JUNIPER NETWORKS | Engineering Simplicity

Contrail Cloud Deployment Guide

Published
2023-07-14

RELEASE
Release 13.7

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

Contrail Cloud Deployment Guide
Release 13.7

YEAR 2000 NOTICE

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

END USER LICENSE AGREEMENT

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

Table of Contents

CHAPTER 1

Deploy Contrail Cloud

Contrail Cloud Overview

IN THIS SECTION

Juniper Networks® Contrail® Cloud is an integrated Telco cloud platform built to run high-performance Network Functions Virtualization with always-on reliability, allowing service providers to deliver innovative services with greater agility. Contrail Cloud features Red Hat® OpenStack and Red Hat Ceph combined with Juniper Networks Contrail Networking™, thereby bridging dynamic cloud orchestration with highly scalable connectivity. Contrail Cloud leverages AppFormix, which has a built-in automation capability powered by machine learning, to monitor the cloud infrastructure and VNFs in the most optimal manner, to remediate any potential failures to ensure adherence to service-level agreements (SLAs).

The following sections define key components of Contrail Cloud and your Contrail Cloud deployment. This will give you a basic understanding of the structure and interrelations of the different Contrail Cloud components, and help you understand what happens during the deployment process. For a detailed understanding of Contrail Cloud beyond the scope of the basic deployment instructions, see Contrail Cloud Reference Architecture Guide.

Contrail Cloud Node Types

Understanding the different node types that are used in Contrail Cloudwill help you configure the YAML files for deployment.

jump host — A bare-metal hypervisor from which all Contrail Cloud deployment scripts are run.

undercloud — A virtual machine (VM) that runs on the jump host. The undercloud deploys and configures the overcloud.

overcloud	The overcloud is considered the production OpenStack Cloud. The overcloud includes resources for the control plane, baremetal storage nodes, and baremetal compute nodes as deployed by the undercloud.
Contrail Command	A VM that runs on the jump host and provides the user interface for Contrail Networking.
control host	A baremetal server on which one or more control VMs run on.
control VMs	VMs that run on one or more control hosts. Control VMs include: • OpenStack Controller • Contrail Controller • Contrail Analytics • Contrail Analytics Database • Contrail TSN (optional) • AppFormix Controller
compute nodes	Baremetal servers on which tenant VMs run. The different compute node types include: • Kernel • DPDK • SR-IOV
storage nodes	Baremetal servers that are loaded with storage drives for running Ceph storage software. The use of Ceph storage is optional.

Contrail Cloud Directory Structure

Contrail Cloud is installed on the jump host and located in **/var/lib/contrail_cloud/**. The following is the Contrail Cloud file directory structure:

- The **appformix** directory holds your AppFormix license and any user-defined AppFormix plugins.
- The **auth** directory stores the overcloudrc (overcloud) and stackrc (undercloud) credential files.
- The **certs** directory holds your SSL certificates for Contrail and OpenStack.

- The **config** directory has all the configuration files for deploying Contrail Cloud. The configuration files are in the YAML (.yaml) file format.
- The **docs** directory contains a copy of the Contrail Cloud Deployment Guide.This guide includes the essential steps for deploying Contrail Cloud.
- The **facts.d** directory contains information about the Contrail Cloud environment and holds the state across Ansible runs to allow the playbooks to remember important deployment information.
- The **introspection** directory contains detailed information about the hardware that is used and how the hardware is configured. This information is useful when planning your overall Contrail Cloud deployment. This information can also be used to verify that NICs are assigned properly, along with VLANs and bonds.
- The **samples** directory contains sample snippets of the various YAML configuration files which can be referenced for deployment. Each snippet details a specific use case configuration. This directory also contains information about the various knobs and configuration options available, and is separated by features and scaling use cases.
- The **scripts** directory contains various required and optional scripts that are used with Contrail Cloud, including pre-deployment verification scripts, deployment scripts, introspection and post-deployment verification tools.
- The **tmp** directory holds the temporary files that are required by Contrail Cloud.

Configuration Files

This section briefly defines each of the configuration files that are used in your Contrail Cloud deployment. For specific configuration examples, see **/var/lib/contrail_cloud/samples** on the jump host. All configuration files are in the YAML file format using the Jinja2 template. All deployment configuration files are located in the **/var/lib/contrail_cloud/config** directory. The following configuration files are used in your Contrail Cloud deployment:

site.yml	This configuration file contains information about the specific undercloud and overcloud instance.
inventory.yml	This configuration file is for baremetal server configuration, such as IPMI and UEFI.
overcloud_nics.yml	You can use this configuration file to configure network options for the overcloud nodes.

control-host-nodes.yml	You can configure the network layout for the control host server(s) in this YAML file.
compute-nodes.yml	Configure this file to assign naming and role mapping for all computes.
storage-nodes.yml	Define the role mapping of the storage nodes.
vault-data.yml	This is an encrypted file that holds all sensitive user data, such as product keys and user data.

NOTE: Contrail Cloud also provides default values for all Contrail Cloud playbooks. The default values can be found in **/var/lib/contrail_cloud/ansible/playbooks/default.yml**. Advanced users may review this file to look for values they wish to override in their **config/site.yml**.

Never modify the **default.yml** file directly. Always apply the values from the **default.yml** into your **config/site.yml**.

RELATED DOCUMENTATION

- Contrail Cloud Documentation
- Contrail Cloud Reference Architecture Guide
- Product Documentation for Red Hat OpenStack Platform 13

Deploy Contrail Cloud

IN THIS SECTION

Prerequisites for Contrail Cloud Deployment

Before you deploy Contrail Cloud, ensure that your system meets the following prerequisites:

- Infrastructure networking
  - Every system must have access to the Contrail Cloud repository satellite. The satellite is used to distribute packages and control software versions.
  - The Contrail Cloud jump host must have access to the Intelligent Platform Management Interface (IPMI) of every managed server.
  - The Contrail Cloud jump host must be in the same broadcast domain as each managed server's management interface to allow Preboot Execution Environment (PXE) booting. When you use multiple networks that use different switching devices per rack, this is accomplished by stretching a VLAN across the interfaces. BOOTP forwarding in the network fabric is not supported. The undercloud is the only DHCP server in this network.

    Additional networks are created for the control plane, tenant traffic, storage access, and storage backend, as described in Red Hat OpenStack Platform director (OSPd) installation and usage.
- Contrail Cloud jump host setup

  The OpenStack undercloud is deployed as a virtual machine on a Linux kernel-based virtual machine (KVM) Contrail Cloud jump host. You must ensure that the KVM host OS:
  - Runs Red Hat Enterprise Linux 7.8 or earlier with only base packages installed. Contrail Cloud will install RHEL 7.9 and all necessary packages as part of the Contrail Cloud installation process.
  - Does not run other virtual machines.
  - Has a network connection that can reach the Contrail Cloud Repository Satellite and has IPMI access to physical hardware.
  - Has a network connection that can be used for provisioning other infrastructure resources.
  - Has at least 500 GB space in the **/var** directory to host virtual machines, packages, and images.
  - Has at least 40 GB RAM and 24 vCPUs.
  - Supports users such as a root user with password-less `sudo` permissions.
  - Provides password-less SSH access in loopback for users with `sudo` permissions.
  - Resolves Internet and satellite sites with DNS.
  - Has time synchronized with an NTP source.

Deployment Sequence for Contrail Cloud Deployment

IN THIS SECTION

The following sections describe how to install, configure, and deploy Contrail Cloud.

Install Contrail Cloud Installer on the Contrail Cloud Host

Send a request to mailto:contrail_cloud_subscriptions@juniper.net regarding the purchase or upgrade of Contrail Cloud. You will receive an e-mail containing:

- A tar file containing the Contrail Cloud installer in **.sh** format. Untar the file to extract the `contrail_cloud_installer.sh` script.
- A unique satellite activation key.
- The satellite DNS name.
- The satellite organization.

You configure site.yml file settings to use the Contrail Cloud Satellite as the repository to access all Contrail Cloud packages including:

- Red Hat OpenStack and Red Hat Enterprise Linux
- Red Hat Ceph Storage
- Contrail Networking
- AppFormix

Ensure that the root user has SSH keys before performing the installation. Create new SSH keys:

```
yes '' | sudo ssh-keygen -t rsa -N ''
```

To create a passphrase-protected key:

```
sudo ssh-keygen -t rsa
```

If a passphrase is set on the SSH key, you can use the ssh-agent to cache the passphrase. For example, as the `contrail` user on the jump host:

```
ssh-agent bash
ssh-add <key_path>   (default: /home/contrail/.ssh/id_rsa)
```

Ensure that the root user can connect via SSH to the localhost without a password. To authorize access (a password might be required the first time):

```
sudo ssh-copy-id localhost
```

Install Contrail Cloud:

1. Untar the `contrail_cloud_installer.sh` on the jump host. The jump host is the Contrail Cloud host, and it is the starting point for deploying Contrail Cloud.
2. Specify the Contrail Cloud activation key by setting the environment variables.

   For example:

   ```
   SATELLITE="contrail-cloud-satellite.juniper.net"
   SATELLITE_KEY="ak-my-account-key"
   SATELLITE_ORG=”ContrailCloud”
   ```

3. Ensure that the Contrail Cloud Installer script has the required permissions to install the Contrail Cloud packages:

```
./contrail_cloud_installer.sh \
--satellite_host ${SATELLITE} \
--satellite_key ${SATELLITE_KEY} \
--satellite_org ${SATELLITE_ORG}
```

   The Contrail Cloud packages are installed in the **/var/lib/contrail_cloud** directory.

4. Define site-specific information in the Ansible variables:

   a. Change the directory to **/var/lib/contrail_cloud/config**.

   b. Copy the sample **/var/lib/contrail_cloud/samples/*.yml** configuration files to the **/var/lib/contrail_cloud/config** directory. You can skip this step if you have existing configuration files in the **config** directory.

   c. Customize the **/var/lib/contrail_cloud/config/site.yml** file with site-specific settings to reflect your environment. Ensure that the following fields are changed for each site:

```
global:
  # List of DNS nameservers
  dns:
    - "8.8.8.8"
  # List of NTP time servers
  ntp:
    - "66.129.255.62"
  # Timezone for all servers
  timezone: 'America/Los_Angeles'
  rhel:
    # Contrail Cloud Activation Key
    satellite:
      #SATELLITE_KEY should be defined in vault-data.yml file
      #SATELLITE_ORG
      organization: "ContrailCloud"
      #SATELLITE_FQDN
      fqdn: contrail-cloud-satellite.juniper.net
  # DNS domain information.
  # Must be unique for every deployment to avoid name conflicts
  domain: "my.unique.domain"

jumphost:
```

```
  network:
    provision:
      # jumphost nic to be used for provisioning (PXE booting) servers
      nic: eno1
```

5. Set the DPDK driver to `vfio-pci` in the **site.yml** file when deploying DPDK on an Intel X710 NIC.

   For a complete matrix of supported NIC and driver mapping, see Contrail Networking NIC Support Matrix.

   NOTE: For Netronome, see: "Appendix C: Deploying Netronome SmartNIC" on page 89.

```
overcloud:
  contrail:
    vrouter:
      dpdk:
        driver: "vfio-pci"
```

6. Prepare Ansible Vault:

   a. Store all sensitive data in Ansible Vault. Credentials are commonly stored in Ansible Vault. Customize the **/var/lib/contrail_cloud/config/vault-data.yml**:

```
ansible-vault edit /var/lib/contrail_cloud/config/vault-data.yml
```

   b. Change the password for the vault-encrypted file after customization.

      The default password is `c0ntrail123`.

```
ansible-vault rekey /var/lib/contrail_cloud/config/vault-data.yml
```

   You can use a plain-text password to create the **/var/lib/contrail_cloud/config/.vault_password** file to avoid being asked for a vault password during deployment. The file should be read-only by the `contrail` user, and it is best practice to delete the file after deployment is finished.

```
sudo chown contrail /var/lib/contrail_cloud/config/.vault_password
sudo chmod 0400 /var/lib/contrail_cloud/config/.vault_password
```

7. Run the Contrail Cloud Ansible provisioning:

   a. Verify that you can establish an SSH connection without specifying a password:

      ```
      sudo ssh localhost true
      ```

   b. Install the Contrail Cloud automation scripts:

      ```
      sudo /var/lib/contrail_cloud/scripts/install_contrail_cloud_manager.sh
      ```

      NOTE: Take the optional step to configure Netronome at this time. For more information, see: "Appendix C: Deploying Netronome SmartNIC" on page 89.

A new user with the username `contrail` is created on the Contrail Cloud host (jump host). The default password is `c0ntrail123`. You should change the default password in your **vault-data.yml** file. SSH keys have been authorized from the `root` user. A new set of SSH keys is generated for this new user. This gives access to the undercloud VM by executing `ssh undercloud`, and also control hosts for the `contrail` user. The overcloud nodes including AppFormix nodes are accessible by the `heat-admin` user, and use a separate pair of keys stored on the undercloud VM, by default.

Contrail Cloud adds entries into **/home/contrail/.ssh/config** directory to provide the username used for each of the overcloud nodes (and the undercloud). Thus, you can `ssh undercloud` or `ssh <address>` without specifying a user.

The `contrail` user keys can be authorized for `heat-admin` when it is defined in the site.yml configuration file:

```
global:
  service_user:
    use_ssh_key_in_overcloud: true
```

Use the `contrail` user to run all subsequent operations in Contrail Cloud from the **/var/lib/contrail_cloud/scripts** directory:

```
su - contrail
cd /var/lib/contrail_cloud/scripts
```

NOTE: The SSH keys are authorized by the `root` user.

Prepare the Deployment Templates

You can validate your configuration files at any time by running the `/var/lib/contrail_cloud/scripts/node-configuration.py` script. This scipt will load all the configuration files, check the syntax, and verify that the structures and values conform to the schema. Different arguments can be used with the Python script depending on what results you are looking for.

- Site settings

  The **/var/lib/contrail_cloud/config/site.yml** file defines the properties for your deployment environment. The properties in this file are unique for every deployment and need to be customized.

  For more information, see *sample site.yml* in "Appendix A: Sample Configuration Files" on page 18.

- Inventory settings

  The inventory defines all the servers that are used by Contrail Cloud. The **/var/lib/contrail_cloud/config/inventory.yml** file contains the descriptions of the inventory.yml file configurations. You can copy a sample inventory file from **/var/lib/contrail_cloud/samples/**.

  For more information, see *sample inventory.yml* in "Appendix A: Sample Configuration Files" on page 18.

- Control hosts settings

  The control hosts run virtual machines for all Contrail Cloud control functions. The following are the various Contrail Cloud control VMs that are created on the control hosts:

  - OpenStack and Ceph Controller
  - Contrail Controller
  - Contrail Analytics
  - Contrail Analytics Database
  - AppFormix Controller

  The **/var/lib/contrail_cloud/config/control-host-nodes.yml** file defines the server and network properties for each control host. To ensure high availability of the control functions, three control hosts must be defined. Hosts must also be defined in the **inventory.yml** file. You can copy a sample **control-hosts.yml** file from the **/var/lib/contrail_cloud/samples/** directory.

  NOTE: The control host systems must have sufficient resources to host the control VMs. The control host systems must have the following minimum specifications:

- 156 GB RAM
- Minimum 100 GB first disk for the operating system
- Minimum 1 TB hard disk for VM storage (multiple SSDs with RAID is recommended)
- Minimum 200 GB SSD drive for VM journals
- Hardware RAID controller set to the appropriate RAID level for your operating environment. The operating environment includes: operating system disk, VM storage, and VM journals.

For more information, see *sample control-host-nodes.yml* in "Appendix A: Sample Configuration Files" on page 18.

- Overcloud network settings

  The overcloud roles are deployed to the control VMs, compute, and storage resources that you have identified. The **/var/lib/contrail_cloud/config/overcloud-nics.yml** file defines the network layout for each role.

  For more information, see *sample overcloud-nics.yml* in "Appendix A: Sample Configuration Files" on page 18.

- Compute node settings

  The compute nodes are used for Nova compute resources. The **/var/lib/contrail_cloud/config/compute-nodes.yml** file defines the compute resources and host aggregates. You also manage host aggregates and match them with availability zones in this file. Nodes must also be defined in the **inventory.yml** file.

  For more information, see *sample compute-nodes.yml* in "Appendix A: Sample Configuration Files" on page 18.

- Storage node settings

  The **/lib/contrail_cloud/config/storage-nodes.yml** file defines the storage nodes that run Ceph storage services. You must define a minimum of three storage hosts to ensure high availability of the storage functions. Nodes must also be defined in the **inventory.yml** file. You can copy a sample **storage-nodes.yml** file from **/var/lib/contrail_cloud/samples/**.

  For more information, see *sample storage-nodes.yml* in "Appendix A: Sample Configuration Files" on page 18.

Add Nodes to the Inventory

The **/var/lib/contrail_cloud/scripts/inventory-assign.sh** script adds all nodes that are defined in the **/var/lib/contrail_cloud/config/inventory.yml** file to the ironic inventory. The nodes added to the ironic inventory are managed by Contrail Cloud.

To add nodes to the ironic inventory:

1. Log in to the Contrail Cloud jump host with the username `contrail` and password `c0ntrail123`.
2. Run the **inventory-assign.sh** script.

   ```
   /var/lib/contrail_cloud/scripts/inventory-assign.sh
   ```

3. Generate a report of the available resource properties.

   These details are helpful when configuring roles, disk devices, and network interfaces. Nodes must be loaded into the Ironic inventory before running the **node-cofiguration.py** script. The **node-cofiguration.py** script is also used to validate configurations against the schema, and can be used after editing any of the configuration files. The report can be generated by executing:

   ```
   /var/lib/contrail_cloud/scripts/node-configuration.py group
   ```

   More detailed reports for a specific resource can be generated (where <resource> is the inventory resource name):

   ```
   /var/lib/contrail_cloud/scripts/node-_configuration.py show -i /var/lib/contrail_cloud/
   introspection/<resource>.introspection
   ```

Deploy Control Hosts

The **control-hosts-deploy.sh** script assigns all nodes that are defined in the **/var/lib/contrail_cloud/config/control-host-nodes.yml** file as control hosts. The hosts are imaged, booted, configured, and prepared to host the overcloud control plane VMs.

To deploy control host roles to the inventory:

1. Log in to the Contrail Cloud jump host with the username `contrail` and password `c0ntrail123`.
2. Run the **control-hosts-deploy.sh** script.

   ```
   /var/lib/contrail_cloud/scripts/control-hosts-deploy.sh
   ```

Create VMs for all Control Roles

The **control-vms-deploy.sh** script imports VM details into the ironic inventory.

To create VMs for control roles:

1. Log in to the Contrail Cloud jump host with the username `contrail` and password `c0ntrail123`.

2. Run the **control-vms-deploy.sh** script.

   `/var/lib/contrail_cloud/scripts/control-vms-deploy.sh`

Assign Compute Hosts

The **compute-nodes-assign.sh** script assigns the Nova compute role for all nodes that are defined in the **/var/lib/contrail_cloud/config/compute-nodes.yml** file.

To assign compute hosts:

1. Log in to the Contrail Cloud jump host with the username `contrail` and password `c0ntrail123`.
2. Run the **compute-nodes-assign.sh** script.

   /var/lib/contrail_cloud/scripts/compute-nodes-assign.sh

Assign Storage Hosts

The **storage-nodes-assign.yml** playbook assigns the Ceph storage role for all nodes that are defined in the **/var/lib/contrail_cloud/config/storage-nodes.yml** file.

To assign storage hosts:

1. Log in to the Contrail Cloud jump host with the username `contrail` and password `c0ntrail123`.
2. Run the **storage-nodes-assign.sh** script.

   /var/lib/contrail_cloud/scripts/storage-nodes-assign.sh

Deploy the OpenStack Cluster

The **openstack-deploy.sh** script deploys the OpenStack overcloud with all control functions and all compute and storage resources that have been defined in the previous playbooks.

To deploy the OpenStack cluster:

1. Log in to the Contrail Cloud jump host with the username `contrail` and password `c0ntrail123`.
2. Run the `validate-node.sh` script to verify that the environment is set correctly.

   Run the `validate-node.sh` script on the jump host in **/var/lib/contrail_cloud/scripts** to validate the YAML configuration files for:

   - Network for OpenStack Controllers, Contrail Controller, Contrail Analytics, and Analytics DB.
   - Networking for controller hosts and compute hosts.
   - Disk resource and configuration validation.
3. Run the **openstack-deploy.sh** script.

   `/var/lib/contrail_cloud/scripts/openstack-deploy.sh`

Deploy the AppFormix Cluster

The **appformix-deploy.sh** script deploys the AppFormix controllers.

When you deploy AppFormix in a highly available cluster with a floating virtual IP (VIP) that is used to service requests from the active elected leader, you must define the *appformix_vip* variable in the **site.yml** file with a valid IP address. If you do not define the variable, you must comment out the *appformix_vip* variable. Copy the AppFormix license file to the **/var/lib/contrail_cloud/appformix/appformix.sig** directory.

To deploy the AppFormix cluster:

1. Log in to the Contrail Cloud jump host with the username `contrail` and password `c0ntrail123`.
2. Run the **appformix-deploy.sh** script.

   ```
   /var/lib/contrail_cloud/scripts/appformix-deploy.sh
   ```

Deploy Contrail Command Web UI

The `install_contrail_command.sh` script deploys the Contrail Command web UI on a virtual machine on the jump host.The Contrail Command web UI can be reached via https:// *<jumphost ip>*:9091.

- Review the /var/lib/contrail_cloud/config/vault-data.yml for Contrail Command authentication details.

Validate the OpenStack Environment

By default, tests that require floating IPs (FIPs) are skipped. The `provision-sdn-gateway.sh` script can be executed before validation to provision SDN gateways and external network that can be used by Tempest. Examples of object definitions can be found at **/var/lib/contrail_cloud/samples/features/provision-sdn-gateway/site.yml**.

The `overcloud-validation.sh` script can be used to run Tempest test collections in newly deployed environments. The script downloads a CirrOS VM image, uploads it to the overcloud, and creates new flavors. After the script execution, results of the test can be found in the undercloud home directory where two files are created:

- **tempest-subunit-smoke.xml**
- **tempest-subunit-full.xml**

The first line of the file shows the number of failures and the total count of conducted tests.

Install VNF Images and Templates

You can use Horizon or OpenStack command-line clients to install Glance images and Heat templates for the VNF services.

Add New Compute and Storage Nodes

To add new compute and storage nodes to an existing environment:

1. Update the **inventory.yml** configuration file.
2. Run the **inventory-assign.sh** script.
3. Update the **compute-nodes.yml** configuration file with the new nodes, and run the **compute-nodes-assign.sh** script.
4. Update the **storage-nodes.yml** configuration file with the new nodes, and run the **storage-nodes-assign.sh** script.
5. Rerun the **openstack-deploy.sh** script.

Gather Logs

This section provides a script for you to run, that will gather into one place, important log, configuration, and status data from your deployed nodes. This script is typically used when you need to find and provide specific information for troubleshooting or while making support calls.

We recommend you use the script after a successful deployment to provide a baseline that can be compared against future upgrades or failures. To archive the configuration, status, and logs from the deployment:

```
/var/lib/contrail_cloud/scripts/collect_data.sh -r all
```

The usage description of the `collect_data.sh` script is as follows:

```
Usage: ./collect_data.sh [-r ROLE ] [ -d ] [ -h|? ]
  -r ROLE    collect data from specific role or environment. Possible values for ROLE:
             jumphost, undercloud, control_hosts, openstack_controllers, contrail_controllers,
             contrail_analytics, contrail_analytics_db, appformix_controllers, contrail_tsn,
             compute_nodes, storage_nodes, all - collect data from all nodes.
             You can specify multiple roles, eg. -r \"undercloud jumphost appformix_controllers\"
  -d         enable debugging messages
  -e         external config
  -h         print usage information
```

RELATED DOCUMENTATION

Appendix A: Sample Configuration Files

IN THIS SECTION

Sample site.yml Configuration File

```
# Copyright 2018 Juniper Networks, Inc. All rights reserved.
# Licensed under the Juniper Networks Script Software License (the "License").
# You may not use this script file except in compliance with the License, which is located at
# http://www.juniper.net/support/legal/scriptlicense/
# Unless required by applicable law or otherwise agreed to in writing by the parties,
# software distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#
#
#

global:
  # List of DNS nameservers
```

```
  dns:
    # Google Public DNS
    - "8.8.8.8"
    - "8.8.4.4"
  # List of NTP time servers
  ntp:
    # public pool.ntp.org
    - "0.pool.ntp.org"
    - "1.pool.ntp.org"
    - "2.pool.ntp.org"
    - "3.pool.ntp.org"
  # Timezone for all servers
  timezone: 'America/Los_Angeles'
  rhel:
    # Contrail Cloud Activation Key
    # These details are provided when you request an activation key from
    # contrail cloud subscriptions <contrail_cloud_subscriptions@juniper.net>
    #
    satellite:
      #SATELLITE_KEY should be defined in vault-data.yml file
      #SATELLITE_ORG
      organization: "ContrailCloud"
      #SATELLITE_FQDN
      fqdn: contrail-cloud-satellite.juniper.net
  # DNS domain information.
  # Must be unique for every deployment to avoid name conflicts.
  # Need not be a registered DNS domain.
  domain: "my.unique.domain"

jumphost:
  network:
    # network used for provisioning (PXE booting) servers
    provision:
      # jumphost nic to be used for provisioning (PXE booting) servers
      nic: eno1

control_hosts:
  # Contains a list of label to disk mappings for roles
  disk_mapping:
    # the control host always uses the "baremetal" role
    baremetal:
           # Mapping of labels to disk devices. The label is assigned to the disk
     # device so that the disk can be referenced by the alias in other
```

```
     # configurations. for example /dev/disk/by-alias/<labels>
     # Each list element contains:
     #   label: label to assign
     # And one of following elements:
     #   name: disk device path (e.g. /dev/sdb)
     #   hctl: alternative notation for disk paths specifying SCSI address (Host, Channel,
Target and Lun) The HCTL can be found with thh lsscsi (or lspci) command or it can be found in
introspection data
     #   wwn:  List of disks WWNs across hosts, which should have the same label e.g.
host1_disk1_wwn, host2_disk1_wwn (WWN can be found in introspection data)
     # Example of wwn usage can be found in samples/features/label-disk/site.yml
      #
      - label: spinning-0
        name: /dev/sdb
      - label: spinning-1
        name: /dev/sdc
      - label: spinning-2
        name: /dev/sdd
      - label: spinning-3
        name: /dev/sde
      - label: ssd-0
        hctl: "0:2:3:0"
  storage:
    # Define a set of disk groups that can be referenced for VM virtual disk allocations
    # These become virsh storage pools on the control host
    # Each pool has:
    #   type: Either "dir" or "logical".
    #       "dir" resides on /var/lib/libvirt/images.
    #       "logical" is a LVM volume placed on the list of "disk".
    #   disk: List of disk devices to use for the pool
    # There is a built-in storage type called "default_dir_pool" which resides on /var/lib/
libvirt/images.
    hdd_storage:
      type: logical
      disk:
        - "/dev/disk/by-alias/spinning-0"
        - "/dev/disk/by-alias/spinning-1"
        - "/dev/disk/by-alias/spinning-2"
        - "/dev/disk/by-alias/spinning-3"
    ssd_storage:
      type: logical
      disk:
        - "/dev/disk/by-alias/ssd-0"
```

```
vm:
  # VM for Openstack Controller role
  control:
    disk:
      # Root disk
      vda:
        # Virsh storage pool (see storage above)
        pool: hdd_storage
  # VMs for ContrailController role
  contrail-controller:
    disk:
      # Root disk
      vda:
        # Virsh storage pool (see storage above)
        pool: hdd_storage
  # VM for ContrailTsn role
  contrail-tsn:
    disk:
      # Root disk
      vda:
        # Virsh storage pool (see storage above)
        pool: hdd_storage
  # VM for ContrailAnalytics role
  contrail-analytics:
    disk:
      # Root disk
      vda:
        # Virsh storage pool (see storage above)
        pool: hdd_storage
  # VM for ContrailAnalyticsDatabase role
  contrail-analytics-database:
    disk:
      # Root disk
      vda:
        # Virsh storage pool (see storage above)
        pool: hdd_storage
      # Analytics database journal (ssd when possible)
      vdb:
        # Virsh storage pool (see storage above)
        pool: ssd_storage
      # Analytics data (large capacity)
      vdc:
        # Virsh storage pool (see storage above)
```

```
          pool: hdd_storage
    # VM for AppFormix controller role
    appformix-controller:
      disk:
        # Root disk
        vda:
          # Virsh storage pool (see storage above)
          pool: hdd_storage

compute_hosts:
  sriov:
    #enable sriov support
    enabled: true
    #enable sriov with dpdk
    # Contrail vrouter mode:
    #   supported values are: dpdk or anything else means kernel vRouter
    mode: dpdk
    #Sriov NumVFs separated by comma
    num_vf:
      - "ens2f1:7"
    #NovaPCIPassthrough settings
    pci_passthrough:
      - devname: "ens2f1"
        physical_network: "sriov1"
  root_disk:
  # Define root disk for the listed ironic profiles.
  # The default of "/dev/sda" will be used if there is no
  # specific profile definition
    ComputeKernel0Hw0:
      name: "/dev/sda"
    ComputeKernel0Hw1:
      name: "/dev/sda"
    ComputeKernel1Hw1:
      name: "/dev/sda"
    ComputeKernel1Hw0:
      name: "/dev/sda"
    ComputeDpdk0Hw2:
      name: "/dev/sda"
    ComputeDpdk1Hw3:
      name: "/dev/sda"
    ComputeSriov0Hw4:
      name: "/dev/sda"
    ComputeSriov1Hw5:
```

```
      name: "/dev/sda"
  resource:
    minimal_disk:
    # This value will be used as the local_gb size for the listed ironic profiles
    # If not defined for a profile then the default will be used
      ComputeKernel0Hw0: 50
      ComputeKernel0Hw1: 50
      ComputeKernel1Hw1: 50
      ComputeKernel1Hw0: 50
      ComputeDpdk0Hw2: 50
      ComputeDpdk1Hw3: 50
      ComputeSriov0Hw4: 50
      ComputeSriov1Hw5: 50

storage_hosts:
  root_disk:
  # Define root disk for the listed ironic profiles.
  # The default of "/dev/sda" will be used if there is no
  # specific profile definition
    CephStorage0Hw6:
      name: "/dev/sda"
    CephStorage1Hw7:
      name: "/dev/sda"

undercloud:
  nova:
    # Nova flavor definitions for roles
    flavor:
      CephStorage0Hw6:
        cpu: 1
        memory: 4
        disk: 40
        ephemeral: 0
      CephStorage1Hw7:
        cpu: 1
        memory: 4
        disk: 40
        ephemeral: 0
      ComputeKernel0Hw0:
        cpu: 8
        memory: 24
        disk: 40
        ephemeral: 0
```

```
      ComputeKernel0Hw1:
        cpu: 8
        memory: 24
        disk: 40
        ephemeral: 0
      ComputeKernel1Hw1:
        cpu: 8
        memory: 24
        disk: 40
        ephemeral: 0
      ComputeKernel1Hw0:
        cpu: 8
        memory: 24
        disk: 40
        ephemeral: 0
      ComputeDpdk0Hw2:
        cpu: 8
        memory: 24
        disk: 40
        ephemeral: 0
      ComputeDpdk1Hw3:
        cpu: 8
        memory: 24
        disk: 40
        ephemeral: 0
      ComputeSriov0Hw4:
        cpu: 8
        memory: 24
        disk: 40
        ephemeral: 0
      ComputeSriov1Hw5:
        cpu: 8
        memory: 24
        disk: 40
        ephemeral: 0

overcloud:
  # Contains a list of label to disk mappings for roles.
  # When Ceph Storage is disabled, compute-related roles (Compute* and
  # ComputeDpdk* roles) will use any disks labeled with
  # "ephemeral-<digits>" for local Nova ephemeral storage.
  disk_mapping:
    ComputeKernel:
```

```
            # Mapping of labels to disk devices. The label is assigned to the disk
     # device so that the disk can be referenced by the alias in other
     # configurations. for example /dev/disk/by-alias/<label>
     # Each list element contains:
     #   label: label to assign
     # And one of following elements:
     #   name: disk device path (e.g. /dev/sdb)
     #   hctl: alternative notation for disk paths specifying SCSI address (Host, Channel,
Target and Lun) The HCTL can be found with thh lsscsi (or lspci) command or it can be found in
introspection data
     #   wwn:  List of disks WWNs across hosts, which should have the same label e.g.
host1_disk1_wwn, host2_disk1_wwn (WWN can be found in introspection data)
     # Example of wwn usage can be found in samples/features/label-disk/site.yml
      - label: ephemeral-0
        hctl: '5:0:0:0'
      - label: ephemeral-1
        hctl: '6:0:0:0'
      - label: ephemeral-2
        hctl: '7:0:0:0'
      - label: ephemeral-3
        hctl: '8:0:0:0'
    ComputeKernel0Hw0:
            # Mapping of labels to disk devices. The label is assigned to the disk
     # device so that the disk can be referenced by the alias in other
     # configurations. for example /dev/disk/by-alias/<label>
     # Each list element contains:
     #   label: label to assign
     # And one of following elements:
     #   name: disk device path (e.g. /dev/sdb)
     #   hctl: alternative notation for disk paths specifying SCSI address (Host, Channel,
Target and Lun) The HCTL can be found with thh lsscsi (or lspci) command or it can be found in
introspection data
     #   wwn:  List of disks WWNs across hosts, which should have the same label e.g.
host1_disk1_wwn, host2_disk1_wwn (WWN can be found in introspection data)
     # Example of wwn usage can be found in samples/features/label-disk/site.yml
      - label: ephemeral-0
        hctl: '5:0:0:0'
      - label: ephemeral-1
        hctl: '6:0:0:0'
      - label: ephemeral-2
        hctl: '7:0:0:0'
      - label: ephemeral-3
        hctl: '8:0:0:0'
```

```
    ComputeKernel1Hw0:
           # Mapping of labels to disk devices. The label is assigned to the disk
     # device so that the disk can be referenced by the alias in other
     # configurations. for example /dev/disk/by-alias/<label>
     # Each list element contains:
     #   label: label to assign
     # And one of following elements:
     #   name: disk device path (e.g. /dev/sdb)
     #   hctl: alternative notation for disk paths specifying SCSI address (Host, Channel,
Target and Lun) The HCTL can be found with thh lsscsi (or lspci) command or it can be found in
introspection data
     #   wwn:  List of disks WWNs across hosts, which should have the same label e.g.
host1_disk1_wwn, host2_disk1_wwn (WWN can be found in introspection data)
     # Example of wwn usage can be found in samples/features/label-disk/site.yml
      - label: ephemeral-0
        hctl: '5:0:0:0'
      - label: ephemeral-1
        hctl: '6:0:0:0'
      - label: ephemeral-2
        hctl: '7:0:0:0'
      - label: ephemeral-3
        hctl: '8:0:0:0'
    ComputeKernel1Hw1:
           # Mapping of labels to disk devices. The label is assigned to the disk
     # device so that the disk can be referenced by the alias in other
     # configurations. for example /dev/disk/by-alias/<label>
     # Each list element contains:
     #   label: label to assign
     # And one of following elements:
     #   name: disk device path (e.g. /dev/sdb)
     #   hctl: alternative notation for disk paths specifying SCSI address (Host, Channel,
Target and Lun) The HCTL can be found with thh lsscsi (or lspci) command or it can be found in
introspection data
     #   wwn:  List of disks WWNs across hosts, which should have the same label e.g.
host1_disk1_wwn, host2_disk1_wwn (WWN can be found in introspection data)
     # Example of wwn usage can be found in samples/features/label-disk/site.yml
      - label: ephemeral-0
        hctl: '5:0:0:0'
      - label: ephemeral-1
        hctl: '6:0:0:0'
      - label: ephemeral-2
        hctl: '7:0:0:0'
      - label: ephemeral-3
```

```
        hctl: '8:0:0:0'
    ComputeKernel0Hw1:
           # Mapping of labels to disk devices. The label is assigned to the disk
     # device so that the disk can be referenced by the alias in other
     # configurations. for example /dev/disk/by-alias/<label>
     # Each list element contains:
     #   label: label to assign
     # And one of following elements:
     #   name: disk device path (e.g. /dev/sdb)
     #   hctl: alternative notation for disk paths specifying SCSI address (Host, Channel,
Target and Lun) The HCTL can be found with thh lsscsi (or lspci) command or it can be found in
introspection data
     #   wwn:  List of disks WWNs across hosts, which should have the same label e.g.
host1_disk1_wwn, host2_disk1_wwn (WWN can be found in introspection data)
     # Example of wwn usage can be found in samples/features/label-disk/site.yml
      - label: ephemeral-0
        hctl: '5:0:0:0'
      - label: ephemeral-1
        hctl: '6:0:0:0'
      - label: ephemeral-2
        hctl: '7:0:0:0'
      - label: ephemeral-3
        hctl: '8:0:0:0'
    ComputeDpdk:
           # Mapping of labels to disk devices. The label is assigned to the disk
     # device so that the disk can be referenced by the alias in other
     # configurations. for example /dev/disk/by-alias/<label>
     # Each list element contains:
     #   label: label to assign
     # And one of following elements:
     #   name: disk device path (e.g. /dev/sdb)
     #   hctl: alternative notation for disk paths specifying SCSI address (Host, Channel,
Target and Lun) The HCTL can be found with thh lsscsi (or lspci) command or it can be found in
introspection data
     #   wwn:  List of disks WWNs across hosts, which should have the same label e.g.
host1_disk1_wwn, host2_disk1_wwn (WWN can be found in introspection data)
     # Example of wwn usage can be found in samples/features/label-disk/site.yml
      - label: ephemeral-0
        hctl: '5:0:0:0'
      - label: ephemeral-1
        hctl: '6:0:0:0'
      - label: ephemeral-2
        hctl: '7:0:0:0'
```

```
      - label: ephemeral-3
        hctl: '8:0:0:0'
    ComputeDpdk0Hw2:
           # Mapping of labels to disk devices. The label is assigned to the disk
     # device so that the disk can be referenced by the alias in other
     # configurations. for example /dev/disk/by-alias/<label>
     # Each list element contains:
     #   label: label to assign
     # And one of following elements:
     #   name: disk device path (e.g. /dev/sdb)
     #   hctl: alternative notation for disk paths specifying SCSI address (Host, Channel,
Target and Lun) The HCTL can be found with thh lsscsi (or lspci) command or it can be found in
introspection data
     #   wwn:  List of disks WWNs across hosts, which should have the same label e.g.
host1_disk1_wwn, host2_disk1_wwn (WWN can be found in introspection data)
     # Example of wwn usage can be found in samples/features/label-disk/site.yml
        hctl: '5:0:0:0'
      - label: ephemeral-1
        hctl: '6:0:0:0'
      - label: ephemeral-2
        hctl: '7:0:0:0'
      - label: ephemeral-3
        hctl: '8:0:0:0'
    ComputeDpdk1Hw3:
           # Mapping of labels to disk devices. The label is assigned to the disk
     # device so that the disk can be referenced by the alias in other
     # configurations. for example /dev/disk/by-alias/<label>
     # Each list element contains:
     #   label: label to assign
     # And one of following elements:
     #   name: disk device path (e.g. /dev/sdb)
     #   hctl: alternative notation for disk paths specifying SCSI address (Host, Channel,
Target and Lun) The HCTL can be found with thh lsscsi (or lspci) command or it can be found in
introspection data
     #   wwn:  List of disks WWNs across hosts, which should have the same label e.g.
host1_disk1_wwn, host2_disk1_wwn (WWN can be found in introspection data)
     # Example of wwn usage can be found in samples/features/label-disk/site.yml
      - label: ephemeral-0
        hctl: '5:0:0:0'
      - label: ephemeral-1
        hctl: '6:0:0:0'
      - label: ephemeral-2
        hctl: '7:0:0:0'
```

```
      - label: ephemeral-3
        hctl: '8:0:0:0'
    ComputeSriov:
           # Mapping of labels to disk devices. The label is assigned to the disk
     # device so that the disk can be referenced by the alias in other
     # configurations. for example /dev/disk/by-alias/<label>
     # Each list element contains:
     #   label: label to assign
     # And one of following elements:
     #   name: disk device path (e.g. /dev/sdb)
     #   hctl: alternative notation for disk paths specifying SCSI address (Host, Channel,
Target and Lun) The HCTL can be found with thh lsscsi (or lspci) command or it can be found in
introspection data
     #   wwn:  List of disks WWNs across hosts, which should have the same label e.g.
host1_disk1_wwn, host2_disk1_wwn (WWN can be found in introspection data)
     # Example of wwn usage can be found in samples/features/label-disk/site.yml
      - label: ephemeral-0
        hctl: '5:0:0:0'
      - label: ephemeral-1
        hctl: '6:0:0:0'
      - label: ephemeral-2
        hctl: '7:0:0:0'
      - label: ephemeral-3
        hctl: '8:0:0:0'
    ComputeSriov0Hw4:
           # Mapping of labels to disk devices. The label is assigned to the disk
     # device so that the disk can be referenced by the alias in other
     # configurations. for example /dev/disk/by-alias/<label>
     # Each list element contains:
     #   label: label to assign
     # And one of following elements:
     #   name: disk device path (e.g. /dev/sdb)
     #   hctl: alternative notation for disk paths specifying SCSI address (Host, Channel,
Target and Lun) The HCTL can be found with thh lsscsi (or lspci) command or it can be found in
introspection data
     #   wwn:  List of disks WWNs across hosts, which should have the same label e.g.
host1_disk1_wwn, host2_disk1_wwn (WWN can be found in introspection data)
     # Example of wwn usage can be found in samples/features/label-disk/site.yml
        hctl: '5:0:0:0'
      - label: ephemeral-1
        hctl: '6:0:0:0'
      - label: ephemeral-2
        hctl: '7:0:0:0'
```

```
      - label: ephemeral-3
        hctl: '8:0:0:0'
    ComputeSriov1Hw5:
           # Mapping of labels to disk devices. The label is assigned to the disk
     # device so that the disk can be referenced by the alias in other
     # configurations. for example /dev/disk/by-alias/<label>
     # Each list element contains:
     #   label: label to assign
     # And one of following elements:
     #   name: disk device path (e.g. /dev/sdb)
     #   hctl: alternative notation for disk paths specifying SCSI address (Host, Channel,
Target and Lun) The HCTL can be found with thh lsscsi (or lspci) command or it can be found in
introspection data
     #   wwn:  List of disks WWNs across hosts, which should have the same label e.g.
host1_disk1_wwn, host2_disk1_wwn (WWN can be found in introspection data)
     # Example of wwn usage can be found in samples/features/label-disk/site.yml
      - label: ephemeral-0
        hctl: '5:0:0:0'
      - label: ephemeral-1
        hctl: '6:0:0:0'
      - label: ephemeral-2
        hctl: '7:0:0:0'
      - label: ephemeral-3
        hctl: '8:0:0:0'
  network:
    # The external network is used for referencing the overcloud APIs from outside the
infrastructure.
    external:
      # Network name used by TripleO Heat Templates
      heat_name: External
      # CIDR (IP/prefix) for the external network subnet
      # Corresponds to the ExternalIpSubnet heat property
      cidr: "10.10.10.64/26"
      # Default route for the external network
      # Corresponds to the ExternalInterfaceDefaultRoute heat property
      gateway: "10.10.10.126"
      # VLAN tag for the external network
      # Corresponds to the ExternalNetworkVlanID heat property
      vlan: 18
      # Floating virtual IP for the Openstack APIs on the external network
      # Corresponds to the PublicVirtualFixedIPs heat property
      vip: "10.10.10.100"
      # DHCP pool for the external network
```

```
      # Be sure that the range is large enough to accommodate all nodes in the external network
      pool:
        # Range start for the DHCP pool
        start: "10.10.10.70"
        # Range end for the DHCP pool
        end: "10.10.10.99"
      # MTU for external network
      # Corresponds to the ExternalNetworkMtu heat property
      mtu: 1500
      # List of roles that can be on this network
      role:
        - Controller
        - AppformixController
    # The internal API network is used for control plane signalling and service API calls
    internal_api:
      # Network name used by TripleO Heat Templates
      heat_name: InternalApi
      # VLAN tag for the internal API network
      # Corresponds to the InternalApiNetworkVlanID heat property
      vlan: 226
      # CIDR (IP/prefix) for the internal api supernet network subnet
      # Corresponds to the InternalApiSupernet heat property
      # Supernet is used in spine/leaf configuration
      # Supernet accommodate all related leaf networks, e.g. internal_api0 and internal_api1
      # Supernet is used to create static routes between leafs
      # Supernet is defined only for main network, not per leafs
      supernet: "172.16.0.0/16"
      # CIDR (IP/prefix) for the internal api network subnet
      # Corresponds to the InternalApiIpSubnet heat property
      cidr: "172.16.0.0/24"
      # Default route for the internal api network
      # Corresponds to the InternalApiInterfaceDefaultRoute heat property
      gateway: 172.16.0.1
      # MTU for internal api network
      # Corresponds to the InternalApiNetworkMtu heat property
      mtu: 1500
      # DHCP pool for the internal api network
      # Be sure that the range is large enough to accommodate all nodes in the internal api
network
      pool:
        # Range start for the DHCP pool
        start: 172.16.0.100
        # Range end for the DHCP pool
```

```
        end: 172.16.0.160
      # Floating virtual IP for the Openstack APIs on the internal api network
      # Corresponds to the InternalApiVirtualFixedIPs heat property
      vip: 172.16.0.90
      # List of roles that can be on this network
      role:
        - Controller
        - ContrailController
        - ContrailAnalytics
        - ContrailAnalyticsDatabase
        - ContrailTsn
        - AppformixController
    # Leaf 0 subnet of the internal_api network
    internal_api0:
      # Network name used by TripleO Heat Templates
      heat_name: InternalApi0
      # VLAN tag for the internal API 0 network
      # Corresponds to the InternalApi0NetworkVlanID heat property
      vlan: 229
      # CIDR (IP/prefix) for the internal api 0 network subnet
      # Corresponds to the InternalApi0IpSubnet heat property
      cidr: "172.16.1.0/24"
      # Default route for the internal api 0 network
      # Corresponds to the InternalApi0InterfaceDefaultRoute heat property
      gateway: 172.16.1.1
      # MTU for internal api 0 network
      # Corresponds to the InternalApi0NetworkMtu heat property
      mtu: 1500
      # DHCP pool for the internal api 0 network
      # Be sure that the range is large enough to accommodate all nodes in the internal api
network
      pool:
        # Range start for the DHCP pool
        start: 172.16.1.100
        # Range end for the DHCP pool
        end: 172.16.1.200
      # List of roles that can be on this network
      role:
        - ComputeDpdk0Hw2
        - ComputeSriov0Hw4
        - ComputeKernel0Hw0
        - ComputeKernel0Hw1
    # Leaf 1 subnet of the internal_api network
```

```
    internal_api1:
      # Network name used by TripleO Heat Templates
      heat_name: InternalApi1
      # VLAN tag for the internal API 1 network
      # Corresponds to the InternalApi1NetworkVlanID heat property
      vlan: 449
      # CIDR (IP/prefix) for the internal api 1 network subnet
      # Corresponds to the InternalApi1IpSubnet heat property
      cidr: "172.16.2.0/24"
      # Default route for the internal api 1 network
      # Corresponds to the InternalApi1InterfaceDefaultRoute heat property
      gateway: 172.16.2.1
      # MTU for internal api 1 network
      # Corresponds to the InternalApi1NetworkMtu heat property
      mtu: 1500
      # DHCP pool for the internal api 1 network
      # Be sure that the range is large enough to accommodate all nodes in the internal api
network
      pool:
        # Range start for the DHCP pool
        start: 172.16.2.100
        # Range end for the DHCP pool
        end: 172.16.2.200
      # List of roles that can be on this network
      role:
        - ComputeDpdk1Hw3
        - ComputeSriov1Hw5
        - ComputeKernel1Hw1
        - ComputeKernel1Hw0
    # The management network is defined for backwards-compatibility in RHOSP and is not
    # used by default by any roles.
    management:
      # Network name used by TripleO Heat Templates
      heat_name: Management
      # VLAN tag for the management network
      # Corresponds to the ManagementNetworkVlanID heat property
      vlan: 225
      # CIDR (IP/prefix) for the network subnet
      # Corresponds to the ManagementIpSubnet heat property
      cidr: "192.168.1.0/24"
      # MTU for the network
      # Corresponds to the ManagementNetworkMtu heat property
      mtu: 1500
```

```
    # DHCP pool for the  network
    # Be sure that the range is large enough to accommodate all nodes in the network
    pool:
      # Range start for the DHCP pool
      start: 192.168.1.100
      # Range end for the DHCP pool
      end: 192.168.1.200
  # The storage network is used for Compute storage access
  storage:
    # Network name used by TripleO Heat Templates
    heat_name: Storage
    # VLAN tag for the storage network
    # Corresponds to the StorageNetworkVlanID heat property
    vlan: 227
    supernet: "172.19.0.0/16"
    cidr: "172.19.0.0/24"
    gateway: 172.19.0.1
    mtu: 1500
    pool:
      start: 172.19.0.100
      end: 172.19.0.200
    # List of roles that can be on this network
    role:
      - Controller
      - ContrailTsn
  # Leaf 0 subnet of the storage network
  storage0:
    # Network name used by TripleO Heat Templates
    heat_name: Storage0
    vlan: 223
    cidr: "172.19.1.0/24"
    gateway: 172.19.1.1
    mtu: 1500
    pool:
      start: 172.19.1.100
      end: 172.19.1.200
    # List of roles that can be on this network
    role:
      - ComputeDpdk0Hw2
      - ComputeSriov0Hw4
      - CephStorage0Hw6
      - ComputeKernel0Hw0
      - ComputeKernel0Hw1
```

```
    # Leaf 1 subnet of the storage network
    storage1:
      # Network name used by TripleO Heat Templates
      heat_name: Storage1
      vlan: 443
      cidr: "172.19.2.0/24"
      gateway: 172.19.2.1
      mtu: 1500
      pool:
        start: 172.19.2.100
        end: 172.19.2.200
      # List of roles that can be on this network
      role:
        - ComputeDpdk1Hw3
        - ComputeSriov1Hw5
        - CephStorage1Hw7
        - ComputeKernel1Hw1
        - ComputeKernel1Hw0
    # The storage management network is used for storage operations such as replication
    storage_mgmt:
      # Network name used by TripleO Heat Templates
      heat_name: StorageMgmt
      # VLAN tag for the storage management network
      # Corresponds to the StorageMgmtNetworkVlanID heat property
      vlan: 224
      supernet: "172.20.0.0/16"
      cidr: "172.20.0.0/24"
      gateway: 172.20.0.1
      mtu: 1500
      pool:
        start: 172.20.0.100
        end: 172.20.0.200
      # List of roles that can be on this network
      role:
        - Controller
    # Leaf 0 subnet of the storage_mgmt network
    storage_mgmt0:
      # Network name used by TripleO Heat Templates
      heat_name: StorageMgmt0
      vlan: 221
      cidr: "172.20.1.0/24"
      gateway: 172.20.1.1
      mtu: 1500
```

```
    pool:
      start: 172.20.1.100
      end: 172.20.1.200
    # List of roles that can be on this network
    role:
      - CephStorage0Hw6
  # Leaf 1 subnet of the storage_mgmt network
  storage_mgmt1:
    # Network name used by TripleO Heat Templates
    heat_name: StorageMgmt1
    vlan: 444
    cidr: "172.20.2.0/24"
    gateway: 172.20.2.1
    mtu: 1500
    pool:
      start: 172.20.2.100
      end: 172.20.2.200
    # List of roles that can be on this network
    role:
      - CephStorage1Hw7
  # The tenant network is used for tenant workload data
  tenant:
    # Network name used by TripleO Heat Templates
    heat_name: Tenant
    # VLAN tag for the tenant network
    # Corresponds to the TenantNetworkVlanID heat property
    vlan: 228
    supernet: "172.18.0.0/16"
    cidr: "172.18.0.0/24"
    gateway: 172.18.0.1
    vrouter_gateway: 172.18.0.1
    mtu: 1500
    pool:
      start: 172.18.0.100
      end: 172.18.0.200
    # List of roles that can be on this network
    role:
      - ContrailController
      - ContrailAnalytics
      - ContrailAnalyticsDatabase
      - ContrailTsn
  # Leaf 0 subnet of the tenant network
  tenant0:
```

```
      # Network name used by TripleO Heat Templates
      heat_name: Tenant0
      vlan: 222
      cidr: "172.18.1.0/24"
      gateway: 172.18.1.1
      vrouter_gateway: 172.18.1.1
      mtu: 1500
      pool:
        start: 172.18.1.100
        end: 172.18.1.200
      # List of roles that can be on this network
      role:
        - ComputeDpdk0Hw2
        - ComputeSriov0Hw4
        - ComputeKernel0Hw0
        - ComputeKernel0Hw1
    # Leaf 1 subnet of the tenant network
    tenant1:
      # Network name used by TripleO Heat Templates
      heat_name: Tenant1
      vlan: 442
      cidr: "172.18.2.0/24"
      gateway: 172.18.2.1
      vrouter_gateway: 172.18.2.1
      mtu: 1500
      pool:
        start: 172.18.2.100
        end: 172.18.2.200
      # List of roles that can be on this network
      role:
        - ComputeDpdk1Hw3
        - ComputeSriov1Hw5
        - ComputeKernel1Hw1
        - ComputeKernel1Hw0

  # Contrail sepcific settings
  #contrail:
  #  aaa_mode: cloud-admin
  #  vrouter:
  #    contrail_settings:
  #      # Settings per profile.
  #      # Profile's contrail_settings replace default settings and should include
  #      # all keys and values which are intended to be exported on given role.
```

```
  #      # When leafs are used it implies per profile configuration as it defines
  #      # VROUTER_GATEWAY for profile by quering node's tenant network for
  #      # vrouter_gateway value.
  #      default:
  #        VROUTER_GATEWAY: 172.16.81.254
  #        BGP_ASN: 64512
  #        LACP_RATE: 1
  #      ComputeKernel1Hw0:
  #        LACP_RATE: 1

  # Information used to generate the SSL certificates for the public Openstack service APIs
  tls:
    #countryName_default
    country: "US"
    #stateOrProvinceName_default
    state: "CA"
    #localityName_default
    city: "Sunnyvale"
    #organizationalUnitName_default
    organization: "JNPR"
    #commonName_default - this is typically the external VIP
    common_name: "10.10.10.90"

ceph:
  # Choice to enable Ceph storage in the overcloud.
  # "true" means that Ceph will be deployed as the backed for Cinder and Glance services.
  # "false" false means that Ceph will not be deployed.
  enabled: true
  # Ceph OSD disk configuration
  osd:
    # Update the Ceph crush map when OSDs are started
    crush_update_on_start: true
    # Ceph OSD disk assignments. The named disks will be exclusively used by Ceph for
persistence.
    # Lvm is a default scenario for ceph deployment with bluestore as a backend.
    # When all named disks are the same type, spinning or solid state, all of them will be used
    # as ceph osds. When disks with mixed types are defined spinning disks will be used as osds
    # and on solid state disks ceph db will be created. For mixed types of disks the automatic
pgp
    # number calculation requires assigning key 'contents' with value 'db' to ssd disks.
    # In below example disks sd[b-e] are spinning disks and sdf is solid state disk.
    default:
      disk:
```

```
        '/dev/sdb':
        '/dev/sdc':
        '/dev/sdd':
        '/dev/sde':
        '/dev/sdf':
          contents: db
    CephStorage0Hw6:
      disk:
        '/dev/sdb':
        '/dev/sdc':
        '/dev/sdd':
        '/dev/sde':
        '/dev/sdf':
          contents: db
    CephStorage1Hw7:
      disk:
        '/dev/sdb':
        '/dev/sdc':
        '/dev/sdd':
        '/dev/sde':
        '/dev/sdf':
          contents: db
  # By default, pgp number is calculated by contrail cloud. If you want, you can give this
parameter
  # by yourself. Use the calculator on the website: https://ceph.com/pgcalc/. Calculator takes
into
  # account also pool utilization. Calculated pgp_num can be introduced in configuration as
below.
  # It's defined per used pool.
  #  pool:
  #    vms:
  #      pgp_num: 32
  #    rbd:
  #      pgp_num: 32
  #    images:
  #      pgp_num: 32
  #    volumes:
  #      pgp_num: 32
  #    backups:
  #      pgp_num: 32
  #
  # Rados Gateway when enabled, which is a default behaviour, creates it's own ceph pools
  # not tracked by contrail cloud. Those pools can be predefined to better control
```

```
  # their sizes. Below pools definitions are not an exhaustive, please consult with
  # https://ceph.com/pgcalc/
  # Pools should have enabled application according to their use.
  # If not changed explicit, pools are created with 'rbd' application assigned.
  # Available options are:
  #   - rbd for the Ceph Block Device
  #   - rgw for the Ceph Object Gateway
  #   - cephfs for the Ceph Filesystem
  # or user defined value for custom application.
  # More details can be found on
  # https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/3/html/
storage_strategies_guide/pools-1#enable-application
  #    .rgw.root:
  #      pgp_num: 16
  #      enabled: true
  #      replica: 3
  #      application: rgw
  #    default.rgw.control:
  #      pgp_num: 16
  #      enabled: true
  #      replica: 3
  #      application: rgw
  #    default.rgw.meta:
  #      pgp_num: 16
  #      enabled: true
  #      replica: 3
  #      application: rgw
  #    default.rgw.log:
  #      pgp_num: 16
  #      enabled: true
  #      replica: 3
  #      application: rgw
  #    default.rgw.buckets.index:
  #      pgp_num: 16
  #      enabled: true
  #      replica: 3
  #      application: rgw
  #    default.rgw.buckets.data:
  #      pgp_num: 16
  #      enabled: true
  #      replica: 3
  #      application: rgw
  #    default.rgw.buckets.non-ec:
```

```
  #      pgp_num: 16
  #      enabled: true
  #      replica: 3
  #      application: rgw

appformix:
  # Set to true if you have multiple control hosts which allows Apformix to run in HA mode
  enable_ha: true
  # Floating virtual IP for the Appformix APIs on the external network, used and required by HA
mode.
  vip: "10.10.10.101"
  keepalived:
    # Set which interface will be used for vrrp
    vrrp_interface: "eth1"
```

Sample inventory.yml Configuration File

```
# Copyright 2018 Juniper Networks, Inc. All rights reserved.
# Licensed under the Juniper Networks Script Software License (the "License").
# You may not use this script file except in compliance with the License, which is located at
# http://www.juniper.net/support/legal/scriptlicense/
# Unless required by applicable law or otherwise agreed to in writing by the parties,
# software distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#
#
#

# Common values shared among group of nodes
ipmi_hardware1: &hardware1
  pm_type: "ipmi"
  pm_user: "{{ vault['inventory_nodes']['hardware1']['pm_user'] }}"
  pm_password: "{{ vault['inventory_nodes']['hardware1']['pm_password'] }}"
  capabilities: "boot_mode:uefi"

# List of baremetal server nodes that can be used for the deploying roles
# Each list item contains:
#    name: logical name to assign this resource (string)
#    pm_addr: IP address for resourceIPMI interface (string)
```

```
#    pm_type: Ironic driver to interface with this resource (typically ipmi) (string)
#    pm_user: IPMI user account (string)
#    pm_password: IPMI account user password (string)
#    capabilities: String of comma separated list of node capabilities.
#                  Capabilities 'profile' and 'boot_option' are managed
#                  by Contrail Cloud and will be omitted. (string)
#                  e.g capabilities: "boot_mode:uefi" set boot mode to uefi
#
# Some values common for nodes can be moved to dedicated section like ipmi_hardware1
# and be referred like this:
#    <<: *hardware1
inventory_nodes:
  - name: "control-host1"
    pm_addr: "10.10.11.58"
    <<: *hardware1
  - name: "control-host2"
    pm_addr: "10.10.11.59"
    <<: *hardware1
  - name: "control-host3"
    pm_addr: "10.10.11.60"
    <<: *hardware1
  - name: "storage1"
    pm_addr: "10.10.11.61"
    <<: *hardware1
  - name: "storage2"
    pm_addr: "10.10.11.62"
    <<: *hardware1
  - name: "storage3"
    pm_addr: "10.10.11.63"
    <<: *hardware1
  - name: "computedpdk1"
    pm_addr: "10.10.11.64"
    <<: *hardware1
  - name: "computedpdk2"
    pm_addr: "10.10.11.65"
    <<: *hardware1
  - name: "compute1"
    pm_addr: "10.10.11.66"
    <<: *hardware1
  - name: "compute2"
    pm_addr: "10.10.11.67"
    <<: *hardware1
  - name: "compute3"
```

```
    pm_addr: "10.10.11.68"
    <<: *hardware1
  - name: "compute4"
    pm_addr: "10.10.11.69"
    <<: *hardware1
  - name: "computesriov1"
    pm_addr: "10.10.11.70"
    <<: *hardware1
  - name: "computesriov2"
    pm_addr: "10.10.11.71"
    <<: *hardware1
```

Sample control-host-nodes.yml Configuration File

```
# Copyright 2018 Juniper Networks, Inc. All rights reserved.
# Licensed under the Juniper Networks Script Software License (the "License").
# You may not use this script file except in compliance with the License, which is located at
# http://www.juniper.net/support/legal/scriptlicense/
# Unless required by applicable law or otherwise agreed to in writing by the parties,
# software distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#
#
#

# List of nodes to use as control host role
# Each list item contains a set of variables which can be referenced
# with "{{ host.<variable> }}" in control_host_nodes_network_config below.
# Other ad-hoc variables can be added as needed.
#    name: name of a node in the inventory (string)
#    hostname: hostname to assign the node after it is imaged (string)
#    control_ip_netmask: static CIDR address on Control Plane network.
#                        Choose a value outside the DHCP range. (string)
#    dns_server1,dns_server2: dns server addresses (string)
#    max_mtu: The largest MTU supported by an interface
#

control_host_nodes:
  - name: "control-host1"
```

```
    control_ip_netmask: "192.168.213.5/24"
    dns_server1: "172.29.143.60"
    dns_server2: "172.29.139.60"
    max_mtu: 9216
  - name: "control-host2"
    control_ip_netmask: "192.168.213.6/24"
    dns_server1: "172.29.143.60"
    dns_server2: "172.29.139.60"
    max_mtu: 9216
  - name: "control-host3"
    control_ip_netmask: "192.168.213.7/24"
    dns_server1: "172.29.143.60"
    dns_server2: "172.29.139.60"
    max_mtu: 9216

# Template for network layout on all control host nodes
# This follows the os-net-config syntax
# See https://github.com/openstack/os-net-config/tree/stable/queens
# variables from control_host_nodes can be refered with "{{ host.variable> }}"
control_host_nodes_network_config:
  - type: ovs_bridge
    name: br-eno1
    use_dhcp: false
    mtu: "{{ overcloud['network']['control']['mtu'] }}"
    addresses:
    -
      ip_netmask: "{{ host.control_ip_netmask }}"
    routes:
      -
        next_hop: "{{ overcloud['network']['control']['gateway'] }}"
        default: true
    dns_servers:
      - "{{ host.dns_server1 }}"
      - "{{ host.dns_server2 }}"
    members:
      - type: interface
        name: eno1
        use_dhcp: false
        mtu: "{{ overcloud['network']['control']['mtu'] }}"
  - type: ovs_bridge
    name: br-eno2
    use_dhcp: false
    mtu: "{{ host.max_mtu }}"
```

```
    members:
      -
        type: interface
        name: eno2
        use_dhcp: false
        mtu: "{{ host.max_mtu }}"
  - type: ovs_bridge
    name: br-bond0
    use_dhcp: false
    mtu: "{{ host.max_mtu }}"
    members:
      -
        type: linux_bond
        name: bond0
        use_dhcp: false
        mtu: "{{ host.max_mtu }}"
        bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast miimon=100"
        members:
          -
            type: interface
            name: ens7f0
            use_dhcp: false
            mtu: "{{ host.max_mtu }}"
            primary: true
          -
            type: interface
            name: ens7f1
            use_dhcp: false
            mtu: "{{ host.max_mtu }}"

control_hosts:
  # The mapping from control host interfaces to the control VM interfaces
  # The first interface (eth0) must always be the Control Plane network to allow the VM to PXE
boot
  # VM interface names must be sequential with no gaps (e.g. eth0, eth1, eth2,...)
  vm_interfaces:
    - interface: eth0
      bridge: br-eno1
    - interface: eth1
      bridge: br-eno2
    - interface: eth2
      bridge: br-bond0
```

Sample overcloud-nics.yml Configuration File

```
Contrail_network_config:
  - type: interface
    name: eth0
    dns_servers:
      get_param: DnsServers
    mtu:
      get_param: ControlPlaneNetworkMtu
    addresses:
    - ip_netmask:
        list_join:
          - '/'
          - - get_param: ControlPlaneIp
            - get_param: ControlPlaneSubnetCidr
    use_dhcp: false
    routes:
    -
      ip_netmask: 169.254.169.254/32
      next_hop:
        get_param: EC2MetadataIp
    -
      default: True
      next_hop:
        get_param: ControlPlaneDefaultRoute
  - type: vlan
    device: eth0
    vlan_id:
      get_param: InternalApiNetworkVlanID
    mtu:
      get_param: InternalApiNetworkMtu
    addresses:
    - ip_netmask:
        get_param: InternalApiIpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
        get_param: InternalApiInterfaceDefaultRoute
  - type: interface
```

```
    name: eth1
    use_dhcp: false
  - type: interface
    name: eth2
    use_dhcp: false
    mtu:
      get_param: TenantNetworkMtu
  - type: vlan
    device: eth2
    vlan_id:
      get_param: TenantNetworkVlanID
    mtu:
      get_param: TenantNetworkMtu
    addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
    routes:
    -
      ip_netmask:
        get_param: TenantSupernet
      next_hop:
        get_param: TenantInterfaceDefaultRoute

Controller_network_config:
  - type: interface
    name: eth0
    dns_servers:
      get_param: DnsServers
    use_dhcp: false
    mtu:
      get_param: ControlPlaneNetworkMtu
    addresses:
    - ip_netmask:
        list_join:
          - '/'
          - - get_param: ControlPlaneIp
            - get_param: ControlPlaneSubnetCidr
    routes:
    -
      ip_netmask: 169.254.169.254/32
      next_hop:
        get_param: EC2MetadataIp
  - type: vlan
```

```
    device: eth0
    vlan_id:
      get_param: StorageNetworkVlanID
    mtu:
      get_param: StorageNetworkMtu
    addresses:
    - ip_netmask:
        get_param: StorageIpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageSupernet
      next_hop:
        get_param: StorageInterfaceDefaultRoute
  - type: vlan
    device: eth0
    vlan_id:
      get_param: StorageMgmtNetworkVlanID
    mtu:
      get_param: StorageMgmtNetworkMtu
    addresses:
    - ip_netmask:
        get_param: StorageMgmtIpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageMgmtSupernet
      next_hop:
        get_param: StorageMgmtInterfaceDefaultRoute
  - type: vlan
    device: eth0
    vlan_id:
      get_param: InternalApiNetworkVlanID
    mtu:
      get_param: InternalApiNetworkMtu
    addresses:
    - ip_netmask:
        get_param: InternalApiIpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
```

```
        get_param: InternalApiInterfaceDefaultRoute
  - type: interface
    name: eth1
    mtu:
      get_param: ExternalNetworkMtu
    addresses:
    - ip_netmask:
        get_param: ExternalIpSubnet
    routes:
    -
      default: True
      next_hop:
        get_param: ExternalInterfaceDefaultRoute
  - type: interface
    name: eth2
    use_dhcp: false

AppformixController_network_config:
  - type: interface
    name: eth0
    dns_servers:
      get_param: DnsServers
    use_dhcp: false
    mtu:
      get_param: ControlPlaneNetworkMtu
    addresses:
    - ip_netmask:
        list_join:
          - '/'
          - - get_param: ControlPlaneIp
            - get_param: ControlPlaneSubnetCidr
    routes:
    -
      ip_netmask: 169.254.169.254/32
      next_hop:
        get_param: EC2MetadataIp
  - type: vlan
    device: eth0
    vlan_id:
      get_param: InternalApiNetworkVlanID
    mtu:
      get_param: InternalApiNetworkMtu
    addresses:
```

```
    - ip_netmask:
        get_param: InternalApiIpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
        get_param: InternalApiInterfaceDefaultRoute
  - type: interface
    name: eth1
    mtu:
      get_param: ExternalNetworkMtu
    addresses:
    - ip_netmask:
        get_param: ExternalIpSubnet
    routes:
    -
      default: True
      next_hop:
        get_param: ExternalInterfaceDefaultRoute
  - type: interface
    name: eth2
    use_dhcp: false

ContrailTsn_network_config:
  - type: interface
    name: eth0
    dns_servers:
      get_param: DnsServers
    mtu:
      get_param: ControlPlaneNetworkMtu
    addresses:
    - ip_netmask:
        list_join:
          - '/'
          - - get_param: ControlPlaneIp
            - get_param: ControlPlaneSubnetCidr
    use_dhcp: false
    routes:
    -
      ip_netmask: 169.254.169.254/32
      next_hop:
        get_param: EC2MetadataIp
```

```
    -
      default: True
      next_hop:
        get_param: ControlPlaneDefaultRoute
  - type: vlan
    device: eth0
    vlan_id:
      get_param: InternalApiNetworkVlanID
    mtu:
      get_param: InternalApiNetworkMtu
    addresses:
    - ip_netmask:
        get_param: InternalApiIpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
        get_param: InternalApiInterfaceDefaultRoute
  - type: interface
    name: eth1
    use_dhcp: false
  - type: interface
    name: eth2
    use_dhcp: false
    mtu:
      get_param: TenantNetworkMtu
  - type: vlan
    device: eth2
    vlan_id:
      get_param: TenantNetworkVlanID
    mtu:
      get_param: TenantNetworkMtu
    use_dhcp: false
  - type: contrail_vrouter
    name: vhost0
    members:
      -
        type: interface
        name:
          str_replace:
            template: vlanVLANID
            params:
```

```
              VLANID: {get_param: TenantNetworkVlanID}
        use_dhcp: false
    mtu:
      get_param: TenantNetworkMtu
    addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
    routes:
    -
      ip_netmask:
        get_param: TenantSupernet
      next_hop:
        get_param: TenantInterfaceDefaultRoute

ComputeKernel0Hw1_network_config:
  - type: interface
    name: nic1
    dns_servers:
      get_param: DnsServers
    use_dhcp: false
    mtu:
      get_param: ControlPlaneNetworkMtu
    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
    device: nic1
    vlan_id:
      get_param: Storage0NetworkVlanID
    mtu:
      get_param: Storage0NetworkMtu
```

```
    addresses:
    - ip_netmask:
        get_param: Storage0IpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageSupernet
      next_hop:
        get_param: Storage0InterfaceDefaultRoute
  - type: vlan
    device: nic1
    vlan_id:
      get_param: InternalApi0NetworkVlanID
    mtu:
      get_param: InternalApi0NetworkMtu
    addresses:
    - ip_netmask:
        get_param: InternalApi0IpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
        get_param: InternalApi0InterfaceDefaultRoute
  - type: interface
    name: nic2
    use_dhcp: false
  - type: linux_bond
    name: bond0
    use_dhcp: false
    bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast updelay=1000
miimon=100"
    mtu:
      get_param: Tenant0NetworkMtu
    members:
    - type: interface
      name: nic3
      primary: true
      mtu:
        get_param: Tenant0NetworkMtu
    - type: interface
      name: nic4
      mtu:
```

```
        get_param: Tenant0NetworkMtu
  - type: vlan
    vlan_id:
      get_param: Tenant0NetworkVlanID
    device: bond0
  - type: contrail_vrouter
    name: vhost0
    use_dhcp: false
    members:
      -
        type: interface
        name:
          str_replace:
            template: vlanVLANID
            params:
              VLANID: {get_param: Tenant0NetworkVlanID}
        use_dhcp: false
    addresses:
    - ip_netmask:
        get_param: Tenant0IpSubnet
    mtu:
      get_param: Tenant0NetworkMtu
    routes:
    -
      ip_netmask:
        get_param: TenantSupernet
      next_hop:
        get_param: Tenant0InterfaceDefaultRoute

ComputeKernel0Hw0_network_config:
  - type: interface
    name: nic1
    dns_servers:
      get_param: DnsServers
    use_dhcp: false
    mtu:
      get_param: ControlPlaneNetworkMtu
    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
    device: nic1
    vlan_id:
      get_param: Storage0NetworkVlanID
    mtu:
      get_param: Storage0NetworkMtu
    addresses:
    - ip_netmask:
        get_param: Storage0IpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageSupernet
      next_hop:
        get_param: Storage0InterfaceDefaultRoute
  - type: vlan
    device: nic1
    vlan_id:
      get_param: InternalApi0NetworkVlanID
    mtu:
      get_param: InternalApi0NetworkMtu
    addresses:
    - ip_netmask:
        get_param: InternalApi0IpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
        get_param: InternalApi0InterfaceDefaultRoute
  - type: interface
    name: nic2
    use_dhcp: false
  - type: linux_bond
```

```
    name: bond0
    use_dhcp: false
    bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast updelay=1000
miimon=100"
    mtu:
      get_param: Tenant0NetworkMtu
    members:
    - type: interface
      name: nic3
      primary: true
      mtu:
        get_param: Tenant0NetworkMtu
    - type: interface
      name: nic4
      mtu:
        get_param: Tenant0NetworkMtu
  - type: vlan
    vlan_id:
      get_param: Tenant0NetworkVlanID
    device: bond0
  - type: contrail_vrouter
    name: vhost0
    use_dhcp: false
    members:
      -
        type: interface
        name:
          str_replace:
            template: vlanVLANID
            params:
              VLANID: {get_param: Tenant0NetworkVlanID}
        use_dhcp: false
    addresses:
    - ip_netmask:
        get_param: Tenant0IpSubnet
    mtu:
      get_param: Tenant0NetworkMtu
    routes:
    -
      ip_netmask:
        get_param: TenantSupernet
      next_hop:
        get_param: Tenant0InterfaceDefaultRoute
```

```
ComputeKernel1Hw0_network_config:
  - type: interface
    name: nic1
    dns_servers:
      get_param: DnsServers
    use_dhcp: false
    mtu:
      get_param: ControlPlaneNetworkMtu
    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
    device: nic1
    vlan_id:
      get_param: Storage1NetworkVlanID
    mtu:
      get_param: Storage1NetworkMtu
    addresses:
    - ip_netmask:
        get_param: Storage1IpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageSupernet
      next_hop:
        get_param: Storage1InterfaceDefaultRoute
  - type: vlan
    device: nic1
    vlan_id:
      get_param: InternalApi1NetworkVlanID
```

```
    mtu:
      get_param: InternalApi1NetworkMtu
    addresses:
    - ip_netmask:
        get_param: InternalApi1IpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
        get_param: InternalApi1InterfaceDefaultRoute
  - type: interface
    name: nic2
    use_dhcp: false
  - type: linux_bond
    name: bond0
    use_dhcp: false
    bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast updelay=1000
miimon=100"
    mtu:
      get_param: Tenant1NetworkMtu
    members:
    - type: interface
      name: nic3
      primary: true
      mtu:
        get_param: Tenant1NetworkMtu
    - type: interface
      name: nic4
      mtu:
        get_param: Tenant1NetworkMtu
  - type: vlan
    vlan_id:
      get_param: Tenant1NetworkVlanID
    device: bond0
  - type: contrail_vrouter
    name: vhost0
    use_dhcp: false
    members:
      -
        type: interface
        name:
          str_replace:
```

```
            template: vlanVLANID
            params:
              VLANID: {get_param: Tenant1NetworkVlanID}
        use_dhcp: false
    addresses:
    - ip_netmask:
        get_param: Tenant1IpSubnet
    mtu:
      get_param: Tenant1NetworkMtu
    routes:
    -
      ip_netmask:
        get_param: TenantSupernet
      next_hop:
        get_param: Tenant1InterfaceDefaultRoute

ComputeKernel1Hw1_network_config:
  - type: interface
    name: nic1
    dns_servers:
      get_param: DnsServers
    use_dhcp: false
    mtu:
      get_param: ControlPlaneNetworkMtu
    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
    device: nic1
    vlan_id:
      get_param: Storage1NetworkVlanID
```

```
    mtu:
      get_param: Storage1NetworkMtu
    addresses:
    - ip_netmask:
        get_param: Storage1IpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageSupernet
      next_hop:
        get_param: Storage1InterfaceDefaultRoute
  - type: vlan
    device: nic1
    vlan_id:
      get_param: InternalApi1NetworkVlanID
    mtu:
      get_param: InternalApi1NetworkMtu
    addresses:
    - ip_netmask:
        get_param: InternalApi1IpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
        get_param: InternalApi1InterfaceDefaultRoute
  - type: interface
    name: nic2
    use_dhcp: false
  - type: linux_bond
    name: bond0
    use_dhcp: false
    bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast updelay=1000
miimon=100"
    mtu:
      get_param: Tenant1NetworkMtu
    members:
    - type: interface
      name: nic3
      primary: true
      mtu:
        get_param: Tenant1NetworkMtu
    - type: interface
```

```
      name: nic4
      mtu:
        get_param: Tenant1NetworkMtu
  - type: vlan
    vlan_id:
      get_param: Tenant1NetworkVlanID
    device: bond0
  - type: contrail_vrouter
    name: vhost0
    use_dhcp: false
    members:
      -
        type: interface
        name:
          str_replace:
            template: vlanVLANID
            params:
              VLANID: {get_param: Tenant1NetworkVlanID}
        use_dhcp: false
    addresses:
    - ip_netmask:
        get_param: Tenant1IpSubnet
    mtu:
      get_param: Tenant1NetworkMtu
    routes:
    -
      ip_netmask:
        get_param: TenantSupernet
      next_hop:
        get_param: Tenant1InterfaceDefaultRoute

ComputeSriov0Hw4_network_config:
  - type: interface
    name: nic1
    dns_servers:
      get_param: DnsServers
    use_dhcp: false
    mtu:
      get_param: ControlPlaneNetworkMtu
    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
    device: nic1
    vlan_id:
      get_param: Storage0NetworkVlanID
    mtu:
      get_param: Storage0NetworkMtu
    addresses:
    - ip_netmask:
        get_param: Storage0IpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageSupernet
      next_hop:
        get_param: Storage0InterfaceDefaultRoute
  - type: vlan
    device: nic1
    vlan_id:
      get_param: InternalApi0NetworkVlanID
    mtu:
      get_param: InternalApi0NetworkMtu
    addresses:
    - ip_netmask:
        get_param: InternalApi0IpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
        get_param: InternalApi0InterfaceDefaultRoute
  - type: interface
    name: nic2
```

```
    use_dhcp: false
  - type: linux_bond
    name: bond0
    use_dhcp: false
    bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast updelay=1000
miimon=100"
    mtu:
      get_param: Tenant0NetworkMtu
    members:
    - type: interface
      name: nic3
      primary: true
      mtu:
        get_param: Tenant0NetworkMtu
    - type: interface
      name: nic4
      mtu:
        get_param: Tenant0NetworkMtu
  - type: vlan
    vlan_id:
      get_param: Tenant0NetworkVlanID
    device: bond0
  - type: contrail_vrouter
    name: vhost0
    use_dhcp: false
    members:
      -
        type: interface
        name:
          str_replace:
            template: vlanVLANID
            params:
              VLANID: {get_param: Tenant0NetworkVlanID}
        use_dhcp: false
    addresses:
    - ip_netmask:
        get_param: Tenant0IpSubnet
    mtu:
      get_param: Tenant0NetworkMtu
    routes:
    -
      ip_netmask:
        get_param: TenantSupernet
```

```
      next_hop:
        get_param: Tenant0InterfaceDefaultRoute


ComputeSriov1Hw5_network_config:
  - type: interface
    name: nic1
    dns_servers:
      get_param: DnsServers
    use_dhcp: false
    mtu:
      get_param: ControlPlaneNetworkMtu
    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
    device: nic1
    vlan_id:
      get_param: Storage1NetworkVlanID
    mtu:
      get_param: Storage1NetworkMtu
    addresses:
    - ip_netmask:
        get_param: Storage1IpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageSupernet
      next_hop:
        get_param: Storage1InterfaceDefaultRoute
  - type: vlan
```

```
    device: nic1
    vlan_id:
      get_param: InternalApi1NetworkVlanID
    mtu:
      get_param: InternalApi1NetworkMtu
    addresses:
    - ip_netmask:
        get_param: InternalApi1IpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
        get_param: InternalApi1InterfaceDefaultRoute
  - type: interface
    name: nic2
    use_dhcp: false
  - type: linux_bond
    name: bond0
    use_dhcp: false
    bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast updelay=1000
miimon=100"
    mtu:
      get_param: Tenant1NetworkMtu
    members:
    - type: interface
      name: nic3
      primary: true
      mtu:
        get_param: Tenant1NetworkMtu
    - type: interface
      name: nic4
      mtu:
        get_param: Tenant1NetworkMtu
  - type: vlan
    vlan_id:
      get_param: Tenant1NetworkVlanID
    device: bond0
  - type: contrail_vrouter
    name: vhost0
    use_dhcp: false
    members:
      -
```

```
        type: interface
        name:
          str_replace:
            template: vlanVLANID
            params:
              VLANID: {get_param: Tenant1NetworkVlanID}
        use_dhcp: false
    addresses:
    - ip_netmask:
        get_param: Tenant1IpSubnet
    mtu:
      get_param: Tenant1NetworkMtu
    routes:
    -
      ip_netmask:
        get_param: TenantSupernet
      next_hop:
        get_param: Tenant1InterfaceDefaultRoute

ComputeDpdk0Hw2_network_config:
  - type: interface
    name: nic1
    dns_servers:
      get_param: DnsServers
    use_dhcp: false
    mtu:
      get_param: ControlPlaneNetworkMtu
    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
```

```
    device: nic1
    vlan_id:
      get_param: Storage0NetworkVlanID
    mtu:
      get_param: Storage0NetworkMtu
    addresses:
    - ip_netmask:
        get_param: Storage0IpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageSupernet
      next_hop:
        get_param: Storage0InterfaceDefaultRoute
  - type: vlan
    device: nic1
    vlan_id:
      get_param: InternalApi0NetworkVlanID
    mtu:
      get_param: InternalApi0NetworkMtu
    addresses:
    - ip_netmask:
        get_param: InternalApi0IpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
        get_param: InternalApi0InterfaceDefaultRoute
  - type: interface
    name: nic2
    use_dhcp: false
  - type: contrail_vrouter_dpdk
    name: vhost0
    vlan_id:
      get_param: Tenant0NetworkVlanID
    driver: "{{ overcloud['contrail']['vrouter']['dpdk']['driver'] }}"
    bond_mode: 4
    bond_policy: layer2+3
    cpu_list: 1,2
    members:
    - type: interface
      name: nic3
```

```
    - type: interface
      name: nic4
    addresses:
    - ip_netmask:
        get_param: Tenant0IpSubnet
    mtu:
      get_param: Tenant0NetworkMtu
    routes:
    -
      ip_netmask:
        get_param: TenantSupernet
      next_hop:
        get_param: Tenant0InterfaceDefaultRoute

ComputeDpdk1Hw3_network_config:
  - type: interface
    name: nic1
    dns_servers:
      get_param: DnsServers
    use_dhcp: false
    mtu:
      get_param: ControlPlaneNetworkMtu
    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
    device: nic1
    vlan_id:
      get_param: Storage1NetworkVlanID
    mtu:
      get_param: Storage1NetworkMtu
```

```
    addresses:
    - ip_netmask:
        get_param: Storage1IpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageSupernet
      next_hop:
        get_param: Storage1InterfaceDefaultRoute
  - type: vlan
    device: nic1
    vlan_id:
      get_param: InternalApi1NetworkVlanID
    mtu:
      get_param: InternalApi1NetworkMtu
    addresses:
    - ip_netmask:
        get_param: InternalApi1IpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
        get_param: InternalApi1InterfaceDefaultRoute
  - type: interface
    name: nic2
    use_dhcp: false
  - type: contrail_vrouter_dpdk
    name: vhost0
    vlan_id:
      get_param: Tenant1NetworkVlanID
    driver: "{{ overcloud['contrail']['vrouter']['dpdk']['driver'] }}"
    bond_mode: 4
    bond_policy: layer2+3
    cpu_list: 1,2
    members:
    - type: interface
      name: nic3
    - type: interface
      name: nic4
    addresses:
    - ip_netmask:
        get_param: Tenant1IpSubnet
```

```
    mtu:
      get_param: Tenant1NetworkMtu
    routes:
    -
      ip_netmask:
        get_param: TenantSupernet
      next_hop:
        get_param: Tenant1InterfaceDefaultRoute

CephStorage0Hw6_network_config:
  - type: interface
    name: nic1
    dns_servers:
      get_param: DnsServers
    use_dhcp: false
    mtu:
      get_param: ControlPlaneNetworkMtu
    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
    device: nic1
    vlan_id:
      get_param: Storage0NetworkVlanID
    mtu:
      get_param: Storage0NetworkMtu
    addresses:
    - ip_netmask:
        get_param: Storage0IpSubnet
    routes:
    -
```

```
      ip_netmask:
        get_param: StorageSupernet
      next_hop:
        get_param: Storage0InterfaceDefaultRoute
  - type: vlan
    device: nic1
    vlan_id:
      get_param: StorageMgmt0NetworkVlanID
    mtu:
      get_param: StorageMgmt0NetworkMtu
    addresses:
    - ip_netmask:
        get_param: StorageMgmt0IpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageMgmtSupernet
      next_hop:
        get_param: StorageMgmt0InterfaceDefaultRoute
  - type: interface
    name: nic2
    use_dhcp: false
  - type: interface
    name: nic3
    use_dhcp: false
  - type: interface
    name: nic4
    use_dhcp: false

CephStorage1Hw7_network_config:
  - type: interface
    name: nic1
    dns_servers:
      get_param: DnsServers
    use_dhcp: false
    mtu:
      get_param: ControlPlaneNetworkMtu
    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
    device: nic1
    vlan_id:
      get_param: Storage1NetworkVlanID
    mtu:
      get_param: Storage1NetworkMtu
    addresses:
    - ip_netmask:
        get_param: Storage1IpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageSupernet
      next_hop:
        get_param: Storage1InterfaceDefaultRoute
  - type: vlan
    device: nic1
    vlan_id:
      get_param: StorageMgmt1NetworkVlanID
    mtu:
      get_param: StorageMgmt1NetworkMtu
    addresses:
    - ip_netmask:
        get_param: StorageMgmt1IpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageMgmtSupernet
      next_hop:
        get_param: StorageMgmt1InterfaceDefaultRoute
  - type: interface
    name: nic2
    use_dhcp: false
  - type: interface
```

```
    name: nic3
    use_dhcp: false
  - type: interface
    name: nic4
    use_dhcp: false
```

Sample compute-nodes.yml Configuration File

```
# Copyright 2018 Juniper Networks, Inc. All rights reserved.
# Licensed under the Juniper Networks Script Software License (the "License").
# You may not use this script file except in compliance with the License, which is located at
# http://www.juniper.net/support/legal/scriptlicense/
# Unless required by applicable law or otherwise agreed to in writing by the parties,
# software distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#
#
#

# Each list item contains:
#    name: name of a node in the inventory (string)
#    profile: name of hardware profile, group of servers (optional, string)
#    leaf: leaf name (optional, string)

# List of nodes to use as compute role using Contrail DPDK vRouter
compute_nodes_dpdk:
  - name: computedpdk1
    leaf: '0'
    profile: hw2
  - name: computedpdk2
    leaf: '1'
    profile: hw3

# List of nodes to use as compute role using Sriov
compute_nodes_sriov:
  - name: computesriov1
    leaf: '0'
    profile: hw4
  - name: computesriov2
```

```
    leaf: '1'
    profile: hw5

# List of nodes to use as compute role using Contrail kernel vRouter
compute_nodes_kernel:
  - name: compute1
    leaf: '0'
    profile: hw0
  - name: compute2
    leaf: '0'
    profile: hw1
  - name: compute3
    leaf: '1'
    profile: hw1
  - name: compute4
    leaf: '1'
    profile: hw0

# Sample list of host aggregates, containing:
# - name of aggregate
# - (optional) Availability Zone
# - (optional) metadata: a list of key:values
# - hosts: list of hosts assigned to aggregate
aggregates:
  rack1:
    az: "az1"
    metadata:
      - location: "DC1A3R1"
      - dc: "eng-prod1"
    hosts:
      - compute1
      - compute4
      - computedpdk1
  rack2:
    az: "az2"
    metadata:
      - location: "DC1A3R2"
    hosts:
      - compute2
      - computesriov1
      - computedpdk2
  rack3:
    az: "az3"
```

```
    metadata:
      - location: "DC1A3R3"
    hosts:
      - compute3
      - computesriov2
  sriov:
    metadata:
      - capabilities: "sriov"
    hosts:
      - computesriov1
      - computesriov2
  dpdk:
    metadata:
      - capabilities: "dpdk"
    hosts:
      - computedpdk1
      - computedpdk2
  kernel:
    metadata:
      - capabilities: "kernel"
    hosts:
      - compute1
      - compute2
      - compute3
      - compute4
```

Sample storage-nodes.yml Configuration File

```
# Copyright 2018 Juniper Networks, Inc. All rights reserved.
# Licensed under the Juniper Networks Script Software License (the "License").
# You may not use this script file except in compliance with the License, which is located at
# http://www.juniper.net/support/legal/scriptlicense/
# Unless required by applicable law or otherwise agreed to in writing by the parties,
# software distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#
#
#
```

```
# List of nodes to use as storage host role
# List item contains:
#    name: name of a node in the inventory (string)
#    profile: name of hardware profile, group of servers (optional, string)
#    leaf: leaf name (optional, string)

storage_nodes:
  - name: storage1
    leaf: '0'
    profile: hw6
  - name: storage2
    leaf: '0'
    profile: hw6
  - name: storage3
    leaf: '1'
    profile: hw7
```

Sample vault-data.yml Configuration File

```
# This config structure can be used to hold information that needs to be encrypted for privacy
# If there is a password stored in /var/lib/contrail_cloud/config/.vault_password then it will
be used
# Otherwise the password can be entered interactively
#
# This file can be edited with the "ansible-vault edit" command
# This file can be re-encrypted with a new password with the "ansible-vault rekey" command
vault:
  global:
    rhel:
      # Contrail Cloud Activation Key
      satellite:
        #SATELLITE_KEY
        key: "PUT_YOUR_KEY_HERE"
    # User account used for all Contrail Cloud automation
    # This account will be created on:
    #    - jumphost
    #    - control hosts
    #    - all overcloud roles
    #    - appformix controllers
```

```
  service_user:
    # Account Name
    name: "contrail"
    # Account Password
    password: "c0ntrail123"
    # Passphrase used to encrypt ssh key of service user.
    # If not defined ssh private key will not be encrypted.
    # ssh_key_passphrase: "c0ntrail123"
rhvm:
  vm:
    # rhvm user name
    user: "contrail"
    # password for the rhvm vm user
    password: "c0ntrail123"
    # root password for the rhvm VM
    root_password: "c0ntrail123"
    # keystone admin password
    admin_password: "c0ntrail123"
    # Passphrase used to encrypt ssh key of rhvm user.
    # If not defined ssh private key will not be encrypted.
    # ssh_key_passphrase: "c0ntrail123"
    vnc:
      # VNC console password for the rhvm VM
      password: "contrail123"
undercloud:
  #Administrator password - default is randomly generated
  #admin_password: "c0ntrail123"
  vm:
    # undercloud user name
    user: "stack"
    # password for the undercloud vm user
    password: "contrail123"
    # root password for the undercloud VM
    root_password: "contrail123"
    # Passphrase used to encrypt ssh key of undercloud user.
    # If not defined ssh private key will not be encrypted.
    # ssh_key_passphrase: "c0ntrail123"
    vnc:
      # VNC console password for the undercloud VM
      password: "contrail123"
overcloud:
  #Administrator password
  admin_password: "c0ntrail123"
```

```
  # Root password used for local login to overcloud nodes through console
  # root_password: "contrail123"
  contrail:
    rabbitmq:
      # contrail rabbitmq user name
      user: "contrail_rabbitmq"
      # contrail rabbitmq user password
      password: "c0ntrail123"
control_hosts:
  vm:
    vnc:
      # VNC console password for all control VMs
      password: "contrail123"
appformix:
  mysql:
    # Approfmix MySQL user account
    user: "appformix"
    # Approfmix MySQL user password
    password: "c0ntrail123"
  rabbitmq:
    # Approfmix RabbitMQ user account
    user: "appformix"
    # Approfmix RabbitMQ user password
    password: "c0ntrail123"
# Credentials used to connect external ceph cluster
#ceph_external:
#  client_key: "CLIENT_KEY"
#  client_user: "openstack"

# List of inventory hardware types that can hold hardware-specific properties
# You can create similar configutations to allow reference from inventory-nodes.yml
inventory_nodes:
  # A sample configuration for a hardware type
  hardware1:
    # IPMI user account for Ironic inevntory resources
    pm_user: "ADMIN"
    # IPMI user password for Ironic inevntory resources
    pm_password: "ADMIN"
  # A sample configuration for a hardware type
  hardware2:
    # IPMI user account for Ironic inevntory resource
    pm_user: "admin"
    # IPMI user password for Ironic inevntory resource
```

```
    pm_password: "admin"
# User defined sensitive data can be stored under 'other' key.
# Schema validation will only check if key,value format is used.
#other:
#  mykey: myvalue
```

RELATED DOCUMENTATION

- Deploy Contrail Cloud | 5

Appendix B: Node Reboot and Health Check

IN THIS SECTION

From time-to-time, you need to manually reboot all the nodes (VMs and hypervisor). Updating the packages does not automatically trigger a subsequent reboot; you must manually reboot the nodes. The reboot sequence presented in Appendix B helps ensure minimal disruption to services. It is equally important to check the health of each service as it is brought back up. This is to maintain a quorum of healthy nodes for high-availability clusters.

The following procedure details the proper sequence and commands for rebooting and checking each service and node. The procedure covers both Juniper Contrail components and Red Hat components for HA environments.

Prerequisites

Before you reboot your nodes:

- Deploy the software update: Contrail Cloud update scripts (role by role).

Update and Reboot the Control Plane Nodes

Run the procedure in the order of starting with non-virtual IP (VIP) nodes, and then repeat the procedure for the nodes with VIP.

> **NOTE**: The sample output lists are deployment-specific. Your list shows output specific to your Contrail Cloud deployment.
>
> All SSH connections to the control hosts need to be done from the jump host to ensure that proper SSH keys are used.

1. (Control Hosts: non-VIP nodes to VIP nodes) Identify all the VMs associated with the hypervisor:

   Run the following commands from the jumphost:

   ```
   su - contrail
   ssh undercloud
   source stackrc
   ```

   192.168.213.10 is the address of the control host in the following sample.

   ```
   for NODE in $(openstack baremetal node list --long -c Name -c "Driver Info" -c "Instance
   UUID" | awk -F "|" '/192.168.213.10/ {print $3}'); do openstack server show ${NODE}  -c
   addresses -f value | cut -d= -f2 ; done
   ```

2. Quiet the Pacemaker service.

   Pacemaker ensures high availability of the cluster and takes action if node quorum is disrupted. Rebooting the nodes disrupts the quorum of nodes, which is why you need to quiet Pacemaker while the reboot is in progress.

a. Disable Pacemaker resources before stopping the Pacemaker service:

```
ansible -i /usr/bin/tripleo-ansible-inventory Controller -m shell --become -a 'pcs status|
awk "/stonith/{print \$1}" | xargs -n 1 pcs resource disable'
```

NOTE: Pacemaker should be brought back up as soon as possible to preserve the integrity of your HA cluster.

b. Identify the node using the following naming format (overcloud*XXX*-ctrl-*N*).

SSH to the node and stop the pcs cluster:

```
ssh <address> sudo pcs cluster stop
```

3. Update the packages for each address that was returned in Step 1:

```
ssh <address> sudo yum update -y
```

4. Shut down each VM that was previously updated:

```
ssh <address> sudo shutdown -h 0
```

5. Disable autostart for the VMs from the jump host:

```
ssh 192.168.213.5 'for vm in appformix-controller contrail-analytics contrail-analytics-
database contrail-controller control; do sudo virsh autostart ${vm} --disable ; done'
```

6. Update the hypervisor after all the VMs have been shut down.

From the jump host, establish an SSH connection to the control host and run the update:

```
ssh <control-host-ip> sudo yum update -y
```

7. Enable autostart for the VMs:

```
ssh 192.168.213.5 'for vm in appformix-controller contrail-analytics contrail-analytics-
database contrail-controller control; do sudo virsh autostart ${vm} ; done'
```

8. Reboot the hypervisor.

   From the jump host, establish an SSH connection to the control host and run the update:

   > **NOTE**: After the reboot, the hypervisor starts all VMs that are associated with it along with the Pacemaker service.

   ```
   ssh <control-host-ip> sudo shutdown -r 0
   ```

   Although the Pacemaker service starts after the hypervisor is rebooted, Pacemaker is still in a *cluster stop* state. You must start the Pacemaker cluster once all services are verified to be up and healthy.

9. Verify that all services are healthy after you reboot the instances per role in the steps above. Once all services have been verified to be healthy, repeat the process above for the remaining hypervisors and associated VMs.

Reboot the undercloud and the jump host:

You need to reboot the undercloud and jump host after rebooting the overcloud.

1. Shut down your undercloud. From the undercloud:

   ```
   sudo shutdown -h 0
   ```

2. Shut down Contrail Command as the `contrail` user from the jump host:

   ```
   ssh command sudo shutdown -h 0
   ```

3. Shutdown and reboot the jump host as the `root` user:

   ```
   sudo shutdown -r 0
   ```

All services come back up after the jump host is rebooted.

Verify Quorum and Node Health

IN THIS SECTION

Purpose

It is important to verify the cluster high availability state and the health state of all the control nodes in the cluster. This sequence verifies that there is a quorum of healthy nodes and that the cluster returns to normal operating status.

Action

This procedure needs to be performed on the target host. Establish an SSH connection to the node, complete the steps, and then exit back out.

To verify all components of an HA cluster:

1. After rebooting the hypervisor, run a system status check from the jump host as the `contrail` user. Run this check until a healthy status is returned:

```
ssh <control host> sudo systemctl list-units --state=failed
```

2. Verify that all VMs have started properly and that no VM is in a failed status:

```
ssh <control host> sudo virsh list -all
```

3. Run a service check from the undercloud for each item on the node until a healthy status is returned:

   a. Check the OpenStack Controller.

Verify the health of the `systemd` services:

```
ansible -i /usr/bin/tripleo-ansible-inventory Controller -m shell -a "sudo systemctl
list-units --state=failed 'openstack*' 'neutron*' 'httpd' 'docker' 'ceph*'"
```

b. Verify the health of Pacemaker from the OpenStack controller:

```
ansible -i /usr/bin/tripleo-ansible-inventory Controller -m shell -a "sudo pcs status"
```

Verify that:

- All cluster nodes are online.
- No resources are stopped on any of the cluster nodes.
- There are no failed Pacemaker actions.

c. Verify the status of Ceph from the OpenStack controller:

```
ansible -i /usr/bin/tripleo-ansible-inventory Controller -m shell -a "sudo ceph status"
```

4. Verify the status of the Contrail Controller. Run the following command from the Contrail controller node:

```
ansible -i /usr/bin/tripleo-ansible-inventory ContrailController -m shell -a "sudo contrail-
status"
```

5. Verify the status of Contrail Analytics. Run the following command from the Contrail Analytics node:

```
ansible -i /usr/bin/tripleo-ansible-inventory ContrailAnalytics -m shell -a "sudo contrail-
status"
```

6. Verify the status of Contrail Analytics Database:

```
ansible -i /usr/bin/tripleo-ansible-inventory ContrailAnalyticsDatabase -m shell -a "sudo
contrail-status"
```

7. Verify the status of AppFormix:

```
ansible -i /usr/bin/tripleo-ansible-inventory AppformixController -m shell -a "curl -s
http://127.0.0.1:9000/appformix/controller/v2.0/status"
```

When successful, this returns the code 200. Any other code that is returned should be considered a failure.

The API output also contains the AppFormix version. This is helpful to verify that the correct version has been installed. See the following sample:

```
{ "Version": "2.19.10-65aa34f7ad", "DBVersion": "70" }
```

NOTE: AppFormix and MongoDB containers need about 60 seconds after a reboot to properly come up and synchronize with each other.

8. To check the MongoDB cluster status:

```
ansible -i /usr/bin/tripleo-ansible-inventory AppformixController --become -m shell -a
'source /opt/appformix/mongo/config/credentials.js ;  echo "rs.status();" | docker exec -i
appformix-mongo mongo admin --username "${USERNAME}" --password "${PASSWORD}"'
```

9. Verify the status of the ToR services node (TSN):

NOTE: The TSN node is an optional feature and might not exist in your environment.

```
ansible -i /usr/bin/tripleo-ansible-inventory ContrailTsn -m shell -a "sudo contrail-status"
```

10. Repeat this process after the next cycle of reboots are performed for the next hypervisor.

NOTE: Always verify that the services and nodes are healthy before proceeding to the reboot procedure for the next hypervisor.

11. Start the Pacemaker cluster that was previously stopped.

Starting the Pacemaker cluster happens last. This is after all updates and reboots have been performed. All nodes must be up at this point so that Pacemaker detects a quorum. This also assumes that all verifications and health checks came back with the proper status.

a. Start the Pacemaker cluster (overcloud*XXX*ctrl-*N*):

```
ssh <address> sudo pcs cluster start --all
```

b. Enable Pacemaker services:

```
ansible -i /usr/bin/tripleo-ansible-inventory Controller -m shell --become -a 'pcs
status| awk "/stonith/{print \$1}" | xargs -n 1 pcs resource enable
```

Reboot Ceph Storage Nodes

Follow this procedure to properly reboot your Ceph storage nodes.

First, disable the Ceph storage cluster, select a storage node, and then reboot it. After the reboot, verify the status of the node. Repeat the reboot process for all Ceph storage nodes. Enable the cluster and then run a final status report to verify that the cluster health is OK.

1. Log in to the OpenStack Controller node (overcloud*XXX*ctrl-*N*) from the undercloud, and disable Ceph storage cluster rebalancing:

```
ssh <openstack controller> sudo ceph osd set noout
ssh <openstack controller> sudo ceph osd set norebalance
```

2. Select the storage node that you want to reboot, and reboot the node from the undercloud:

```
ssh <storage-node> sudo shutdown -r 0
```

3. Wait until the storage node reboots, and then verify the status from the OpenStack controller:

NOTE: The cluster takes time to return to normal. Do not proceed with this procedure until the cluster returns to a normal state.

```
ssh <openstack controller> sudo ceph status
```

Check that the pgmap reports all placement groups (PGs) as normal (active and clean).

4. Repeat step 2 and step 3 until all storage nodes have been rebooted. Proceed only when Step 3 returns an “active+clean” status.

NOTE: You must ensure that Ceph is healthy before you proceed to reboot the other storage nodes.

5. After rebooting all storage nodes, log in to the OpenStack controller node, and enable Ceph Storage cluster rebalancing:

NOTE: Rebalance only if Step 3 returns an “active+clean” status.

```
ssh <openstack controller> sudo ceph osd unset noout
ssh <openstack controller> sudo ceph osd unset norebalance
```

6. Perform a final status check on the OpenStack controller to verify the health of the cluster:

```
ssh <openstack controller> sudo ceph status
```

Reboot Compute Nodes

Follow this procedure to properly reboot your compute nodes.

First, select the compute node that you want to reboot, and then disable it so that it doesn’t provision new instances. Migrate all instances to another compute node. Reboot the compute node, and enable it once the reboot is complete.

1. Log in to the undercloud as the `stack` user.

```
source ~/stackrc
```

2. View the list of all compute nodes in your deployment.

```
openstack server list | grep comp
```

3. Find the compute node that you want to reboot.
4. From the undercloud, select the compute node, and disable it.

```
$ source ~/overcloudrc
(overcloud) $ openstack compute service list | grep node
(overcloud) $ openstack compute service set <compute-node> nova-compute --disable
```

5. Migrate all instances off the compute node. Identify all the instances by pulling a list.

```
openstack server list --host <compute-node> --all-projects
```

6. Choose one of the following methods to migrate your instances, that best fits your needs.
   a. Migrate the instance to a specific host:

```
openstack server migrate <service> --live <compute-node> --wait
```

   b. Let `nova-scheduler` automatically select the compute node:

```
nova live-migration [instance-id]
```

   c. Live migrate all instances at once:

```
nova host-evacuate-live <compute-node>
```

7. Wait until migration is complete, and then confirm that the migration was successful:

```
openstack server list --host [hostname] --all-projects
```

8. Finish migrating the instances until all instances have been removed from the compute node.

9. Log in to the compute node from the undercloud, and reboot it:

```
ssh <compute> sudo shutdown -r 0
```

10. Wait until the reboot is finished, and then enable the compute node:

```
openstack compute service set [hostname] nova-compute --enable
```

11. Verify that the compute node is enabled:

```
openstack compute service list
```

12. Repeat for all remaining compute nodes in your deployment.

RELATED DOCUMENTATION

Deploy Contrail Cloud | 5

Rebooting The Overcloud

Appendix C: Deploying Netronome SmartNIC

This appendix contains information to configure and deploy Netronome SmartNIC in your Contrail Cloud environment.

The procedure below will show you:

- How to get the Netronome containers.
- How to configure the roles and properties for Netronome including profiles.
- How to deploy Netronome in Contrail Cloud.

Prepare for Netronome deployment.

You need to configure specific parameters in your YAML configuration files before deploying Netronome in your Contrail Cloud environment. The following will demonstrate which files you need to configure and will give you examples for each section. Complete samples for Netronome deployment can be referenced in **samples/features/netronome_support/**. This folder will have Netronome specific

examples for each required YAML configuration file listed within. Make sure to copy the necessary configuration samples into the appropriate YAML configuration files for the changes to take effect.

The following procedure will prepare your Contrail Cloud environment for Netronome deployment:

1. Review the samples in **samples/features/netronome_support/**.
2. Compute nodes with Netronome cards must include “Smartnic” in the profile name to enable Agilio services. Add the “Smartnic” profile name to your **config/compute_nodes.yml**:

```
compute_nodes_kernel:
  - name: compute1
    leaf: '0'
    profile: Smartnic0
```

3. Provide information about the Agilio containers version and source. Modify your **config/site.yml** overcloud section to include “smartnic:” per the format found in the **samples/features/netronome_support/site.yml** file. Here you will select container source and tag, as seen below:

```
overcloud:
  smartnic:
    vrouter_tag: '2.39-rhel-queens'
    forwarder_tag: '2.39-rhel-queens'
    vrouter_path: docker.io/netronomesystems/agilio-vrouter
    forwarder_path: docker.io/netronomesystems/virtio-forwarder
```

   The information you entered above will be used by **scripts/netronome_containers_upload.sh** later in the procedure.
4. Modify Netronome-enabled compute nodes in the overcloud extra configurations section of **config/site.yml** to reflect hugepages and kernel parameters for the compute nodes using Netronome cards.

```
overcloud:
  extra_config:
    ComputeKernel0Smartnic0Parameters:
      # Force writing of kernel parameters
      KernelArgs: "intel_iommu=on iommu=pt isolcpus=1,2"
      ContrailVrouterHugepages2MB: "8192"
      PHYSICAL_INTERFACE: nfp_p0
```

Deploy Netronome in your Contrail Cloud Environment.

The previous section set the parameters for your Netronome deployment. This section will now take you through the procedure for the script needed to deploy Netronome.

Start your Netronome deployment after you have completed the undercloud node deployment (**scripts/install_contrail_cloud_manager.sh**). The steps below will load the Netronome Agilio containers:

1. The script below will perform the Docker login and download Agilio containers from the source you provided in **config/site.yml** from the previous section. The process will then upload the containers to the local registry on the undercloud.

   > **NOTE**: The username and password used below is for accessing the plugins in the registry and is supplied by Netronome. This is not the same as your Contrail Cloud login credentials.

   ```
   scripts/netronome_containers_upload.sh -u <USER> -p <PASSWORD>
   ```

2. You can now proceed with the rest of the deployment process as stated in the Contrail Cloud Deployment Guide.

RELATED DOCUMENTATION

- Contrail Cloud Documentation
- Contrail Networking NIC Support Table
- Contrail Cloud Reference Architecture Guide
- Contrail Cloud Release Notes

Appendix D: Remove a Ceph Storage Node

Use this procedure to remove a Ceph storage node from a Ceph cluster. Ceph storage node removal is handled as a Red Hat process rather than an end-to-end Contrail Cloud process. However, this procedure will demonstrate the removal of a storage node from an environment in the context of Contrail Cloud.

Before you begin, ensure that the remaining nodes in the cluster will be sufficient for keeping the required amount of pgs and replicas for your Ceph storage cluster. Ensure that both Ceph cluster and overcloud stack are healthy. For checking the health of your overcloud, see Verify Quorum and Node Health.

All examples in this procedure come from a lab setting to demonstrate storage removal within the context of Contrail Cloud. Sample output in the provided examples will differ from the information in your specific cloud deployment. In the examples used for this procedure, “storage3” will be the targeted node for removal.

Remove the storage node:

1. Find the connection between the bare metal server and the overcloud server. The output from the command below shows us that the serer we are looking for is “overcloud8st-cephstorageblue1-0”. This information will be used later in the procedure.

```
(undercloud) [stack@undercloud ~]$ openstack ccloud nodemap list
+--------------------------------+---------------+------------+----------------+
| Name                           | IP            | Hypervisor | Hypervisor IP  |
+--------------------------------+---------------+------------+----------------+
| overcloud8st-cc-2              | 192.168.213.54 | controler2 | 192.168.213.6  |
| overcloud8st-cc-1              | 192.168.213.51 | controler3 | 192.168.213.7  |
| overcloud8st-cc-0              | 192.168.213.55 | controler1 | 192.168.213.5  |
| overcloud8st-ca-1              | 192.168.213.72 | controler1 | 192.168.213.5  |
| overcloud8st-ca-0              | 192.168.213.60 | controler3 | 192.168.213.7  |
| overcloud8st-ca-2              | 192.168.213.53 | controler2 | 192.168.213.6  |
| overcloud8st-cadb-1            | 192.168.213.71 | controler1 | 192.168.213.5  |
| overcloud8st-afxctrl-0         | 192.168.213.69 | controler2 | 192.168.213.6  |
| overcloud8st-afxctrl-1         | 192.168.213.52 | controler3 | 192.168.213.7  |
| overcloud8st-afxctrl-2         | 192.168.213.58 | controler1 | 192.168.213.5  |
| overcloud8st-cadb-0            | 192.168.213.65 | controler2 | 192.168.213.6  |
| overcloud8st-ctrl-0            | 192.168.213.73 | controler2 | 192.168.213.6  |
| overcloud8st-ctrl-1            | 192.168.213.63 | controler1 | 192.168.213.5  |
| overcloud8st-ctrl-2            | 192.168.213.59 | controler3 | 192.168.213.7  |
| overcloud8st-cephstorageblue1-0 | 192.168.213.62 | storage3   | 192.168.213.62 |
| overcloud8st-compdpdk-0        | 192.168.213.56 | compute1   | 192.168.213.56 |
| overcloud8st-cephstorageblue2-0 | 192.168.213.61 | storage2   | 192.168.213.61 |
| overcloud8st-cephstorageblue2-1 | 192.168.213.80 | storage1   | 192.168.213.80 |
| overcloud8st-cadb-2            | 192.168.213.74 | controler3 | 192.168.213.7  |
+--------------------------------+---------------+------------+----------------+
```

2. From the undercloud as the `heat-admin` user, SSH to any of the openstack controllers and then run `sudo ceph -s` to verify that the Ceph cluster is healthy:

```
[root@overcloud8st-ctrl-1 ~]# sudo ceph -s
  cluster:
```

```
    id:     a98b1580-bb97-11ea-9f2b-525400882160
    health: HEALTH_OK
```

3. Find the OSDs that reside on the server to be removed (overcloud8st-cephstorageblue1-0). We identify osd.2, osd.3, osd.6, and osd.7 from the example below:

```
[root@overcloud8st-ctrl-1 ~]# sudo ceph osd tree
ID CLASS WEIGHT   TYPE NAME                                STATUS REWEIGHT PRI-AFF
-1       10.91638 root default
-3        3.63879     host overcloud8st-cephstorageblue1-0
 2   hdd  0.90970         osd.2                                up  1.00000 1.00000
 3   hdd  0.90970         osd.3                                up  1.00000 1.00000
 6   hdd  0.90970         osd.6                                up  1.00000 1.00000
 7   hdd  0.90970         osd.7                                up  1.00000 1.00000
-7        3.63879     host overcloud8st-cephstorageblue2-0
 1   hdd  0.90970         osd.1                                up  1.00000 1.00000
 4   hdd  0.90970         osd.4                                up  1.00000 1.00000
 8   hdd  0.90970         osd.8                                up  1.00000 1.00000
10   hdd  0.90970         osd.10                               up  1.00000 1.00000
-5        3.63879     host overcloud8st-cephstorageblue2-1
 0   hdd  0.90970         osd.0                                up  1.00000 1.00000
 5   hdd  0.90970         osd.5                                up  1.00000 1.00000
 9   hdd  0.90970         osd.9                                up  1.00000 1.00000
11   hdd  0.90970         osd.11                               up  1.00000 1.00000
```

4. While still logged in to the openstack controller mark osd.2, osd.3, osd.6, and osd.7 as non-operational:

```
[root@overcloud8st-ctrl-1 ~]# sudo ceph osd out 2
marked out osd.2.
[root@overcloud8st-ctrl-1 ~]# sudo osd out 3
marked out osd.3.
[root@overcloud8st-ctrl-1 ~]# sudo ceph osd out 6
marked out osd.6.
[root@overcloud8st-ctrl-1 ~]# sudo ceph osd out 7
marked out osd.7.
```

From the undercloud as the `heat-admin` user, SSH to any of the openstack controllers and then run `sudo ceph -s` to verify that the Ceph cluster returns a “health_ok” state before you continue.

5. From the undercloud as the `heat-admin` user, SSH to Ceph node overcloud8st-cephstorageblue1-0, and stop the OSD services:

```
[root@overcloud8st-cephstorageblue1-0 ~]# sudo systemctl stop  ceph-osd@2.service
[root@overcloud8st-cephstorageblue1-0 ~]# sudo systemctl stop  ceph-osd@3.service
[root@overcloud8st-cephstorageblue1-0 ~]# sudo systemctl stop  ceph-osd@6.service
[root@overcloud8st-cephstorageblue1-0 ~]# sudo systemctl stop  ceph-osd@7.service
```

From the undercloud as the `heat-admin` user, SSH to any of the openstack controllers and then run `sudo ceph -s` to verify that the Ceph cluster returns a “health_ok” state before you continue.

6. From the undercloud as the `heat-admin` user, SSH back into the controller and remove further information about the OSDs from overcloud8st-cephstorageblue1-0:

```
[root@overcloud8st-ctrl-1 ~]# sudo ceph osd crush remove osd.2
removed item id 2 name 'osd.2' from crush map
[root@overcloud8st-ctrl-1 ~]# sudo ceph osd crush remove osd.3
removed item id 3 name 'osd.3' from crush map
[root@overcloud8st-ctrl-1 ~]# sudo ceph osd crush remove osd.6
removed item id 6 name 'osd.6' from crush map
[root@overcloud8st-ctrl-1 ~]# sudo ceph osd crush remove osd.7
removed item id 7 name 'osd.7' from crush map


[root@overcloud8st-ctrl-1 ~]# sudo ceph auth del osd.2
updated
[root@overcloud8st-ctrl-1 ~]# sudo ceph auth del osd.3
updated
[root@overcloud8st-ctrl-1 ~]# sudo ceph auth del osd.6
updated
[root@overcloud8st-ctrl-1 ~]# sudo ceph auth del osd.7
updated


[root@overcloud8st-ctrl-1 ~]# sudo ceph osd rm 2
removed osd.2
[root@overcloud8st-ctrl-1 ~]# sudo ceph osd rm 3
removed osd.3
[root@overcloud8st-ctrl-1 ~]# sudo ceph osd rm 6
removed osd.6
[root@overcloud8st-ctrl-1 ~]# sudo ceph osd rm 7
removed osd.7
```

```
[root@overcloud8st-ctrl-1 ~]# sudo ceph osd crush rm overcloud8st-cephstorageblue1-0
```

From the undercloud as the `heat-admin` user, SSH to any of the openstack controllers and then run `sudo ceph -s` to verify that the Ceph cluster returns a “health_ok” state before you continue.

7. From the undercloud VM find the ID of the Ceph storage node:

```
(undercloud) [stack@undercloud ~]$ openstack server list | grep overcloud8st-
cephstorageblue1-0
| 7ee9be4f-efda-4837-a597-a6554027d0c9 | overcloud8st-cephstorageblue1-0 | ACTIVE |
ctlplane=192.168.213.62 | overcloud-full | CephStorageBlue1
```

8. Initiate a removal using the node ID from the previous step:

```
(undercloud) [stack@undercloud ~]$ openstack overcloud node delete --stack overcloud
7ee9be4f-efda-4837-a597-a6554027d0c9
```

From the undercloud as the `heat-admin` user, SSH to any of the openstack controllers and then run `sudo ceph -s` to verify that the Ceph cluster returns a “health_ok” state before you continue.

9. Verify that the bare metal node is in a state of `power off` and `available`:

```
(undercloud) [stack@undercloud ~]$ openstack baremetal node list | grep storage3
| 05bbab4b-b968-4d1d-87bc-a26ac335303d | storage3 | None  | power off  | available | False |
```

10. From the jump host as the `contrail` user mark the storage node with ‘status: deleting’ so the Ceph profile will be removed from it. Add the ‘status: deleting’ to the **storage-nodes.yml** file for storage3 and then run the script **storage-nodes-assign.sh**.

```
[contrail@5a6s13-node1 contrail_cloud]$ cat config/storage-nodes.yml
storage_nodes:
  - name: storage1
    profile: blue2
  - name: storage2
    profile: blue2
  - name: storage3
    profile: blue1
    status: deleting

[contrail@5a6s13-node1 contrail_cloud]$ ./scripts/storage-nodes-assign.sh
```

From the undercloud as the `heat-admin` user, SSH to any of the openstack controllers and then run `sudo ceph -s` to verify that the Ceph cluster returns a “health_ok” state before you continue.

11. From the jump host as the `contrail` user, run **openstack-deploy.sh** to regenerate the templates to reflect the current state:

```
[contrail@5a6s13-node1 contrail_cloud]$ ./scripts/openstack-deploy.sh
```

From the undercloud as the `heat-admin` user, SSH to any of the openstack controllers and then run `sudo ceph -s` to verify that the Ceph cluster returns a “health_ok” state before you continue.

If the goal is to remove the bare metal node completely, use the following additional procedure:

1. Edit the **config/storage-nodes.yml** file and remove the bare metal node.
2. Edit the **inventory.yml** file and include the ‘status: deleting’ to the node to be removed:

```
[contrail@5a6s13-node1 contrail_cloud]$ cat config/inventory.yml
...
inventory_nodes:
  - name: "storage3"
    pm_addr: "10.84.129.184"
    status: deleting
    <<: *common
```

3. Run the `inventory-assign.sh` script:

```
[contrail@5a6s13-node1 contrail_cloud]$ ./scripts/inventory-assign.sh
```

From the undercloud as the `heat-admin` user, SSH to any of the openstack controllers and then run `sudo ceph -s` to verify that the Ceph cluster returns a “health_ok” state before you continue.

4. Verify the bare metal node has been removed. Enter the following command to view the list of nodes:

```
(undercloud) [stack@undercloud ~]$ openstack ccloud nodemap list |grep storage
 | overcloud8st-cephstorageblue2-1 | 192.168.213.80 | storage1 | 192.168.213.80 |
| overcloud8st-cephstorageblue2-0 | 192.168.213.61 | storage2 | 192.168.213.61 |
```

Appendix E: Remove a Compute Node

This procedure details how to remove a compute node from the overcloud in order to scale down a deployed environment.

Before you begin:

- Migrate all workloads to different hypervisors before you begin the compute node removal procedure.
- The process below uses a hypothetical situation created in a lab environment. Keep in mind that the names, addresses, and other information will be different in your specific environment.

Remove a compute node from your environment:

In this situation, we want to remove a compute node in the context of the undercloud that is called `host1`. This is based on the entry found in the **/var/lib/contrail_cloud/config/inventory.yml** configuration file.

1. From the undercloud, run:

```
source /home/stack/overcloudrc && openstack ccloud nodemap list --hypervisor host1
```

   Your output will return something similar to below:

```
+--------------------------+---------------+-------------+----------------+
| Name                     | IP            | Hypervisor  | Hypervisor IP  |
+--------------------------+---------------+-------------+----------------+
| overcloudx72-compkernel-1 | 192.168.213.64 | host1       | 192.168.213.64 |
+--------------------------+---------------+-------------+----------------+
```

   Take note of the name `overcloudx72-compkernel-1`. This is the hostname of `host1`, in context of the overcloud.

2. Run the following command from the undercloud to prevent workloads from being scheduled on the given hypervisor:

```
source /home/stack/overcloudrc && openstack compute service set --disable $(openstack
hypervisor list --matching overcloudx72-compkernel-1 -c "Hypervisor Hostname" -f value)
nova-compute
```

   We have now marked `overcloudx72-compkernel-1` as disabled, and all future workloads will not be placed on this target hypervisor.

3. Verify that the hypervisor no longer holds any workloads. Run the following command from the undercloud:

```
source /home/stack/overcloudrc && openstack hypervisor list --matching  overcloudx72-
compkernel-1 -c "Hypervisor Hostname" -f value | xargs -n 1 openstack server list --host
```

   This command should return an empty result.

   If the hypervisor has remaining workloads, you need to migrate that workload before continuing.

4. From the jumphost, modify your **/var/lib/contrail_cloud/config/compute-nodes.yml** configuration file by adding `status: "deleting"` under the node to be deleted. See below for an example entry in the **compute-nodes.yml** configuration file:

```
compute_nodes_kernel:
  name: host1
  status: "deleting"
```

5. From the jumphost, run:

```
/var/lib/contrail_cloud/scripts/compute-remove.sh -d
```

   It is important to note that this will remove the compute node from the overcloud. However, the compute node will still be shown on the hypervisor, and will be reported as down. It is normal for the node to remain in the output of “nova hypervisor-list” after being removed from the overcloud. Upon completion of this procedure, the compute node is effectively considered removed from your deployed environment.

6. Remove the node entry from the satellite. From the jumphost, run:

```
/var/lib/contrail_cloud/scripts/satellite6hosts.py --user USERNAME --password PASSWORD --
satellite contrail-cloud-satellite.juniper.net delete --host overcloudx72-
compkernel-1.env.example.com
```

   Replace `USERNAME` and `PASSWORD` with your username and password.

   Replace `env.example.com` with the `global["domain"]` value found in your **site.yml** configuration file.

7. Completely remove the node entry from your **compute-nodes.yml** configuration file.
8. Delete the node from your undercloud inventory.

From the jumphost, modify your **/var/lib/contrail_cloud/config/inventory.yml** configuration file by adding the line `status: "deleting"` under the node in question. See below for an example entry in the **inventory.yml** configuration file:

```
- name: "host1"
    status: "deleting"
    pm_addr: "10.10.10.10"
    <<: *hardware1
```

9. Run the **inventory-assign.sh** script to complete the deletion of the node:

```
/var/lib/contrail_cloud/scripts/inventory-assign.sh -d
```

10. Validate that the node has been removed from your inventory.

    From the undercloud, run:

```
source /home/stack/stackrc && openstack baremetal node list
```

    The output should no longer show `host1`.

11. Finalize the procedure by removing the host entry from your **/var/lib/contrail_cloud/config/inventory.yml** configuration file.

RELATED DOCUMENTATION

- Migrating Workloads