JUNIPER NETWORKS

DAY ONE: JUNIPER CONTRAIL® AND RED HAT OPENSTACK 16 ARCHITECTURE DEPLOYMENT

By Fady Philip

DAY ONE: JUNIPER CONTRAIL AND RED HAT OPENSTACK 16 ARCHITECTURE DEPLOYMENT

This five-chapter Day One book is designed to take you from basic knowledge of OpenStack and SDN into some deeper areas around Juniper Contrail plus Red Hat OpenStack architecture deployment details. It reviews the various architectures and then takes a closer look at the Neutron project, the open source project for networking in OpenStack that handles its network-as-a-service abilities. The author reviews the many related acronyms that might confuse those working with OpenStack and Juniper Contrail. By the end you have are using RHOSP 16 and Juniper Contrail in a step-by-step deployment that illustrates how to manipulate different templates for customized overcloud deployment.

"This book covers concepts of Red Hat OpenStack and Juniper Contrail architecture and deployment. It is packed with examples that makes it a go-to resource even if you never worked with Red Hat before. Highly recommended! I, for one, learned a great deal."

Christian Scholz, Senior Consultant, Axians

IT'S DAY ONE AND YOU HAVE A JOB TO DO:

- Understand and describe Juniper Contrail and RHOSP architectures.
- Design Customized overcloud
- Deploy Customized Contrail + RHOSP

Juniper Networks Books are focused on network reliability and efficiency. Peruse the complete library at www.juniper.net/dayone.

JUNIPER NETWORKS

Day One: Juniper Contrail and Red Hat OpenStack 16 Architecture Deployment

by Fady Philip

JUNIPER NETWORKS

Published by Juniper Networks Books
Authors: Fady Philip
Technical Reviewers: Greg Houde, Christian Scholz, Victor Gonzalez
Editor in Chief: Patrick Ames
Copyeditor: Nancy Koerbel

Version History: v1, January 2022
2 3 4 5 6 7 8 9 10

About the Author
Fady Philip has over 15 years of experience in the networking industry. He has wide experience across multiple technology domains with a unique blend of routing, security, datacenter, mobile packet core, SDN/NFV, and cloud skills. He is certified JNCIE-Cloud#23. He was a former end-to end lead Solutions Architect who worked with global large telecom service providers. In his current role at Juniper as SDN/NFV Product Consultant, he enjoys assisting customers build, maintain, operate, and optimize Juniper Contrail plus VIM in addition to transfering knowledge to whomever needs competence in these related areas.

Author's Acknowledgments
I'd love to thank my wife Haidy who gave me the time and support to write this book and to work after hours. I'd love to also thank my manager Murtaza Asgerali who inspired me by the idea and gave full support from the start to the end. Above all I owe the full gratitude to God who gave me his grace to understand and absorb the concepts, the strength to carry out the writing with patience, and surrounded me with perfect people to support on all levels.

Welcome to Day One

This book is part of the *Day One* library, produced and published by Juniper Networks. *Day One* books cover Juniper Networks network administration with straightforward explanations, step-by-step instructions, and practical examples that are easy to follow.

- Download a free PDF edition at https://www.juniper.net/dayone.
- Purchase the paper edition at Vervante Corporation (www.vervante.com).

Key Juniper Contrail Resources

- https://www.juniper.net/documentation/en_US/release-independent/solutions/information-products/pathway-pages/sg-010-contrail-networking-arch-guide.pdf
- https://github.com/tnaganawa/tungstenfabric-docs/blob/master/TungstenFabricPrimer.md#overall-picture
- https://docs.openstack.org/tripleo-docs/latest/install/introduction/architecture.html

Key Red Hat Resources

- https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/

What You Need to Know Before Reading This Book

- Basic knowledge of RHEL,YAML, OpenStack, and Juniper Contrail.
- Basic networking knowledge

What You Will Be Able To Do After Reading This Book

- Understand and describe Juniper Contrail and RHOSP architectures.
- Design customized overclouds.
- Deploy customized Juniper Contrail plus RHOSP.

Chapter 1

Introduction

This five-chapter *Day One* book is designed to take you from basic knowledge of OpenStack and SDN into some deeper areas around Contrail plus Red Hat OpenStack Architecture and deployment details.

Workflow

In this chapter, the OpenStack project is explained with focus on neutron (networking) project and how Contrail is integrated into the overall stack.

In Chapter 2 TripleO architecture and deployment methodology is explained as a prelude for Chapter 3, which contains Red Hat OpenStack architecture (TripleO-based OpenStack).

In Chapter 4 the evolution of Contrail architecture and its main building blocks are explained, and Chapter 5 explains RHOSP16 plus Contrail in a step-by-step deployment and how to manipulate different templates for customized overcloud deployment.

Let's start with some basic but very important-to-know theories.

What is OpenStack?

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a data center, all managed and provisioned through APIs with common authentication mechanisms.

Cloud in its simplest definition is a pool of storage with nearby compute cycles. Cloud networks are built to be used by many types of users for different purposes, however, as a rule you need to operate these networks anytime and anywhere, hence the need for cloud operating systems that users can interact with to operate the cloud.

Clouds typically do not have all their resources in one location. The resources might be distributed among different physical locations with few restrictions on the distance between those physical locations, while the operational experience should be seamless to the users and their workloads. This necessary agility for seamless cloud usage mandates the need for agile networking from provisioning, modification, optimization, and fault management perspectives.

The OpenStack cloud operating system allows admins and tenant users to manage their compute, storage, and networking resources, according to their needs, via a set of tools and APIs. It's broken up into services that allow cloud providers to plug and play components. Moreover, it's an open source project that allows developers to contribute and customize internal services that add, remove, and enhance services according to their organization's needs.

The OpenStack map in Figure 1.1 provides an at-a-glance overview of the OpenStack landscape and where those services fit and how they work together. The various services and projects that appear in OpenStack map work interdependently to form the cloud operating system.

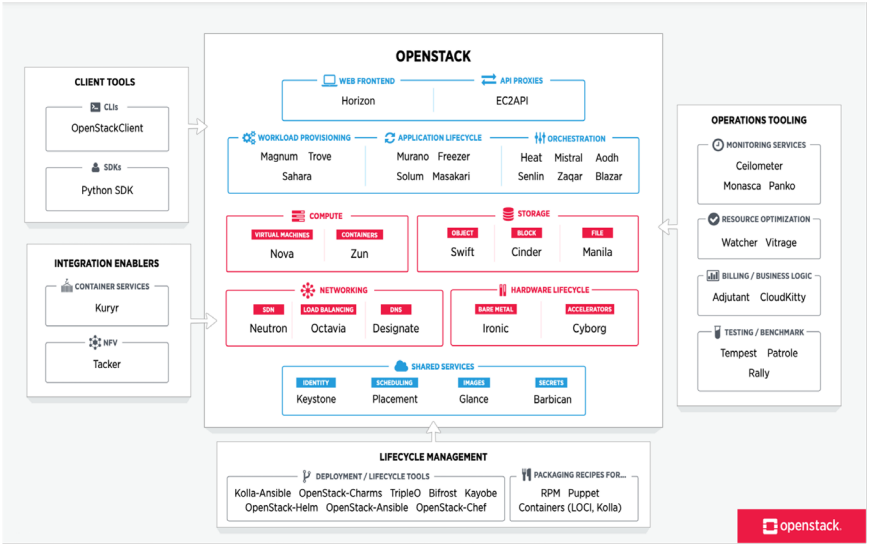

Figure 1.1 *OpenStack Map*

Table 1.1 lists the OpenStack Core components and their functions.

Table 1.1 OpenStack Core Components

Service	Name	Description
Dashboard	horizon	Web browser-based dashboard that you use to manage OpenStack services.
Identity	keystone	Centralized service for authentication and authorization of OpenStack services and for managing users, projects, and roles.
Openstack Networking	neutron	Provides connectivity between the interfaces of OpenStack services.
Block Storage	cinder	Manages persistent block storage volumes for virtual machines.
Compute	nova	Manages and provisions virtual machines running on hypervisor nodes.
Image	glance	Registry service that you use to store resources such as virtual machine images and volume snapshots.
Object Storage	swift	Allows users to store and retrieve files and arbitrary data.
Telemetry	ceilometer	Provides measurements of cloud resources.
Orchestration	heat	Template-based orchestration engine that supports automatic creation of resource stacks.

Since this book investigates Juniper Contrail plus Red Hat OpenStack, let's have a closer look at the Neutron project, the open source project for networking in OpenStack that handles its network-as-a-service abilities, and some of the related acronyms that might confuse those working with OpenStack and Contrail.

ML2 Drivers

The Modular Layer 2 (ML2) neutron plug-in is a framework allowing OpenStack Networking to simultaneously use the variety of Layer 2 networking technologies found in complex real-world data centers. The ML2 framework distinguishes between the two kinds of drivers that can be configured as shown in Figure 1.2.

Type Drivers

These drivers define how an OpenStack network is technically realized, an example being VXLAN. Each available network type is managed by an ML2 type driver. Type drivers maintain any needed type-specific network state. They validate the type-specific information for provider networks and are responsible for the allocation of free segments in project networks.

Mechanism Drivers

These drivers define the mechanism to access an OpenStack network of a certain type, an example being the Open vSwitch mechanism driver. The mechanism driver is responsible for taking the information established by the type driver and ensuring that it is properly applied given the specific networking mechanisms that have been enabled. Mechanism drivers can utilize Level 2 agents (via RPC) and interact directly with external devices or controllers.

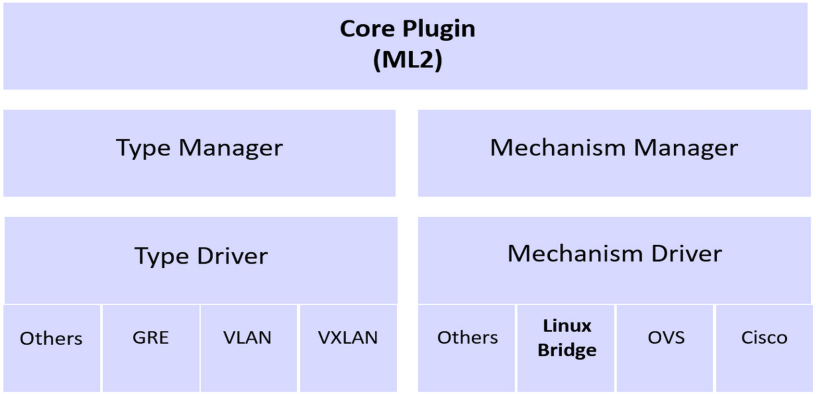

Figure 1.2 *ML2 Plug-in Architecture*

OpenStack Plug-ins

The OpenStack ecosystem is wide and deep and growing more so every day. The value of DevStack (let's consider it the main code of OpenStack environment) is that it's simple enough for one to clearly understand what it's doing. And yet we'd like to support as much of the OpenStack ecosystem as possible. We do that with plug-ins.

DevStack plug-ins are bits of bash code that live outside the DevStack tree. They are called through a strong contract, so these plug-ins can be sure that they will continue to work in the future as DevStack evolves.

After we have defined those basic acronyms let's have a closer look to the Neutron project in OpenStack.

Networking (Neutron) Concepts

OpenStack Networking (Neutron) manages all networking facets for the Virtual Networking Infrastructure (VNI) and the access layer aspects of the Physical Networking Infrastructure (PNI) in your OpenStack environment. OpenStack Networking enables projects to create advanced virtual network topologies which may include services such as a firewall, a load balancer, and a virtual private network (VPN).

Networking provides networks, subnets, and routers as object abstractions. Each abstraction has functionality that mimics its physical counterpart: networks contain subnets and routers that route traffic between different subnets and networks.

Any given networking setup has at least one *external network*. Unlike the other networks, the external network is not merely a virtually defined network. Instead, it represents a view into a slice of the physical, external network accessible outside the OpenStack installation. IP addresses on the external network are accessible by anyone physically on the outside network.

Each plug-in that Networking uses has its own concepts. While not vital to operating the VNI and OpenStack environment, understanding these concepts can help you set up Networking. All Networking installations use a core plug-in and a security group plug-in.

OpenStack Networking Neutron Service Overview

OpenStack Networking Neutron allows you to create and attach interface devices managed by other OpenStack services to networks. Plug-ins can be implemented to accommodate different networking equipment and software, providing flexibility to the OpenStack architecture and its deployment.

Neutron provides an API that lets you define network connectivity and addressing within the cloud. Its networking service enables operators to leverage different networking technologies to power their cloud networking. Neutron also provides an API to configure and manage a variety of network services ranging from Level 3 forwarding and Network Address Translation (NAT), to load balancing, perimeter firewalls, and VPNs.

Neutron includes the following components that are illustrated in Figure 1.3 and Figure 1.4:

- API server: The OpenStack Networking API includes support for Layer 2 networking and IP Address Management (IPAM), as well as an extension for a Layer 3 router construct that enables routing between Layer 2 networks and gateways to external networks. OpenStack Networking includes a growing list of plug-ins that enable interoperability with various commercial and open

source network technologies, including routers, switches, virtual switches, and software-defined networking (SDN) controllers.

- OpenStack Networking plug-ins and agents: Plug and unplug ports, create networks or subnets, and provide IP addressing with these plug-ins and agents that differ depending on the vendor and technologies used in various clouds. OpenStack Networking ships with plug-ins and agents for Cisco virtual and physical switches, NEC OpenFlow products, Open vSwitch, Linux bridging, and the VMware NSX product. The common agents are L3 (Layer 3), DHCP (dynamic host IP addressing), and a plug-in agent.
- Messaging queue: This is used by most OpenStack Networking installations to route information between the Neutron-server and various agents. The queue acts as a database to store networking state for particular plug-ins. OpenStack Networking mainly interacts with OpenStack Compute to provide networks and connectivity for its instances.

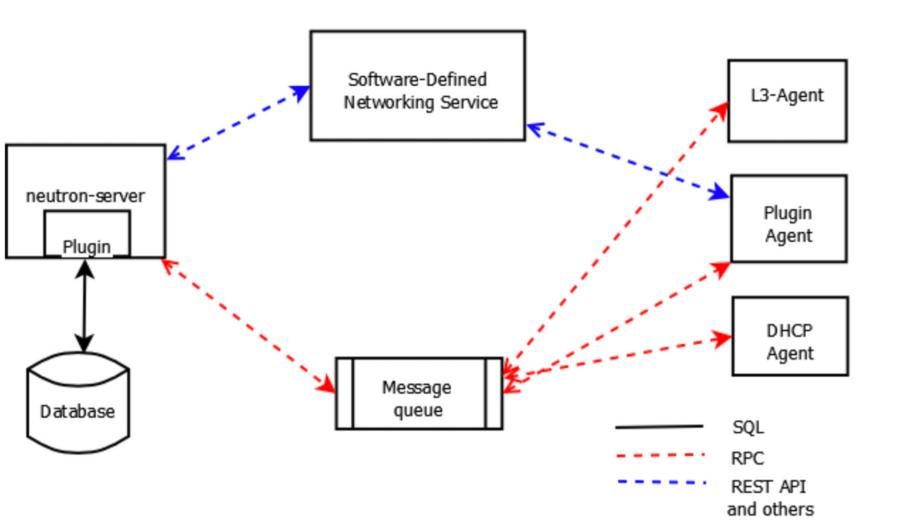

Figure 1.3 *OpenStack Networking Neutron Architecture*

You might notice a building block in Figure 1.3 called *Software-Defined Networking Service* and in Figure 1.4 it is obvious that it has vendor plug-ins under core plug-ins, and vendor drivers under mechanism drivers. This leads us to a very revealing question: what is the relation between Neutron and Contrail and where are the stitching points?

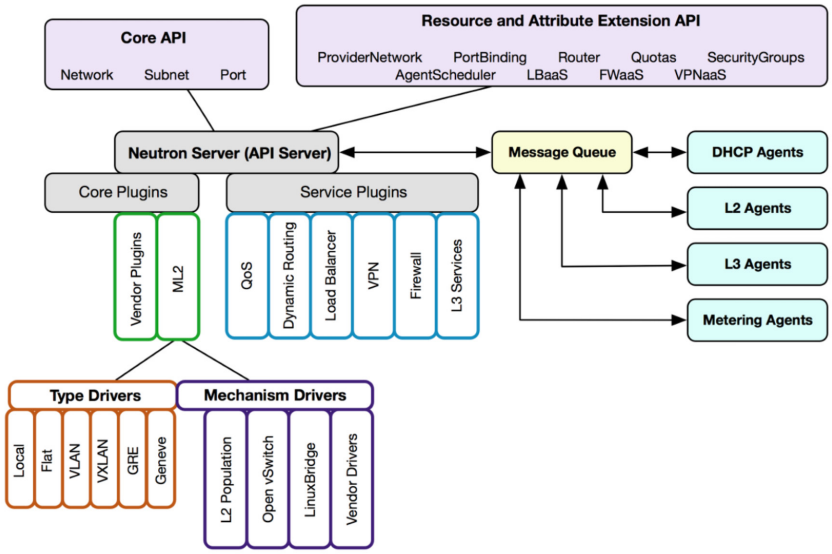

Figure 1.4 Detailed Neutron Architecture

Juniper Contrail/OpenContrail Deployment Models with OpenStack

OpenContrail has two deployment models with OpenStack: the monolithic plug-in and the ML2-based plug-in. Let's review each.

Single Plug-in Method (Monolithic Plug-in)

The single plug-in method, also known as the *monolithic plug-in,* is replacing the Neutron ML2 core plug-in with the Contrail plug-in. This method of deployment is used in commercial Contrail (Juniper Contrail). It gives full-feature support via the monolithic plug-in but you can't use any of the ML2 plugin-based extensions (as they have been totally replaced). You can see in Figure 1.5 that Contrail is deployed as the core plug-in instead of ML2 in the monolithic plug-in deployment.

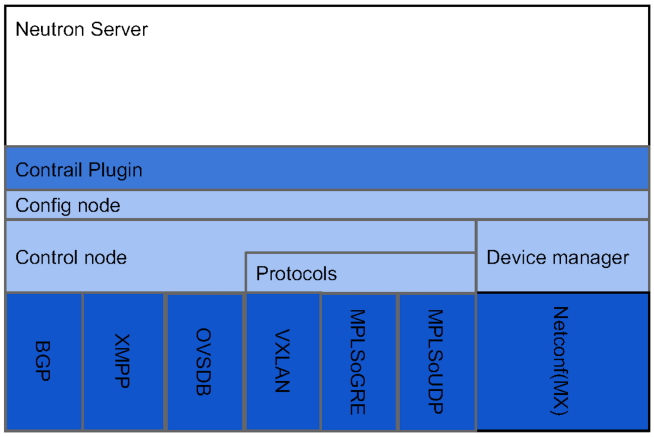

Figure 1.5 *Monolithic Deployment of Juniper Contrail*

ML2-based Plug-in

As previously mentioned, ML2-based plug-in deployment supports many types of plug-ins (drivers) at the same time. For example, OpenContrail could be added as a vendor driver under mechanism driver, or some of Juniper Contrail's advanced features could be added as a service plug-in (but not all of them compared to the commercial Contrail itself). You can see in Figure 1.6 that OpenContrail is deployed as one of the mechanism drivers in the ML2-based plug-in.

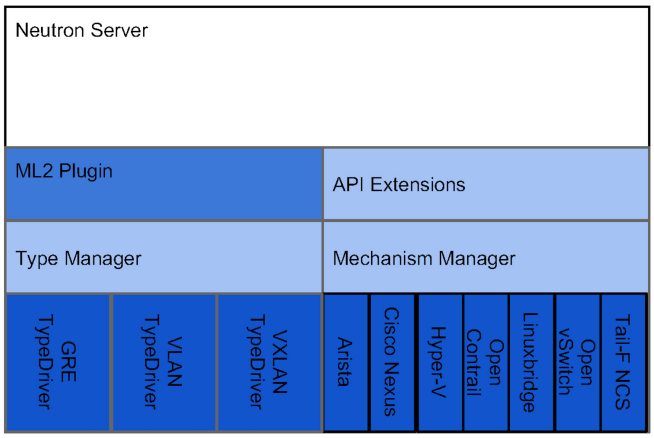

Figure 1.6 *ML2-based Plug-in Deployment of OpenContrail*

When putting all the pieces together it should be obvious how Juniper Contrail is

stitched together with the OpenStack Neutron project. Chapter 2 discusses the second piece of the journey, which is the TripleO architecture and deployment methodology as a base for Red Hat OpenStack product.

Chapter 2

TripleO Architecture and Deployment Methodology

There are many deployment methods for OpenStack with different tools and packaging recipes to help install and maintain the life cycle of OpenStack deployments. In this chapter we will describe OpenStack tripleO project architecture and deployment methodology as basis for Red Hat OpenStack.

What is TripleO?

TripleO is the friendly name for *OpenStack on OpenStack*. It is an official OpenStack project with the goal of allowing you to deploy and manage a production cloud onto bare metal hardware using a subset of existing OpenStack components. This means you deploy a compact OpenStack deployment on one box or VM (known as *undercloud*), with a subset of OpenStack services and use that to deploy, manage, and upgrade a production grade OpenStack on bare metal servers. See Figure 2.1.

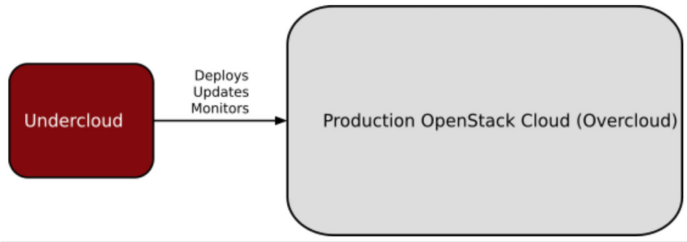

Figure 2.1 *High Level Concept of TripleO Project*

With TripleO, you start by creating an "undercloud" (a deployment cloud) that will contain the necessary OpenStack components to deploy and manage an *overcloud* (a workload cloud). The overcloud is the deployed solution and can represent a cloud for any purpose.

TripleO leverages several existing core components of OpenStack including Nova, Ironic, Neutron, Heat, Glance, and Ceilometer to deploy OpenStack on bare metal hardware (see Figure 2.2). Nova and Ironic are used in the undercloud to manage bare metal instances that comprise the infrastructure for the overcloud. Neutron is utilized to provide a networking environment that deploys the overcloud, machine images are stored in Glance, and Ceilometer collects metrics about your overcloud.

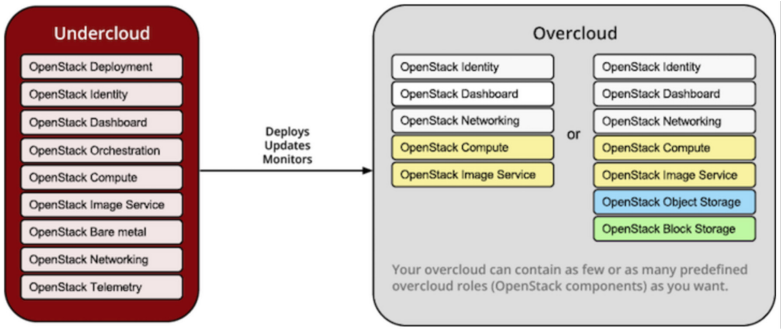

Figure 2.2 *Logical Overview of Undercloud and Overcloud Components*

Figure 2.3 illustrates a physical view of how the undercloud may be hosted on one physical server and the overcloud distributed across many physical servers.

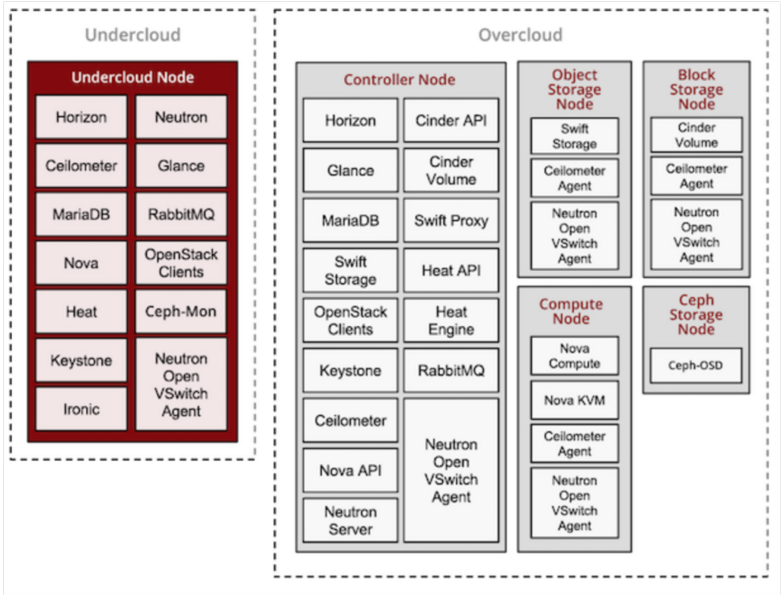

Figure 2.3 *Physical View of Undercloud/Overcloud Placement on Physical Servers*

Let's first take a deeper look into the undercloud and its main components, which are used to deploy a production overcloud.

Undercloud

The undercloud is the main management node containing the deployment and management toolset. It is a single-system OpenStack installation that includes components for provisioning and managing the OpenStack nodes that form your OpenStack environment (the overcloud). The components that form the undercloud have multiple functions:

- *Environment planning*: The undercloud includes planning functions that you can use to create and assign certain node roles. The undercloud includes a default set of nodes – compute, controller, and various storage roles. You can also design custom roles. Additionally, you can select which OpenStack platform services to include on each node role, which provides a method to model new node types or isolate certain components on their own host.
- *Bare metal system control*: The undercloud uses the out-of-band management interface, usually Intelligent Platform Management Interface (IPMI), of each node for power management control and a PXE-based service to discover hardware attributes and install OpenStack on each node. You can use this feature to provision bare metal systems as OpenStack nodes.
- *Orchestration*: The undercloud contains a set of YAML templates that represent a set of plans for your environment. The undercloud imports these plans and follows their instructions to create the resulting OpenStack environment. The plans also include hooks that you can use to incorporate your own customizations as points in your environment creation process.

Undercloud Components

The undercloud uses OpenStack components as its base toolset. Each component operates within a separate container on the undercloud:

- OpenStack Identity (Keystone): Provides authentication and authorization for the director components.
- OpenStack Bare Metal (Ironic) and OpenStack Compute (nova): Manages bare metal nodes.
- OpenStack Networking (Neutron) and Open vSwitch: Control networking for bare metal nodes.
- OpenStack Image Service (Glance): Stores images that undercloud writes to bare metal machines.
- OpenStack Orchestration (Heat) and Puppet: Provides orchestration and configuration of nodes after undercloud writes the overcloud image to disk.
- OpenStack Telemetry (Ceilometer): Performs monitoring and data collection. Telemetry also includes the following components:

- OpenStack Telemetry Metrics (Gnocchi): Provides a time series database for metrics.
- OpenStack Telemetry Alarm (AODH): Provides an alarm component for monitoring.
- OpenStack Telemetry Event Storage (Panko): Provides event storage for monitoring.
- OpenStack Workflow Service (Mistral): Provides a set of work flows for certain undercloud-specific actions, such as importing and deploying plans.
- OpenStack Messaging Service (Zaqar): Provides a messaging service for the OpenStack Workflow Service.
- OpenStack Object Storage (Swift): Provides object storage for various OpenStack Platform components, including:
- Image storage for OpenStack Image Service
- Introspection data for OpenStack Bare Metal
- Deployment plans for OpenStack Workflow Service

Those undercloud components work together to provide the toolset of services to provision, manage and upgrade overcloud. Their mutual goal is to facilitate and execute the sequence of actions for the required workflow to be done. Let's take the TripleO deployment workflow as an example.

Deployment Workflow Overview

1. Environment Preparation

- Prepare your environment (bare metal or virtual)
- Install undercloud

In the first place, you need to check that your environment is ready. TripleO can deploy OpenStack into bare metal as well as virtual environments. You need to make sure that your environment satisfies minimum requirements for the given environment type and that the networking is correctly set up. The next step is to install undercloud on a physical server or a VM, depending on your choice.

2. Undercloud Data Preparation

- Create images to establish the overcloud
- Register hardware nodes with undercloud
- Inspect hardware
- Create flavors (node profiles)

For the four steps that need to be completed in this stage, you need to have the images on the undercloud that will be installed on each of the Overcloud nodes, preferably in QCOW2 format. They are stored in the Glance component of the undercloud (as listed previously in *Undercloud Components)*.

Then undercloud needs to register the overcloud nodes, whether they are physical servers or VMs. It uses the power management information of each node and the MAC address of the provisioning NIC in JSON or YAML format to get control of the power management for each server.

Then it collects information about the hardware of each server and stores it into swift storage objects to be used at a later stage in the deployment. This process is called *introspection*.

As a last step at this stage, node flavors on undercloud need to be created to match the needed Overcloud role and map them to specific nodes. For example, you could create controller flavors that matches specific nodes to place the controller role on the intended nodes. This is the same for other roles; node placement can also be completed by manual node tagging.

Creating non-existent VM flavors on the undercloud has a slightly different effect than when they are used in overcloud. In undercloud they act as a constraint for node selection.

3. Deployment Planning

Deployment planning means manipulating many templates containing deployment information to include all parameters related to the task:

- Configure overcloud roles
    1. Assign flavor (node profile to match desired hardware specs) or use direct node tagging
    2. Assign image (provisioning image)
    3. Size the role (how many instances to deploy)
- Configure service parameters

After registering overcloud nodes to undercloud, the next deployment step is to use them to place overcloud roles (controllers, computes, storage) and their related configuration. To achieve that goal, you need to manipulate the many templates and environment files on the undercloud (these are discussed in detail in subsequent chapters). In this chapter you'll see the logic behind this manipulation based on the deployment planning.

The first thing you need know is which node should carry which overcloud role. For example, you need to determine where you need to place controllers, computes, and storage. How many instances of each do you need in overcloud deployment and what configuration parameters are needed specifically for each role?

For node placement there are two ways to achieve that goal. One of them is using flavors to schedule specific overcloud roles to specific node specs and letting undercloud select it for you according to the nodes matching the flavor specs. The other way is to use direct node tagging. This can be done by tagging specific nodes and manipulating the environment files to use the nodes with specific tags for specific roles.

Images used for specific roles, and the number of instances of that role, are also a configuration parameter inside deployment templates and environment files should be configured as part of your deployment planning.

After completing all the major tasks needed in deployment planning, heat templates and environment files are being used by Heat, Neutron, Nova, Glance, and Ironic to start overcloud deployment.

4. Deployment

The final step is to kick off your deployment using Heat to do the following steps:

- Heat will use Nova to identify and reserve the appropriate nodes
- Nova will use Ironic to start up nodes and install the correct images

Deployment to physical servers occurs through a collaboration of Heat, Nova, Neutron, Glance, and Ironic.

Heat uses the deployment templates and environment files to create the overcloud stack. The *stack* is Heat's term for the applications that it creates. The overcloud stack is considered a stack of smaller stacks. Heat is getting all needed information from templates and environment files to create objects and the overall stack.

Heat uses Nova to schedule overcloud roles to overcloud physical servers, which are selected either by associated flavor constrains or node tagging. At this point, when a specific physical server is selected for placing a specific role Ironic starts to do the node provisioning. It retrieves the role-associated OS image from Glance to use it in node booting and provisioning.

After node provisioning, overcloud nodes gather configuration metadata from the Heat template configuration files. At this stage Heat uses puppet manifests to configure the services on the nodes. Puppet runs in multiple steps, so that after each step there can be tests triggered to check the progress of the deployment and allow for easier debugging.

At the end of the process, assuming no parameters contradict or conflict with each other, you should have a working production overcloud according to the deployment planning parameters in your environment files.

Chapter 3

RHOSP Architecture

RHOSP is the OpenStack platform developed by Red Hat based on the TripleO open source project. This chapter examines this RHOSP architecture in light of the previous TripleO architecture that you saw in Chapter 2, by describing the undercloud/director services/containers/templates/environment files, the different overcloud nodes, and the related services/containers and underlay networking requirements for communication between different nodes and services.

Red Hat OpenStack Director

The multiple functions and components of TripleO were described in Chapter 2. In Red Hat Director there is no difference. Undercloud is following the TripleO project. To achieve the described function and services architecture stated earlier, Director has containers to run the needed services (see Figure 3.1).

```
(undercloud) [stack@tripleo-director ~]$ sudo podman ps
CONTAINER ID  IMAGE                                                                COMMAND               CREATED      STATUS          PORTS  NAMES
78a9131ee376  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-nova-compute-ironic:16.1  kolla_start           7 months ago  Up 8 days ago          nova_compute
362922081d0ef  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-ironic-inspector:16.1     kolla_start           7 months ago  Up 8 days ago          ironic_inspector_dnsmasq
04222e83b091  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-ironic-inspector:16.1     kolla_start           7 months ago  Up 8 days ago          ironic_inspector
b11affa68f36  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-ironic-pxe:16.1           kolla_start           7 months ago  Up 8 days ago          ironic_pxe_http
7a5faeaff44e  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-ironic-pxe:16.1           /bin/bash -c BIND...  7 months ago  Up 8 days ago          ironic_pxe_tftp
4b3860b3bc50  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-ironic-neutron-agent:16.1  kolla_start           7 months ago  Up 8 days ago          ironic_neutron_agent
4fe963f17150  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-ironic-conductor:16.1      kolla_start           7 months ago  Up 8 days ago          ironic_conductor
ea84e3f057b5  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-nistral-api:16.1          kolla_start           7 months ago  Up 8 days ago          mistral_api
c85791365aa6  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-neutron-openvswitch-agent:16.1  kolla_start           7 months ago  Up 8 days ago          neutron_ovs_agent
a675820bbbd1  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-neutron-l3-agent:16.1     kolla_start           7 months ago  Up 8 days ago          neutron_l3_agent
4551e51f3af7  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-neutron-dhcp-agent:16.1   kolla_start           7 months ago  Up 8 days ago          neutron_dhcp
6769a083de66  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-ironic-api:16.1           kolla_start           7 months ago  Up 8 days ago          ironic_api
dc9a9bcacc59  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-nova-api:16.1             kolla_start           7 months ago  Up 8 days ago          nova_api_cron
2d4710fc8880  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-swift-proxy-server:16.1   kolla_start           7 months ago  Up 8 days ago          swift_proxy
52e311a3fe9d  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-nova-api:16.1             kolla_start           7 months ago  Up 8 days ago          nova_api
a7e90db4eac7  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-glance-api:16.1           kolla_start           7 months ago  Up 8 days ago          glance_api
684d88b85172  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-placement-api:16.1        kolla_start           7 months ago  Up 8 days ago          placement_api
5ac3c5659dcf  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-zaqar-wsgi:16.1           kolla_start           7 months ago  Up 8 days ago          zaqar_websocket
48bb6e46feeb  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-zaqar-wsgi:16.1           kolla_start           7 months ago  Up 8 days ago          zaqar
070a08d633c9  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-swift-object:16.1         kolla_start           7 months ago  Up 8 days ago          swift_rsync
2f7ea687fc17  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-swift-object:16.1         kolla_start           7 months ago  Up 8 days ago          swift_object_updater
93c2023e07ba  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-swift-object:16.1         kolla_start           7 months ago  Up 8 days ago          swift_object_server
60e66f0fb2a8  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-swift-proxy-server:16.1   kolla_start           7 months ago  Up 8 days ago          swift_object_expirer
1cd21ffe7b57  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-swift-container:16.1      kolla_start           7 months ago  Up 8 days ago          swift_container_updater
30be838e9cf0  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-swift-container:16.1      kolla_start           7 months ago  Up 8 days ago          swift_container_server
bd8e5275d68e  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-swift-account:16.1       kolla_start           7 months ago  Up 8 days ago          swift_account_server
67731013fdf7  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-swift-account:16.1       kolla_start           7 months ago  Up 8 days ago          swift_account_reaper
4a983bcc0a16  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-nova-scheduler:16.1      kolla_start           7 months ago  Up 8 days ago          nova_scheduler
fa1951a693d3  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-nova-conductor:16.1      kolla_start           7 months ago  Up 8 days ago          nova_conductor
c99ab196791a  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-neutron-server:16.1      kolla_start           7 months ago  Up 8 days ago          neutron_api
d4dc3667c339  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-mistral-executor:16.1    kolla_start           7 months ago  Up 8 days ago          mistral_executor
7a83d20cc1d6  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-mistral-event-engine:16.1  kolla_start           7 months ago  Up 8 days ago          mistral_event_engine
d02756d43d10  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-nistral-engine:16.1      kolla_start           7 months ago  Up 8 days ago          mistral_engine
3998bf2f4c6e1  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-cron:16.1                kolla_start           7 months ago  Up 8 days ago          logrotate_crond
1d01f8030943  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-heat-engine:16.1         kolla_start           7 months ago  Up 8 days ago          heat_engine
709d770f6969  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-heat-api:16.1           kolla_start           7 months ago  Up 8 days ago          heat_api_cron
d2feeb7c6898  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-heat-api:16.1           kolla_start           7 months ago  Up 8 days ago          heat_api
962f5a377412  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-keystone:16.1           kolla_start           7 months ago  Up 8 days ago          keystone
a8b44255f1b3  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-iscsid:16.1             kolla_start           7 months ago  Up 8 days ago          iscsid
f834a07fdefa  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-mariadb:16.1             kolla_start           7 months ago  Up 8 days ago          mysql
6726380b244f  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-haproxy:16.1             kolla_start           7 months ago  Up 8 days ago          haproxy
ce97b864895c  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-memcached:16.1           kolla_start           7 months ago  Up 8 days ago          memcached
81aa78677ae0  tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-keepalived:16.1          /usr/local/bin/ko...  7 months ago  Up 8 days ago          keepalived
(undercloud) [stack@tripleo-director ~]$
```

Figure 3.1 *RHOSP 16 Undercloud Containers*

As you can see, the containers that exist on undercloud are related to the needed services to deploy the overcloud.

The question here is how do you instruct those services in the correct way and with the parameters needed to deploy the overcloud stack, as the complex Heat stack consists of small stacks and objects? The short answer is by using *deployment templates* and *environment files*, however, understanding how to provide these parameters is too complex if you are not at least familiar with the overall process. Let's explain and clarify the main files needed to deploy and customize overcloud.

Director Templates and Environment Files

The TripleO templates and environment files are the most important part as they are the core files that enable the Red Hat OpenStack plus Contrail networking stack deployment, update, and upgrade.

RHOSPd Directory Structure and Templates

Red Hat OpenStack Director uses the Heat template to create the overcloud stack. The purpose of the template is to define the resources that form the stack and the configuration of these resources.

Figure 3.2 is an example of overcloud stack hierarchy created by issuing the `OpenStack stack list –nested` command.

```
+--------------------------------------+--------------------------------------------------------------------+-----------------+----------------------+----------------------+--------------------------------------+
| ID                                   | Stack Name                                                         | Stack Status    | Creation Time        | Updated Time         | Parent                               |
+--------------------------------------+--------------------------------------------------------------------+-----------------+----------------------+----------------------+--------------------------------------+
| bd72f2b7-8d78-4f85-be71-c9fe5f3c5da8 | overcloud-AllNodesDeploySteps-wqihzbunmpbs-ContrailControllerExtraConfigPost-6kskzss6rjme | fec63d928ed3466c89ac052ebfc78f40 | UPDATE_COMPLETE | 2021-10-06T09:08:05Z | 2021-10-07T09:12:23Z | 1925092a-8710-4db7-9f88-f43e29b52ff5 |
| 2e957779-4097-4f62-b403-18cc658e1e70 | overcloud-AllNodesDeploySteps-wqihzbunmpbs-ComputeExtraConfigPost-2rlvgqtarqvd | fec63d928ed3466c89ac052ebfc78f40 | UPDATE_COMPLETE | 2021-10-06T09:08:04Z | 2021-10-07T09:12:22Z | 1925092a-8710-4db7-9f88-f43e29b52ff5 |
| 0463b3e4-60a9-4d7c-aaf6-c9dfddcbe7d5 | overcloud-AllNodesDeploySteps-wqihzbunmpbs-ControllerExtraConfigPost-2zcfbye5rab6 | fec63d928ed3466c89ac052ebfc78f40 | UPDATE_COMPLETE | 2021-10-06T09:08:00Z | 2021-10-07T09:12:18Z | 1925092a-8710-4db7-9f88-f43e29b52ff5 |
| 1925092a-8710-4db7-9f88-f43e29b52ff5 | overcloud-AllNodesDeploySteps-wqihzbunmpbs                         | fec63d928ed3466c89ac052ebfc78f40 | UPDATE_COMPLETE | 2021-10-06T09:07:57Z | 2021-10-07T09:12:15Z | c651a396-69c4-467d-9e5b-aa2ae4fa9875 |
| 4058654e-9696-46d0-b172-44d2e3b443f1 | overcloud-ComputeIpListMap-zpmbucks2vyz                            | fec63d928ed3466c89ac052ebfc78f40 | UPDATE_COMPLETE | 2021-10-06T09:07:55Z | 2021-10-07T09:11:13Z | c651a396-69c4-467d-9e5b-aa2ae4fa9875 |
| f5a0ff7d-a9f6-4592-8b34-d674cb55c35c | overcloud-ControllerIpListMap-4thrq2nybwl2                         | fec63d928ed3466c89ac052ebfc78f40 | UPDATE_COMPLETE | 2021-10-06T09:07:53Z | 2021-10-07T09:12:13Z | c651a396-69c4-467d-9e5b-aa2ae4fa9875 |
| 8381940f-9768-479a-9154-2914246da566 | overcloud-Compute-wyq4pxlhgbgo-1-o5kc7xlx5iei-NodeExtraConfig-f3cdmdp5576c | fec63d928ed3466c89ac052ebfc78f40 | UPDATE_COMPLETE | 2021-10-06T09:07:50Z | 2021-10-07T09:11:07Z | 9fe79f45-0104-45d5-8736-9fc5f77c289e |
| 287b5382-724b-449d-96a7-daf8b3922456 | overcloud-Controller-ouoftocyvmuw-0-5b3puvtsqk2f-NodeExtraConfig-kvydamq4rp6i | | | | | |
```

Figure 3.2 Sample Overcloud Stack Hierarchy

After installing Red Hat OpenStack director you find OpenStack TripleO Heat templates in the `/usr/share/OpenStack-tripleo-heat-templates/` directory (see Figure 3.3). The version of the installed template is dependent on the TripleO version client, so check the template version to know what version you're using.

```
(undercloud) [stack@tripleo-director ~]$ ls /usr/share/openstack-tripleo-heat-templates/ -al
total 304
drwxr-xr-x.  17 root root  4096 Feb 20  2021 .
drwxr-xr-x. 125 root root  4096 Feb 23  2021 ..
-rw-r--r--.   1 root root  1512 Aug 21  2020 all-nodes-validation.yaml
-rw-r--r--.   1 root root 27362 Aug 21  2020 capabilities-map.yaml
drwxr-xr-x.   5 root root    73 Feb 20  2021 ci
drwxr-xr-x.   3 root root  4096 Feb 20  2021 common
-rw-r--r--.   1 root root  1572 Dec  2  2020 config-download-software.yaml
-rw-r--r--.   1 root root  1574 Dec  2  2020 config-download-structured.yaml
drwxr-xr-x.   4 root root  4096 Feb 20  2021 container_config_scripts
-rw-r--r--.   1 root root   676 Aug 21  2020 default_passwords.yaml
drwxr-xr-x.   3 root root   213 Feb 20  2021 deployed-server
drwxr-xr-x.  69 root root  4096 Feb 20  2021 deployment
drwxr-xr-x.  17 root root  8192 Feb 20  2021 environments
drwxr-xr-x.   5 root root    63 Feb 20  2021 extraconfig
drwxr-xr-x.   2 root root   260 Feb 20  2021 firstboot
-rw-r--r--.   1 root root   220 Aug 21  2020 j2_excludes.yaml
-rw-r--r--.   1 root root  4050 Aug 21  2020 net-config-bond.j2.yaml
-rw-r--r--.   1 root root  3763 Aug 21  2020 net-config-bridge.j2.yaml
-rw-r--r--.   1 root root  4021 Aug 21  2020 net-config-linux-bridge.j2.yaml
-rw-r--r--.   1 root root  3031 Aug 21  2020 net-config-noop.j2.yaml
-rw-r--r--.   1 root root  5101 Aug 21  2020 net-config-standalone.j2.yaml
-rw-r--r--.   1 root root  4173 Aug 21  2020 net-config-static-bridge.j2.yaml
-rw-r--r--.   1 root root  4251 Aug 21  2020 net-config-static-bridge-with-external-dhcp.j2.yaml
-rw-r--r--.   1 root root  3921 Aug 21  2020 net-config-static.j2.yaml
-rw-r--r--.   1 root root  4987 Aug 21  2020 net-config-undercloud.j2.yaml
drwxr-xr-x.   6 root root   203 Feb 20  2021 network
-rw-r--r--.   1 root root  5717 Aug 21  2020 network_data_dashboard.yaml
-rw-r--r--.   1 root root  8901 Sep  5  2020 network_data_ganesha.yaml
-rw-r--r--.   1 root root 10356 Aug 21  2020 network_data_routed.yaml
-rw-r--r--.   1 root root  8932 Aug 21  2020 network_data_subnets_routed.yaml
-rw-r--r--.   1 root root    52 Aug 21  2020 network_data_undercloud.yaml
-rw-r--r--.   1 root root  7751 Aug 21  2020 network_data.yaml
-rw-r--r--.   1 root root 45863 Aug 21  2020 overcloud.j2.yaml
-rw-r--r--.   1 root root 22122 Dec  2  2020 overcloud-resource-registry-puppet.j2.yaml
-rw-r--r--.   1 root root   198 Dec  2  2020 plan-environment.yaml
drwxr-xr-x.   2 root root    68 Feb 20  2021 plan-samples
drwxr-xr-x.   4 root root    94 Feb 20  2021 puppet
drwxr-xr-x.   2 root root  4096 Feb 20  2021 roles
-rw-r--r--.   1 root root  4221 Aug 21  2020 roles_data_undercloud.yaml
-rw-r--r--.   1 root root 17280 Dec  2  2020 roles_data.yaml
drwxr-xr-x.   2 root root    36 Feb 20  2021 scripts
drwxr-xr-x.   2 root root  4096 Feb 20  2021 tools
drwxr-xr-x.   2 root root    26 Feb 20  2021 validation-scripts
```

Figure 3.3 *Openstack TripleO Heat Template Directory Structure*

As shown in Figure 3.3, the TripleO Heat template directory contains subdirectories that also contain subsequent directories and files to provide the code logic for deploying overcloud stack and its resources.

You have seen the OpenStack templates that provide the code logic for deploying the overcloud stack and its resources, however, when it comes to Contrail plus RHOSP deployment, there should be a similar code logic to provide a way of deploying Contrail as an SDN plug-in into the stack.

Contrail has also its Heat templates to provide the method to create and configure Contrail nodes and resources as an integral part of the Overcloud stack. Contrail heat templates are downloadable from https://support.juniper.net/support/downloads/?p=contrail-networking.

Select the target version, which in our case is the RHEL 8 plus RHOSP16 OOO Heat templates. Figure 3.4 is an example of the Contrail Heat template structure.

```
(undercloud) [stack@tripleo-director ~]$ ls contrail-tripleo-heat-templates -al
total 128
drwxr-xr-x.  9 stack stack  4096 Dec 16  2020 .
drwx------. 16 stack stack  4096 Oct 19 11:35 ..
drwxr-xr-x.  3 stack stack    22 Dec 16  2020 deployment
drwxr-xr-x.  3 stack stack    22 Dec 16  2020 environments
drwxr-xr-x.  4 stack stack    43 Dec 16  2020 extraconfig
drwxr-xr-x.  8 stack stack   163 Dec 16  2020 .git
-rw-r--r--.  1 stack stack    25 Dec 16  2020 .gitignore
-rw-r--r--.  1 stack stack   129 Dec 16  2020 .gitreview
-rw-r--r--.  1 stack stack 11358 Dec 16  2020 LICENSE
drwxr-xr-x.  3 stack stack    20 Dec 16  2020 network
-rw-r--r--.  1 stack stack 32623 Dec 16  2020 README.md
drwxr-xr-x.  2 stack stack    30 Dec 16  2020 roles
-rw-r--r--.  1 stack stack 22965 Dec 16  2020 roles_data_contrail_aio.yaml
-rw-r--r--.  1 stack stack 14634 Dec 16  2020 roles_data_contrail_ml2.yaml
-rw-r--r--.  1 stack stack 25874 Dec 16  2020 roles_data_contrail.yaml
drwxr-xr-x.  3 stack stack    22 Dec 16  2020 tools
```

Figure 3.4 Contrail TripleO Heat Templates Directory Structure

It is obvious that both the Contrail Heat templates and the OpenStack Heat templates have the same directory structure from the main subdirectories perspective. This makes it easy to merge them together to form one template directory to deploy Contrail plus Red Hat OpenStack without struggling how to merge those two different code templates.

After downloading Contrail TripleO Heat templates into undercloud, merge both OpenStack and Contrail templates into one new directory to be used as an overcloud stack template directory. This will provide both OpenStack and Contrail deployment logic.

Environment Files

Environment files look no different than the deployment templates. It's why many operations and deployment engineers are not able to distinguish between environment files and deployment templates at first glance.

While environment files are also a kind of template, they have a different purpose that provides customization for the Heat templates and pass the needed parameters to fulfill your specific customized overcloud setup. The templates provide a way of creating a controller node with the services and the containers needed to run on it, while environment files provide information about how many controllers you want in the deployment and where to place them on physical servers.

NOTE There are a huge number of possible customizations and subsequently lots of environment files that could be used. This *Day One* book will attempt to explain the most important files used later in its deployment chapters.

The deploy command used to kick off this book's deployment is:

```
openstack overcloud deploy --templates ~/tripleo-heat-templates \
  -e ~/containers-prepare-parameter.yaml \
  -e ~/tripleo-heat-templates/environments/network-isolation.yaml \
  -e ~/tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
  -e ~/tripleo-heat-templates/environments/contrail/nodes-placement.yaml \
  -e ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
  -e ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
  -e ~/tripleo-heat-templates/environments/ips-from-pool-all.yaml \
  -e ~/tripleo-heat-templates/environments/fixed-ip-vips.yaml \
  -e ~/rhsm.yaml \
  -r ~/tripleo-heat-templates/roles_data_contrail.yaml
```

In this command there are three different types of arguments and parameters for the purpose of passing the customized information to the deployment workflow:

- `--templates` which pass the Heat template file directories containing the code logic for OpenStack plus Contrail.
- `-e` which passes the environment files used to customize the overcloud deployment.
- `-r` which passes the different node roles and each role's related services.

Let's explore the different files, how to use them, and their related files in case of nested calling. Some of these files are not found by default on the environment directory and are generated from the template processing step in the middle of deployment. We'll walk through how to do this later in the deployment chapters.

Containers-prepare-parameter.yaml

This undercloud configuration requires initial registry configuration to determine where to obtain and how to store images. For that purpose, before installing undercloud, you need to generate the file that contains the information about the registry and where those images are going to be stored in overcloud deployment. Undercloud uses this information to obtain container images for OpenStack services. Here's a sample file:

```
(undercloud) [stack@tripleo-director ~]$ cat ~/containers-prepare-parameter.yaml
# Generated with the following on 2021-02-20T11:54:36.995567
#
#   openstack tripleo container image prepare default --local-push-destination --output-env-
file containers-prepare-parameter.yaml
#

parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    excludes:
      - ceph
      - prometheus
    set:
      ceph_alertmanager_image: ose-prometheus-alertmanager
```

```
    ceph_alertmanager_namespace: registry.redhat.io/openshift4
    ceph_alertmanager_tag: 4.1
    ceph_grafana_image: rhceph-4-dashboard-rhel8
    ceph_grafana_namespace: registry.redhat.io/rhceph
    ceph_grafana_tag: 4
    ceph_image: rhceph-4-rhel8
    ceph_namespace: registry.redhat.io/rhceph
    ceph_node_exporter_image: ose-prometheus-node-exporter
    ceph_node_exporter_namespace: registry.redhat.io/openshift4
    ceph_node_exporter_tag: v4.1
    ceph_prometheus_image: ose-prometheus
    ceph_prometheus_namespace: registry.redhat.io/openshift4
    ceph_prometheus_tag: 4.1
    ceph_tag: latest
    name_prefix: openstack-
    name_suffix: ''
    namespace: registry.redhat.io/rhosp-rhel8
    neutron_driver: null
    rhel_containers: false
    tag: '16.1'
  tag_from_label: '{version}-{release}'
ContainerImageRegistryCredentials:
  registry.redhat.io:
    username: 'password'
```

Table 3.1 lists the most important parameters in this file.

Table 3.1 Key Parameters in containers-prepare-parameter.yaml

Parameter	Description
Push Destination	Defines the name space of the registry that you want to push images to. Setting it to true means you are going to use undercloud registry to store container images.
Neutron Driver	Defines the neutron driver you are going to use. By default it is set to ovn. As we use Contrail in our deployment it should be set to null.
Tag	Defines the version number of container images you need to pull it from Red Hat registry to be used in deployment. In this case we are using 16.1.
Container Image Registery Credentials	Defines the container image registry you pull images from, in this case "registery.redhat.io" and the username and password to log in to that registry.

Network-isolation.yaml

The overcloud assigns services to the provisioning network by default, however, the director can divide overcloud network traffic into isolated networks. To host specific types of network traffic in isolation you need to use the network-isolation.yaml file. This file doesn't exist by default. It is the result of processing templates with the composable roles file discussed later in this chapter. After the template processing director renders the network-isolation.j2.yaml file it produces the network isolation file. Here's a sample of the network isolation file:

```
(undercloud) [stack@tripleo-director tripleo-heat-templates]$ cat ~/tripleo-heat-templates/
environments/network-isolation.yaml
# Enable the creation of Neutron networks for isolated Overcloud
# traffic and configure each role to assign ports (related
# to that role) on these networks.
resource_registry:
  # networks as defined in network_data.yaml
  OS::TripleO::Network::Storage: ../network/storage.yaml
  OS::TripleO::Network::StorageMgmt: ../network/storage_mgmt.yaml
  OS::TripleO::Network::InternalApi: ../network/internal_api.yaml
  OS::TripleO::Network::Tenant: ../network/tenant.yaml
  OS::TripleO::Network::External: ../network/external.yaml
  OS::TripleO::Network::Management: ../network/management.yaml

  # Port assignments for the VIPs
  OS::TripleO::Network::Ports::StorageVipPort: ../network/ports/storage.yaml
  OS::TripleO::Network::Ports::StorageMgmtVipPort: ../network/ports/storage_mgmt.yaml
  OS::TripleO::Network::Ports::InternalApiVipPort: ../network/ports/internal_api.yaml
  OS::TripleO::Network::Ports::ExternalVipPort: ../network/ports/external.yaml
  OS::TripleO::Network::Ports::RedisVipPort: ../network/ports/vip.yaml
  OS::TripleO::Network::Ports::OVNDBsVipPort: ../network/ports/vip.yaml

  # Port assignments by role, edit role definition to assign networks to roles.
  # Port assignments for the Controller
  OS::TripleO::Controller::Ports::StoragePort: ../network/ports/storage.yaml
  OS::TripleO::Controller::Ports::StorageMgmtPort: ../network/ports/storage_mgmt.yaml
  OS::TripleO::Controller::Ports::InternalApiPort: ../network/ports/internal_api.yaml
  OS::TripleO::Controller::Ports::ExternalPort: ../network/ports/external.yaml
  # Port assignments for the Compute
  OS::TripleO::Compute::Ports::StoragePort: ../network/ports/storage.yaml
  OS::TripleO::Compute::Ports::InternalApiPort: ../network/ports/internal_api.yaml
  OS::TripleO::Compute::Ports::TenantPort: ../network/ports/tenant.yaml
  # Port assignments for the ContrailController
  OS::TripleO::ContrailController::Ports::InternalApiPort: ../network/ports/internal_api.yaml
  OS::TripleO::ContrailController::Ports::TenantPort: ../network/ports/tenant.yaml
  OS::TripleO::ContrailController::Ports::ExternalPort: ../network/ports/external.yaml
  # Port assignments for the ContrailControlOnly()
  OS::TripleO::ContrailControlOnly::Ports::InternalApiPort: ../network/ports/internal_api.yaml
  OS::TripleO::ContrailControlOnly::Ports::TenantPort: ../network/ports/tenant.yaml
  # Port assignments for the ContrailAnalytics
  OS::TripleO::ContrailAnalytics::Ports::InternalApiPort: ../network/ports/internal_api.yaml
  OS::TripleO::ContrailAnalytics::Ports::TenantPort: ../network/ports/tenant.yaml
  # Port assignments for the ContrailAnalyticsDatabase
  OS::TripleO::ContrailAnalyticsDatabase::Ports::InternalApiPort: ../network/ports/internal_api.
yaml
  OS::TripleO::ContrailAnalyticsDatabase::Ports::TenantPort: ../network/ports/tenant.yaml
  # Port assignments for the ContrailDpdk
  OS::TripleO::ContrailDpdk::Ports::StoragePort: ../network/ports/storage.yaml
  OS::TripleO::ContrailDpdk::Ports::InternalApiPort: ../network/ports/internal_api.yaml
  OS::TripleO::ContrailDpdk::Ports::TenantPort: ../network/ports/tenant.yaml
  # Port assignments for the ContrailSriov
  OS::TripleO::ContrailSriov::Ports::StoragePort: ../network/ports/storage.yaml
  OS::TripleO::ContrailSriov::Ports::InternalApiPort: ../network/ports/internal_api.yaml
  OS::TripleO::ContrailSriov::Ports::TenantPort: ../network/ports/tenant.yaml
  # Port assignments for the ContrailTsn
  OS::TripleO::ContrailTsn::Ports::StoragePort: ../network/ports/storage.yaml
  OS::TripleO::ContrailTsn::Ports::InternalApiPort: ../network/ports/internal_api.yaml
  OS::TripleO::ContrailTsn::Ports::TenantPort: ../network/ports/tenant.yaml
  # Port assignments for the ContrailControllerIssu
  OS::TripleO::ContrailControllerIssu::Ports::InternalApiPort: ../network/ports/internal_api.yaml
  OS::TripleO::ContrailControllerIssu::Ports::TenantPort: ../network/ports/tenant.yaml
```

You can see that the file contains the resource registry for all networking resources needed to be created for the overcloud, including networks and subnets, services, VIPs ports, and nodes ports (for example, controller storage port, Contrail controller tenant port). Every resource points to the yaml files that contain that resource's definition structure.

Contrail-plugins.yaml

This file contains the logic of the Contrail plugin integration and configuration of different services, pre-networking configurations, disabled neutron services, and OVN driver and services parameters. This file is a standard file and in most cases there is no need to modify any parameter within it:

```
(undercloud) [stack@tripleo-director tripleo-heat-templates]$ cat ~/tripleo-heat-templates/environments/
contrail/contrail-plugins.yaml
# A Heat environment file which can be used to enable OpenContrail
# # extensions, configured via puppet
resource_registry:
  OS::TripleO::Services::ContrailAnalytics: ../../deployment/contrail/contrail-analytics.yaml
  OS::TripleO::Services::ContrailAnalyticsAlarm: ../../deployment/contrail/contrail-analytics-alarm.yaml
  OS::TripleO::Services::ContrailAnalyticsDatabase: ../../deployment/contrail/contrail-analytics-database.
yaml
  OS::TripleO::Services::ContrailAnalyticsSnmp: ../../deployment/contrail/contrail-analytics-snmp.yaml
  OS::TripleO::Services::ContrailConfigDatabase: ../../deployment/contrail/contrail-config-database.yaml
  OS::TripleO::Services::ContrailConfig: ../../deployment/contrail/contrail-config.yaml
  OS::TripleO::Services::ContrailControl: ../../deployment/contrail/contrail-control.yaml
  OS::TripleO::Services::ContrailControlOnly: ../../deployment/contrail/contrail-control-only.yaml
  OS::TripleO::Services::ContrailWebui: ../../deployment/contrail/contrail-webui.yaml
  OS::TripleO::Services::ContrailVrouter: ../../deployment/contrail/contrail-vrouter.yaml
  OS::TripleO::Services::ContrailDpdk: ../../deployment/contrail/contrail-vrouter-dpdk.yaml
  OS::TripleO::Services::ContrailSriov: ../../deployment/contrail/contrail-vrouter-sriov.yaml
  OS::TripleO::Services::ContrailTsn: ../../deployment/contrail/contrail-vrouter-tsn.yaml
  OS::TripleO::Services::NeutronCorePlugin: ../../deployment/contrail/contrail-neutron-container-plugin.
yaml
  OS::TripleO::Services::ContrailHeatPlugin: ../../deployment/contrail/contrail-heat-container-plugin.yaml
  OS::TripleO::Services::ComputeNeutronCorePlugin: ../../deployment/contrail/contrail-vrouter.yaml
  OS::TripleO::Services::ContrailCertmongerUser: OS::Heat::None
  OS::TripleO::Services::ContrailIssu: OS::Heat::None
  OS::TripleO::Services::ContrailIssuControl: OS::Heat::None
  OS::TripleO::Services::ContrailIssuCompute: OS::Heat::None
  OS::TripleO::Services::ContrailIssuScript: OS::Heat::None
  OS::TripleO::ContrailIssuInit: OS::Heat::None
  OS::TripleO::ContrailVrouterIssu: OS::Heat::None
  OS::TripleO::ContrailHeatIssu: OS::Heat::None
  OS::TripleO::ContrailNeutronIssu: OS::Heat::None
  OS::TripleO::Compute::PreNetworkConfig: ../../extraconfig/pre_network/contrail/compute_pre_network.yaml
  OS::TripleO::ContrailAio::PreNetworkConfig: ../../extraconfig/pre_network/contrail/compute_pre_network.
yaml
  OS::TripleO::ContrailTsn::PreNetworkConfig: ../../extraconfig/pre_network/contrail/contrail_tsn_pre_
network.yaml
  OS::TripleO::ContrailDpdk::PreNetworkConfig: ../../extraconfig/pre_network/contrail/contrail_dpdk_pre_
network.yaml
  OS::TripleO::ContrailSriov::PreNetworkConfig: ../../extraconfig/pre_network/contrail/contrail_sriov_pre_
network.yaml
  OS::TripleO::ControllerExtraConfigPre: ../../extraconfig/pre_deploy/contrail/contrail-init.yaml
```

```
  OS::TripleO::ContrailControllerExtraConfigPre: ../../extraconfig/pre_deploy/contrail/contrail-init.yaml
  OS::TripleO::ContrailControlOnlyExtraConfigPre: ../../extraconfig/pre_deploy/contrail/contrail-init.yaml
  OS::TripleO::ContrailAnalyticsExtraConfigPre: ../../extraconfig/pre_deploy/contrail/contrail-init.yaml
  OS::TripleO::ContrailAnalyticsAlarmExtraConfigPre: ../../extraconfig/pre_deploy/contrail/contrail-init.
yaml
  OS::TripleO::ContrailAnalyticsDatabaseExtraConfigPre: ../../extraconfig/pre_deploy/contrail/contrail-init.
yaml
  OS::TripleO::ContrailAnalyticsSnmpExtraConfigPre: ../../extraconfig/pre_deploy/contrail/contrail-init.yaml
  OS::TripleO::ComputeExtraConfigPre: ../../extraconfig/pre_deploy/contrail/contrail-init.yaml
  OS::TripleO::ContrailDpdkExtraConfigPre: ../../extraconfig/pre_deploy/contrail/contrail-init.yaml
  OS::TripleO::ContrailSriovExtraConfigPre: ../../extraconfig/pre_deploy/contrail/contrail-init.yaml
  OS::TripleO::ContrailTsnExtraConfigPre: ../../extraconfig/pre_deploy/contrail/contrail-init.yaml
  OS::TripleO::ContrailAioExtraConfigPre: ../../extraconfig/pre_deploy/contrail/contrail-init.yaml

  # Agilio by default is disabled
  OS::TripleO::Services::AgilioVrouter: OS::Heat::None

  # Disable OVN services
  OS::TripleO::Services::OVNController: OS::Heat::None
  OS::TripleO::Services::OVNDBs: OS::Heat::None
  OS::TripleO::Services::OVNMetadataAgent: OS::Heat::None
  OS::TripleO::Network::Ports::OVNDBsVipPort: ../../network/ports/noop.yaml

  # Disabling Neutron services
  OS::TripleO::Docker::NeutronMl2PluginBase: OS::Heat::None
  OS::TripleO::Services::NeutronOvsAgent: OS::Heat::None
  OS::TripleO::Services::ComputeNeutronOvsAgent: OS::Heat::None
  OS::TripleO::Services::NeutronL3Agent: OS::Heat::None
  OS::TripleO::Services::NeutronMetadataAgent: OS::Heat::None
  OS::TripleO::Services::ComputeNeutronL3Agent: OS::Heat::None
  OS::TripleO::Services::ComputeNeutronMetadataAgent: OS::Heat::None
  OS::TripleO::Services::NeutronDhcpAgent: OS::Heat::None

parameter_defaults:
  NeutronCorePlugin: neutron_plugin_contrail.plugins.opencontrail.contrail_plugin.
NeutronPluginContrailCoreV2
  NeutronServicePlugins: ''
  NeutronTunnelTypes: ''
  NeutronMechanismDrivers: ''
  NeutronTypeDrivers: ''
  NeutronNetworkType: ''
  NeutronPluginExtensions: ''
  NeutronFlatNetworks: ''
  NeutronTunnelIdRanges: ''
  NeutronNetworkVLANRanges: ''
  NeutronVniRanges: ''

  NeutronApiOptVolumes:
    - '/var/lib/contrail/plugin_contrail_neutron:/opt/plugin:ro'
    - '/etc/contrail/ssl:/etc/contrail/ssl:ro'
  NeutronApiOptEnvVars:
      PYTHONPATH: '/opt/plugin/python3/site-packages'
  NovaComputeOptEnvVars:
      PATH: '/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin/:/opt/plugin/bin'
  NovaComputeOptVolumes:
    - /var/lib/contrail/vrouter_port_control:/opt/plugin/bin:ro
    - /var/lib/contrail:/var/lib/contrail:z
    - /var/lib/contrail/vif_nova_contrail:/usr/lib/python3.6/site-packages/nova_contrail_vif:ro
```

```
    - /var/lib/contrail/vif_nova_contrail_egg:/usr/lib/python3.6/site-packages/nova_contrail_
vif-0.1-py2.7.egg-info:ro
    - /var/lib/contrail/vif_plug_vrouter:/usr/lib/python3.6/site-packages/vif_plug_vrouter:ro
    - /var/lib/contrail/vif_plug_contrail_vrouter:/usr/lib/python3.6/site-packages/vif_plug_contrail_
vrouter:ro
  HeatEngineOptVolumes:
    - '/etc/contrail/ssl:/etc/contrail/ssl:ro'
    - '/var/lib/contrail/plugin_contrail_heat:/opt/plugin:ro'
  HeatEngineOptEnvVars:
      PYTHONPATH: '/opt/plugin/python3/site-packages'
  HeatEnginePluginDirs:
    - '/opt/plugin/site-packages/vnc_api/gen/heat/resources'
    - '/opt/plugin/site-packages/contrail_heat/resources'
```

nodes-placement.yaml

This environment file contains the information about each node count. A definition of each node and node-related networks and services that exist in another file, roles_data_contrail.yaml, are discussed later in this chapter. The environment file also contains the tag information to schedule the node in the target server, as well as server naming:

```
(undercloud) [stack@tripleo-director tripleo-heat-templates]$ cat ~/tripleo-heat-templates/
environments/contrail/nodes-placement.yaml
parameter_defaults:
##############################################################
# Number of instances of each kind of role to be deployed #
##############################################################
  ControllerCount: 1
  ContrailControllerCount: 1
  ContrailAnalyticsCount: 1
  ContrailAnalyticsDatabaseCount: 1
  ComputeCount: 2
  ContrailDpdkCount: 0

##############################################################
# Role with BM Node Mapping in case of per nodeid placement #
##############################################################

 ControllerSchedulerHints:
    'capabilities:node': 'controller-%index%'
  ContrailControllerSchedulerHints:
    'capabilities:node': 'contrailcontroller-%index%'
  ContrailControlOnlySchedulerHints:
    'capabilities:node': 'contrailcontrolonly-%index%'
  ContrailAnalyticsSchedulerHints:
    'capabilities:node': 'contrailanalytics-%index%'
  ContrailAnalyticsDatabaseSchedulerHints:
    'capabilities:node': 'contrailanalyticsdatabase-%index%'
  ComputeSchedulerHints:
    'capabilities:node': 'compute-%index%'
  ContrailDpdkSchedulerHints:
    'capabilities:node': 'contraildpdk-%index%'
  ContrailSriovSchedulerHints:
    'capabilities:node': 'contrailsriov-%index%'
  ContrailDpdkSriovSchedulerHints:
    'capabilities:node': 'contraildpdksriov-%index%'
```

```
  ContrailControllerIssuSchedulerHints:
    'capabilities:node': 'contrailissu-%index%'
############################
# Openstack Servers naming #
############################

  HostnameMap:
    overcloud-controller-0: jnprct01
    overcloud-controller-1: jnprct02
    overcloud-controller-2: jnprct03

    overcloud-contrailcontroller-0: jnprcc01
    overcloud-contrailcontroller-1: jnprcc02
    overcloud-contrailcontroller-2: jnprcc03

    overcloud-contrailcontrolonly-0: jnprcco01
    overcloud-contrailcontrolonly-1: jnprcco02
    overcloud-contrailcontrolonly-2: jnprcco03

    overcloud-contrailanalytics-0: jnprca01
    overcloud-contrailanalytics-1: jnprca02
    overcloud-contrailanalytics-2: jnprca03

    overcloud-contrailanalyticsdatabase-0: jnprcadb01
    overcloud-contrailanalyticsdatabase-1: jnprcadb02
    overcloud-contrailanalyticsdatabase-2: jnprcadb03

    overcloud-compute-0: jnprcp01
    overcloud-compute-1: jnprcp02
    overcloud-compute-2: jnprcp03
```

The parameters in this file are sorted into three categories:

- The number of instances per node role
- Scheduler hints based on node tagging
- Instance naming

According to the role file, Table 3.2 lists the roles in our deployment.

Table 3.2 Nodes Placement Usage

Parameter	Role	Description
ControllerCount	Controller	This parameter describes the number of Openstack controller instances in our example. We used one Openstack controller.
ContrailControllerCount	ContrailController	This parameter describes the number of Contrail controller instances in our example. We used one Contrail Controller.

ContrailAnalyticsCount	ContrailAnalytics	This parameter describes the number of Contrail analytics instances in our example we used one Contrail Analytics.
ContrailAnalyticsDatabaseCount	ContrailAnalyticsDatabase	This parameter describes the number of Contrail analytics DB. We used one Contrail Analytics DB.
ComputeCount	Compute	This parameter describes the number of kernel vrouter computes. We used two kernel vrouter computes.
ContrailDpdkCount	ContrailDpdk	This parameter describes the number of DPDK vrouter compute instances in the example. We used zero DPDK vrouter computes.

The second type of parameter is *scheduler hints*. In our example we use node tagging to schedule Overcloud roles to specific nodes. That means putting a specific tag to each node used to carry specific Overcloud roles and then scheduling the role to the node using that tag. Table 3.3 provides an explanation to scheduler hints usage

Table 3.3 Scheduler Hints Usage

Parameter	Tag	Description
ControllerSchedulerHints	'capabilities:node': 'controller-%index%'	Openstack controller instances are scheduled to nodes tagged with 'capabilities:node': 'controller-%index%'.
ContrailControllerSchedulerHints	'capabilities:node': 'contrailcontroller-%index%'	Contrail controller instances are scheduled to nodes tagged with 'capabilities:node': 'contrailcontroller-%index%'
ContrailAnalyticsSchedulerHints	'capabilities:node': 'contrailanalytics-%index%'	Contrail Analytics instances are scheduled to nodes tagged with 'capabilities:node': 'contrailanalytics-%index%'
ContrailAnalyticsDatabaseSchedulerHints	'capabilities:node': 'contrailanalyticsdatabase-%index%'	Contrail analytics DB instances are scheduled to nodes tagged with 'capabilities:node': 'contrailanalyticsdatabase-%index%'
ComputeSchedulerHints	'capabilities:node': 'compute-%index%'	Kernel vrouter compute instances are scheduled to nodes tagged with 'capabilities:node': 'compute-%index%'

NOTE Index could be 0 or 1 or etc., based on instance numbering.

The third category of parameters in this file is instances naming and it is very straight forward as you can see in Table 3.4.

Table 3.4 Instance Naming Usage

Param	Name	Description
HostnameMap		Provides customized naming to different overcloud roles .
overcloud-controller-0	jnprct01	Under HostnameMap parameters there are many key value pairs to customize Overcloud instances naming. This entry is to name overcloud-controller-0 "jnprct01".
overcloud-contrailcontroller-0	jnprcc01	Xxxxx?

Contrail-services.yaml

This file contains many parameters to fine tune Contrail plus RHOSP deployment, so we are going to discuss the most important ones for the deployment to be successful. Some of the parameters are for performance tuning and are beyond the scope of this book.

The most important parameters in this file could be grouped into three main categories:

- ServiceNetMap
- Contrail registry information
- Contrail settings

The ServiceNetMap parameter determines the network types you use for each service. By default, each OpenStack service is assigned to a default network type in the resource registry. These services are bound to IP addresses within the network type's assigned network. The parameter overrides the default assignment and is needed whenever you use custom networks.

Contrail registry contains information about container images needed for the deployment of both RHOSP and Contrail. It also conveys whether the registry is secure or not, as well as the username and password to use. Most importantly it contains the tag of the targeted Contrail release.

Contrail settings contains basic information, the most important of which is basic configuration information like vRouter gateway, encapsulation priority for overlay traffic, controllers BGP AS, etc.

Here is a sample `contrail-services` file followed by Table 3.5 listing usage notes for the different parameters:

```
(undercloud) [stack@tripleo-director ~]$ cat ~/tripleo-heat-templates/environments/contrail/
contrail-services.yaml
parameter_defaults:
  ServiceNetMap:
    ContrailDatabaseNetwork: internal_api
    ContrailAnalyticsNetwork: internal_api
    ContrailAnalyticsAlarmNetwork: internal_api
    ContrailAnalyticsDatabaseNetwork: internal_api
    ContrailAnalyticsSnmpNetwork: internal_api
    ContrailConfigNetwork: internal_api
    ContrailControlNetwork: tenant
    ContrailControlOnlyNetwork: tenant
    ContrailWebuiNetwork: internal_api
    ContrailVrouterNetwork: tenant
    ContrailCertmongerUserNetwork: internal_api
###################################
# Security topics (RBAC activation) #
###################################

  NeutronMetadataProxySharedSecret: secret
  AAAMode: rbac

#############################################################
# Contrail Registry used for Contrail Containers download #
#############################################################

  ContrailRegistryInsecure: true
  DockerInsecureRegistryAddress:
   - 192.168.24.1:8787
   - tripleo-director.ctlplane.localdomain:8787
  ContrailRegistry: 192.168.24.1:8787/contrail
  ContrailImageTag: 2011.138

#########################################################################
# Bare Metal Disk Size requirement - installation will fail if not met #
#########################################################################

  ContrailAnalyticsDBMinDiskGB: 20
  ContrailConfigDBMinDiskGB: 20

####################
# Hugepages Tuning #
####################

  # Kernel vRouter hugepages setup
  ContrailVrouterHugepages2MB: 2096
  ContrailVrouterHugepages1GB: 4

############################
# VROUTER SPECIFIC TUNING #
############################

#  ContrailVrouterModuleOptions:
#     Vrouter kernel module options. This string is passed to agent container as env variable VROUTER_
```

```
MODULE_OPTIONS
#     it is saved as options in /etc/modprobe.d/vrouter.conf

  ContrailVrouterModuleOptions: "vr_nexthops=32768"

########################
# HOST CPU FINE TUNING #
########################

# Tuned profile configuration
#   TunedProfileName -  Name of tuned profile
#   IsolCpusList     -  Logical CPUs list to be isolated from the host process (applied via cpu-
partitioning tuned).
#                    It is mandatory to provide isolated cpus for tuned to achive optimal performance.
#                    Example: "3-8,12-15,18"
# These parameters are to be set per a role


  ComputeParameters:
    TunedProfileName: "cpu-partitioning"
    IsolCpusList: "1-7"
    NovaVcpuPinSet: "1-4"
    NovaComputeCpuSharedSet: "5-7"
    KernelArgs: " isolcpus=1-7"

######################################################
# Customer Contrail container configuration settings #
######################################################

  ContrailSettings:
    VROUTER_GATEWAY: 20.0.0.1
    ANALYTICS_STATISTICS_TTL: 2
    ANALYTICS_DATA_TTL: 2

# 16 bit BGP Autonomous System number for the cluster.
    BGP_ASN: 64512

# Supported values are true/false. When true, system will automatically create BGP peering mesh with
all control-nodes that have same BGP AS number as global AS number.
    BGP_AUTO_MESH: true

# Encapsulation priority order
    ENCAP_PRIORITY: 'MPLSoUDP,VXLAN,MPLSoGRE'

# Supported values are true/false. To be set to false in order to retrict introspect listining port on a
single IP.
    INTROSPECT_LISTEN_ALL: False

# Redis password to avoid any unauthenticated access
    REDIS_SERVER_PASSWORD: test123

# Vrouter dpdk options
    LACP_RATE: 0
```

Table 3.5 Contrail-services File Usage

Parameter	Option	Value	Description
ServiceNetMap			This parameter defines the services of network mapping by means of the options defined under it.
	ContrailDatabaseNetwork	internal_api	This means that Contrail DB services should use the internal_api network for communication
	ContrailControlNetwork	tenant	This means that Contrail control services should use tenant network for communication. Note that thhe same applies for all other options listed in the example.
ContrailRegistryInsecure		true	This means insecure registry is used for the deployment.
DockerInsecureRegistry-Address:			This parameter defines the registry address for OpenStack container images.
	tripleo-director.ctlplane.localdomain:8787		The OpenStack container image FQDN.
ContrailRegistry		192.168.24.1:8787/ contrail	This parameter defines the registry address for Contrail container images.
ContrailImageTag		2011.138	This parameter defines the tag of the Contrail targeted release for the deployment. It should match exactly the registry container image tags.
ContrailSettings:			This parameter defines the basic Contrail settings by means of the following options defined under it.
	VROUTER_GATEWAY	20.0.0.1	This option defines the vRouter gateway which is the gateway of the vRouter agent of every compute. It should match the GW of the tenant network.
	BGP_ASN	64512	This option defines the BGP autonomous system number of the Contrail controller
	BGP_AUTO_MESH	True	This option makes the Controller form full mesh BGP sessions if set to true.
	ENCAP_PRIORITY	MLSoUDP, VXLAN, MPLSoGRE	This option sets the encapsulation priority preference of the overlay traffic between vRouters and inbetween vRouters and the physical infrastructure.

Contrail-net.yaml

This file contains information about networking configuration, namely:

- Nic config templates for each node
- Networks prefix ranges
- Networks related routes
- VLANs
- Network services such as DNS, and NTP

Here is a sample contrail-net.yaml file followed by Table 3.6 listing the included parameters and how to use them:

```
(undercloud) [stack@tripleo-director ~]$ 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::ContrailAnalytics::Net::SoftwareConfig: ../../network/config/contrail/contrail-ca-nic-
config.yaml
  OS::TripleO::ContrailAnalyticsDatabase::Net::SoftwareConfig: ../../network/config/contrail/
contrail-ca-nic-config.yaml
  OS::TripleO::Compute::Net::SoftwareConfig: ../../network/config/contrail/compute-bond-nic-config.
yaml
  OS::TripleO::ContrailDpdk::Net::SoftwareConfig: ../../network/config/contrail/contrail-dpdk-bond-
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
  OS::TripleO::ContrailControllerIssu::Net::SoftwareConfig: ../../network/config/contrail/contrail-
controller-nic-config.yaml
  OS::TripleO::ContrailAio::Net::SoftwareConfig: ../../network/config/contrail/controller-nic-config-
aio.yaml

parameter_defaults:
  # Customize all these values to match the local environment
  TenantNetCidr: 20.0.0.0/24
  InternalApiNetCidr: 20.1.0.0/24
  ExternalNetCidr: 20.2.0.0/24
  StorageNetCidr: 20.3.0.0/24
  StorageMgmtNetCidr: 20.4.0.0/24

  # CIDR subnet mask length for provisioning network
  ControlPlaneSubnetCidr: '24'

  # Allocation pools
  TenantAllocationPools: [{'start': '20.0.0.200', 'end': '20.0.0.239'}]
  InternalApiAllocationPools: [{'start': '20.1.0.200', 'end': '20.1.0.239'}]
```

```
ExternalAllocationPools: [{'start': '20.2.0.200', 'end': '20.2.0.239'}]
StorageAllocationPools: [{'start': '20.3.0.200', 'end': '20.3.0.239'}]
StorageMgmtAllocationPools: [{'start': '20.4.0.200', 'end': '20.4.0.239'}]

# Routes
ControlPlaneDefaultRoute: 192.168.24.1
InternalApiDefaultRoute: 20.1.0.1
ExternalInterfaceDefaultRoute: 20.2.0.1

# Vlans
TenantNetworkVlanID: 700
InternalApiNetworkVlanID: 710
ExternalNetworkVlanID: 720
StorageNetworkVlanID: 730
StorageMgmtNetworkVlanID: 740

# Services
EC2MetadataIp: 192.168.24.1 # Generally the IP of the Undercloud
DnsServers: ["192.168.24.100"]
NtpServer: ntp.juniper.net
TimeZone: 'Europe/Paris'
```

Table 3.6 Contrail-net.yaml File Usage

Parameter	Option	Value	Description
resource_ registry:			Use this is to define each network resource related to every node.
	OS::TripleO::Controller ::Net::SoftwareConfig:	../../network/ config/contrail/ controller-nic-config.yaml	This option defines that it is the networking resource of OpenStack controller while the value reveals the location of the yaml containing the networking resource structure. Note: The location of the nic config yaml file is a relative location to the conrail-net.yaml file.
	OS::TripleO::ContrailController ::Net::SoftwareConfig	../../network/ config/contrail/ contrail-controller-nic-config.yaml	This option defines that it is the networking resource of the Contrail controller while the value reveals the location of the yaml file containing the networking resource structure.

	OS::TripleO::ContrailAnalytics ::Net::SoftwareConfig	../../network/ config/contrail/ contrail-ca-nic-config.yaml	This option defines that it is the networking resource of the Contrail analytics node while the value reveals the location of the yaml file containing the networking resource structure.
	OS::TripleO::ContrailAnalytics Database::Net::SoftwareConfig	../../network/ config/contrail/ contrail-ca-nic-config.yaml	This option defines that it is the networking resource of the Contrail analytics DB node while the value reveals the location of the yaml file containing the networking resource structure.
	OS::TripleO::Compute ::Net::SoftwareConfig	../../network/ config/contrail/ compute-bond-nic-config.yaml	This option defines that it is the networking resource of the kernel vRouter compute node while the value reveals the location of the yaml file containing the networking resource structure.
parameter_ defaults:			This defines the customized parameters to override the default parameters in the network-data.yaml file containing the default networking configuration for RHOSP deployment.
	TenantNetCidr:	20.0.0.0/24	Defines the tenant network prefix range.
	InternalApiNetCidr	20.1.0.0/24	Defines the internal-api network prefix range.
	ExternalNetCidr	20.2.0.0/24	Defines the external network prefix range.
	StorageNetCidr	20.3.0.0/24	Defines the storage network prefix range.
	StorageMgmtNetCidr	20.4.0.0/24	Defines the storage management network prefix range.
	ControlPlaneSubnetCidr	'24'	Defines the CIDR subnet mask length for the provisioning network.
	ControlPlaneDefaultRoute	192.168.24.1	Defines the default route next hop of the provisioning network. Note: You can define any default route to any network in this file and then use it in the nic yaml file.
	InternalApiDefaultRoute	20.1.0.1	Defines the default route next hop of the internel API network.

	ExternalInterfaceDefaultRoute	20.2.0.1	Defines the default route next hop of the external network
	TenantNetworkVlanID	700	Defines the VLAN tag associated with the tenant network. Note: If you use different composable network structure you can define VLANs related to that structure by following the right syntax.
	InternalApiNetworkVlanID	710	Defines the VLAN tag associated with the internal API network.
	ExternalNetworkVlanID	720	Defines the VLAN tag associated with the external network.
	StorageNetworkVlanID	730	Defines the VLAN tag associated with the storage network.
	StorageMgmtNetworkVlanID	740	Defines the VLAN tag associated with the storage management network.
	DnsServers	192.168.24.100	Defines the DNS server IP address.
	NtpServer	ntp.juniper.net	Defines the NTP server. Note: It is very important to have a functioning cluster. NTP FQDN should be resolvable by DNS.
	TimeZone	'Europe/Paris'	Defines the time zone.

Whew, after that you should know how to use the file. However, there is one more thing you need to be familiarized with to be able to customize the NIC configuration of every node. As explained in Table 3.6, the networking resources are defined and correlated with the NIC configuration template under the resource registry at the beginning of the file. Let's take the first line as an example:

```
OS::TripleO::Controller::Net::SoftwareConfig: ../../network/config/contrail/controller-nic-config.
yaml
```

This example tells you that the controller networking resources should be according to the template `controller-nic-config`. To be able to achieve the needed configuration for the controller NICs you should modify `controller-nic-config` according to your target configuration. Here's a snippet from the `controller-nic-config` file:

```
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: 169.254.169.254/32
                  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
```

You can see that under the network_config you can find a list of configurations for different interfaces/VLAN configurations bound to NIC devices. NIC 1 represents the first NIC in the server and NIC 2 represents the second, and so on. You can configure native interfaces or VLAN tags. The example has the combination of a native interface for the provisioning network (control network) and VLAN-tagged interfaces on the same device (NIC 1 in this example). Our example shows the configuration template should result in the controller node with the NIC configuration as show Figure 3.5.

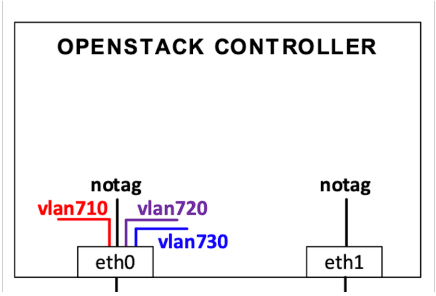

Figure 3.5 Controller Node with the NIC Configuration

Fixed-ip-vips.yaml

This file defines the VIP ports' YAML templates and the values of fixed VIP parameters. The file is optional to use, however it's recommended that it be used in production deployments instead of having random VIPs. Below is an example:

```
(undercloud) [stack@tripleo-director ~]$ cat ~/tripleo-heat-templates/environments/fixed-ip-vips.
yaml
# This template allows the IPs to be preselected for each VIP. Note that
# this template should be included after other templates which affect the
# network such as network-isolation.yaml.

resource_registry:
  OS::TripleO::Network::Ports::ExternalVipPort: ../network/ports/external.yaml
  OS::TripleO::Network::Ports::InternalApiVipPort: ../network/ports/internal_api.yaml
  OS::TripleO::Network::Ports::StorageVipPort: ../network/ports/storage.yaml
  OS::TripleO::Network::Ports::StorageMgmtVipPort: ../network/ports/storage_mgmt.yaml
  OS::TripleO::Network::Ports::RedisVipPort: ../network/ports/vip.yaml
  OS::TripleO::Network::Ports::OVNDBsVipPort: ../network/ports/vip.yaml

parameter_defaults:
  # Set the IP addresses of the VIPs here.
  # NOTE: we will eventually move to one VIP per service
  #
  PublicVirtualFixedIPs:      [{'ip_address':'20.2.0.240'}]
  InternalApiVirtualFixedIPs: [{'ip_address':'20.1.0.240'}]
  ControlFixedIPs:            [{'ip_address':'192.168.24.240'}]
  StorageVirtualFixedIPs:     [{'ip_address':'20.3.0.240'}]
  StorageMgmtVirtualFixedIPs: [{'ip_address':'20.4.0.240'}]
  RedisVirtualFixedIPs:       [{'ip_address':'20.1.0.241'}]
```

Ips-from-pool-all.yaml

This file defines the OpenStack port configurations of each node and pre-allocates fixed IPs to the ports instead of random assignment. Here's a snippet of the file content:

```
(undercloud) [stack@tripleo-director ~]$ cat ~/tripleo-heat-templates/environments/ips-from-pool-all.
yaml
# Environment file demonstrating how to pre-assign IPs to all node types
resource_registry:
  OS::TripleO::Controller::Ports::ExternalPort: ../network/ports/external_from_pool.yaml
  OS::TripleO::Controller::Ports::InternalApiPort: ../network/ports/internal_api_from_pool.yaml
  OS::TripleO::Controller::Ports::StoragePort: ../network/ports/storage_from_pool.yaml
  OS::TripleO::Controller::Ports::StorageMgmtPort: ../network/ports/storage_mgmt_from_pool.yaml
  OS::TripleO::Controller::Ports::TenantPort: ../network/ports/tenant_from_pool.yaml
  # Management network is optional and disabled by default
  #OS::TripleO::Controller::Ports::ManagementPort: ../network/ports/management_from_pool.yaml

  OS::TripleO::Compute::Ports::ExternalPort: ../network/ports/noop.yaml
  OS::TripleO::Compute::Ports::InternalApiPort: ../network/ports/internal_api_from_pool.yaml
  OS::TripleO::Compute::Ports::StoragePort: ../network/ports/storage_from_pool.yaml
  OS::TripleO::Compute::Ports::StorageMgmtPort: ../network/ports/noop.yaml
  OS::TripleO::Compute::Ports::TenantPort: ../network/ports/tenant_from_pool.yaml
  #OS::TripleO::Compute::Ports::ManagementPort: ../network/ports/management_from_pool.yaml

  OS::TripleO::CephStorage::Ports::ExternalPort: ../network/ports/noop.yaml
  OS::TripleO::CephStorage::Ports::InternalApiPort: ../network/ports/noop.yaml
  OS::TripleO::CephStorage::Ports::StoragePort: ../network/ports/storage_from_pool.yaml
  OS::TripleO::CephStorage::Ports::StorageMgmtPort: ../network/ports/storage_mgmt_from_pool.yaml
  OS::TripleO::CephStorage::Ports::TenantPort: ../network/ports/noop.yaml
  #OS::TripleO::CephStorage::Ports::ManagementPort: ../network/ports/management_from_pool.yaml

  OS::TripleO::ObjectStorage::Ports::ExternalPort: ../network/ports/noop.yaml
  OS::TripleO::ObjectStorage::Ports::InternalApiPort: ../network/ports/internal_api_from_pool.yaml
  OS::TripleO::ObjectStorage::Ports::StoragePort: ../network/ports/storage_from_pool.yaml
  OS::TripleO::ObjectStorage::Ports::StorageMgmtPort: ../network/ports/storage_mgmt_from_pool.yaml
  OS::TripleO::ObjectStorage::Ports::TenantPort: ../network/ports/noop.yaml
  #OS::TripleO::ObjectStorage::Ports::ManagementPort: ../network/ports/management_from_pool.yaml

  OS::TripleO::BlockStorage::Ports::ExternalPort: ../network/ports/noop.yaml
  OS::TripleO::BlockStorage::Ports::InternalApiPort: ../network/ports/internal_api_from_pool.yaml
  OS::TripleO::BlockStorage::Ports::StoragePort: ../network/ports/storage_from_pool.yaml
  OS::TripleO::BlockStorage::Ports::StorageMgmtPort: ../network/ports/storage_mgmt_from_pool.yaml
  OS::TripleO::BlockStorage::Ports::TenantPort: ../network/ports/noop.yaml
  #OS::TripleO::BlockStorage::Ports::ManagementPort: ../network/ports/management_from_pool.yaml

parameter_defaults:
  ControllerIPs:
    ctlplane:
    - 192.168.24.11
    - 192.168.24.12
    - 192.168.24.13
    tenant:
    - 20.0.0.11
    - 20.0.0.12
    - 20.0.0.13
    internal_api:
    - 20.1.0.11
```

```
    - 20.1.0.12
    - 20.1.0.13
    external:
    - 20.2.0.11
    - 20.2.0.12
    - 20.2.0.13
    storage:
    - 20.3.0.11
    - 20.3.0.12
    - 20.3.0.13
    storage_mgmt:
    - 20.4.0.11
    - 20.4.0.12
    - 20.4.0.13
    management:
    - 20.5.0.11
    - 20.5.0.12
    - 20.5.0.13
```

In the beginning of the file you can see the definition of the OpenStack ports on each node and the YAML template used to obtain the port configuration. This is followed by the default parameters to be used to assign fixed IPs related to each network per node. For the purpose of brevity we shorten the output to include only the controller IPs. You can include IPs for each node. Actually, this is what has been done in our book's deployment.

Rhsm.yaml

This file is crucial as it contains the mandatory Red Hat subscription information to obtain the needed images for every component. This file also includes the repos needed to complete the deployment. Here is a sample file taken from our deployment and Table 3.7 to help define the sample:

```
(undercloud) [stack@tripleo-director ~]$ cat ~/rhsm.yaml
parameter_defaults:
  RhsmVars:
    rhsm_repos:
      - fast-datapath-for-rhel-8-x86_64-rpms
      - openstack-16.1-for-rhel-8-x86_64-rpms
      - satellite-tools-6.5-for-rhel-8-x86_64-rpms
      - ansible-2-for-rhel-8-x86_64-rpms
      - rhel-8-for-x86_64-highavailability-rpms
      - rhel-8-for-x86_64-appstream-rpms
      - rhel-8-for-x86_64-baseos-rpms
    rhsm_username: "username"
    rhsm_password: "password"
    rhsm_org_id: "6963238"
    rhsm_pool_ids: "8a85f99b77b0c6850177ceb51fdd237b"
```

Table 3.7 The rhsm.yaml File Usage

Option	Value	Description
Rhsm_repos	- fast-datapath-for-rhel-8-x86_64-rpms - openstack-16.1-for-rhel-8-x86_64-rpms - satellite-tools-6.5-for-rhel-8-x86_64-rpms - ansible-2-for-rhel-8-x86_64-rpms - rhel-8-for-x86_64-highavailability-rpms - rhel-8-for-x86_64-appstream-rpms - rhel-8-for-x86_64-baseos-rpms	List of repos needed for the deployment to be enabled.
rhsm_username		The username for registration.
rhsm_password		The password for registration.
rhsm_org_id		The organization to use for registration. To locate this ID, run sudo subscription-manager orgs from the undercloud node. Enter your Red Hat credentials when prompted, and use the resulting Key value.
rhsm_pool_ids		This is the subscription pool ID to use. Use this if not auto-attaching subscriptions. To locate this ID, run sudo subscription-manager list --available --all --matches="*OpenStack*" from the undercloud node and use the resulting Pool ID value.

Roles_data_contrail.yaml

This file defines the roles in the deployment according to:

- Networks attached to the role
- Services included in the role

In other words, this file is where you put the information about how networks should exist in a specific role and the services related to that role. Be careful because you need to know exactly what to you do if you change the services structure. Moreover, you need to have the exact match between each network's role and its NIC template configuration. And if you put a network under controller definition that does not exist in the controller NIC template, your deployment will fail.

Here is a snippet from the file including a role example (as it is too long to include all roles):

```
###############################################################################
# Role: Controller                                                            #
###############################################################################
```

```
- name: Controller
  description: |
    Controller role that has all the controler services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 0
  tags:
    - primary
    - controller
  networks:
    External:
      subnet: external_subnet
    InternalApi:
      subnet: internal_api_subnet
    Storage:
      subnet: storage_subnet
    StorageMgmt:
      subnet: storage_mgmt_subnet
#    Tenant:
#      subnet: tenant_subnet
  # For systems with both IPv4 and IPv6, you may specify a gateway network for
  # each, such as ['ControlPlane', 'External']
  default_route_networks: ['External']
  HostnameFormatDefault: '%stackname%-controller-%index%'
  # Deprecated & backward-compatible values (FIXME: Make parameters consistent)
  # Set uses_deprecated_params to True if any deprecated params are used.
  uses_deprecated_params: True
  deprecated_param_extraconfig: 'controllerExtraConfig'
  deprecated_param_flavor: 'OvercloudControlFlavor'
  deprecated_param_image: 'controllerImage'
  deprecated_nic_config_name: 'controller.yaml'
  update_serial: 1
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AodhApi
    - OS::TripleO::Services::AodhEvaluator
    - OS::TripleO::Services::AodhListener
    - OS::TripleO::Services::AodhNotifier
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::BarbicanApi
    - OS::TripleO::Services::BarbicanBackendSimpleCrypto
    - OS::TripleO::Services::BarbicanBackendDogtag
    - OS::TripleO::Services::BarbicanBackendKmip
    - OS::TripleO::Services::BarbicanBackendPkcs11Crypto
    - OS::TripleO::Services::BootParams
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CeilometerAgentCentral
    - OS::TripleO::Services::CeilometerAgentNotification
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephGrafana
    - OS::TripleO::Services::CephMds
    - OS::TripleO::Services::CephMgr
    - OS::TripleO::Services::CephMon
    - OS::TripleO::Services::CephRbdMirror
    - OS::TripleO::Services::CephRgw
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::CinderApi
    - OS::TripleO::Services::CinderBackendDellPs
    - OS::TripleO::Services::CinderBackendDellSc
```

```
- OS::TripleO::Services::CinderBackendDellEMCPowermax
- OS::TripleO::Services::CinderBackendDellEMCSc
- OS::TripleO::Services::CinderBackendDellEMCUnity
- OS::TripleO::Services::CinderBackendDellEMCVMAXISCSI
- OS::TripleO::Services::CinderBackendDellEMCVNX
- OS::TripleO::Services::CinderBackendDellEMCVxFlexOS
- OS::TripleO::Services::CinderBackendDellEMCXtremio
- OS::TripleO::Services::CinderBackendDellEMCXTREMIOISCSI
- OS::TripleO::Services::CinderBackendNetApp
- OS::TripleO::Services::CinderBackendPure
- OS::TripleO::Services::CinderBackendScaleIO
- OS::TripleO::Services::CinderBackendVRTSHyperScale
- OS::TripleO::Services::CinderBackendNVMeOF
- OS::TripleO::Services::CinderBackup
- OS::TripleO::Services::CinderHPELeftHandISCSI
- OS::TripleO::Services::CinderScheduler
- OS::TripleO::Services::CinderVolume
- OS::TripleO::Services::Clustercheck
- OS::TripleO::Services::Collectd
- OS::TripleO::Services::ContainerImagePrepare
- OS::TripleO::Services::DesignateApi
- OS::TripleO::Services::DesignateCentral
- OS::TripleO::Services::DesignateProducer
- OS::TripleO::Services::DesignateWorker
- OS::TripleO::Services::DesignateMDNS
- OS::TripleO::Services::DesignateSink
- OS::TripleO::Services::ContrailCertmongerUser
- OS::TripleO::Services::ContrailHeatPlugin
- OS::TripleO::Services::Docker
- OS::TripleO::Services::Ec2Api
- OS::TripleO::Services::Etcd
- OS::TripleO::Services::ExternalSwiftProxy
- OS::TripleO::Services::GlanceApi
- OS::TripleO::Services::GnocchiApi
- OS::TripleO::Services::GnocchiMetricd
- OS::TripleO::Services::GnocchiStatsd
- OS::TripleO::Services::HAproxy
- OS::TripleO::Services::HeatApi
- OS::TripleO::Services::HeatApiCloudwatch
- OS::TripleO::Services::HeatApiCfn
- OS::TripleO::Services::HeatEngine
- OS::TripleO::Services::Horizon
- OS::TripleO::Services::IpaClient
- OS::TripleO::Services::Ipsec
- OS::TripleO::Services::IronicApi
- OS::TripleO::Services::IronicConductor
- OS::TripleO::Services::IronicInspector
- OS::TripleO::Services::IronicPxe
- OS::TripleO::Services::IronicNeutronAgent
- OS::TripleO::Services::Iscsid
- OS::TripleO::Services::Keepalived
- OS::TripleO::Services::Kernel
- OS::TripleO::Services::Keystone
- OS::TripleO::Services::LoginDefs
- OS::TripleO::Services::ManilaApi
- OS::TripleO::Services::ManilaBackendCephFs
- OS::TripleO::Services::ManilaBackendIsilon
- OS::TripleO::Services::ManilaBackendNetapp
```

```
- OS::TripleO::Services::ManilaBackendUnity
- OS::TripleO::Services::ManilaBackendVNX
- OS::TripleO::Services::ManilaBackendVMAX
- OS::TripleO::Services::ManilaScheduler
- OS::TripleO::Services::ManilaShare
- OS::TripleO::Services::Memcached
- OS::TripleO::Services::MetricsQdr
- OS::TripleO::Services::MistralApi
- OS::TripleO::Services::MistralEngine
- OS::TripleO::Services::MistralExecutor
- OS::TripleO::Services::MistralEventEngine
- OS::TripleO::Services::Multipathd
- OS::TripleO::Services::MySQL
- OS::TripleO::Services::MySQLClient
- OS::TripleO::Services::NeutronApi
- OS::TripleO::Services::NeutronBgpVpnApi
- OS::TripleO::Services::NeutronSfcApi
- OS::TripleO::Services::NeutronCorePlugin
- OS::TripleO::Services::NeutronDhcpAgent
- OS::TripleO::Services::NeutronL2gwAgent
- OS::TripleO::Services::NeutronL2gwApi
- OS::TripleO::Services::NeutronL3Agent
- OS::TripleO::Services::NeutronLinuxbridgeAgent
- OS::TripleO::Services::NeutronMetadataAgent
- OS::TripleO::Services::NeutronML2FujitsuCfab
- OS::TripleO::Services::NeutronML2FujitsuFossw
- OS::TripleO::Services::NeutronOvsAgent
- OS::TripleO::Services::NeutronVppAgent
- OS::TripleO::Services::NeutronAgentsIBConfig
- OS::TripleO::Services::NovaApi
- OS::TripleO::Services::NovaConductor
- OS::TripleO::Services::NovaIronic
- OS::TripleO::Services::NovaMetadata
- OS::TripleO::Services::NovaScheduler
- OS::TripleO::Services::NovaVncProxy
- OS::TripleO::Services::ContainersLogrotateCrond
- OS::TripleO::Services::OctaviaApi
- OS::TripleO::Services::OctaviaDeploymentConfig
- OS::TripleO::Services::OctaviaHealthManager
- OS::TripleO::Services::OctaviaHousekeeping
- OS::TripleO::Services::OctaviaWorker
- OS::TripleO::Services::OpenStackClients
- OS::TripleO::Services::OVNDBs
- OS::TripleO::Services::OVNController
- OS::TripleO::Services::Pacemaker
- OS::TripleO::Services::PankoApi
- OS::TripleO::Services::PlacementApi
- OS::TripleO::Services::OsloMessagingRpc
- OS::TripleO::Services::OsloMessagingNotify
- OS::TripleO::Services::Podman
- OS::TripleO::Services::Rear
- OS::TripleO::Services::Redis
- OS::TripleO::Services::Rhsm
- OS::TripleO::Services::Rsyslog
- OS::TripleO::Services::RsyslogSidecar
- OS::TripleO::Services::SaharaApi
- OS::TripleO::Services::SaharaEngine
- OS::TripleO::Services::Securetty
```

```
- OS::TripleO::Services::SkydiveAgent
- OS::TripleO::Services::SkydiveAnalyzer
- OS::TripleO::Services::Snmp
- OS::TripleO::Services::Sshd
- OS::TripleO::Services::SwiftProxy
- OS::TripleO::Services::SwiftDispersion
- OS::TripleO::Services::SwiftRingBuilder
- OS::TripleO::Services::SwiftStorage
- OS::TripleO::Services::Timesync
- OS::TripleO::Services::Timezone
- OS::TripleO::Services::TripleoFirewall
- OS::TripleO::Services::TripleoPackages
- OS::TripleO::Services::Tuned
- OS::TripleO::Services::Vpp
- OS::TripleO::Services::Zaqar
```

As you know, the two main components are the networks related to the role and the services. In this external example, you can see internal API, and storage and storage management are the networks that should exist in OpenStack controller. This should exactly match the networks in the controller-nic-confic file. You can verify it with the previous file in the Contrail-net.yaml section of this chapter.

Overcloud Architecture

The overcloud stack is the production OpenStack cluster that forms one resource pool of compute, storage, and networking resources to carry out the production workloads and connect them to each other or to the outside world. Considering RHOSP plus OpenStack, the overcloud stack consists of the following node types.

- OpenStack Controller Node
- Contrail Controller Node
- Contrail Analytics
- Contrail Analytics DB
- Compute Nodes
- Storage Nodes (optional)

Each node provides specific services related to its function. To make this possible, every node consists of different pods and containers to provide its related services. The collection of node containers is what gives the node its identity as a controller of compute, storage, etc..

The deployment in this book uses five types of overcloud nodes in the stack:

- OpenStack Controller
- Contrail Controller

- Contrail Analytics
- Contrail Analytics DB
- Kernel vRouter Compute

Figure 3.6 shows the overcloud stack node architecture and the containers mapped to each node. For simplicity, only one node of each type is shown in the diagram. However, you can use as many as needed of each type. In HA deployments, at least three instances of every controlling function are needed and as many computes as you may require based on needed scale.

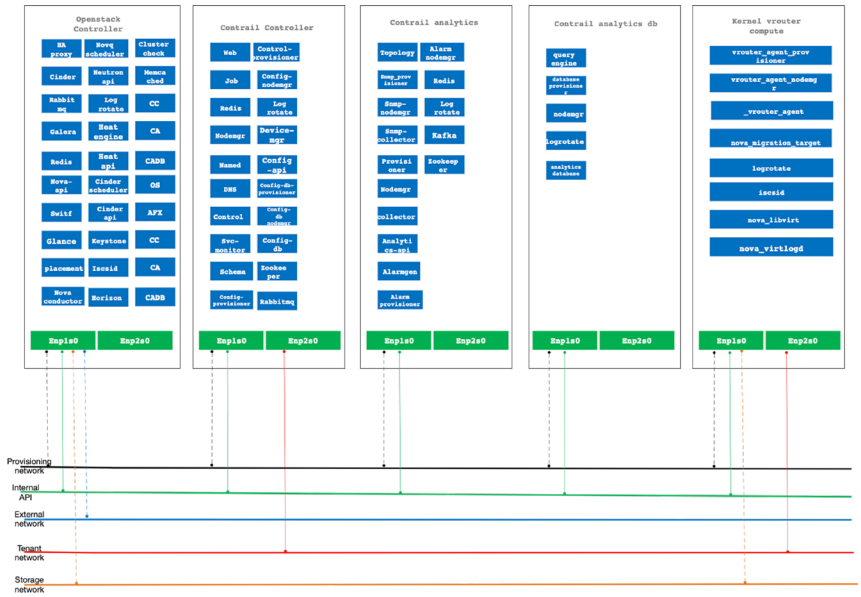

Figure 3.6 Overcloud Architecture

RHOSP Overcloud Networking Requirements

For the different RHOSP overcloud components to operate successfully there needs to be a specific networking requirement. Those networks work as the underlay networks for both signaling traffic between different components and to carry overlay workload traffic. Table 3.8 summarizes the networking requirements.

Table 3.8 Networking Requirements for RHOSP Components

Network name	Description
Internal API	Carries the signaling traffic between different OpenStack projects when they call each other to fulfill a complete job. For example, during VM creation, nova service sends API calls to the neutron API server, Cinder API, etc. The traffic for these calls is carried over the internal API network.
External network	Carries user traffic to the horizon dashboard and user traffic to external APIs.
Provisioning network	Carries undercloud-overcloud control plane traffic during deployment and management. During deployment, overcloud nodes use it for fetching container images. Undercloud also uses it for booting PXE overcloud nodes.
Tenant network	Carries overlay traffic between workloads and between workloads and external world.
Storage network	Access to storage resources from compute and controller nodes.
Management network	Accesses nodes directly from the remote servers without the need to log in to undercloud, then accesses through a provisioning network. It is an optional network.

It should be obvious that not all nodes need to be connected to all networks and it depends on the node function and its related services that are mapped to use the relevant networks. This was previously observed in the ServiceNetMap parameter in the environment files.

In this book, deployment networks are mapped to NICs for testing purposes, however for production grade deployments, it is preferable to use bond interfaces to place internal APIs to external tenant networks. This ensures high availability of the stack and avoids affecting network functionality due to one link failure.

Chapter 4

Juniper Contrail Architecture

Juniper Contrail is an SDN solution developed by Juniper based on the Tungsten Fabric open source project. Contrail has many internal components working together to provide its complete functionality. It is very important when using Contrail to know the different pieces, the functionality of each piece, and how they work together. This chapter examines Juniper Contrail architecture, containers, and the functionality of each component.

Contrail Architecture

As shown in Figure 4.1, Contrail's high-level architecture consists of two parts that generally follow the SDN methodology of having a centralized control function and distributed forwarding.

You can see the logically centralized control functions and the distributed forwarding function in Figure 4.1. The control part of the Contrail system consists of several parts, each performing a specific function. The next sections examine the containerized architecture of Juniper Contrail and the function of each control component.

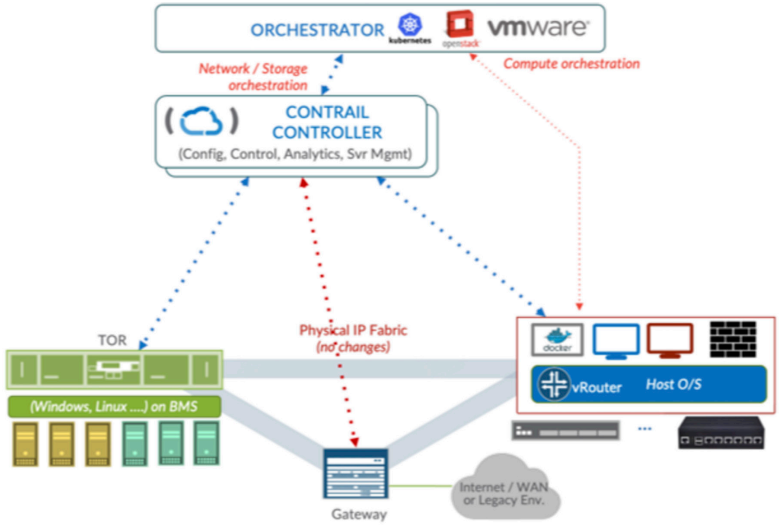

Figure 4.1 *Contrail Architecture Overview*

Contrail Microservices Architecture

Contrail uses microservice architecture based on containers. The microservices are grouped into pods that correspond to roles that are assigned to servers during deployment, as discussed in Chapter 3. In RHOSP plus Contrail deployment you can use composable roles that enable you to separate functions into individual roles and place them on targeted physical servers. For example, you can have controller functions that contain controller, configuration DB, configuration, and webUI on one server as one node or you can segregate them into multiple nodes.

The relationship of microservices to pods is shown in Figure 4.2 and Figure 4.3.

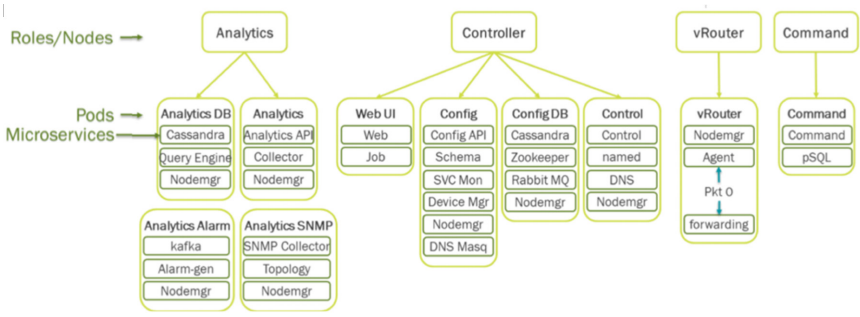

Figure 4.2 *Contrail Microservices Architecture*

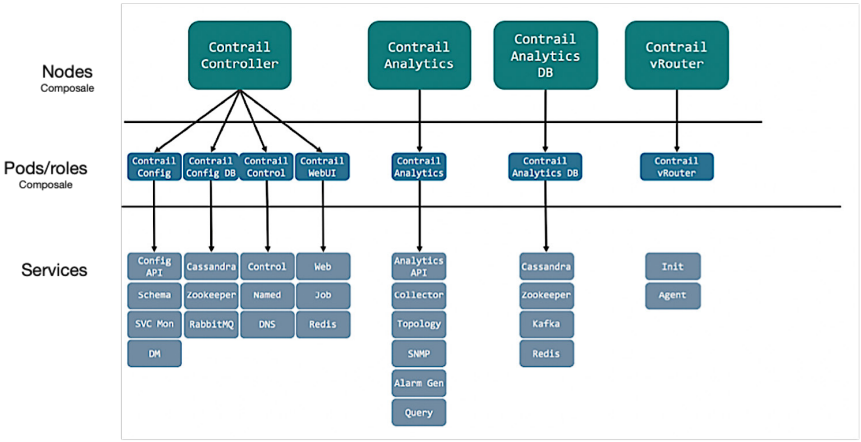

Figure 4.3 *Contrail Microservices Architecture – Containers' Structure*

There are seven roles in the Contrail system

- Contrail control
- Contrail config
- Contrail config DB
- Contrail webUI
- Contrail analytics
- Contrail analytics DB
- Contrail vRouter

It is not mandatory to have the seven roles in each deployment. You can omit some of the roles like the analytics role, but do realize it might impact functionalities.

Contrail Components

Let's examine in more detail some of these Contrail main roles.

Contrail Control

Contrail control role consists of three main services: DNS, named, and control service. Control service receives the configuration from the configuration nodes. It exchanges routes with vRouter via XMPP and other controllers in case you have a high availability setup via BGP. Contrail control also exchanges routes with physical infrastructure via BGP.

Contrail DNS serves vDNS info with XMPP and vRouter and will perform DNS tasks based on that input, while contrail-named is used for external DNS query to vDNS entry.

The architecture of the Contrail control role is shown in Figure 4.4.

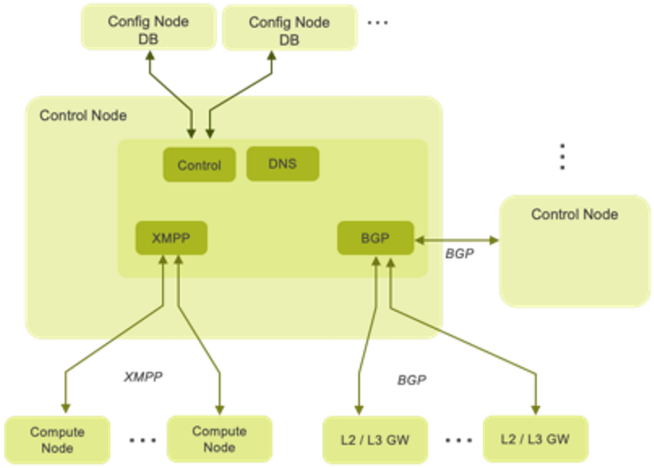

Figure 4.4 *Contrail Control Internal Architecture*

Contrail Config and Config DB

Config and config database roles are separate roles, however, they are tightly coupled together from a functionality aspect.

The config node receives the config over the northbound interface via the REST API from orchestrators or from any other application. It receives information as high-level data models, while the config DB is used to store all configuration data objects in the database. See Figure 4.5.

Config node services are:

- The REST API server provides the north-bound interface to an orchestration system or other application. It is responsible for storing and loading permanent data to and from the database.
- Schema transformer converts abstract config parameters, such as logical-router, network-policy, and service-chain, into the verbiage of a L3 VPN. It learns about changes in the high-level data model via the message Bus and *compiles* these changes into the low-level data model.

- Svc-monitor serves various services that have to use external processes internally, such as HAProxy load balancer, v1 service-chain instances based on the Nova API, and iptables MASQUERADE for SNAT. Internally, the vRouter-agent has some logic to kick in HAProxy or set iptables MASQUERADE. Svc-monitor will kick in that logic when a related service is defined.
- Device manager manages physical devices such as routers and switches.
- RabbitMQ is open source message broker software that implements the Advanced Message Queuing Protocol (AMQP). It serves as a message bus to facilitate communications amongst internal components.

Config DB services are:

- Cassandra, which is the main component in the database for persistent data storage. It is a high-performance distributed database designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.
- Zookeeper provides coordination services for distributed applications. Contrail uses the Apache ZooKeeper process to maintain synchronization of the Contrail configuration, analytics, and database running on the different instances of controllers. Services like service monitor, device monitor, and Contrail schema work in active-backup mode. This helps in the election of active service. It locks resources to avoid data inconsistency and helps allocate resources like IP addresses. In the case of concurrent requests from multiple services for the same resource, it makes sure the resources are uniquely utilized.

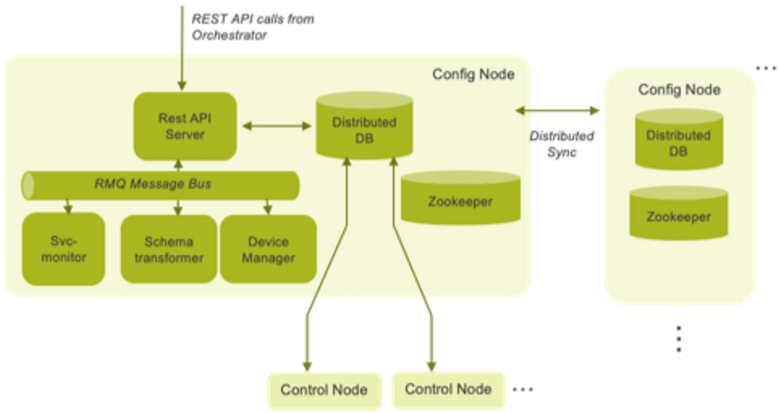

Figure 4.5 Config and Config DB Internal Architecture

Contrail vRouter

Contrail vRouter is the distributed forwarding part of the Contrail system. The main functionality of the vRouter is to perform all related forwarding actions, including traffic steering, QOS marking and scheduling, policy actions, NATing, multicast, traffic mirroring, traffic load balancing, and almost all other forwarding plane functionality. For all networking engineers it is exactly like PFE in the router. vRouter replaces the Linux bridge or OVS module in the hypervisor kernel.

vRouter implementation uses the same concept of L3VPN for traffic segregation between different virtual networks or VM NICs. The vRouter schematic diagram is shown in Figure 4.6.

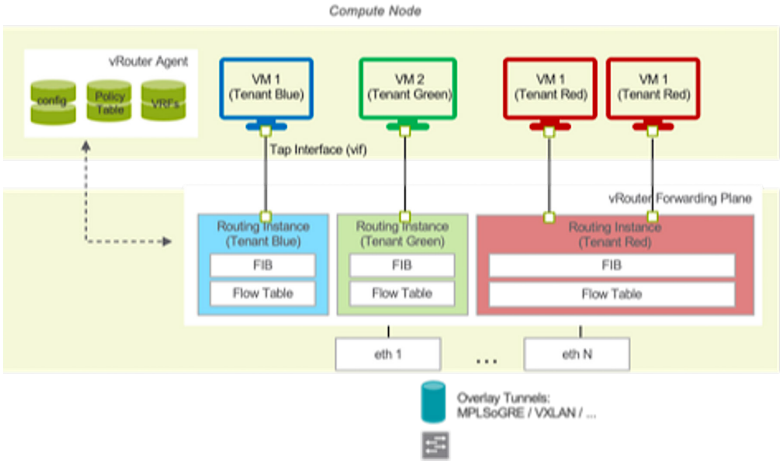

Figure 4.6 *Contrail vRouter*

Now that you have a clear view of the main core components of the Contrail system, let's explain the step-by-step deployment procedures for RHOSP 16 plus Contrail 2011.138 to build a production-grade cloud.

Chapter 5

Step-by-Step Deployment for RHOSP 16 Plus Contrail 2011.138

This chapter walks you through a detailed step-by-step deployment using the environment files information discussed in Chapter 3.

We describe the setup and the target overcloud environment to deploy and then create a step-by-step tutorial for the procedure. Let's start.

Contrail Networking Plus RHOSP Node Roles

As described in Chapter 3, the book's deployment uses composable roles defined by means of environment files. The following are the roles used in the deployment:

- OpenStack Controller (1 instance)
- Contrail Controller (1 instance)
- Contrail Analytics (1 instance)
- Contrail Analytics DB (1 instance)
- Kernel vRouter Compute (2 instances)

Networking Requirements

Isolated networks are also used as they are the preferred way to achieve traffic isolation and segregation on different networks instead of having everything running on a provisioning network.

In this deployment we use:

- Provisioning network
- Internal API
- Tenant network
- External network
- Storage network

For the provisioning network some mandatory conditions need to be met:

- The same NIC of every overcloud node should be in the same broadcast domain of the provisioning network. This deployment uses NIC 1 to carry the provisioning network.
- The provisioning network NIC should be untagged on overcloud nodes.
- The provisioning NIC on undercloud should be in a different NIC than the NIC used in remote connectivity because it loses connectivity during the director installation and is bridged to the OVS created during the undercloud installation.
- You should know the MAC address of each provisioning NIC on each node as it's part of the needed information to register overcloud nodes.

In addition to the mentioned overcloud networks, IPMI connectivity between all overcloud nodes and the undercloud node should be in place. Figure 5.1 shows the nodes and networking architecture followed in the deployment.

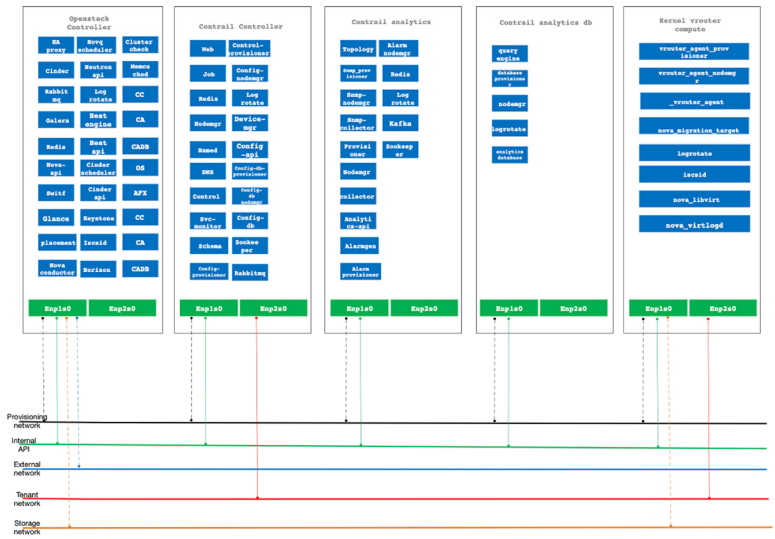

Figure 5.1 Overcloud Networking Connectivity

Step-by-step Procedure

The step-by-step procedures assumes:

1. The correct connectivity according to the previously mentioned connectivity details in Figure 5.1.

2. Undercloud server readiness with RHEL 8.2 as OS image.

3. You're using undercloud as the registry.

Creating user stack on undercloud and configuring it to use password-less sudo privilege

This snippet user stack has been created on undercloud and a directory for images has been created after switching to user stack:

```
[root@tripleo-director ~]# useradd stack
[root@tripleo-director ~]# passwd stack
[root@tripleo-director ~]# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack
[root@tripleo-director ~]# chmod 0440 /etc/sudoers.d/stack
[root@tripleo-director ~]# su - stack
[stack@tripleo-director ~]$
[stack@tripleo-director ~]$ mkdir ~/images
```

Registering the undercloud and attaching subscriptions

Because it is not free software you should register the undercloud node by running `subscription-manager` and attach a valid Red Hat OpenStack platform subscription. This is a mandatory step to get needed access to fetch the necessary images for deployment:

```
[stack@tripleo-director ~]$ sudo subscription-manager register
```

The command will prompt you for a subscription username and password. You need to enter them to register your system.

Now locate a suitable pool ID for RHOSP by issuing the following command. You can expect similar output to that shown here. If you have too many pools, please select the one most suitable to you.

Note that the output exact pool ID has been omitted and replaced with `xxxxx`, however you should expect to have specific pool ID:

```
[stack@tripleo-director ~]$ sudo subscription-manager list --available --all --matches="Red Hat
OpenStack"
+-------------------------------------------+
   Available Subscriptions
+-------------------------------------------+
Subscription Name:   Red Hat OpenStack Platform, Standard Support (4 Sockets, NFR, Partner Only)
Provides:          Red Hat Enterprise Linux FDIO Early Access (RHEL 7 Server)
```

```
                Red Hat Ansible Engine
                Red Hat OpenStack Certification Test Suite
                Red Hat CodeReady Linux Builder for Power, little endian
                Red Hat Enterprise Linux Fast Datapath
                Red Hat CloudForms
                Red Hat Software Collections Beta (for RHEL Server for IBM Power LE)
                Red Hat Enterprise Linux Advanced Virtualization Beta
                Red Hat Beta
                Red Hat Enterprise Linux Fast Datapath (for RHEL Server for IBM Power LE)
                dotNET on RHEL (for RHEL Server)
                Red Hat Certification (for RHEL Server)
                Red Hat CodeReady Linux Builder for x86_64 - Extended Update Support
                Red Hat Ceph Storage Calamari
                Red Hat Developer Tools (for RHEL Server for IBM Power LE)
                Red Hat OpenStack Beta Certification Test Suite
                Red Hat Enterprise Linux High Availability (for IBM Power LE) - Extended Update
Support
                Red Hat OpenStack Beta for IBM Power LE
                Red Hat Enterprise Linux Fast Datapath Beta for Power, little endian
                Red Hat Software Collections (for RHEL Server)
                Red Hat Enterprise Linux for Power, little endian - Extended Update Support
                Red Hat Enterprise Linux for Power, little endian
                Red Hat Developer Tools (for RHEL Server for IBM Power 9)
                Red Hat Enterprise Linux Server (for IBM Power LE) - Update Services for SAP Solutions
                Red Hat Enterprise Linux for Power 9
                Red Hat Enterprise Linux Fast Datapath Beta for x86_64
                Red Hat Enterprise MRG Messaging
                Red Hat Enterprise Linux for x86_64
                Red Hat Ceph Storage MON
                Red Hat Enterprise Linux FDIO (RHEL 7 Server)
                Red Hat Software Collections (for RHEL Server for IBM Power 9)
                dotNET on RHEL Beta (for RHEL Server)
                Red Hat CodeReady Linux Builder for x86_64
                Red Hat Ceph Storage
                Red Hat Software Collections (for RHEL Server for IBM Power LE)
                Red Hat Enterprise Linux Atomic Host Beta
                Red Hat Developer Tools Beta (for RHEL Server for IBM Power 9)
                Red Hat OpenStack Beta
                Red Hat Enterprise Linux Load Balancer (for RHEL Server)
                Red Hat Enterprise Linux High Availability for Power, little endian
                Red Hat Enterprise Linux High Availability for x86_64
                Red Hat Single Sign-On
                Red Hat Enterprise Linux Advanced Virtualization
                Red Hat Developer Tools Beta (for RHEL Server for IBM Power LE)
                Red Hat CloudForms Beta
                Red Hat Enterprise Linux High Availability for x86_64 - Extended Update Support
                Red Hat Software Collections Beta (for RHEL Server for IBM Power 9)
                Red Hat OpenStack
                Red Hat Enterprise Linux Atomic Host
                Red Hat OpenStack for IBM Power
                Red Hat Enterprise Linux for Real Time for NFV
                Red Hat Software Collections Beta (for RHEL Server)
                Red Hat Enterprise Linux Server
                Red Hat Enterprise Linux for x86_64 - Extended Update Support
SKU:             SER0505
Contract:         12568131
Pool ID:          xxxxxxxxxxxxxxxxxxxxxxxxx
Provides Management: No
```

```
Available:            Unlimited
Suggested:            1
Service Type:         L1-L3
Roles:
Service Level:        Standard
Usage:
Add-ons:
Subscription Type:    Standard
Starts:               01/14/2021
Ends:                 01/13/2022
Entitlement Type:     Virtual
```

After locating the pool ID attach undercloud to the RHOSP16 entitlement:

```
[stack@tripleo-director ~]$ sudo subscription-manager attach --pool=xxxxxxxxxxx
Successfully attached a subscription for: Red Hat OpenStack Platform, Standard Support (4 Sockets, NFR,
Partner Only)
```

Now lock undercloud to RHEL 8.2:

```
[stack@tripleo-director ~]$ sudo subscription-manager release --set=8.2
```

Release set to: 8.2

Enabling Repos For the Undercloud

Now disable all default repos and enable only the required repos that contain all needed packages for director installation:

```
[stack@tripleo-director ~]$ sudo subscription-manager repos --disable=*
[stack@tripleo-director ~]$ sudo subscription-manager repos --enable=rhel-8-for-x86_64-baseos-eus-
rpms --enable=rhel-8-for-x86_64-appstream-eus-rpms --enable=rhel-8-for-x86_64-highavailability-eus-
rpms --enable=ansible-2.9-for-rhel-8-x86_64-rpms --enable=openstack-16.1-for-rhel-8-x86_64-rpms
--enable=fast-datapath-for-rhel-8-x86_64-rpms --enable=advanced-virt-for-rhel-8-x86_64-rpms

[stack@tripleo-director ~]$ sudo subscription-manager repos --disable=*

Repository 'codeready-builder-for-rhel-8-x86_64-eus-rpms' is disabled for this system.
Repository 'satellite-tools-6.8-for-rhel-8-x86_64-debug-rpms' is disabled for this system.
Repository 'rh-gluster-3-client-for-rhel-8-x86_64-debug-rpms' is disabled for this system.
Repository 'rh-gluster-3-client-for-rhel-8-x86_64-source-rpms' is disabled for this system.
Repository 'openstack-16.1-for-rhel-8-x86_64-source-rpms' is disabled for this system.
----- output truncated


[stack@tripleo-director ~]$ sudo subscription-manager repos --enable=rhel-8-for-x86_64-baseos-eus-
rpms --enable=rhel-8-for-x86_64-appstream-eus-rpms --enable=rhel-8-for-x86_64-highavailability-eus-
rpms --enable=ansible-2.9-for-rhel-8-x86_64-rpms --enable=openstack-16.1-for-rhel-8-x86_64-rpms
--enable=fast-datapath-for-rhel-8-x86_64-rpms --enable=advanced-virt-for-rhel-8-x86_64-rpms
Repository 'rhel-8-for-x86_64-baseos-eus-rpms' is enabled for this system.
Repository 'rhel-8-for-x86_64-appstream-eus-rpms' is enabled for this system.
Repository 'rhel-8-for-x86_64-highavailability-eus-rpms' is enabled for this system.
Repository 'ansible-2.9-for-rhel-8-x86_64-rpms' is enabled for this system.
Repository 'openstack-16.1-for-rhel-8-x86_64-rpms' is enabled for this system.
Repository 'fast-datapath-for-rhel-8-x86_64-rpms' is enabled for this system.
Repository 'advanced-virt-for-rhel-8-x86_64-rpms' is enabled for this system.
```

Set the container-tools repository module to version 2.0 by issuing:

```
[stack@tripleo-director ~]$ sudo dnf module disable -y container-tools:rhel8
[stack@tripleo-director ~]$ sudo dnf module enable -y container-tools:2.0

[stack@tripleo-director ~]$ sudo dnf module disable -y container-tools:rhel8
Updating Subscription Management repositories.
Red Hat Enterprise Linux 8 for x86_64 - AppStream - Extended Update Support (RPMs)        6.5 MB/s |  19 MB     00:02
Red Hat Enterprise Linux 8 for x86_64 - BaseOS - Extended Update Support (RPMs)           11 MB/s |  26 MB     00:02
Advanced Virtualization for RHEL 8 x86_64 (RPMs)                                        791 kB/s | 922 kB     00:01
Red Hat Enterprise Linux 8 for x86_64 - High Availability - Extended Update Support (RPMs)  1.6 MB/s | 2.0 MB     00:01
Fast Datapath for RHEL 8 x86_64 (RPMs)                                                  151 kB/s | 145 kB     00:00
Red Hat OpenStack Platform 16.1 for RHEL 8 x86_64 (RPMs)                                  1.5 MB/s | 1.8 MB     00:01
Red Hat Ansible Engine 2.9 for RHEL 8 x86_64 (RPMs)                                       1.1 MB/s | 1.2 MB     00:01
Only module name is required. Ignoring unneeded information in argument: 'container-tools:rhel8'
Dependencies resolved.
================================================================================================================
================================================================================================================
 Package                                  Architecture                        Version
Repository                           Size
================================================================================================================
================================================================================================================
Disabling modules:
 container-tools

Transaction Summary
================================================================================================================
================================================================================================================

Complete!


[stack@tripleo-director ~]$ sudo dnf module enable -y container-tools:2.0
Updating Subscription Management repositories.
Last metadata expiration check: 0:00:14 ago on Sat 20 Feb 2021 11:40:57 AM EST.
Dependencies resolved.
================================================================================================================
================================================================================================================
Package                                 Architecture                              Version
Repository                               Size
================================================================================================================
================================================================================================================
Enabling module streams:
 container-tools                                                                  2.0

Transaction Summary
================================================================================================================
================================================================================================================

Complete!
```

Set the virt repository module to version 8.2:

```
[stack@tripleo-director ~]$ sudo dnf module disable -y virt:rhel
[stack@tripleo-director ~]$ sudo dnf module enable -y virt:8.2


[stack@tripleo-director ~]$ sudo dnf module disable -y virt:rhel
```

```
Updating Subscription Management repositories.
Last metadata expiration check: 0:00:42 ago on Sat 20 Feb 2021 11:40:57 AM EST.
Only module name is required. Ignoring unneeded information in argument: 'virt:rhel'
Dependencies resolved.
================================================================================
================================================================================
 Package                          Architecture                  Version
Repository                    Size
================================================================================
================================================================================
Disabling modules:
 virt

Transaction Summary
================================================================================
================================================================================

Complete!


[stack@tripleo-director ~]$ sudo dnf module enable -y virt:8.2
Updating Subscription Management repositories.
Last metadata expiration check: 0:00:55 ago on Sat 20 Feb 2021 11:40:57 AM EST.
Dependencies resolved.
================================================================================
================================================================================
Package                               Architecture                  Version
Repository                     Size
================================================================================
================================================================================
Enabling module streams:
 virt                                                                8.2

Transaction Summary
================================================================================
================================================================================

Complete!
```

Perform an update on your system to ensure that you have the latest base system packages:

```
[stack@tripleo-director ~]$ sudo dnf update -y
[stack@tripleo-director ~]$ sudo reboot

[stack@tripleo-director ~]$ sudo dnf update -y
Updating Subscription Management repositories.
Last metadata expiration check: 0:01:08 ago on Sat 20 Feb 2021 11:40:57 AM EST.
Dependencies resolved.
================================================================================
================================================================================
Package                                 Architecture      Version     Repository
Size
================================================================================
================================================================================
Installing:
 kernel                                  x86_64            4.18.0-193.41.1.el8_2
rhel-8-for-x86_64-baseos-eus-rpms    2.8 M
```

```
 kernel-core                                      x86_64                4.18.0-193.41.1.el8_2
rhel-8-for-x86_64-baseos-eus-rpms    28 M
 kernel-modules                                   x86_64                4.18.0-193.41.1.el8_2
rhel-8-for-x86_64-baseos-eus-rpms    24 M
Upgrading:
 NetworkManager                                   x86_64                1:1.22.8-7.el8_2
rhel-8-for-x86_64-baseos-eus-rpms     2.3 M
---- output truncated
Installed:
  grub2-tools-efi-1:2.02-87.el8_2.2.x86_64                kernel-4.18.0-193.41.1.el8_2.x86_64
kernel-core-4.18.0-193.41.1.el8_2.x86_64       kernel-modules-4.18.0-193.41.1.el8_2.x86_64
  linux-firmware-20191202-99.gite8a0f4c9.el8_2.noarch

Complete!

[stack@tripleo-director ~]$ sudo reboot
3.4. Installing director packages
Install packages relevant to Red Hat OpenStack Platform director.

Procedure

client_loop: send disconnect: Broken pipe
```

Install Director Packages

Now you need to install command line tools for director installation and configuration. Here the print output shows `python3-tripleoclient` installation procedure:

```
[stack@tripleo-director ~]$ sudo dnf install -y python3-tripleoclient
Updating Subscription Management repositories.
/usr/lib/python3.6/site-packages/dateutil/parser/_parser.py:70: UnicodeWarning: decode() called on
unicode string, see https://bugzilla.redhat.com/show_bug.cgi?id=1693751
  instream = instream.decode()

Last metadata expiration check: 0:06:57 ago on Tue 23 Nov 2021 06:58:45 AM EST.
Dependencies resolved.
================================================================================================
================================================================================================
 Package                                  Architecture          Version       Repository
Size
================================================================================================
================================================================================================
Installing:
 python3-tripleoclient                      noarch              12.3.2-1.20210407123431.
ae58329.el8ost                openstack-16.1-for-rhel-8-x86_64-rpms              550 k
----- truncated output -----

Complete!
```

Prepare Container Image

This next necessary step prepares `containers-prepare-parameter.yaml` which was explained in Chapter 3:

```
[stack@tripleo-director ~]$ sudo openstack tripleo container image prepare default \
>   --local-push-destination \
```

```
>  --output-env-file containers-prepare-parameter.yaml
# Generated with the following on 2021-11-23T07:10:54.535011
#
#   openstack tripleo container image prepare default --local-push-destination --output-env-
file containers-prepare-parameter.yaml
#

parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    set:
      ceph_alertmanager_image: ose-prometheus-alertmanager
      ceph_alertmanager_namespace: registry.redhat.io/openshift4
      ceph_alertmanager_tag: 4.1
      ceph_grafana_image: rhceph-4-dashboard-rhel8
      ceph_grafana_namespace: registry.redhat.io/rhceph
      ceph_grafana_tag: 4
      ceph_image: rhceph-4-rhel8
      ceph_namespace: registry.redhat.io/rhceph
      ceph_node_exporter_image: ose-prometheus-node-exporter
      ceph_node_exporter_namespace: registry.redhat.io/openshift4
      ceph_node_exporter_tag: v4.1
      ceph_prometheus_image: ose-prometheus
      ceph_prometheus_namespace: registry.redhat.io/openshift4
      ceph_prometheus_tag: 4.1
      ceph_tag: latest
      name_prefix: openstack-
      name_suffix: ''
      namespace: registry.redhat.io/rhosp-rhel8
      neutron_driver: ovn
      rhel_containers: false
      tag: '16.1'
    tag_from_label: '{version}-{release}'
```

You can see that the output shows the default generated file. You need to customize the file according to your setup and add pieces of information regarding the Red Hat container registry username and password. Here, `ceph` is not part of the deployment, so it is excluded in the deployment file. Also, as we are using Contrail as an SDN component, the neutron driver would be set to null instead of ONV. The final file is shown below:

```
[stack@tripleo-director ~]$ cat containers-prepare-parameter.yaml
# Generated with the following on 2021-02-20T11:54:36.995567
#
#   openstack tripleo container image prepare default --local-push-destination --output-env-
file containers-prepare-parameter.yaml
#
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    excludes:
      - ceph
      - prometheus
    set:
      ceph_alertmanager_image: ose-prometheus-alertmanager
```

```
      ceph_alertmanager_namespace: registry.redhat.io/openshift4
      ceph_alertmanager_tag: 4.1
      ceph_grafana_image: rhceph-4-dashboard-rhel8
      ceph_grafana_namespace: registry.redhat.io/rhceph
      ceph_grafana_tag: 4
      ceph_image: rhceph-4-rhel8
      ceph_namespace: registry.redhat.io/rhceph
      ceph_node_exporter_image: ose-prometheus-node-exporter
      ceph_node_exporter_namespace: registry.redhat.io/openshift4
      ceph_node_exporter_tag: v4.1
      ceph_prometheus_image: ose-prometheus
      ceph_prometheus_namespace: registry.redhat.io/openshift4
      ceph_prometheus_tag: 4.1
      ceph_tag: latest
      name_prefix: openstack-
      name_suffix: ''
      namespace: registry.redhat.io/rhosp-rhel8
      neutron_driver: null
      rhel_containers: false
      tag: '16.1'
    tag_from_label: '{version}-{release}'
  ContainerImageRegistryCredentials:
    registry.redhat.io:
      username: 'password'
```

Configuring and Installing Director

Before installing director there are some settings that need to be put in `undercloud.conf` config file that director reads from the home directory of the user stack. Copy the default template to the home directory:

```
[stack@tripleo-director ~]$ cp \
>   /usr/share/python-tripleoclient/undercloud.conf.sample \
>   ~/undercloud.conf
```

Then edit the file to add the needed configuration. In this deployment the information has been put at the end of the file under `ctlplane-subnet`. Specify the container image file:

```
container_images_file = /home/stack/containers-prepare-parameter.yaml

[ctlplane-subnet]

cidr = 192.168.24.0/24
dhcp_start = 192.168.24.200
dhcp_end = 192.168.24.219
inspection_iprange = 192.168.24.220,192.168.24.239
gateway = 192.168.24.1
```

Now run the OpenStack undercloud install, but first log in to the Red Hat registry using the `podman` command as shown. Sometimes it causes errors in the undercloud installation because it's not able to log in and fetch images:

```
[stack@tripleo-director ~]$ podman login registry.redhat.io
Username:
```

```
Password:
Login Succeeded!

[stack@tripleo-director ~]$ openstack undercloud install
Using /tmp/undercloud-disk-space.yaml0b5zn_qyansible.cfg as config file
[WARNING]: log file at /usr/share/ansible/validation-playbooks/ansible.log is not writeable and we
cannot create it, aborting

Success! The validation passed for all hosts:
* undercloud
Running: sudo --preserve-env openstack tripleo deploy --standalone --standalone-role Undercloud
--stack undercloud --local-domain=localdomain --local-ip=192.168.24.1/24 --templates=/usr/share/
openstack-tripleo-heat-templates/ --networks-file=network_data_undercloud.yaml --heat-native -e /
usr/share/openstack-tripleo-heat-templates/environments/undercloud.yaml -e /usr/share/openstack-
tripleo-heat-templates/environments/use-dns-for-vips.yaml -e /usr/share/openstack-tripleo-heat-
templates/environments/podman.yaml -e /usr/share/openstack-tripleo-heat-templates/environments/
services/ironic.yaml -e /usr/share/openstack-tripleo-heat-templates/environments/services/ironic-
inspector.yaml -e /usr/share/openstack-tripleo-heat-templates/environments/services/mistral.yaml -e
/usr/share/openstack-tripleo-heat-templates/environments/services/zaqar-swift-backend.yaml -e /usr/
share/openstack-tripleo-heat-templates/environments/disable-telemetry.yaml -e /usr/share/openstack-
tripleo-heat-templates/environments/services/tempest.yaml -e /usr/share/openstack-tripleo-heat-
templates/environments/public-tls-undercloud.yaml --public-virtual-ip 192.168.24.2 --control-
virtual-ip 192.168.24.3 -e /usr/share/openstack-tripleo-heat-templates/environments/ssl/
tls-endpoints-public-ip.yaml -e /usr/share/openstack-tripleo-heat-templates/environments/services/
undercloud-haproxy.yaml -e /usr/share/openstack-tripleo-heat-templates/environments/services/
undercloud-keepalived.yaml --deployment-user stack --output-dir=/home/stack --cleanup -e /home/stack/
tripleo-config-generated-env-files/undercloud_parameters.yaml -e /usr/share/openstack-tripleo-heat-
templates/environments/tripleo-validations.yaml --log-file=install-undercloud.log -e /usr/share/
openstack-tripleo-heat-templates/undercloud-stack-vstate-dropin.yaml
/usr/lib64/python3.6/importlib/_bootstrap.py:219: ImportWarning: can't resolve package from __spec__
or __package__, falling back on __name__ and __path__
  return f(*args, **kwds)
The heat stack undercloud action is CREATE
--- output truncated ---

Install artifact is located at /home/stack/undercloud-install-20211123151416.tar.bzip2

#############################################################

Deployment successful!

#############################################################

Writing the stack virtual update mark file /var/lib/tripleo-heat-installer/update_mark_undercloud

#############################################################

The Undercloud has been successfully installed.

Useful files:

Password file is at /home/stack/undercloud-passwords.conf
The stackrc file is at ~/stackrc

Use these files to interact with OpenStack services, and
```

```
ensure they are secured.

############################################################

[stack@tripleo-director ~]$
```

Now that the director installation is complete, it's time for overcloud preparation and installation.

Obtaining Images for Overcloud Nodes

Several disk images are required for the director to provision overcloud nodes:

- An introspection kernel and ramdisk for the bare metal system introspection over the PXE boot.
- A deployment kernel and ramdisk for system provisioning and deployment.
- An overcloud kernel, ramdisk, and full image, which form a base overcloud system that is written to the hard disk of the node.

The following procedures show how to obtain the disk images:

```
[stack@tripleo-director ~]$ source stackrc
(undercloud) [stack@tripleo-director ~]$
(undercloud) [stack@tripleo-director ~]$ sudo dnf install rhosp-director-images rhosp-director-
images-ipa
Updating Subscription Management repositories.
/usr/lib/python3.6/site-packages/dateutil/parser/_parser.py:70: UnicodeWarning: decode() called on
unicode string, see https://bugzilla.redhat.com/show_bug.cgi?id=1693751
  instream = instream.decode()

Last metadata expiration check: 0:08:14 ago on Tue 23 Nov 2021 10:15:00 AM EST.
Dependencies resolved.
================================================================================================
================================================================================================
 Package                                    Architecture               Version
Repository                            Size
================================================================================================
================================================================================================
Installing:
 rhosp-director-images                         noarch                     16.1-20210430.3.el8ost
openstack-16.1-for-rhel-8-x86_64-rpms        9.5 k
---- output truncated ----
Installed products updated.

Installed:
  binutils-2.30-73.el8_2.1.x86_64                                dwz-0.12-9.el8.x86_64     efi-srpm-
macros-3-2.el8.noarch
  elfutils-0.178-7.el8.x86_64                                elfutils-debuginfod-client-0.178-7.
el8.x86_64                gc-7.6.4-3.el8.x86_64
  gdb-headless-8.2-11.el8.x86_64                              ghc-srpm-macros-1.4.2-7.el8.noarch
go-srpm-macros-2-16.el8.noarch
  guile-5:2.0.14-7.el8.x86_64                                libatomic_ops-7.6.2-3.el8.x86_64
libbabeltrace-1.5.4-2.el8.x86_64
```

```
  libipt-1.6.1-8.el8.x86_64                                    libtool-ltdl-2.4.6-25.el8.x86_64
ocaml-srpm-macros-5-4.el8.noarch
  octavia-amphora-image-x86_64-16.1-20210430.3.el8ost.noarch         openblas-srpm-macros-2-2.el8.
noarch                          patch-2.7.6-11.el8.x86_64
  perl-srpm-macros-1-25.el8.noarch                              qt5-srpm-macros-5.12.5-3.el8.noarch
redhat-rpm-config-122-1.el8.noarch
  rhosp-director-images-16.1-20210430.3.el8ost.noarch               rhosp-director-images-ipa-16.1-
20210430.3.el8ost.noarch         rhosp-director-images-ipa-x86_64-16.1-20210430.3.el8ost.noarch
  rhosp-director-images-x86_64-16.1-20210430.3.el8ost.noarch         rhosp-release-16.1.6-1.el8ost.
noarch                          rpm-build-4.14.2-38.el8_2.x86_64
  rpmdevtools-8.10-7.el8.noarch                                 rust-srpm-macros-5-2.el8.noarch
unzip-6.0-43.el8.x86_64
  zip-3.0-23.el8.x86_64                                        zstd-1.4.2-2.el8.x86_64

Complete!
```

Extract the image archives to the images directory in the home directory of the stack user (/home/stack/images):

```
(undercloud) [stack@tripleo-director ~]$ cd ~/images
(undercloud) [stack@tripleo-director images]$ for i in /usr/share/rhosp-director-images/overcloud-
full-latest-16.1.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-16.1.tar; do tar
-xvf $i; done
overcloud-full.qcow2
overcloud-full.initrd
overcloud-full.vmlinuz
overcloud-full-rpm.manifest
overcloud-full-signature.manifest
ironic-python-agent.initramfs
ironic-python-agent.kernel
```

Now import these images into director:

```
(undercloud) [stack@tripleo-director images]$ openstack overcloud image upload --image-path /home/
stack/images/
Image "overcloud-full-vmlinuz" was uploaded.
+--------------------------------------+------------------------+-------------+---------+--------+
|                  ID                  |          Name          | Disk Format |   Size  | Status |
+--------------------------------------+------------------------+-------------+---------+--------+
| 0c72fdeb-9776-41ac-be28-9198a12d5851 | overcloud-full-vmlinuz |     aki     | 8924528 | active |
+--------------------------------------+------------------------+-------------+---------+--------+
Image "overcloud-full-initrd" was uploaded.
+--------------------------------------+-----------------------+-------------+----------+--------+
|                  ID                  |          Name         | Disk Format |   Size   | Status |
+--------------------------------------+-----------------------+-------------+----------+--------+
| 3d671d3e-fe4d-46a3-b6e0-183826f0008a | overcloud-full-initrd |     ari     | 73075454 | active |
+--------------------------------------+-----------------------+-------------+----------+--------+
Image "overcloud-full" was uploaded.
+--------------------------------------+----------------+-------------+------------+--------+
|                  ID                  |      Name      | Disk Format |    Size    | Status |
+--------------------------------------+----------------+-------------+------------+--------+
| 4a9d8723-9601-42c6-9764-ef1d2bc6804a | overcloud-full |    qcow2    | 1113063424 | active |
+--------------------------------------+----------------+-------------+------------+--------+
```

Specify the named server if you intend for the overcloud to resolve external host names:

```
(undercloud) [stack@tripleo-director images]$ undercloud_nameserver=8.8.8.8
(undercloud) [stack@tripleo-director images]$ openstack subnet set `openstack subnet show ctlplane-
subnet -c id -f value` --dns-nameserver ${undercloud_nameserver}
(undercloud) [stack@tripleo-director images]$ openstack subnet show ctlplane-subnet
+-------------------+--------------------------------------------------------------------------------------
--------------------------------------------------------------------------+
| Field             | Value                                                                                 \|
+-------------------+--------------------------------------------------------------------------------------
--------------------------------------------------------------------------+
| allocation_pools  | 192.168.24.200-192.168.24.219                                                         |
| cidr              | 192.168.24.0/24                                                                       |
| created_at        | 2021-11-23T15:13:40Z                                                                  |
| description       |                                                                                       |
| dns_nameservers   | 8.8.8.8                                                                               |
| enable_dhcp       | True                                                                                  |
| gateway_ip        | 192.168.24.1                                                                          |
| host_routes       |                                                                                       |
| id                | 94348f7b-e491-467d-8548-5d079683220c                                                  |
| ip_version        | 4                                                                                     |
| ipv6_address_mode | None                                                                                  |
| ipv6_ra_mode      | None                                                                                  |
| location        | cloud='', project.domain_id=, project.domain_name='Default', project.id='39579e76
448640899bdc058917aeae2a', project.name='admin', region_name='', zone= |
| name              | ctlplane-subnet                                                                       |
| network_id        | e575b76f-d347-4232-8f43-165b9acd4cf1                                                  |
| prefix_length     | None                                                                                  |
| project_id        | 39579e76448640899bdc058917aeae2a                                                      |
| revision_number   | 1                                                                                     |
| segment_id        | None                                                                                  |
| service_types     |                                                                                       |
| subnetpool_id     | None                                                                                  |
| tags              |                                                                                       |
| updated_at        | 2021-11-23T15:35:37Z                                                                  |
+-------------------+--------------------------------------------------------------------------------------
--------------------------------------------------------------------------+
```

Red Hat Enterprise Linux 8.2 no longer includes the docker-distribution package, which installed a Docker Registry v2. To maintain the compatibility and the same level of feature, the director installation creates an Apache web server with a vhost called `image-serve` to provide a registry. This registry also uses port 8787/TCP with SSL disabled. The Apache-based registry is not containerized, meaning you must run the following command to restart the registry:

```
(undercloud) [stack@tripleo-director images]$ sudo systemctl restart httpd
```

Registering Nodes for the Overcloud

The next step is to register overcloud nodes to the undercloud to be recognized and thus be able to provision the overcloud stack. The JSON or YAML template needs to be created manually which contains hardware information and power management details. This template is needed to define overcloud bare metal nodes to undercloud.

Here is the template used in this book's deployment. MAC addresses shown under ports and the address hierarchy is the MAC address of the provisioning NIC of each server. The pm type, user, password, address and port are the power management information for each bare metal node. The NIC MAC addresses and power management information are critical for the node registration to be successful:

```
(undercloud) [stack@tripleo-director ~]$ cat instackenv.json
        {
           "nodes":[
              {
                 "ports":[
                    {
                       "address":"52:54:00:f1:0a:86"
                    }
                 ],
                 "name":"control_1",
                 "cpu":"8",
         "memory":"32768",
         "disk":"100",
         "arch":"x86_64",
                 "pm_type":"ipmi",
                 "pm_user":"admin",
                 "pm_password":"admin",
                 "pm_addr":"10.85.28.15",
                 "pm_port":"16201"
              },
              {
                 "ports":[
                    {
                       "address":"52:54:00:42:20:09"
                    }
                 ],
                 "name":"contrail_controller_1",
         "cpu":"8",
         "memory":"32768",
         "disk":"100",
                 "pm_type":"ipmi",
                 "pm_user":"admin",
                 "pm_password":"admin",
                 "pm_addr":"10.85.28.15",
                 "pm_port":"16202"
              },
              {
                 "ports":[
                    {
                       "address":"52:54:00:81:d9:2e"
                    }
                 ],
                 "name":"contrail_analytics_database_1",
                 "cpu":"4",
         "memory":"16348",
         "disk":"100",
         "arch":"x86_64",
                 "pm_type":"ipmi",
                 "pm_user":"admin",
                 "pm_password":"admin",
                 "pm_addr":"10.85.28.15",
```

```
            "pm_port":"16204"
         },
         {
            "ports":[
               {
                  "address":"52:54:00:aa:97:91"
               }
            ],
            "name":"contrail_analytics_1",
            "cpu":"4",
      "memory":"16348",
      "disk":"100",
      "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"admin",
            "pm_addr":"10.85.28.15",
            "pm_port":"16205"
         },
         {
            "ports":[
               {
                  "address":"52:54:00:46:ab:c1"
               }
            ],
            "name":"compute_2",
            "cpu":"12",
      "memory":"16348",
      "disk":"100",
      "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"admin",
            "pm_addr":"10.85.28.15",
            "pm_port":"16212"
         },
         {
            "ports":[
               {
                  "address":"52:54:00:f2:2f:c8"
               }
            ],
            "name":"compute_1",
            "cpu":"12",
      "memory":"16348",
      "disk":"100",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"admin",
            "pm_addr":"10.85.28.15",
            "pm_port":"16211"
         }
      ]
   }
```


Use this file to register each node into director as follows:

```
(undercloud) [stack@tripleo-director ~]$ openstack overcloud node import instackenv.json
```

```
Waiting for messages on queue 'tripleo' with no timeout.


6 node(s) successfully moved to the "manageable" state.
Successfully registered node UUID 693e0718-e628-4714-8600-08cc77f4b82b
Successfully registered node UUID 086a6cf4-0a2d-4ee7-aca3-f910e57fb8e2
Successfully registered node UUID 847f069f-cac6-4ec7-ab54-25532c9db821
Successfully registered node UUID 1a25797f-a829-447a-8105-63248f12660a
Successfully registered node UUID 6187bc91-821a-4f9a-b4ee-9654472a1c3e
Successfully registered node UUID 97b39ff4-3d1f-4109-8312-2a39e35c356e
(undercloud) [stack@tripleo-director ~]$
(undercloud) [stack@tripleo-director ~]$
```

You can verify it's successfully registered by running the following command, and you should expect to see similar output:

```
(undercloud) [stack@tripleo-director ~]$ openstack baremetal node list
+--------------------------------------+-------------------------------+---------------+-------------
+--------------------+-------------+
| UUID                                 | Name                          | Instance UUID | Power State | Provisioning
State | Maintenance |
+--------------------------------------+-------------------------------+---------------+-------------
+--------------------+-------------+
| 693e0718-e628-4714-8600-08cc77f4b82b | control_1                     | None          | power on    |
manageable         | False       |
| 086a6cf4-0a2d-4ee7-aca3-f910e57fb8e2 | contrail_controller_1         | None          | power on    |
manageable         | False       |
| 847f069f-cac6-4ec7-ab54-25532c9db821 | contrail_analytics_database_1 | None          | power on    |
manageable         | False       |
| 1a25797f-a829-447a-8105-63248f12660a | contrail_analytics_1          | None          | power on    |
manageable         | False       |
| 6187bc91-821a-4f9a-b4ee-9654472a1c3e | compute_2                     | None          | power on    |
manageable         | False       |
| 97b39ff4-3d1f-4109-8312-2a39e35c356e | compute_1                     | None          | power on    |
manageable         | False       |
+--------------------------------------+-------------------------------+---------------+-------------
+--------------------+-------------+
```

Now run the introspection process to boot an introspection agent over the PXE on each node. The purpose is to collect hardware data from the nodes and store it in the director to use for various purposes during deployment:

```
(undercloud) [stack@tripleo-director ~]$ openstack overcloud node introspect --all-manageable --provide
Waiting for introspection to finish...
Waiting for messages on queue 'tripleo' with no timeout.
Introspection of node completed:086a6cf4-0a2d-4ee7-aca3-f910e57fb8e2. Status:SUCCESS. Errors:None
Introspection of node completed:1a25797f-a829-447a-8105-63248f12660a. Status:SUCCESS. Errors:None
Introspection of node completed:847f069f-cac6-4ec7-ab54-25532c9db821. Status:SUCCESS. Errors:None
Introspection of node completed:97b39ff4-3d1f-4109-8312-2a39e35c356e. Status:SUCCESS. Errors:None
Introspection of node completed:693e0718-e628-4714-8600-08cc77f4b82b. Status:SUCCESS. Errors:None
Introspection of node completed:6187bc91-821a-4f9a-b4ee-9654472a1c3e. Status:SUCCESS. Errors:None
Successfully introspected 6 node(s).

Introspection completed.
Waiting for messages on queue 'tripleo' with no timeout.
6 node(s) successfully moved to the "available" state.
```

Adding Flavor for Contrail Nodes

Undercloud already has flavors created for OpenStack roles, so we need to create some other flavors for Contrail nodes.

This output shows the undercloud flavors before creating new flavors:

```
(undercloud) [stack@tripleo-director ~]$ openstack flavor list
+--------------------------------------+---------------+------+------+-----------+-------+---------
--+
| ID                                   | Name          |  RAM | Disk | Ephemeral | VCPUs | Is Public |
+--------------------------------------+---------------+------+------+-----------+-------+---------
--+
| 58c09563-94ab-4adb-8327-741e4e23a22a | swift-storage | 4096 |   40 |         0 |     1 | True      |
| 5a65c9f2-5d35-4275-80ac-f57070922c84 | ceph-storage  | 4096 |   40 |         0 |     1 | True      |
| 758e0dfc-b3ac-45b6-8219-8fd04e5147d6 | compute       | 4096 |   40 |         0 |     1 | True      |
| 9a7c7e6b-16c5-4c00-a6f3-490d6f908bfb | block-storage | 4096 |   40 |         0 |     1 | True      |
| af4cf0be-e39f-47f3-a6a6-7c6564f1db87 | control       | 4096 |   40 |         0 |     1 | True      |
| f762b817-1d03-4a39-8e48-f8a53c8bd9ca | baremetal     | 4096 |   40 |         0 |     1 | True      |
+--------------------------------------+---------------+------+------+-----------+-------+---------
--+
```

And to create additional flavors for Contrail nodes:

```
(undercloud) [stack@tripleo-director ~]$ for i in compute-dpdk \
> compute-kernel \
> contrail-dpdk \
> compute-sriov \
> contrail-controller \
> contrail-analytics \
> contrail-database \
> contrail-analytics-database; do
> openstack flavor create $i --ram 4096 --vcpus 1 --disk 40
> openstack flavor set --property "capabilities:boot_option"="local" \
> --property "capabilities:profile"="${i}" ${i}
> openstack flavor set --property resources:CUSTOM_BAREMETAL=1 --property resources:DISK_
GB='0' --property resources:MEMORY_MB='0' --property resources:VCPU='0' ${i}
> done
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| disk                       | 40                                   |
| id                         | 2bb86839-ca3a-49d0-8581-31a1ef3fd9df |
| name                       | compute-dpdk                         |
| os-flavor-access:is_public | True                                 |
| properties                 |                                      |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 1                                    |
+----------------------------+--------------------------------------+

+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
```

```
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| disk                       | 40                                   |
| id                         | d00d19bb-7b42-410a-8429-48ab4dc06e00 |
| name                       | compute-kernel                       |
| os-flavor-access:is_public | True                                 |
| properties                 |                                      |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 1                                    |
+----------------------------+--------------------------------------+
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| disk                       | 40                                   |
| id                         | e6dc6f96-5ca6-4de3-b2f6-27dc3f831f8a |
| name                       | contrail-dpdk                        |
| os-flavor-access:is_public | True                                 |
| properties                 |                                      |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 1                                    |
+----------------------------+--------------------------------------+
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| disk                       | 40                                   |
| id                         | e48de251-cc48-4f0f-ba33-b8e5e080d6ff |
| name                       | compute-sriov                        |
| os-flavor-access:is_public | True                                 |
| properties                 |                                      |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 1                                    |
+----------------------------+--------------------------------------+
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| disk                       | 40                                   |
| id                         | fd12c274-88ea-4062-ae85-ef5544a7f773 |
| name                       | contrail-controller                  |
| os-flavor-access:is_public | True                                 |
| properties                 |                                      |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 1                                    |
+----------------------------+--------------------------------------+
```

```
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| disk                       | 40                                   |
| id                         | b8bf963b-b98b-41a3-bc36-2620f36f28a8 |
| name                       | contrail-analytics                   |
| os-flavor-access:is_public | True                                 |
| properties                 |                                      |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 1                                    |
+----------------------------+--------------------------------------+
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| disk                       | 40                                   |
| id                         | 7703c667-5f3d-4336-9367-e55d61d0d6e7 |
| name                       | contrail-database                    |
| os-flavor-access:is_public | True                                 |
| properties                 |                                      |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 1                                    |
+----------------------------+--------------------------------------+
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| disk                       | 40                                   |
| id                         | 72024fb2-3b5e-4add-a001-0f983edeec15 |
| name                       | contrail-analytics-database          |
| os-flavor-access:is_public | True                                 |
| properties                 |                                      |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 1                                    |
+----------------------------+--------------------------------------+
(undercloud) [stack@tripleo-director ~]$
```

Here is the flavor list output after creating the new flavors:

```
(undercloud) [stack@tripleo-director ~]$ openstack flavor list
+--------------------------------------+----------------------------+------+------+-----------+-------+-----------+
| ID                                   | Name                       |  RAM | Disk | Ephemeral | VCPUs | Is Public |
+--------------------------------------+----------------------------+------+------+-----------+-------+-----------+
| 2bb86839-ca3a-49d0-8581-31a1ef3fd9df | compute-dpdk               | 4096 |   40 |         0 |     1 | True      |
| 58c09563-94ab-4adb-8327-741e4e23a22a | swift-storage              | 4096 |   40 |         0 |     1 | True      |
| 5a65c9f2-5d35-4275-80ac-f57070922c84 | ceph-storage               | 4096 |   40 |         0 |     1 | True      |
| 72024fb2-3b5e-4add-a001-0f983edeec15 | contrail-analytics-database | 4096 |   40 |         0 |     1 | True      |
| 758e0dfc-b3ac-45b6-8219-8fd04e5147d6 | compute                    | 4096 |   40 |         0 |     1 | True      |
| 7703c667-5f3d-4336-9367-e55d61d0d6e7 | contrail-database          | 4096 |   40 |         0 |     1 | True      |
| 9a7c7e6b-16c5-4c00-a6f3-490d6f908bfb | block-storage              | 4096 |   40 |         0 |     1 | True      |
```

```
| af4cf0be-e39f-47f3-a6a6-7c6564f1db87 | control                | 4096 |   40 |         0 |     1 | True      |
| b8bf963b-b98b-41a3-bc36-2620f36f28a8 | contrail-analytics     | 4096 |   40 |         0 |     1 | True      |
| d00d19bb-7b42-410a-8429-48ab4dc06e00 | compute-kernel         | 4096 |   40 |         0 |     1 | True      |
| e48de251-cc48-4f0f-ba33-b8e5e080d6ff | compute-sriov          | 4096 |   40 |         0 |     1 | True      |
| e6dc6f96-5ca6-4de3-b2f6-27dc3f831f8a | contrail-dpdk          | 4096 |   40 |         0 |     1 | True      |
| f762b817-1d03-4a39-8e48-f8a53c8bd9ca | baremetal              | 4096 |   40 |         0 |     1 | True      |
| fd12c274-88ea-4062-ae85-ef5544a7f773 | contrail-controller    | 4096 |   40 |         0 |     1 | True      |
+--------------------------------------+------------------------+------+------+-----------+-------+-----------+
```

Template Directory Creation

Now let's create the directory for templates and environment files to have both Red Hat OpenStack and Contrail template files.

Copy tripleO heat template files to new directory:

```
(undercloud) [stack@tripleo-director ~]$ cp -r /usr/share/openstack-tripleo-heat-templates/ tripleo-
heat-templates
```

Download and copy the Contrail heat templates from https://support.juniper.net/support/downloads.

The TAR downloaded file is shown here:

```
(undercloud) [stack@tripleo-director ~]$ tar -xzvf contrail-tripleo-heat-templates-2011.138.tgz
contrail-tripleo-heat-templates/
contrail-tripleo-heat-templates/.git/
contrail-tripleo-heat-templates/.git/hooks/
contrail-tripleo-heat-templates/.git/hooks/pre-applypatch.sample
--- output truncated ---
```

Then merge the Contrail tripleO heat template directory files with the OpenStack tripleO heat templates into one directory:

```
(undercloud) [stack@tripleo-director ~]$ cp -r contrail-tripleo-heat-templates/* tripleo-heat-
templates/
(undercloud) [stack@tripleo-director ~]$
```

Create rhsm.yaml file

You need to manually create the rhsm.yaml file. The rhsm.yaml file was explained in Chapter 4.

Create and Upload OpenStack Containers

In this step you create a file that contains a list of the needed OpenStack container images for overcloud deployment, and then upload all the container images to the undercloud registry:

```
(undercloud) [stack@tripleo-director ~]$ sudo openstack tripleo container image prepare -e ~/
containers-prepare-parameter.yaml -e ~/rhsm.yaml > ~/overcloud_containers.yaml

(undercloud) [stack@tripleo-director ~]$ sudo openstack overcloud container image upload --config-
file ~/overcloud_containers.yaml
(undercloud) [stack@tripleo-director ~]$
```

Here is the sample output file:

```
(undercloud) [stack@tripleo-director ~]$ cat overcloud_containers.yaml
# Generated with the following on 2021-11-23T17:44:13.299808
#
#   openstack tripleo container image prepare -e /home/stack/containers-prepare-parameter.yaml -e /home/stack/rhsm.
yaml
#

parameter_defaults:
  ContainerAodhApiImage: tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-aodh-api:16.1
  ContainerAodhConfigImage: tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-aodh-api:16.1
  ContainerAodhEvaluatorImage: tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-aodh-evaluator:16.1
  ContainerAodhListenerImage: tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-aodh-listener:16.1
  ContainerAodhNotifierImage: tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-aodh-notifier:16.1
  ContainerBarbicanApiImage: tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-barbican-api:16.1
  ContainerBarbicanConfigImage: tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-barbican-api:16.1
  ContainerBarbicanKeystoneListenerConfigImage: tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-
barbican-keystone-listener:16.1
  ContainerBarbicanKeystoneListenerImage: tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-barbican-
keystone-listener:16.1
  ContainerBarbicanWorkerConfigImage: tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-barbican-
worker:16.1
  ContainerBarbicanWorkerImage: tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-barbican-
worker:16.1
--- output truncated ---
```

To verify that the OpenStack container images have been uploaded to the local registry on undercloud, run the following command:

```
(undercloud) [stack@tripleo-director ~]$ openstack tripleo container image list
+-----------------------------------------------------------------------------------------+
| Image Name                                                                              |
+-----------------------------------------------------------------------------------------+
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-nova-compute:16.1   |
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-gnocchi-metricd:16.1            |
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-ironic-pxe:16.1                 |
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-aodh-evaluator:16.1             |
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-rsyslog:16.1                    |
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-cinder-volume:16.1              |
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-neutron-server-ovn:16.1         |
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-keepalived:16.1                 |
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-zaqar-wsgi:16.1                 |
--- output truncated ---
```

Create and Upload the Contrail Containers

As the deployment is for Contrail plus RHOSP, you need not only RHOSP container images but also Contrail container images. So, you need to repeat the previ-

ous step, but this time create and upload Contrail container images.

The script `upload_container.sh` in `tripleo-heat-templates/tools/contrail/` should be used for pulling Contrail container images from the registry and to push them to the local registry. The command has four options:

- `-r` to specify container registry to pull contrail containers
- `-u` to specify username of container registry
- `-p` to specify password of container registry
- `-t` to specify release tag

```
(undercloud) [stack@tripleo-director ~]$ cd tripleo-heat-templates/tools/contrail/
undercloud) [stack@tripleo-director contrail]$ ./upload_containers.sh -r hub.juniper.net/contrail -t
123 -u "juniper registry username" -p juniper registry password"  -t 2011.138
login to remote registry: hub.juniper.net/contrail
Login Succeeded!
pull hub.juniper.net/contrail/contrail-analytics-alarm-gen:2011.138
Trying to pull hub.juniper.net/contrail/contrail-analytics-alarm-gen:2011.138...
Getting image source signatures
Copying blob 2bd25ca12457 [======================================] 1.8KiB / 1.8KiB
Copying blob 917d33f53d9c [======================================] 131b / 131b
Copying blob 1390e99d4353 [======================================] 17.6KiB / 17.6KiB
Copying blob 3df39cfcab46 [======================================] 1.2KiB / 1.2KiB
--- output truncated ---
```

You can verify that the Contrail containers have been uploaded to the undercloud registry by issuing `openstack tripleo container image list` once again:

```
(undercloud) [stack@tripleo-director contrail]$ openstack tripleo container image list
+-----------------------------------------------------------------------------------------------+
| Image Name
+-----------------------------------------------------------------------------------------------+
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-neutron-metadata-agent-ovn:16.1   |
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-heat-api:16.1       |
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-heat-api-cfn:16.1     |
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-octavia-api:16.1      |
| docker://tripleo-director.ctlplane.localdomain:8787/rhosp-rhel8/openstack-nova-conductor:16.1                |
--- output truncated ---
| docker://tripleo-director.ctlplane.localdomain:8787/contrail/contrail-openstack-compute-init:2011.138       |
| docker://tripleo-director.ctlplane.localdomain:8787/contrail/contrail-openstack-neutron-init:2011.138       |
| docker://tripleo-director.ctlplane.localdomain:8787/contrail/contrail-nodemgr:2011.138       |
| docker://tripleo-director.ctlplane.localdomain:8787/contrail/contrail-analytics-snmp-topology:2011.138      |
| docker://tripleo-director.ctlplane.localdomain:8787/contrail/contrail-controller-config-schema:2011.138
```

Node Tagging

There are two ways of scheduling overcloud roles to specific nodes. One of them is by flavor and the other is by *node tagging*. In this deployment node tagging is used. Node tagging is simply attaching a tag to the node and then referencing that

tag in the node-placement.yaml environment file in the scheduler hints of specific roles. This is how you instruct undercloud to map specific roles to specific nodes. The node-placement.yaml file was explained in Chapter 3.

Run the following commands to tag bare metal nodes with the same tags referenced in the nodes-placement.yaml file:

```
(undercloud) [stack@tripleo-director contrail]$  openstack baremetal node set  compute_2 --property
capabilities='node:compute-1,boot_option:local,boot_mode:bios'
(undercloud) [stack@tripleo-director contrail]$  openstack baremetal node set contrail_analytics_
database_1 --property capabilities='node:contrailanalyticsdatabase-0,boot_option:local,boot_
mode:bios'
(undercloud) [stack@tripleo-director contrail]$
(undercloud) [stack@tripleo-director contrail]$ openstack baremetal node set  contrail_analytics_1
--property capabilities='node:contrailanalytics-0,boot_option:local,boot_mode:bios'
(undercloud) [stack@tripleo-director contrail]$ openstack baremetal node set  contrail_controller_1
--property capabilities='node:contrailcontroller-0,boot_option:local,boot_mode:bios'
(undercloud) [stack@tripleo-director contrail]$  openstack baremetal node set  control_1 --property
capabilities='node:controller-0,boot_option:local,boot_mode:bios'
```

The highlighted tags match the tags specified in the scheduler hints in node-placement.yaml file:

```
(undercloud) [stack@tripleo-director ~]$ cat tripleo-heat-templates/environments/contrail/nodes-
placement.yaml
parameter_defaults:
  ControllerSchedulerHints:
    'capabilities:node': 'controller-%index%'
  ContrailControllerSchedulerHints:
    'capabilities:node': 'contrailcontroller-%index%'
  ContrailAnalyticsSchedulerHints:
    'capabilities:node': 'contrailanalytics-%index%'
  ContrailAnalyticsDatabaseSchedulerHints:
    'capabilities:node': 'contrailanalyticsdatabase-%index%'
  ComputeSchedulerHints:
    'capabilities:node': 'compute-%index%'

--- output truncated ---
```

Environment Files Customization

We already had a walk-through with the environment files in Chapter 3, so we are not going to explain them again here. Instead, this section gives you some tips about the correlation between different parameters in different files in order to clear ambiguities and prevent errors that could lead to a failed deployment trial.

Here is a list of environment role files used in the deployment:

- network-isolation.yaml
- contrail-plugins.yaml
- nodes-placement.yaml

- contrail-services.yaml
- contrail-net.yaml
- ips-from-pool-all.yaml
- fixed-ip-vips.yaml
- rhsm.yaml
- roles_data_contrail.yaml

NOTE Naming of environment and roles files are not standard and you can change them according to your needs.

Contrail-net.yaml and roles_data_contrail.yaml correlation to nic config templates

In contrail-net.yaml you find how overcloud NICs should be configured by mapping each role to a specific NIC template configuration file. You can also find information about which networks should be configured per node role in the roles_data_contrail.yaml file.

To avoid any errors, you need to have consistency between the two files and the corresponding NIC config templates. In a role file you'll find networks mapped to the role as show below:

```
###############################################################################
# Role: Compute                                                               #
###############################################################################
- name: Compute
  description: |
    Basic Compute Node role
  CountDefault: 0
  # # Create external Neutron bridge (unset if using ML2/OVS without DVR)
  # tags:
  #   - external_bridge
  networks:
    InternalApi:
      subnet: internal_api_subnet
    Tenant:
      subnet: tenant_subnet
    Storage:
      subnet: storage_subnet
```

In contrail-net you'll find compute NICs mapped to specific NIC configuration templates as shown here:

```
(undercloud) [stack@tripleo-director ~]$ 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::ContrailAnalytics::Net::SoftwareConfig: ../../network/config/contrail/contrail-ca-
nic-config.yaml
  OS::TripleO::ContrailAnalyticsDatabase::Net::SoftwareConfig: ../../network/config/contrail/
contrail-ca-nic-config.yaml
  OS::TripleO::Compute::Net::SoftwareConfig: ../../network/config/contrail/compute-bond-nic-config.
yaml
  OS::TripleO::ContrailDpdk::Net::SoftwareConfig: ../../network/config/contrail/contrail-dpdk-bond-
nic-config.yaml
```

The NIC template file `compute-bond-nic-config.yaml` should now exist in the respective location and should also have a correct configuration that takes into consideration the three networks defined under the compute role in the roles file:

```
cat tripleo-heat-templates/network/config/contrail/compute-bond-nic-config.yaml
           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: 169.254.169.254/32
                  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: linux_bond
                name: bond0
                bonding_options: "miimon=100 lacp_rate=slow mode=4 xmit_hash_policy=layer2+3"
                mtu: 9000
```

```
      use_dhcp: false
      members:
       -
         type: interface
         mtu: 9000
         use_dhcp: false
         name: nic2
       -
         type: interface
         mtu: 9000
         use_dhcp: false
         name: nic3
    - type: vlan
      vlan_id:
        get_param: TenantNetworkVlanID
      mtu: 9000
      use_dhcp: false
      device: bond0
    - type: contrail_vrouter
      name: vhost0
      mtu: 9000
      use_dhcp: false
      members:
        -
          type: interface
          name:
            str_replace:
              template: vlanVLANID
              params:
                VLANID: {get_param: TenantNetworkVlanID}
          mtu: 9000
          use_dhcp: false
      addresses:
      - ip_netmask:
          get_param: TenantIpSubnet
```

If any of the three files have an inconsistency, the deployment will fail.

Templates Processing

After customizing all the environment files, the last step before issuing the deploy command is to process templates to generate fixed and VIP ports configuration files and render the network-isolation file. Here's the procedure:

```
(undercloud) [stack@tripleo-director tools]$ ./process-templates.py \
> -r ~/tripleo-heat-templates/roles_data_contrail.yaml \
> -p ~/tripleo-heat-templates/
jinja2 rendering normal template net-config-bond.j2.yaml
rendering j2 template to file: /home/stack/tripleo-heat-templates/net-config-bond.yaml
jinja2 rendering normal template net-config-bridge.j2.yaml
rendering j2 template to file: /home/stack/tripleo-heat-templates/net-config-bridge.yaml
jinja2 rendering normal template net-config-linux-bridge.j2.yaml
rendering j2 template to file: /home/stack/tripleo-heat-templates/net-config-linux-bridge.yaml
jinja2 rendering normal template net-config-noop.j2.yaml
rendering j2 template to file: /home/stack/tripleo-heat-templates/net-config-noop.yaml
```

```
--- output truncated ---
```

Deploy Overcloud

This is the last and final step to deploy overcloud. However, there is a major difference between RHOSP16 compared to RHOSP13 in stack creation steps. In RHOSP13 deployment is successful when you see the *stack create complete* message. However, in RHOSP 16 the stack create complete is not the final message because the logic is different in release 16. In RHOSP16, after the stack create complete message Ansible is doing its job in configuring pretty much everything, so you need to wait until you get a message like the output here:

```
(undercloud) [stack@tripleo-director ~]$   openstack overcloud deploy --templates ~/tripleo-heat-
templates \
>     -e ~/containers-prepare-parameter.yaml \
>     -e ~/tripleo-heat-templates/environments/network-isolation.yaml \
>     -e ~/tripleo-heat-templates/environments/contrail/contrail-plugins.yaml \
>     -e ~/tripleo-heat-templates/environments/contrail/nodes-placement.yaml \
>     -e ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
>     -e ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
>     -e ~/tripleo-heat-templates/environments/ips-from-pool-all.yaml \
>     -e ~/tripleo-heat-templates/environments/fixed-ip-vips.yaml \
>     -e ~/rhsm.yaml \
>     -r ~/tripleo-heat-templates/roles_data_contrail.yaml
Removing the current plan files
Uploading new plan files
Temporary Swift GET/PUT URL parameters have successfully been updated.
Plan updated.
Processing templates in the directory /tmp/tripleoclient-yvz_i_tl/tripleo-heat-templates
WARNING: Following parameter(s) are defined but not currently used in the deployment plan. These
parameters may be valid but not in use due to the service or deployment configuration.
ContrailDpdkHugepages1GB, ContrailDpdkHugepages2MB, ContrailDpdkSriovIPs, ContrailDpdkSriovParameters,
ContrailDpdkSriovSchedulerHints, ContrailRegistryInsecure, ContrailSriovHugepages1GB,
ContrailSriovHugepages2MB, NeutronFlatNetworks, NeutronNetworkType, NeutronNetworkVLANRanges,
NeutronPluginExtensions, NeutronTunnelIdRanges, NeutronTunnelTypes, NeutronTypeDrivers,
NeutronVniRanges, RhsmVars
Deploying templates in the directory /tmp/tripleoclient-yvz_i_tl/tripleo-heat-templates
Initializing overcloud plan deployment
Creating overcloud Heat stack
--- output truncated ---
2021-03-29 17:57:52Z [overcloud.AllNodesDeploySteps.ContrailAnalyticsPostConfig]: CREATE_COMPLETE
state changed
2021-03-29 17:57:52Z [overcloud.AllNodesDeploySteps.ControllerPostConfig]: CREATE_COMPLETE  state
changed
2021-03-29 17:57:52Z [overcloud.AllNodesDeploySteps.ContrailAnalyticsDatabasePostConfig]: CREATE_IN_
PROGRESS  state changed
2021-03-29 17:57:52Z [overcloud.AllNodesDeploySteps.ContrailAnalyticsDatabasePostConfig]: CREATE_
COMPLETE  state changed
2021-03-29 17:57:52Z [overcloud.AllNodesDeploySteps]: CREATE_COMPLETE  Stack CREATE completed
successfully
2021-03-29 17:57:53Z [overcloud.AllNodesDeploySteps]: CREATE_COMPLETE  state changed
--- output truncated ---
--- output truncated ---
PLAY RECAP *********************************************************************
```

```
jnprca01               : ok=340  changed=176  unreachable=0    failed=0    skipped=1136 rescued=0
ignored=0
jnprcadb01             : ok=317  changed=170  unreachable=0    failed=0    skipped=1159 rescued=0
ignored=0
jnprcc01               : ok=343  changed=179  unreachable=0    failed=0    skipped=1133 rescued=0
ignored=0
jnprcp01               : ok=375  changed=203  unreachable=0    failed=0    skipped=1156 rescued=0
ignored=0
jnprcp02               : ok=372  changed=203  unreachable=0    failed=0    skipped=1156 rescued=0
ignored=0
jnprct01               : ok=385  changed=222  unreachable=0    failed=0    skipped=1159 rescued=0
ignored=0
Monday 29 March 2021  14:58:35 -0400 (0:00:00.061)       0:58:06.039 **********
===============================================================================
Wait for containers to start for step 2 using paunch ------------------- 539.55s
Wait for container-puppet tasks (generate config) to finish ------------ 354.40s
Wait for puppet host configuration to finish --------------------------- 118.45s
Wait for containers to start for step 5 using paunch ------------------- 113.01s
Pre-fetch all the containers ------------------------------------------- 103.78s
Run puppet on the host to apply IPtables rules ------------------------- 97.20s
Wait for puppet host configuration to finish --------------------------- 92.98s
Wait for puppet host configuration to finish --------------------------- 90.46s
Wait for container-puppet tasks (bootstrap tasks) for step 4 to finish - 89.96s
Wait for puppet host configuration to finish --------------------------- 89.67s
Wait for puppet host configuration to finish --------------------------- 86.59s
Wait for container-puppet tasks (bootstrap tasks) for step 3 to finish - 86.49s
Pre-fetch all the containers ------------------------------------------- 84.13s
Wait for containers to start for step 3 using paunch ------------------- 74.07s
Pre-fetch all the containers ------------------------------------------- 65.61s
Wait for containers to start for step 4 using paunch ------------------- 62.33s
tripleo-kernel : Reboot after kernel args update ----------------------- 61.10s
Run async deployment ContrailNodeInitDeployment ------------------------ 42.20s
Pre-fetch all the containers ------------------------------------------- 40.60s


Host 20.2.0.240 not found in /home/stack/.ssh/known_hosts
Overcloud Endpoint: http://20.2.0.240:5000
Overcloud Horizon Dashboard URL: http://20.2.0.240:80/dashboard
Overcloud rc file: /home/stack/overcloudrc
```

Now you can play with overcloud after sourcing the overcloud credential file and logging in to overcloud:

```
(undercloud) [stack@tripleo-director ~]$ source
(undercloud) [stack@tripleo-director ~]$ source overcloudrc
(overcloud) [stack@tripleo-director ~]$ openstack server list

(overcloud) [stack@tripleo-director ~]$
```