
- [Configuring the Control Node with BGP | 333](#)
- [Configuring MD5 Authentication for BGP Sessions | 345](#)
- [Configuring Transport Layer Security-Based XMPP in Contrail | 347](#)
- [Configuring Graceful Restart and Long-lived Graceful Restart | 349](#)

Configuring Role and Resource-Based Access Control

IN THIS SECTION

- [Contrail Role and Resource-Based Access \(RBAC\) Overview | 324](#)
- [API-Level Access Control | 325](#)
- [Object Level Access Control | 326](#)
- [Configuration | 327](#)
- [Upgrading from Previous Releases | 329](#)
- [Configuring RBAC Using the Contrail User Interface | 329](#)
- [RBAC Resources | 332](#)

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

IN THIS SECTION

- [Parameter: aaa-mode | 327](#)
- [Parameter: cloud_admin_role | 327](#)
- [Global Read-Only Role | 328](#)
- [Parameter Changes in /etc/neutron/api-paste.ini | 329](#)

This section describes the parameters used in Contrail RBAC.

Parameter: aaa-mode

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

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

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

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

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

Parameter: cloud_admin_role

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

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

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

Configuration Files with Cloud Admin Credentials

The following configuration files contain **cloud_admin_role** credentials:

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

Changing Cloud Admin Configuration Files

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

1. Change the configuration files with the new information.

2. Restart the following:

- API server
service supervisor-config restart
- Neutron server
service neutron-server restart
- WebUI
service supervisor-webui restart

Global Read-Only Role

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

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

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

Setting the Global Read-Only Role

To set the global read-only role:

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

```
/etc/contrail/contrail-api.conf  
global_read_only_role = <new-admin-read-role>
```

2. Restart the **contrail-api** service:

```
service contrail-api restart
```

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

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

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

Upgrading from Previous Releases

The **multi_tenancy** parameter is deprecated.. 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
```

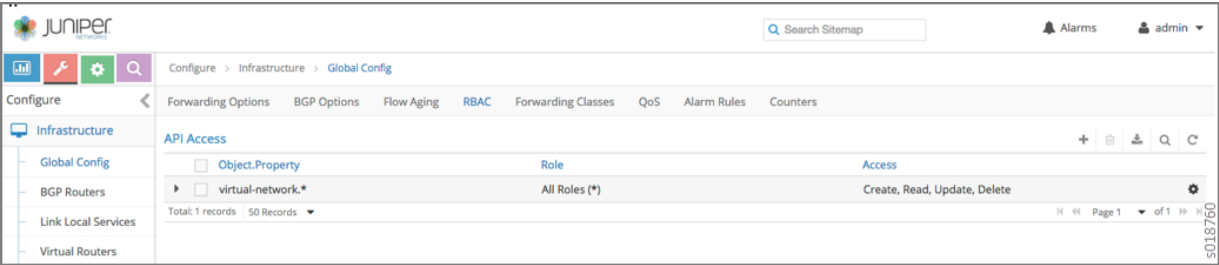
3. Restart services by restarting the container.

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 50 on page 330](#).

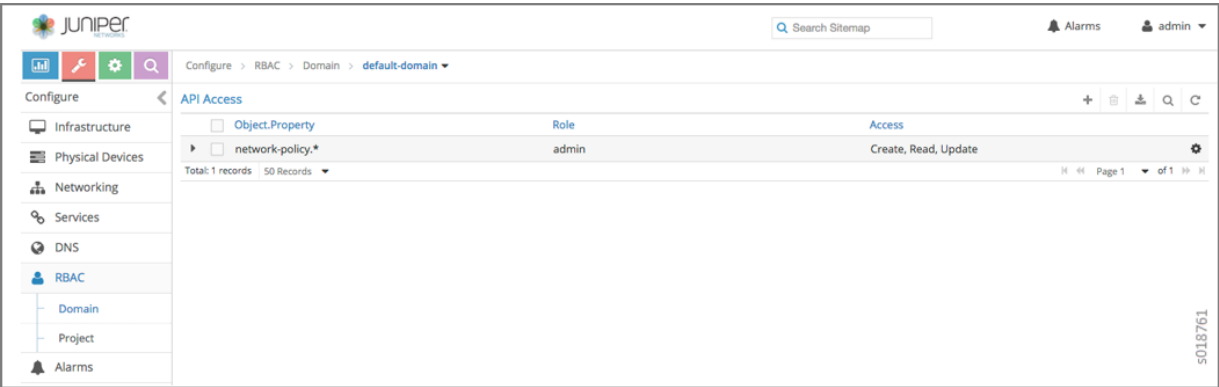
Figure 50: RBAC Global Level



Configuring RBAC at the Domain Level

To configure RBAC at the domain level, navigate to **Configure > RBAC > Domain**, see [Figure 51 on page 330](#).

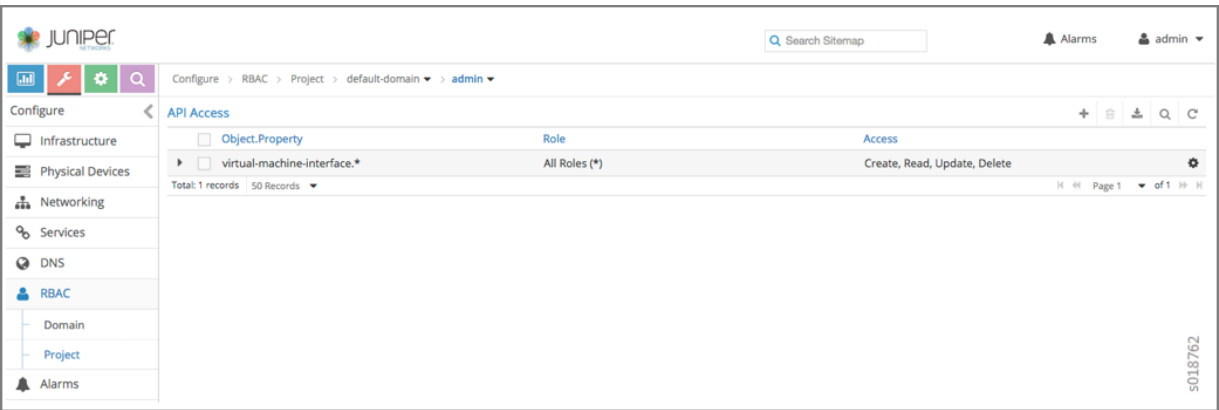
Figure 51: RBAC Domain Level



Configuring RBAC at the Project Level

To configure RBAC at the project level, navigate to **Configure > RBAC > Project**, see [Figure 52 on page 330](#).

Figure 52: 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 53 on page 331](#).

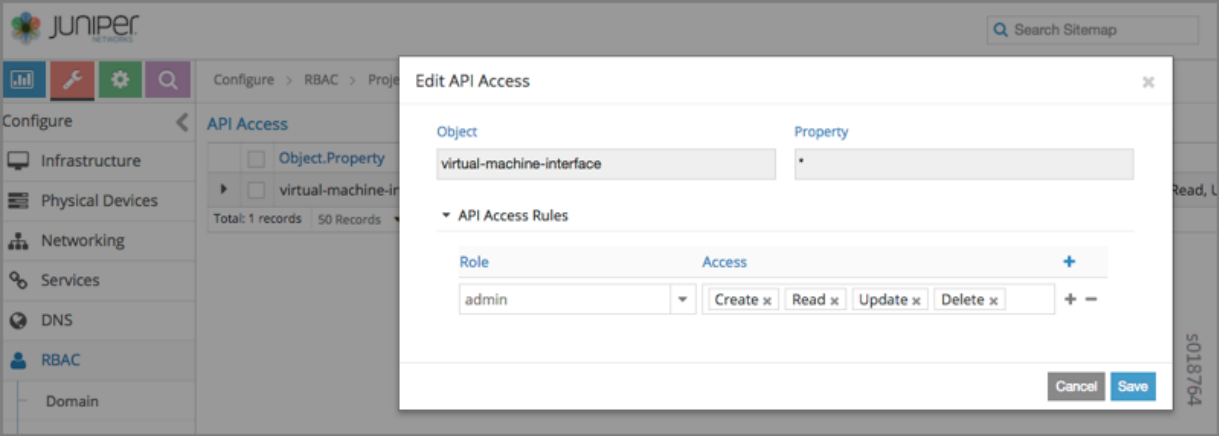
Figure 53: RBAC Details API Access



Creating or Editing API Level Access

Clicking create, edit, or insert after activates the Edit API Access popup window, where you enter the details for the API Access Rules. Enter the user type in the Role field, and use the + icon in the Access field to enter the types of access allowed for the role, including, Create, Read, Update, Delete, and so on, see [Figure 54 on page 331](#).

Figure 54: 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 55 on page 332](#).

Figure 55: Edit Object Level Access

The screenshot shows a web-based 'Edit' dialog for object level access. It features two tabs: 'Network' and 'Permissions'. The 'Permissions' tab is active, displaying the following fields and controls:

- Owner:** A text field containing the UUID `e5071271c48b432b9ca42572600bf1f6`.
- Owner Permissions:** Three buttons labeled 'Read x', 'Write x', and 'Refer x'.
- Global Share Permissions:** A button labeled 'Select Permissions'.
- Share List:** A section with a dropdown arrow and a table. The table has two columns: 'Project' and 'Permissions'. A plus sign (+) is visible to the right of the table header.

At the bottom right of the dialog, there are 'Cancel' and 'Save' buttons. A vertical label 's018765' is visible on the right side of the dialog.

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`
- `application_policy_set`

In `/etc/contrail/contrail-analytics-api.conf`, in the section **DEFAULTS**, the parameter `aaa_mode` now supports `rbac` as one of the values.

Configuring the Control Node with BGP

IN THIS SECTION

- [Configuring the Control Node from Contrail Web UI | 335](#)
- [Configuring the Control Node with BGP from Contrail Command | 341](#)

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.

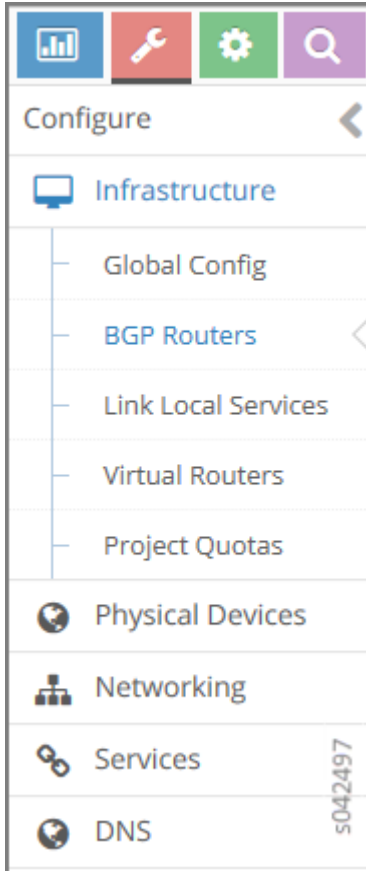
These topics provide instructions to configure the Control Node with BGP.

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 56 on page 336](#).

Figure 56: Configure> Infrastructure > BGP Routers



A summary screen of the control nodes and BGP routers is displayed; see [Figure 57 on page 336](#).

Figure 57: BGP Routers Summary

Configure > Infrastructure > BGP Routers

BGP Routers

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

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

- (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.
- 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 58 on page 338](#).

Figure 58: 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 18 on page 338](#).

Table 18: Create BGP Router Fields

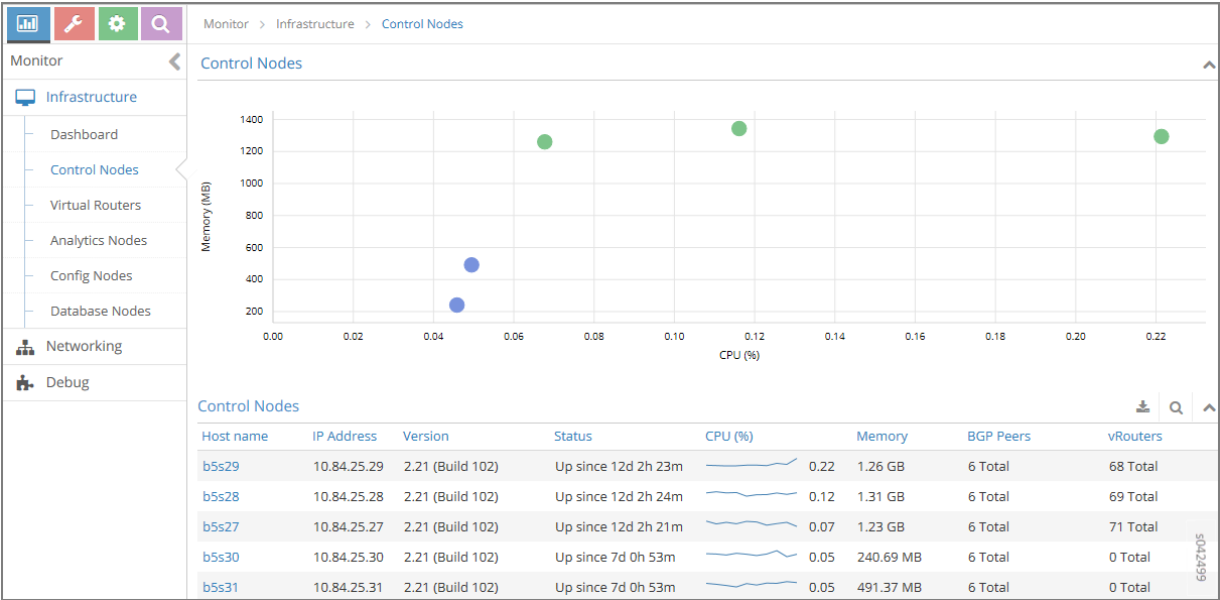
Field	Description
Hostname	Enter a name for the node being added.
Vendor ID	Required for external peers. Populate with a text identifier, for example, "MX-0". (BGP peer only)
IP Address	The IP address of the node.
Router ID	Enter the router ID.
Autonomous System	Enter the AS number in the range 1-65534 for the node. (BGP peer only)

Table 18: Create BGP Router Fields (*continued*)

Field	Description
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.

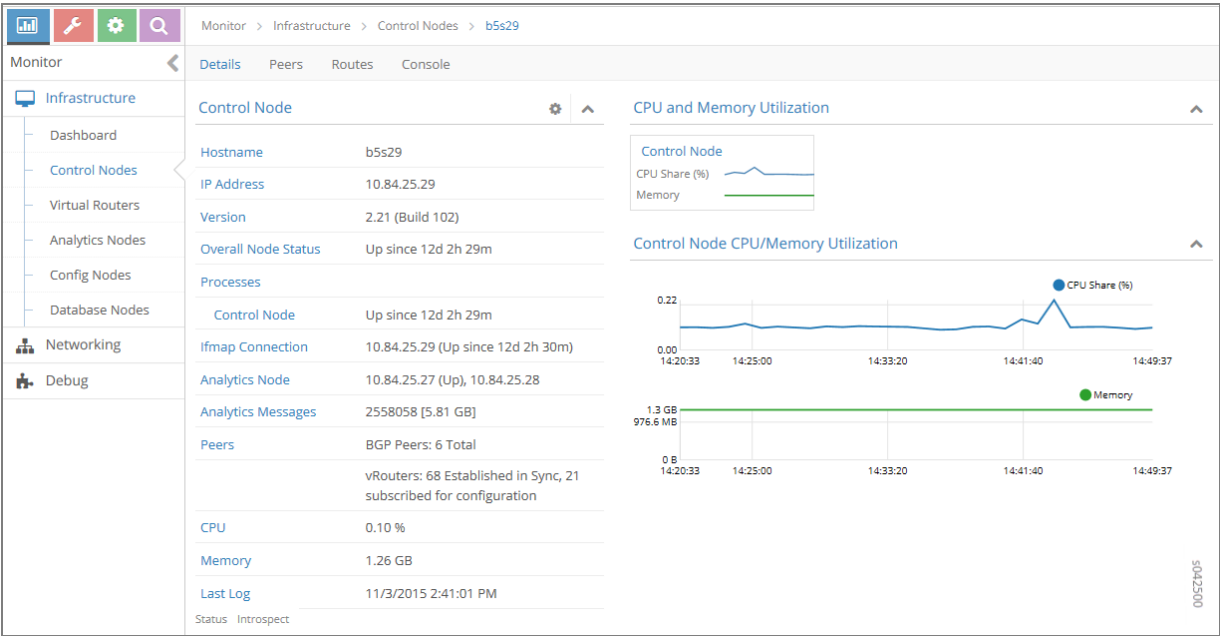
5. Click **Save** to add each node that you create.
6. To configure an existing node as a peer, select it from the list in the **Available Peers** box, then click >> to move it into the **Configured Peers** box.
Click << to remove a node from the **Configured Peers** box.
7. You can check for peers by selecting **Monitor > Infrastructure > Control Nodes**; see [Figure 59 on page 340](#).

Figure 59: Control Nodes



In the **Control Nodes** window, click any hostname in the memory map to view its details; see [Figure 60 on page 340](#).

Figure 60: Control Node Details



8. Click the **Peers** tab to view the peers of a control node; see [Figure 61 on page 341](#).

Figure 62: BGP Routers List

INFRASTRUCTURE > Cluster > Advanced

default-domain > default-project

Admin

< Back < Global Config Virtual Routers **BGP Routers** Control Node Zones Quality of Service Security Encryption Link Local Services Endpoints >

BGP Routers

Search Refresh Delete Create

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

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

4. Select the desired router from the check box and click the **Edit** icon.
- The **BGP** tab in the **Edit BGP Router** page is displayed.
5. (Optional) The global AS number is 64512 by default. You can change the AS number of a router according to your requirement on the **BGP Routers** tab. Select the desired router and click the **Edit** icon. In the **Edit BGP Router** tab enter the new AS number in the range 1-65535.
6. Click the **Create** button on the **BGP Routers** tab. The **Create BGP Router** window is displayed. See [Figure 63 on page 343](#).

Figure 63: Create BGP Router

7. In the **Create BGP Router** page, populate the fields with values to create your system. See [Table 19 on page 343](#).

Table 19: 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.
Autonomous System (AS)	Enter the AS number for the node in the range 1-65535. (BGP peer only)
BGP Router ASN	Enter the Local-AS number, specific to the associated peers.

Table 19: Create BGP Router (*continued*)

Fields	Description
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	
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.

Table 19: Create BGP Router (continued)

Fields	Description
Physical Router	Select the the physical router from the list.

- Click **Create** to complete add each node.
- 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.

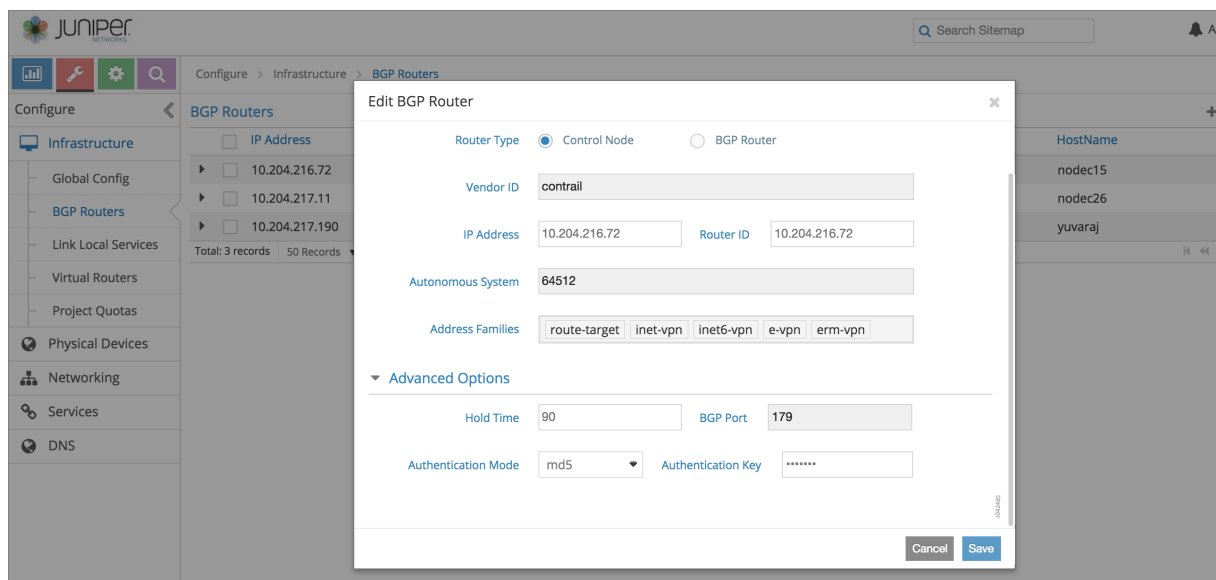
- If the **md5** key is not included in the provisioning, and the node is already provisioned, you can run the following script with an argument for md5:

```
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.
 - a. Connect to the node's IP address at port 8080 (<node_ip>:8080) and select **Configure->Infrastructure->BGP Routers**. As shown in [Figure 64 on page 346](#), a list of BGP peers is displayed.

Figure 64: Edit BGP Router Window



- b. 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.
- c. Scroll down the window and select **Advanced Options**.
- d. 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

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
<code>xmpp_server_cert</code>	Path to the node's public certificate	<code>/etc/contrail/ssl/certs/server.pem</code>
<code>xmpp_server_key</code>	Path to server's or node's private key	<code>/etc/contrail/ssl/private/server-privkey.pem</code>
<code>xmpp_ca_cert</code>	Path to CA certificate	<code>/etc/contrail/ssl/certs/ca-cert.pem</code>

Parameter	Description	Default
<code>xmpp_auth_enable=true</code>	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
<code>xmpp_server_cert</code>	Path to the node's public certificate	<code>/etc/contrail/ssl/certs/server.pem</code>
<code>xmpp_server_key</code>	Path to server's/node's private key	<code>/etc/contrail/ssl/certs/server-privkey.pem</code>
<code>xmpp_ca_cert</code>	Path to CA certificate	<code>/etc/contrail/ssl/certs/ca-cert.pem</code>
<code>xmpp_dns_auth_enable=true</code>	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
<code>xmpp_server_cert</code>	Path to the node's public certificate	<code>/etc/contrail/ssl/certs/server.pem</code>
<code>xmpp_server_key</code>	Path to server's/node's private key	<code>/etc/contrail/ssl/private/server-privkey.pem</code>
<code>xmpp_ca_cert</code>	Path to CA certificate	<code>/etc/contrail/ssl/certs/ca-cert.pem</code>
<code>xmpp_auth_enable=true</code> <code>xmpp_dns_auth_enable=true</code>	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

- [Application of Graceful Restart and Long-lived Graceful Restart | 349](#)
- [BGP Graceful Restart Helper Mode | 350](#)
- [Feature Highlights | 350](#)
- [XMPP Helper Mode | 350](#)
- [Configuration Parameters | 351](#)
- [Cautions for Graceful Restart | 352](#)
- [Configuring Graceful Restart with the Contrail User Interface | 353](#)

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 65 on page 353](#).

Figure 65: Configuring Graceful Restart

