Intel® Data Center Blocks for Cloud – Red Hat* OpenStack* Platform with Red Hat Ceph* Storage

Reference Architecture

Guide for deploying a private cloud based on Red Hat OpenStack* Platform with Red Hat Ceph Storage using Intel® Server Products.

Rev 1.0

January 2017

Intel® Server Products and Solutions

<Blank page>

Document Revision History

Date	Revision	Changes
January 2017	1.0	Initial release.

Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Learn more at Intel.com, or from the OEM or retailer.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel, the Intel logo, and Xeon are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Table of Contents

List of Figures

List of Tables

1. Executive Summary

This reference architecture document provides assistance to Intel channel partners deploying private clouds using Intel ingredients, including guidance on server, network, and software configurations. Specifically, this document explains how to successfully deploy a Software Defined Infrastructure on a private cloud based on Red Hat* OpenStack* Platform using Intel® Server Products such as server boards, chassis, solid state drives, network interface cards, and RAID controllers.

This document is targeted at solution architects, system engineers, chief technology officers, chief information officers, and network and system administrators.

This document is intended to be the first of two parts:

- Reference Architecture: Part 1 – Software and hardware stack definition, network topology and configuration, nodes configuration, and best known methods on deploying a private cloud.
- Reference Architecture: Part 2 –Benchmarks, results, fine tuning, and key recommendations on how to achieve these results.

Some important notes:

- Intel may make updates to this document as necessary.
- This is not intended to be the sole source of information on Red Hat OpenStack Platform or Intel® Data Center Blocks for Cloud.
- This is not intended to be a complete or detailed step-by-step implementation guide. Procedures may vary for each implementation scenario and it is not possible to cover all details in a single document.

2. Red Hat* OpenStack* Platform Overview

Intel has long been a partner of the open source community and is currently an OpenStack Foundation Platinum Member (https://www.openstack.org/foundation/companies/). As of the publication of this document, Intel is among the top ten OpenStack contributors according to Stack Alytics (http://stackalytics.com/).

Red Hat OpenStack Platform offers an integrated solution to create and manage a reliable and secure Infrastructure-as-a-Service (IaaS) cloud using available hardware to provide computing, storage, and networking resources. This reference architecture uses Red Hat* Enterprise Linux* 7.2 or 7.3 and Red Hat* OpenStack* Platform 9.

The recommendations in this document are not applicable to all workloads, scenarios, or use cases, but the intention is to replicate a fairly common environment across multiple organizations, based on recent research.

- OpenStack users share how their deployments stack up - http://superuser.openstack.org/articles/openstack-users-share-how-their-deployments-stack-up
- User survey identifies leading industries and business drivers for OpenStack adoption - http://superuser.openstack.org/articles/user-survey-identifies-leading-industries-and-business-drivers-for-openstack-adoption
- OpenStack User Survey: A snapshot of OpenStack users' attitudes and deployments - http://www.openstack.org/assets/survey/April-2016-User-Survey-Report.pdf

OpenStack deployments tend to start small in number of nodes, cores and storage, and grow with demands and organization maturity and skillsets. Typical OpenStack deployments have heterogeneous servers to perform controller, compute, and storage roles. Each role drives specific software and hardware demands.

Note: Hyper-converged solutions, where compute, network and storage are converged in servers, are possible in Red Hat OpenStack Platform 10 as a Technology Preview. The new feature, known as Hyper-Converged Infrastructure (HCI), co-locates compute and block storage services on the same node.

This reference architecture covers each of these roles in more detail including how they interact with each other and their impacts on high-availability, performance, fault tolerance, and other aspects of the deployment and operations.

2.1 Red Hat* OpenStack* Platform Architecture Overview

The Red Hat OpenStack Platform cloud employs a set of services working together to control computing, storage, and networking resources. The cloud can be managed through a web-based interface or an extensive application programming interface (API). The diagram in Figure 1 provides an overview of OpenStack services and how they interact with each other.

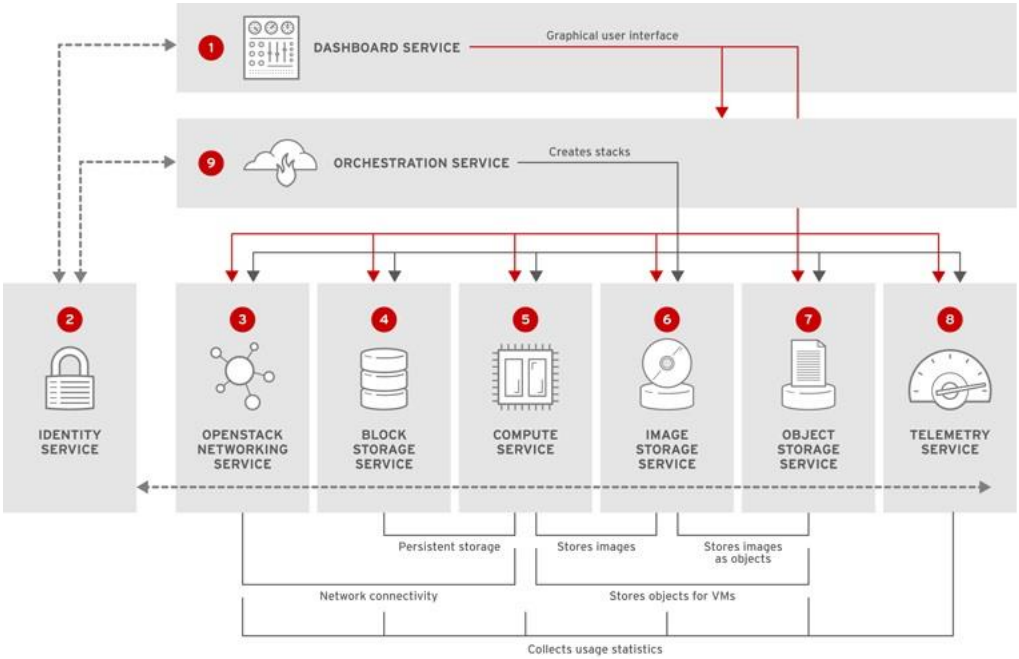


Figure 1. OpenStack service overview

Each service plays a unique and important role, briefly described in the following sections. The service code name is given in parentheses for each service. For more detailed information on these services, please refer to Red Hat OpenStack Platform 9 Architecture Guide.

2.1.1 Dashboard Service (Horizon)

The dashboard is the main graphical user interface (GUI) for users and administrators. Depending on the users' role, the user can perform several tasks from the dashboard such as creating and launching new instances (virtual machines); accessing a global view of resources; and managing the virtual network, virtual storage, and security rules. Access an overview of all resources available from the **Project** or **Admin** menus. A screenshot of the overview is shown in Figure 2.

Note: The Admin menu provides access to all projects. In this reference architecture, the whole infrastructure is created under a single project. This document does not explore multi-tenant, multi-project IaaS.

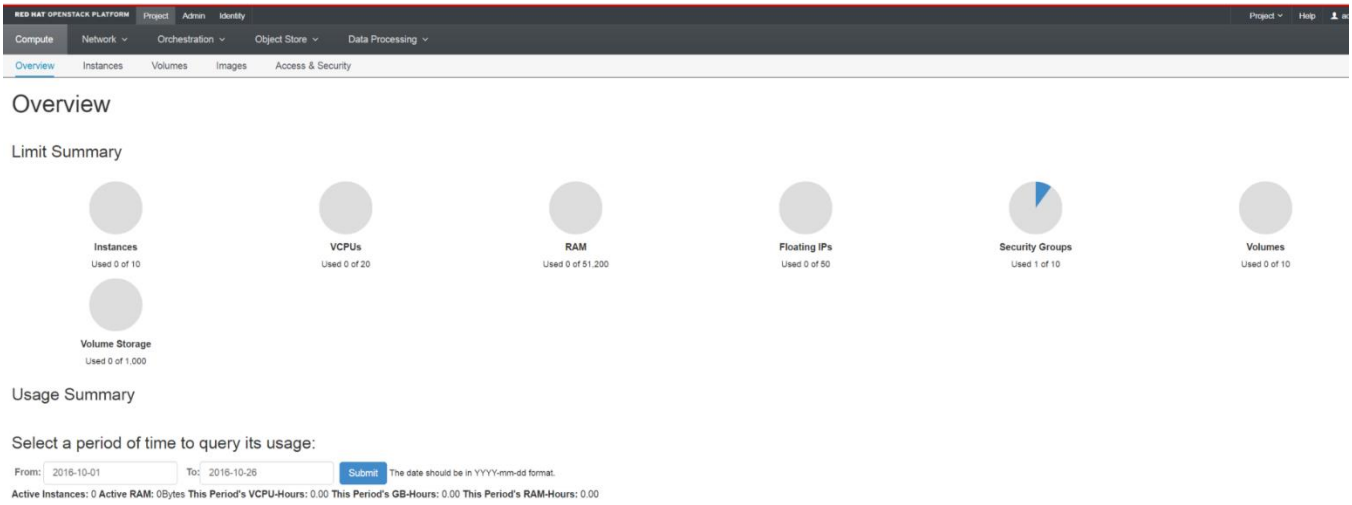


Figure 2. Red Hat OpenStack Platform dashboard overview screenshot

2.1.2 Identity Service (Keystone)

The identity service module provides multiple authentication and authorization mechanisms such as username and password and token-based access.

2.1.3 Networking Service (Neutron)

The networking service module creates the overlay network on top of the physical network. It consists of virtual network infrastructure elements and provides the framework to configure networks, subnets, and routers. Depending on the complexity of your deployment, the networking service can also provide virtual firewalls, virtual load-balancers, or a virtual private network (VPN).

2.1.4 Block Storage Service (Cinder)

The block storage service manages persistent block storage for virtual storage. For this reference architecture, Red Hat Ceph storage is the backend storage solution.

2.1.5 Compute Service (Nova)

The compute service is one of the core modules for any OpenStack deployment consisting of all virtual machines, also called instances. View all compute nodes in the hypervisor section of the dashboard. This reference architecture uses five compute nodes as shown in Figure 3.

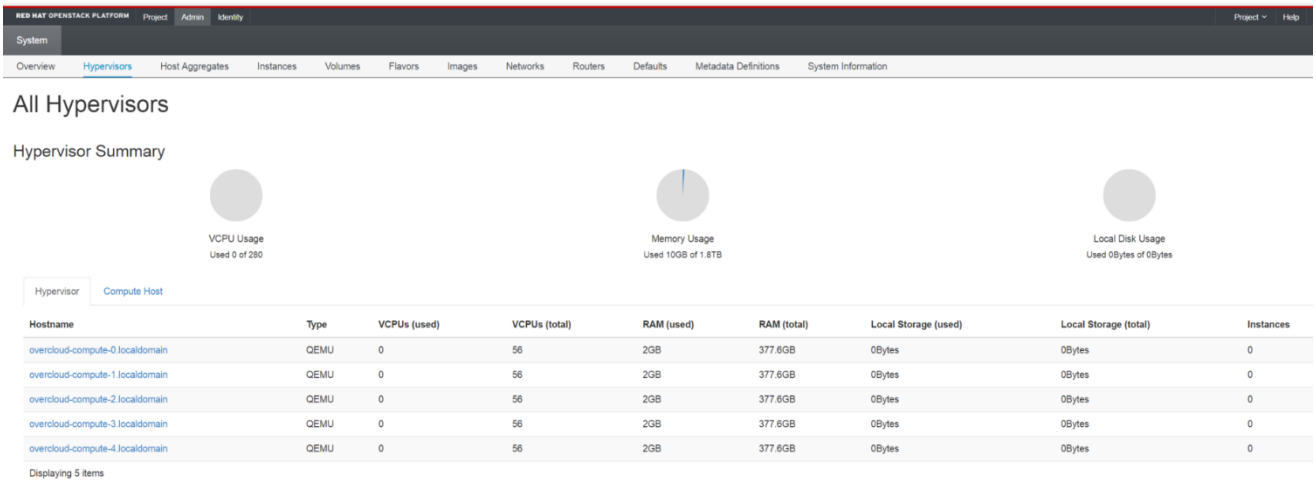


Figure 3. Red Hat OpenStack Platform hypervisors summary

2.1.6 Image Service (Glance)

The image service is a registry of all virtual disk images and templates. This service is crucial from initial server operating system (OS) installation, to the pre-boot execution environment (PXE), to deployment of virtual machines. The image service can also be used as a snapshot repository.

2.1.7 Object Storage Service (Swift)

The object storage service is a repository for objects such as videos, images, files and virtual machine images. For this reference architecture, Red Hat Ceph Storage is the backend storage solution.

2.1.8 Telemetry Service (Ceilometer)

The telemetry service module provides data for system monitoring, alerts, and tenant/customer resource usage for billing.

2.1.9 Orchestration Service (Heat)

The orchestration service provides templates for describing cloud applications that can be deployed over the OpenStack infrastructure. It has the capability of interacting with all other modules and can be used by configuration management tools such as Puppet* and Ansible*, making deployment easier and faster.

2.1.10 Bare Metal Provisioning (Ironic)

For this reference architecture, bare metal provisioning is used to deploy the Red Hat OpenStack Platform. It relies on Intelligent Platform Management Interface (IPMI) to discover the nodes and, through PXE boot, install the images and prepare the nodes for deployment.

2.2 Red Hat* OpenStack* Platform Director

For this reference architecture, Red Hat OpenStack Platform director was used to install and manage the complete OpenStack cloud. Red Hat OpenStack Platform director installs OpenStack on one node to further deploy a full and scalable OpenStack environment. This process is based on the OpenStack project TripleO (OpenStack-On-OpenStack).

Red Hat OpenStack Platform director consists of the undercloud and overcloud. The undercloud is the director node itself that should be installed on a separate physical server. The overcloud is the OpenStack cloud environment. After the director is set up and configured, it deploys the OpenStack controller, compute, and storage nodes according to the users' specification. The diagram in Figure 4 illustrates these concepts.

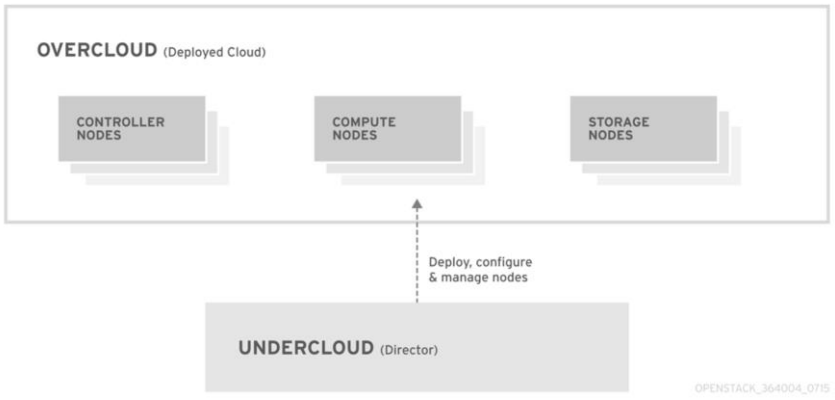


Figure 4. Undercloud and overcloud concepts

Note: Red Hat OpenStack Platform director needs a dedicated server for potential scalable environments to prevent falling short on resources. If an environment is too small, one option is to use Red Hat OpenStack director in a virtual machine to avoid the overhead of having a dedicated node.

Red Hat OpenStack Platform director provides a single place to start the deployment and configure all controller, compute, and storage nodes. It permits scaling out each component independently, making the OpenStack deployment very flexible. It uses several known tools to perform each task. Its major OpenStack components are not different from the Red Hat OpenStack Platform framework described in the previous section.

- Bare Metal (ironic) and Compute (nova)
- Networking (neutron) and Open vSwitch*
- Image Service (glance)
- Orchestration (heat) and Puppet*
- Telemetry (ceilometer), Telemetry Metrics (gnocchi) and Telemetry Alarming (aodh)
- Identity (keystone)
- MariaDB* as the database
- RabbitMQ* as the messaging queue for all components

Note: Red Hat OpenStack Platform director uses a terminal-based command line interface (CLI) instead of a graphical user interface. A certain level of Linux CLI expertise is needed.

3. Intel® Server System Configuration

The controller, compute, and storage that are part of the overcloud vary according to each implementation. The minimum recommendation for a high-available and robust deployment includes three controller nodes, three compute nodes, and three storage nodes.

For this reference architecture, a total of 12 nodes are used including:

- (1) undercloud node (same configuration as the control node)
- (3) controller nodes
- (5) compute nodes
- (3) storage nodes

3.1 Server Chassis Configuration

To fulfill all 12 nodes, this reference architecture uses three 2U Intel® Server Chassis that support up to four nodes each. The server distribution in the rack aims to avoid a single point of failure by locating one controller and one storage node in each physical server chassis as shown in Figure 5.

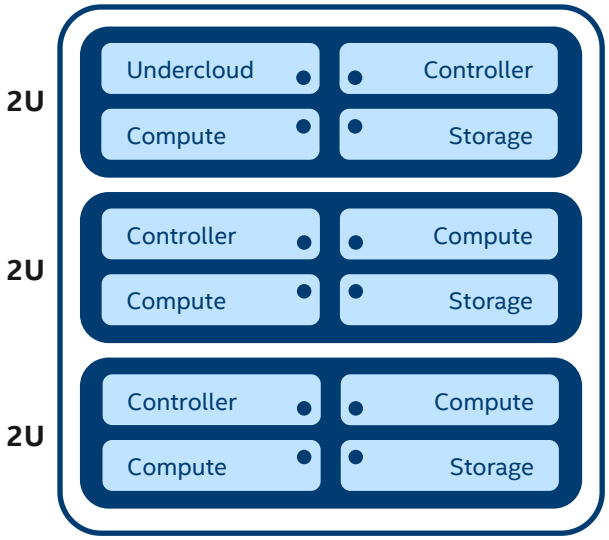


Figure 5. Example server distribution

Note: The same configuration is applicable to single node 1U and 2U server systems. Refer to Intel® Data Center Blocks for Cloud at http://www.intel.com/content/www/us/en/data-center-blocks/cloud/cloud-blocks.html for details.

Specifically, this configuration uses the four node Intel® Server Chassis H2224XXKR2 with a redundant power supply.

Table 1. Server chassis specification

Description	Product Type	Intel Product Code	MM#	Quantity
Intel® Server Chassis H2224XXKR2	Chassis Only	H2224XXKR2	943478	3

Figure 6. Intel® Server Chassis H2224XXKR2 w/ 4 Nodes + 2 PSUs Installed

3.2 Node Configuration

The complete configuration for each node is detailed in the tables below.

Note: All nodes use bridge boards that connect each compute module to six drive bays in the front of the chassis. These bridge boards are IT Mode and LSI 3008.

Table 2. Controller node specification

Component	Description	Intel Product Code	MM#	Quantity
Compute module	Intel® Compute Module HNS2600TP24SR See section 3.2.1 for details	HNS2600TP24SR	945609	1
Processor	Intel® Xeon® E5-2660 v4 (14 Cores, 35M Cache, 2.00 GHz)	CM8066002031201	947617	2
Memory	32 GB DDR4 DIMM (384GB total)	-	-	12
Network adapter	10 GbE SFP+ Dual Port Intel® Ethernet Converged Network Adapter	X710DA2	933206	1
Boot device	Intel® SSD DC S3500 Series (340GB, M.2)	SSDSCKHB340G401	932266	1
Remote management module	Intel® Remote Management Module 4 Lite	AXXRMM4LITE2	946514	1
Local storage	Intel® SSD DC S3710 Series (800 GB, SATA)	SSDSC2BA800G401	937745	2

Table 3. Compute node specification

Component	Description	Intel Product Code	MM#	Quantity
Compute module	Intel® Compute Module HNS2600TP24SR See section 3.2.1 for details	HNS2600TP24SR	945609	1
Processor	Intel® Xeon® E5-2660 v4 (14 Cores, 35M Cache, 2.00 GHz)	CM8066002031201	947617	2
Memory	32 GB DDR4 DIMM (384GB total)	-	-	12
Network adapter	10 GbE SFP+ Dual Port Intel® Ethernet Converged Network Adapter	X710DA2	933206	1
Boot device	Intel® SSD DC S3500 Series (340GB, M.2)	SSDSCKHB340G401	932266	1
Remote management module	Intel® Remote Management Module 4 Lite	AXXRMM4LITE2	946514	1
Local storage	Intel® SSD DC P3700 Series (1.6TB, PCIe* NVMe*)	SSDPE2MD016T401	933081	2

Table 4. Storage node specification

Component	Description	Intel Product Code	MM#	Quantity
Compute module	Intel® Compute Module HNS2600TP24SR See section 3.2.1 for details	HNS2600TP24SR	945609	1
Processor	Intel® Xeon® E5-2640 v4 (10 Cores, 25M Cache, 2.40 GHz)	CM8066002032701	948123	2
Memory	16 GB DDR4 DIMM (128 GB total)	-	-	8
Network adapter	10 GbE SFP+ Dual Port Intel® Ethernet Converged Network Adapter	X710DA2	933206	1
Boot device	Intel® SSD DC S3500 Series (340GB, M.2)	SSDSCKHB340G401	932266	1
Remote management module	Intel® Remote Management Module 4 Lite	AXXRMM4LITE2	946514	1
Data storage (all-flash node)	Intel® SSD DC S3710 Series (800 GB, SATA)	SSDSC2BA800G401	937745	5
Data storage (hybrid node)	Seagate* Enterprise Capacity 2.5 HDD (2TB, SAS)	ST2000NX0433	-	5
Journaling storage	Intel® SSD DC P3700 Series (1.6TB, PCIe* NVMe*)	SSDPE2MD016T401	933081	1

Note: The storage nodes can be either all-flash or hybrid. The difference is the data capacity drives which are SATA SSDs for all-flash and SAS HDDs for hybrid. In both cases, the journaling drive is NVMe*.

3.2.1 Compute Module

As indicated in the previous section, all nodes use the Intel® Compute Module HNS2600TP24SR. This integrated compute module includes:

- (1) Intel® Server Board S2600TPR with two 1Gb ports (Intel® Ethernet Controller I350);
- (1) 12 Gb/s bridge board (FHWKPTPBGB24);
- (1) node power board (FH2000NPB24);
- (1) one slot PCIe* x16 riser card (FHW1U16RISER2);
- (1) front 1U passive heat sinks (FXXEA84X106HS);
- (1) rear 1U passive heat sink (FXXCA91X91HS);
- (3) 4056 dual rotor fan (FXX4056DRFAN2);
- (1) PCIe* x16 rIOM riser and rIOM carrier board kit (AXXKPTPM2IOM);
- (1) dual SFP+ port 10GBASE-T I/O module (AXX10GBNIAIOM);
- (1) air duct;
- (1) 1U node tray

3.2.2 Red Hat certification

All Intel® Server Systems in this reference architecture are Red Hat* certified. For example, the integrated compute module certifications are shown in Figure 7.

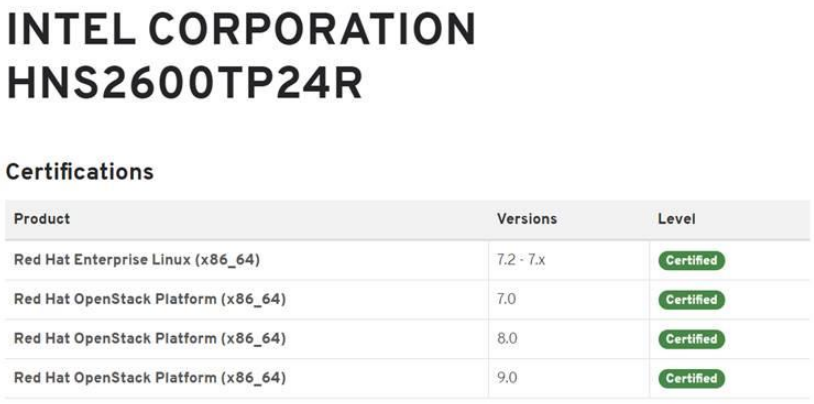


Figure 7. Red Hat* certification for HNS2600TP24R compute module

For a complete list of certified Intel Server Systems, please refer to the Red Hat Ecosystem at https://access.redhat.com/ecosystem/.

Note: Components such as network interface cards and storage drives are not listed in the Red Hat* Ecosystem, but are fully supported and validated by Intel on Intel Server Systems.

4. Planning A Solution

The planning stage is the most important part of a private cloud deployment.

The following sections cover some basics of business and technical planning. The questions posed are not meant to be complete; there are many more questions and details to be considered based on user/customer demands, business and application requirements and regulatory constraints, to name a few.

4.1 Basic Business Requirements

- What is the total budget?
- What applications are going to be run?
- What is the private cloud business model (IaaS, Platform as a Service (PaaS), Software as a Service (SaaS), Everything as a Service (XaaS))?
- Who are the team members and what are their skillsets? Project manager?
- Who are the major sponsors?
- What is the timeline?
- Are there any auditing implications? Any standard or compliance frameworks (such as HIPAA or PCI)?
- What are the success criteria?
- Hybrid Cloud integration?

4.2 Technical Requirements

- What applications are going to be run on the private cloud or Ceph cluster?
- What is the expected security level (hardware, IaaS, application)?
- What is the expected growth for compute, storage, network?
- Is there any external centralized log system?
- Are there any billing requirements?

4.2.1 Compute

- What are the characteristics of the applications?
- Will the applications benefit more from core counts, high frequency, or a balance between the two?
- What is the expected average size of the virtual machines (VMs)/instances?
- How many VMs/instances are expected to run on each node?
- What is the expected oversubscription ratio for cores?
- What is the expected failure tolerance?

4.2.2 Network

- What is the expected throughput?
- Does the network use Network Interface Card (NIC) teaming/bonding?
- What is the overlay network? Which protocol does it use?
- What are the features of the physical network? What is the port technology (1GbE, 10GbE, 25GbE, 40GbE)?
- What are the protocols that must be supported?
- What are the implications of integrating with the current environment?
- What is the expected failure tolerance?
- Are there any encryption requirements?
- Is there any expected or required application latency?
- Are there any virtual functions to be deployed such as virtual firewall, vRouter, or vLoadBalance?
- Is there a connection from Top of Rack (TOR) to aggregate switch?

4.2.3 Storage

- What types of data are to be stored on the Ceph cluster?
- Should the Ceph cluster be optimized for capacity and performance?
- What should the usable storage capacity be?
- What is the expected growth rate (month/year)?
- Is there any expected or required application latency?
- Is there an expected minimum input/output operations per second (IOPS) that the cluster should support?
- How much throughput should the cluster support?
- How much data replication (reliability level) is needed?
- What is the backup strategy?
- What is the expected failure tolerance?

4.3 Planning the Environment

In addition to having a clear understanding of the goals, scope, applications, and workloads, it is important to know the environment intimately. The following key items should be very clear before starting the deployment.

- Server hardware:
  - Details of the CPU, memory, boot device, journal and capacity drives and how they are recognized by the OS, and NIC.
  - Latest drivers, firmware, BIOS, BMC, known bugs, and release notes that can impact the deployment.
  - Component vendor websites, forums, bug trackers, and other similar resources.
- Physical network topology:
  - Ports and supported features of the switches.
  - Current Virtual LANs (VLANs) and any new VLANs that are needed.
  - Out of band management solution (such as Intelligent Platform Management Interface (IPMI) or Redfish* API).
  - Details for a new or existing PXE network and VLAN or independent physical network.
  - Network IP addressing plan for each VLAN.
- Virtual network topology
  - Details of OpenStack* Network Service (neutron), OpenVirtual Switch (OVS), VXLAN, and VLAN.
  - Integration with physical network (if any).
  - Control and data plane details.
- Monitoring, metric, alert, alarm, and backup plans.
- Security needs (such as Trusted Platform Module (TPM), Secure Boot, Geotagging).

Most importantly, always have a troubleshooting plan for situations where the deployment does not work as expected. The more you know about the environment, the better and, hopefully, faster you can solve unexpected issues. Make sure to have the appropriate support from your solution, hardware, and software providers.

5. Network Topology Considerations

Network topology is critical and must be considered before cloud deployment. Once the cloud is deployed, network topology changes are not easily made and can impact the overall private cloud infrastructure. Be sure to run through the planning stage in section 4 and discuss the network topology with several different people before deployment.

In this reference architecture, the recommendation is to have multiple network interfaces configured with VLAN separation to make sure the environment is scalable and robust. While it may be more than enough at the onset, this configuration is highly scalable and efficient and is designed to grow with the cloud environment. Refer to Figure 8 for a diagram of the recommended network topology.

Note: Ensure the network topology is correct by checking the network switch configuration and making sure the right cabling is in place, the appropriate VLANs are created and set, and trunk ports are configured

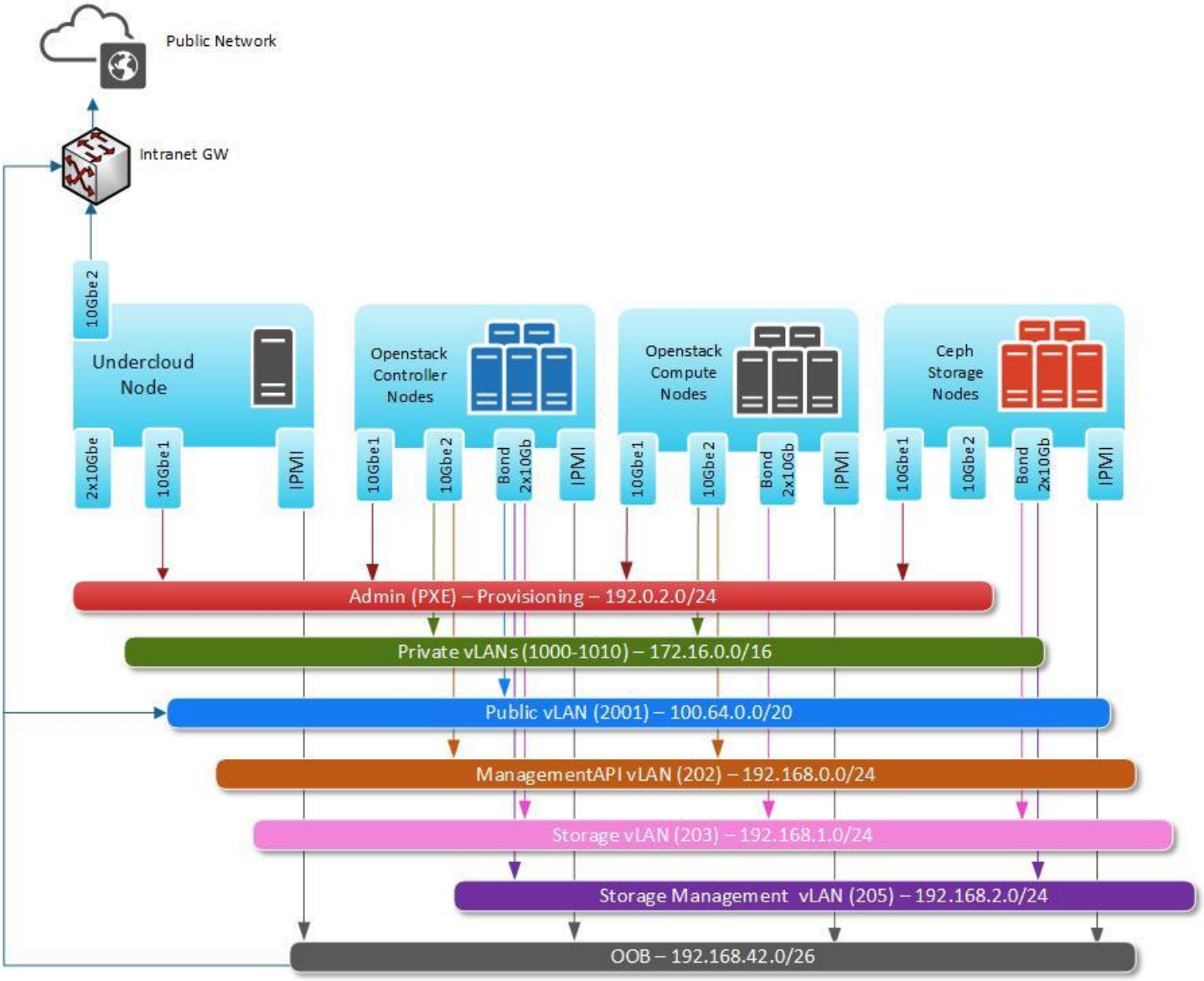


Figure 8. Recommended network topology

The network interfaces shown in Figure 8 include:

- Out of Band (OOB) – Connected through the Intelligent Platform Management Interface (IPMI), this network interface ensures connectivity to the servers for remote power and is used by ironic for node discovery.
- Provisioning – Red Hat OpenStack Platform director uses this network traffic to deploy new nodes over PXE boot and orchestrate the installation of overcloud bare metal servers. After deployment it is also the default gateway for non-controller nodes.
- Private, Public, Management API, Storage, and Storage Management – These are typical OpenStack network types with implemented Open vSwitch* as service provider network (virtual switch) and Virtual Extensible LAN (VXLAN) isolation for tenant network.

Note: Storage Management VLAN is used for storage replication purposes whereas Storage VLAN is the one used to server to compute and controller nodes.

Table 5. Network summary

	Mappings	Total Physical Interfaces	Total VLANs
Flat Network (No 802.1Q VLAN tagging)	• Provisioning (PXE)	1	N/A
External Network	• Public • Out of Band (OOB)	2 (including 1 bonded interface)	1
Isolated Networks	• Tenant Network (Private) • Internal API (Management API) • Storage • Storage Management	2 (including 1 bonded interface)	4 (this number can vary depending on your tenant deployment)

Note: Red Hat OpenStack Platform consists of several technologies to implement high-availability (HA). The HA backend is Pacemaker cluster manager, which adds the ability to detect failures of OpenStack components. The procedure herein is suitable for a Red Hat OpenStack Platform environment deployed using Red Hat OpenStack Platform director, and configured in a fully HA state.

Details must be provided to Red Hat OpenStack Platform director via scripts. This provides a method for mapping OpenStack network types to certain subnets or VLANs, depending on how they are defined and associated with the nodes based on their roles. These traffic types include:

- Internal API (Management API)
- Storage
- Storage Management
- Tenant (Private)
- External / Floating
- Management
- Provisioning

Any unassigned networks are automatically assigned to the same subnet as the provisioning network.

The cables used to make those connections are shown in Table 6.

Table 6. Cable types for physical interfaces

Interface	Cable type
10GbE SFP+	CAB-SFP-SFP-3M 10GBase-CR SFP+ 3-Meter Copper Twinax Cable
1GbE (IPMI)	Cat 5e 1GB RJ45

In the recommended network topology, each overcloud node uses four 10 GbE interfaces:

- ens802f0
- ens802f1
- ens785f0 and ens785f1 in bond configuration

The name of interfaces may be different depending on PCI slot configuration. The following tables specify how each node in each chassis uses their interfaces and how they are physically connected to the switch.

Table 7. NIC-OS correlation

NIC (2x10Gbps - SFP+)	RHEL Name	Driver
Intel X520	ens802f0/ens802f1	ixgbe
Intel X710	ens785f0/ens785f1	i40e

Table 8. Switch ports for OpenStack* nodes

Node		Switch Port
Director		Eth 45-48
Compute	1	Eth 1-4
	2	Eth 29 - 32
	3	Eth 41 - 44
	4	Eth 37 - 40
	5	Eth 13 - 16
Controller	1	Eth 17 - 20
	2	Eth 25 - 28
	3	Eth 21-24
Storage	1	Eth 5 - 8
	2	Eth 9 - 12
	3	Eth 33 - 36

Table 9. OpenStack* network type assignment

NIC Order	Director	Controller Node	Compute Node	Storage Node
ens802f0	Admin (PXE)	Admin(PXE)	Admin(PXE)	Admin(PXE)
ens802f1	Public	Private	Private	
		Management API	Management API	
Bond ens785f0 & ens785f1		External (Public API)	Storage	Storage Management
		Floating IP		Storage
		Storage		
		Storage Management		

Table 10. VLANs for OpenStack* network assignments

Network Name	VLAN
Admin (PXE) (native VLAN)	201*
Public	2001
Private	1000-1010
Management API	202
Storage	203
Storage Management	205

For bond configuration, use port channels in active/active mode without Link Aggregation Control Protocol (LACP) on switch; LACP with Open vSwitch-based bonds is problematic and unsupported. An alternative is to use LACP with standard Linux kernel mode bonds and Open vSwitch on top. Find more details at https://github.com/novacain1/redhat-sepialab/blob/master/templates/nic-configs/compute.yaml#L101-L168.

The routing and switching architecture should achieve expectation of workloads with network redundancy. Using all-flash storage Ceph nodes quickly reaches network limit. Using Open vSwitch implementation of bond can speed up network and flows by customizing optional settings, such as:

- `other_config:bond-miimon-interval=100`
- `other_config:bond-rebalance-interval=10000`
- `prevent flapping by other_config:bond_updelay=1000`

In `/etc/sysconfig/network-scripts/ifcfg-bond1` file in default overcloud deployment, only common settings are used:

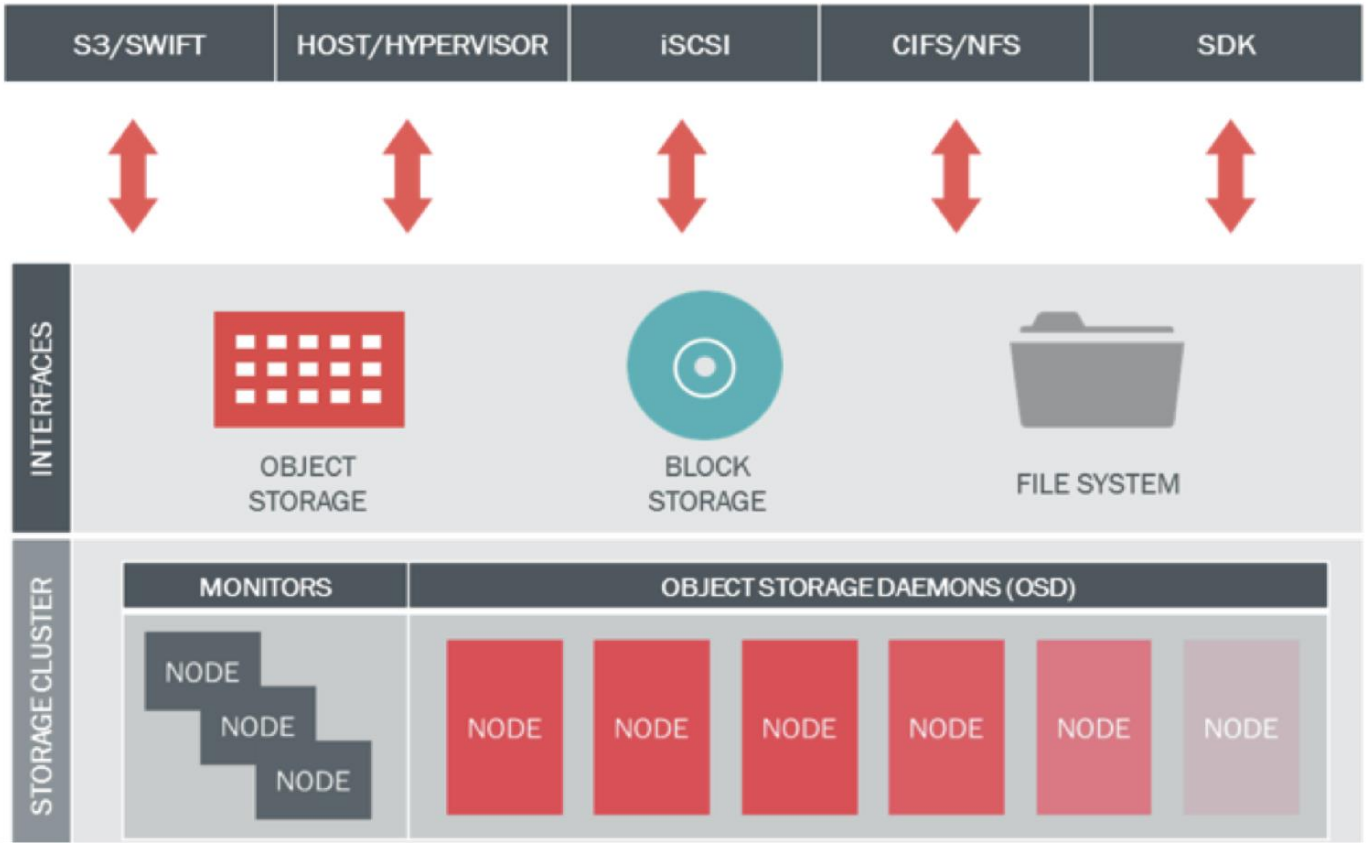
```
DEVICE=bond1
ONBOOT=yes
HOTPLUG=no
NM_CONTROLLED=no
PEERDNS=no
DEVICETYPE=ovs
TYPE=OVSPort
OVS_BRIDGE=br-storage
DEVICETYPE=ovs
TYPE=OVSBond
BOND_IFACES="ens785f0 ens785f1"
OVS_OPTIONS="bond_mode=balance-slb"
```

RedHat OpenStack Platform 9 uses Open vSwitch version 2.4.0. Refer to the Open vSwitch Manual for more details.

6. Storage Considerations

One key element of any storage solution is the planning phase outlined in section 4. For this architecture, Red Hat* Ceph* storage is being used. Ceph is a distributed storage solution that supports object, block, and file system storage. It is a flexible and powerful storage solution. Its foundation is RADOS (Reliable Autonomic Distributed Object Store).

Ceph can integrate with S3/Swift, Hypervisor, iSCSI, CIFS, NFS, and more with interfaces to support object, block, and file system storage types as shown in Figure 9.



Source: http://ceph.com/planet/zero-to-hero-guide-for-ceph-cluster-planning/

Figure 9. Overview of Ceph* storage

Two key components are Ceph monitors and Ceph Object Storage Daemons (OSDs).

Monitors are the guardians of the master copy of the cluster map. In order for a Ceph client to read or write data, it must contact a Ceph monitor to obtain the current and latest cluster map. A Ceph cluster can operate with a single Ceph monitor, but it becomes a single point of failure and, without it, Ceph does not work. The recommendation is to have a minimum of three Ceph monitors. Best practices recommend an odd number to have a better consensus/quorum and to guarantee the cluster state consistency among all monitors.

Ceph OSDs are a key element in charge of data storage, replication, recovery, backfilling, and rebalancing. They also provide monitoring information to Ceph monitors. Despite the minimum requirement to have at least two Ceph OSDs, a minimum of three is strongly recommended to have minimum failure tolerance. By default, a Ceph cluster keeps three replicas of each object in different OSDs. Ceph OSDs typically correspond to file systems on physical disks, and one OSD per hard disk is generally a good fit. More OSDs per hard disk must be carefully analyzed, including partitions and sizes, to avoid impacting overall Ceph availability, reliability, and performance.

Some basic rules of thumb and recommendations are below.

- Run OSDs on a dedicated storage node (server with multiple disks); actual data is stored in the form of objects.
- Multiple OSDs per disk make sense in SSD solutions. In HDD solutions, this configuration leads to increased contention, increased latency, and reduced performance.
- For each terabyte of storage space, the node should have 1GB of RAM for rebalancing, backfilling, and recovery.
- OSD journals should run on separate disks, minimally on SATA SSDs and ideally on NVMe* SSDs.
- Run OSD monitors on separate low-cost dedicated hardware, since they are not resource intensive. They may run concurrently with other processes such as Ceph client.

Three key concepts of CEPH are pools, CRUSH, and Placement Groups (PGs).

Pools are logical partitions to store objects. They determine the number of object replicas and the number of placement groups (PGs) in the pool. You can either replicate or use erasure coded in pools. This option will define the desired reliability and fault tolerance and what makes sense from an application and cost model perspective.

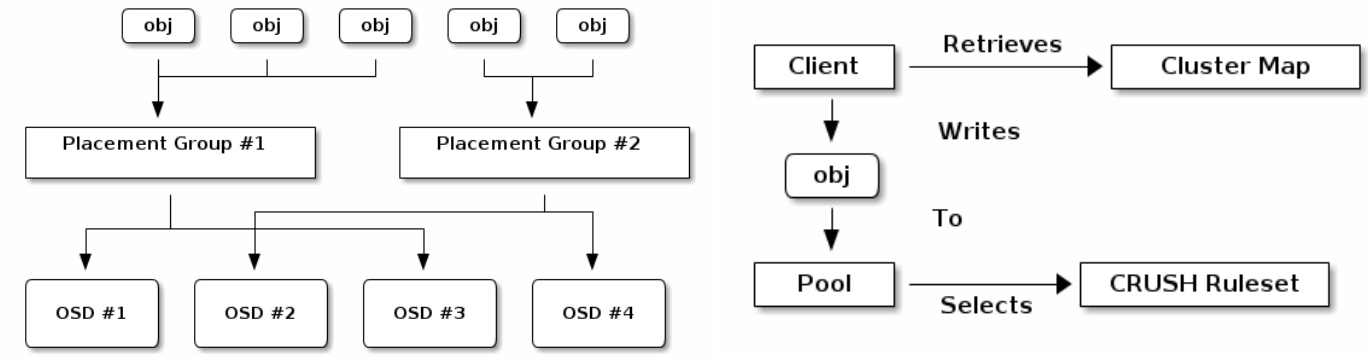
The **CRUSH** algorithm determines how to store and retrieve data by computing data storage locations. The list of OSDs, list of rules (ruleset), and list of "potential" physical device aggregations are contained in CRUSH maps. CRUSH knows the underlying physical layer and, therefore, can define the best strategy on where to place data to avoid device failures, for example. All this information is encoded in the cluster map, and CRUSH placement algorithm leverages that to replicate it across multiple and different failure domains while keeping a balanced data distribution.

Placement Groups (PGs) are the indirection layers that address object mapping to OSDs, and thus to physical disks. This helps reduce the tracking of per-object metadata, which would make the system unusable shortly after deployment. The number of PGs can reduce the per-OSD load on the cluster, but it costs more CPU cycles as well. Some initial recommended numbers are:

- Less than 5 OSDs, set `pg_num` to 128
- Between 5 and 10 OSDs, set `pg_num` to 512
- Between 10 and 50 OSDs, set `pg_num` to 1024

It is recommended to use the placement group calculator available at http://ceph.com/pgcalc/ or refer to the Red Hat Ceph Storage Strategies Guide.

Figure 10 illustrates the roles of PGs, CRUSH and pools in Ceph storage.



Sources: http://docs.ceph.com/docs/giant/architecture/ and https://access.redhat.com/documentation/en/red-hat-ceph-storage/1.3/paged/architecture-guide/.

Figure 10. Roles of pools, CRUSH, and placement groups in Ceph storage

7. Deploying the Solution

7.1 Installing the Undercloud

7.1.1 Prerequisites

The first step to create a Red Hat OpenStack Platform environment is to install the director on the undercloud system. The installation requires a minimal installation of Red Hat Enterprise Linux 7 on a physical system and registered system via Red Hat* Subscription Manager with an active OpenStack subscription.

Register via the Red Hat Subscription Manager:

```
# subscription-manager register
Registering to: subscription.rhn.redhat.com:443/subscription
# Username:
# Password:
The system has been registered with ID: xxxxxxx-xxxxx-xxxx-xxxx-xxxxxxxxxxxx
# subscription-manager attach --auto
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux High Availability (for RHEL Server)
Status:       Subscribed
Product Name: Red Hat Enterprise Linux Server
Status:       Subscribed
```

7.1.2 Install the director user

The director installation process requires a non-root user. Use command below to create a new user:

```
[root@director ~]# useradd stack
[root@director ~]# passwd stack  # specify a password
```

Disable password requirements for this user when using sudo:

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

Switch to the new stack user:

```
[root@director ~]# su - stack
[stack@director ~]$
```

Continue the director installation as the stack user.

7.1.3 Create directories for templates and images

The director uses system images and heat templates to create the overcloud environment. According to Red Hat recommendations, create separate folders for images and templates to keep files organized:

```
$ mkdir ~/images
$ mkdir ~/templates
```

7.1.4 Set the system hostname

The director requires a fully qualified domain name for its installation and configuration process. Check the hostname of your host:

```
$ hostname    # Checks the base hostname
$ hostname -f # Checks the long hostname (FQDN)
```

Use `hostnamectl` to set a hostname if required:

```
$ sudo hostnamectl set-hostname manager.example.com
$ sudo hostnamectl set-hostname --transient manager.example.com
```

The director also requires an entry for the system's hostname and base name in `/etc/hosts`. For example, if the system is named `manager.example.com`, then `/etc/hosts` requires an entry such as:

```
127.0.0.1   manager.example.com manager localhost localhost.localdomain
localhost4 localhost4.localdomain4
```

7.1.5 Install director packages

Disable all default repositories, and then enable the required Red Hat Enterprise Linux repositories:

```
$ sudo subscription-manager repos --disable=*
```

These repositories contain packages the director installation requires.

```
$ sudo subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-
7-server-extras-rpms --enable=rhel-7-server-openstack-9-rpms --enable=rhel-
7-server-openstack-9-director-rpms --enable=rhel-7-server-rh-common-rpms
```

Perform an update on your system to make sure you have the latest base system packages:

```
$ sudo yum update -y
$ sudo reboot
```

Use the following command to install the required command line tools for director installation and configuration:

```
$ sudo yum install -y python-tripleoclient
```

7.1.6 Configure the director

The director installation process requires certain settings to determine network configurations. The settings are stored in a template located in the stack user's home directory as `undercloud.conf`. A copy of this configuration file can be found in Appendix B.

Red Hat provides a basic template to help determine the required settings for your installation. Copy this template to the stack user's home directory:

```
$ cp /usr/share/instack-undercloud/undercloud.conf.sample ~/undercloud.conf
```

Customize the template with your specific parameters such as `local_ip`, `network_gateway`, `undercloud_public_vip`, `undercloud_admin_vip`, `undercloud_service_certificate`, `local_interface`, `network_cidr` and others. For details on these parameters, refer to section B.1.

To prepare a certificate file to use for OpenStack service SSL connections, first create key and certificate pair to act as the certificate authority:

```
$ openssl genrsa -out ca.key.pem 4096
$ openssl req  -key ca.key.pem -new -x509 -days 7300 -extensions v3_ca -out
ca.crt.pem
```

This creates a certificate authority (CA) file called `ca.crt.pem`.

Run the following commands to generate the SSL/TLS key (`server.key.pem`), which is used at different points to generate the undercloud or overcloud certificates:

```
$ openssl genrsa -out server.key.pem 2048
```

Create a certificate signing request for either the undercloud or overcloud:

```
Copy the default OpenSSL configuration file for customization.
$ cp /etc/pki/tls/openssl.cnf .
```

An example of the types of parameters to modify include:

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req

[req_distinguished_name]
countryName = Country Name (2 letter code)
countryName_default = PL
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Intel
localityName = Locality Name (eg, city)
localityName_default = Gdansk
organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = DCG
commonName = Common Name
commonName_default = 192.0.2.2
commonName_max = 64

[ v3_req ]
# Extensions to add to a certificate request
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names


[alt_names]
IP.1 = 192.0.2.2
DNS.1 = 192.0.2.2
DNS.2 = manager.example.com
DNS.3 = 192.0.2.3
```

To initialize the serial file and the `index.txt` file in Linux*, enter the following command:

```
cd /etc/pki/CA/
$ echo "01" > serial
$ touch index.txt
```

These files are used by the CA to maintain its database of certificate files.

The `index.txt` file initially must be completely empty, not even containing white space.

Run the following command to generate certificate signing request (`server.csr.pem`):

```
$ openssl req -config openssl.cnf -key server.key.pem -new -out
server.csr.pem
```

Then use the `server.csr.pem` file to create the SSL/TLS certificate for your undercloud or overcloud:

```
$ openssl ca -config openssl.cnf -extensions v3_req -days 3650 -in
server.csr.pem -out server.crt.pem -cert ca.crt.pem
```

This results in a certificate named `server.crt.pem`. Use this certificate with SSL/TLS key to enable SSL/TLS on the undercloud. Run the following command to combine the certificate and key:

```
$ cat server.crt.pem server.key.pem > undercloud.pem
```

This creates an `undercloud.pem` file. Specify the location of this file in the `undercloud_service_certificate` option in the `undercloud.conf` file.

This file also requires a special SELinux context so that the HAProxy* tool can read it. Use the following example as a guide:

```
$ sudo mkdir /etc/pki/instack-certs
$ sudo cp ~/undercloud.pem /etc/pki/instack-certs/.
$ sudo semanage fcontext -a -t etc_t "/etc/pki/instack-certs(/.*)?"
$ sudo restorecon -R /etc/pki/instack-certs
```

In addition, make sure to add your certificate authority to the undercloud's list of trusted certificate authorities so that different services within the undercloud have access to the certificate authority:

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust extract
```

Next, customize the `undercloud.conf` file with the values for these parameters to suit your network, such as:

```
local_ip = 192.0.2.1/24
network_gateway = 192.0.2.1
undercloud_public_vip = 192.0.2.2
undercloud_admin_vip = 192.0.2.3
local_interface = ens802f0
network_cidr = 192.0.2.0/24
masquerade_network = 10.22.110.0/24
dhcp_start = 192.0.2.5
dhcp_end = 192.0.2.24
inspection_interface = br-ctlplane
inspection_iprange = 192.0.2.100,192.0.2.120
```

When complete, save the file and run the following command:

```
$ openstack undercloud install
```

This launches the director's configuration script. The director installs additional packages and configures its services to suit the settings in the `undercloud.conf` file. When complete, two files are generated: `undercloud-passwords.conf` and `stackrc`. To initialize the stack user to use the command line tools, run the following command:

```
$ source ~/stackrc
```

7.1.7 Tuning the undercloud

As documented in the Red Hat OpenStack Platform Director Installation and Usage Guide (https://access.redhat.com/documentation/en/red-hat-openstack-platform/9/paged/director-installation-and-usage/chapter-10-troubleshooting-director-issues), the `openstack-heat-engine` and `openstack-heat-api` services might consume too many resources at times. If so, set `max_resources_per_stack=-1` in `/etc/heat/heat.conf` and restart the heat services:

```
$ sudo systemctl restart openstack-heat-engine openstack-heat-api
```

Sometimes the director might not have enough resources to perform concurrent node provisioning. The default is 10 nodes at the same time. To reduce the number of concurrent nodes, set the

`max_concurrent_builds` parameter in `/etc/nova/nova.conf` to a value less than 10 and restart the nova services:

```
$ sudo systemctl restart openstack-nova-api openstack-nova-scheduler
```

Edit the `/etc/my.cnf.d/server.cnf` file. Some recommended values to tune include:

- `max_connections = 4096` – Number of simultaneous connections to the database. The default is usually 8M and an ideal value is 20M for the undercloud.
- `innodb_additional_mem_pool_size = 20` – The size in bytes of a memory pool the database uses to store data dictionary information and other internal data structures. The default is usually 8M and an ideal value is 20M for the undercloud.
- `innodb_buffer_pool_size = 128M` – The size in bytes of the buffer pool, the memory area where the database caches table and index data. The default is usually 128M and an ideal value is 1000M for the undercloud.
- `innodb_flush_log_at_trx_commit = 1` – Controls the balance between strict ACID compliance for commit operations and higher performance that is possible when commit-related I/O operations are rearranged and done in batches. Set to 1.
- `innodb_lock_wait_timeout = 50` – The length of time in seconds that a database transaction waits for a row lock before giving up. Set to 50.
- `innodb_max_purge_lag = 10000` – This variable controls how to delay `INSERT`, `UPDATE`, and `DELETE` operations when purge operations are lagging. Set to 10000.
- `innodb_thread_concurrency = 0` – The limit of concurrent operating system threads. Ideally, provide at least two threads for each CPU and disk resource. For example, if using a quad-core CPU and a single disk, use ten threads. Zero means infinite concurrency. It allows the InnoDB* storage engine to decide the best number of concurrency workers to launch and address.

Ensure that heat has enough workers to perform an overcloud creation. Usually, this depends on how many CPUs the undercloud has. To manually set the number of workers, edit the `/etc/heat/heat.conf` file, set the `num_engine_workers` parameter to the number of workers you need (ideally four), and restart the heat engine:

```
$ sudo systemctl restart openstack-heat-engine
$ sudo systemctl restart mariadb
```

7.1.8 Obtaining overcloud node images

The director requires several disk images for provisioning overcloud nodes. All needed images can be obtained from the `rhosp-director-images` and `rhosp-director-images-ipa` packages:

```
$ sudo yum install rhosp-director-images rhosp-director-images-ipa
```

Extract the archives to the images directory on the stack user's home (`/home/stack/images`):

```
$ cd ~/images
$ for i in /usr/share/rhosp-director-images/overcloud-full-latest-9.0.tar
/usr/share/rhosp-director-images/ironic-python-agent-latest-9.0.tar; do tar
-xvf $i; done
```

Import these images into the director:

```
$ openstack overcloud image upload --image-path /home/stack/images/
```

This uploads the following images into the undercloud: `bm-deploy-kernel`, `bm-deploy-ramdisk`, `overcloud-full`, `overcloud-full-initrd`, `overcloud-full-vmlinuz`. These are the images for deployment and the overcloud. The script also installs the introspection images on the director's PXE server.

View a list of the images in the CLI:

```
[stack@director ~]$ openstack image list
+--------------------------------------+------------------------+--------+
| ID                                   | Name                   | Status |
+--------------------------------------+------------------------+--------+
| 507ab208-e862-4644-8125-9bfdb22d25ad | bm-deploy-ramdisk      | active |
| 3bedb051-f933-41b6-9454-69f7dc40834f | bm-deploy-kernel       | active |
| 27ec6a8c-7070-40ec-b2b8-dc063dda304a | overcloud-full         | active |
| e0f63d47-9024-40ea-b6e7-2cbdf44ef1e5 | overcloud-full-initrd  | active |
| e2af929f-a337-44fe-bbdc-5d2d78044c9c | overcloud-full-vmlinuz | active |
+--------------------------------------+------------------------+--------+
```

The list will not show the introspection PXE images. The director copies these files to `/httpboot`.

```
[stack@director ~]$ ls -l /httpboot
total 442112
-rwxr-xr-x. 1 ironic ironic   5157696 Sep 27 16:49 agent.kernel
-rw-r--r--. 1 ironic ironic 447551200 Sep 27 16:49 agent.ramdisk
-rw-r--r--. 1 ironic ironic       759 Oct 31 10:56 boot.ipxe
-rw-r--r--. 1 ironic ironic       428 Sep 27 16:43 inspector.ipxe
drwxr-xr-x. 2 ironic ironic         6 Oct 31 11:01 pxelinux.cfg
```

7.1.9 Setting a nameserver on the undercloud's neutron subnet

Overcloud nodes require a nameserver so that they can resolve hostnames through DNS. For a standard overcloud without network isolation, the nameserver is defined using the undercloud's neutron subnet. Use the following commands to define the nameserver for the environment:

```
$ neutron subnet-list
$ neutron subnet-update [subnet-uuid] --dns-nameserver [nameserver-ip]
```

7.1.10 Verify the undercloud

To perform the undercloud installation checks, complete the following steps:

Check `/etc/resolv.conf`:

```
[stack@director ~]$ cat /etc/resolv.conf
# Generated by NetworkManager
search pcsd.local
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Check the control plane bridge:

```
[stack@director ~]$ ip a
10: br-ctlplane: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UNKNOWN
    link/ether 00:1e:67:e2:52:7c brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global br-ctlplane
       valid_lft forever preferred_lft forever
    inet 192.0.2.3/32 scope global br-ctlplane
       valid_lft forever preferred_lft forever
    inet 192.0.2.2/32 scope global br-ctlplane
       valid_lft forever preferred_lft forever
    inet6 fe80::21e:67ff:fee2:527c/64 scope link
       valid_lft forever preferred_lft forever
```

A new bridge `br-ctlplane` should have been created as part of the undercloud install on the PXE interface as shown below. Validate MAC and IP's.

```
[stack@director ~]$ ifconfig br-ctlplane
br-ctlplane: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.0.2.1  netmask 255.255.255.0  broadcast 192.0.2.255
        inet6 fe80::21e:67ff:fee2:527c  prefixlen 64  scopeid 0x20<link>
        ether 00:1e:67:e2:52:7c  txqueuelen 0  (Ethernet)
        RX packets 14400305  bytes 4028842922 (3.7 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11761870  bytes 159296193973 (148.3 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[stack@director ~]$ neutron net-list
+--------------------------------------+----------+--------------------------------------+
| id                                   | name     | subnets                              |
+--------------------------------------+----------+--------------------------------------+
| d751c2d4-8188-4bbd-ac5c-886ca33b88b9 | ctlplane | 37286e10-ad3f-4f3f-9516-acffc0c272cc
|
|                                      |          | 192.0.2.0/24                         |
+--------------------------------------+----------+--------------------------------------+
```

7.2 Installing the Overcloud

7.2.1 Create a node definition template and register blank nodes in the director

Set roles of bare metal machines for your nodes. This reference architecture uses:

- (3) controller nodes
- (5) compute nodes
- (3) storage nodes

The number of nodes required depends on the type of overcloud you intend to create. These machines also must comply with the requirements set for each node type. For these requirements, please refer to the Red Hat OpenStack Platform Director Installation and Usage Guide (https://access.redhat.com/documentation/en/red-hat-openstack-platform/9/paged/director-installation-and-usage/chapter-2-requirements#sect-Overcloud_Requirements). These nodes do not require an operating system. The director deploys a Red Hat Enterprise Linux 7 image to each node.

To prepare a node definition template, you need the MAC addresses of the IPMI NIC, IPMI credentials, the IP address of IPMI, and not necessarily the node capabilities.

Table 11. Provisioning network IP

Node Name	Interface / IP Adress	MAC Adress	IPMI Addres
Director	ens802f0 / 192.0.2.1	aa:aa:aa:aa:aa:aa	192.168.42.22
Controller	ens802f0 / DHCP defined	xx:xx:xx:xx:xx:11	192.168.42.xx
Compute	ens802f0 / DHCP defined	xx:xx:xx:xx:xx:22	192.168.42.xx
Storage	ens802f0 / DHCP defined	xx:xx:xx:xx:xx:33	192.168.42.xx

Note: By using the provisioning interface on the director node and the `local_ip` and `network_gateway` parameters, it configures the system to act as the gateway for all the nodes.

The director requires a node definition template. This file (`instackenv.json`) uses the JSON format file, and contains the hardware and power management details for your nodes. Short version with assigned profiles for a couple of nodes looks like :

```
{
    "nodes":[
        {
            "mac":[
                "00:1e:67:e2:52:74"
            ],
            "pm_type":"pxe_ipmitool",
            "pm_user":"root",
            "pm_password":"strange_password",
            "pm_addr":"192.168.42.30",
            "capabilities": "profile:control,boot_option:local"
        },
        {
            "mac":[
                "00:1e:67:d1:8b:d0"
            ],
            "pm_type":"pxe_ipmitool",
            "pm_user":"root",
            "pm_password":"strange_password",
            "pm_addr":"192.168.42.32",
            "capabilities": "profile:control,boot_option:local"
        },
   ...
}
```

After creating the template, save the file to the stack user's home directory (`/home/stack/instackenv.json`), then import it into the director using the following command:

```
$ openstack baremetal import --json ~/instackenv.json
```

This imports the template and registers each node from the template into the director.

Assign the kernel and ramdisk images to all nodes:

```
$ openstack baremetal configure boot
```

The nodes are now registered and configured in the director. View a list of these nodes in the CLI:

```
$ ironic node-list
```

The results should show all nodes in available state:

```
[stack@director ~]$ ironic node-list
+--------------------------------------+------+----------+-------------+-------------+
| UUID                                 | Name | Instance | Power State | Provisioning|
|                                      |      |   UUID   |             |    State    |
+--------------------------------------+------+----------+-------------+-------------+
| ebe49216-b725-4221-a115-a2f799cd0a3e | None |          | power off   | available   |
| 96952265-672f-44ac-a7ad-3e021be7e401 | None |          | power off   | available   |
| ef94aeea-af61-4951-a128-bba5bd7a925e | None |          | power off   | available   |
| b2bdb068-ac0d-43e7-a8f1-59dad4f93839 | None |          | power off   | available   |
| f39bab79-c470-4290-a889-aabaf5b70773 | None |          | power off   | available   |
| 50aa713c-613e-4aac-a9f9-3115e3af0325 | None |          | power off   | available   |
| 2607dfe9-8c35-460e-9804-ff54fa4b7c57 | None |          | power off   | available   |
| 9511f044-a5e6-41fa-ac6d-dd48a841b7c0 | None |          | power off   | available   |
| fdb93df7-7811-40d6-b009-ae8d9b45b802 | None |          | power off   | available   |
| 9419e1ca-9a2e-40c2-bb70-a0634a9cee2e | None |          | power off   | available   |
| cb18332a-dc6b-4e88-a847-615ce0c94d80 | None |          | power off   | available   |
+--------------------------------------+------+----------+-------------+-------------+
```

7.2.2 Inspect hardware of all nodes

Run the following command to inspect the hardware attributes of each node:

```
$ openstack baremetal introspection bulk start
```

Monitor the progress of the introspection using the following command in a separate terminal window:

```
$ sudo journalctl -l -u openstack-ironic-inspector -u openstack-ironic-
inspector-dnsmasq -u openstack-ironic-conductor –f
```

During introspection each node boots an introspection agent over PXE, which collects all hardware data and sends it back to the director. Depending on the amount of nodes, it takes around 15 minutes for all 11 nodes.

Check if all nodes have successfully finished introspection in bulk mode:

```
$ openstack baremetal introspection bulk status
```

Every node should have `True` in the finished field. The error field contains an error message if introspection failed, or `None` if introspection succeeded for this node. If the introspection does not finish in one hour, the result times out. In most cases this happens due to a misconfiguration or BIOS settings. Check `/var/log/ironic/*` files to understand and fix any issues.

To check IPMI connectivity, simply perform check status:

```
ipmitool -I lanplus -H <ipmi address> -U admin -P <password> chassis power
status
```

After resolving all issues, retry introspection for selected node in three steps:

1. Put ironic node in manage state:

   ```
   $ ironic node-set-provision-state [NODE UUID] manage
   ```

2. Perform a new introspection:

   ```
   $ openstack baremetal introspection start [NODE UUID]
   ```

3. When introspection is finished, change status back to available:

   ```
   $ ironic node-set-provision-state [NODE UUID] provide
   ```

Sometimes when inspection hangs and times out, ironic does not clean temporary images for introspection, which prevents clean boot for new introspection. The temporary images (grayed) can be found in `httpboot` director:

```
$ [root@director ~]# ll /httpboot/
total 442112
drwxr-xr-x. 2 ironic ironic        60 Oct  9 21:36 19bb9c35-6f0e-40cf-acb1-
1f43ef94d1ff
drwxr-xr-x. 2 ironic ironic        60 Oct 17 14:52 2a2f5d96-a9f7-4825-b287-
5e05f881a10b
drwxr-xr-x. 2 ironic ironic        60 Oct  9 21:36 54613069-e2e8-43aa-88b4-
0635574de11f
-rwxr-xr-x. 1 root   root     5157696 Sep 27 16:49 agent.kernel
-rw-r--r--. 1 root   root   447551200 Sep 27 16:49 agent.ramdisk
-rw-r--r--. 1 ironic ironic       759 Oct 17 14:52 boot.ipxe
-rw-r--r--. 1 root   root         428 Sep 27 16:43 inspector.ipxe
drwxr-xr-x. 2 ironic ironic        92 Oct 17 16:40 pxelinux.cfg
```

The workaround for this issue is to delete them manually.

To fully clean all discovered nodes by ironic, remove the file `discovered.sqlite`:

```
$ sudo rm /var/lib/ironic-discoverd/discoverd.sqlite
$ ls -al /var/lib/ironic-discoverd/discoverd.sqlite
sudo systemctl restart openstack-ironic-discoverd
sudo systemctl status openstack-ironic-discovered
```

7.2.3 Tag nodes into roles

After registering and inspecting the hardware of each node, tag them into specific profiles. These profile tags match nodes to flavors, and in turn, the flavors are assigned to a deployment role. Default profile flavors are `compute`, `control`, `swift-storage`, `ceph-storage`, and `block-storage`.

To tag a node into a specific profile, add a profile option to the properties/capabilities parameter for each node. For example, to tag nodes to use controller and compute profiles, respectively, use the following commands:

```
$ ironic node-update 58c3d07e-24f2-48a7-bbb6-6843f0e8ee13 add
properties/capabilities='profile:compute,boot_option:local'
$ ironic node-update 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 add
properties/capabilities='profile:control,boot_option:local'
```

Note: In this reference architecture these commands are not needed because these tags are included in the JSON file as explained below.

In this setup, the flavors are assigned in the `instackenv.json` file (grayed section).

```
{
            "mac":[
                "00:1e:67:d1:8b:d0"
            ],
            "pm_type":"pxe_ipmitool",
            "pm_user":"root",
            "pm_password":"strange_password",
            "pm_addr":"192.168.42.32",
            "capabilities": "profile:control,boot_option:local"
```

```
    },
```

Check currently assigned profiles:

```
[stack@director ~]$ openstack overcloud profiles list
+--------------------------------------+-----------+-----------------+-----------------+
| Node UUID                            | Node Name | Provision State | Current Profile |
+--------------------------------------+-----------+-----------------+-----------------+
| ebe49216-b725-4221-a115-a2f799cd0a3e |           | active          | control         |
| 96952265-672f-44ac-a7ad-3e021be7e401 |           | active          | control         |
| ef94aeea-af61-4951-a128-bba5bd7a925e |           | active          | control         |
| b2bdb068-ac0d-43e7-a8f1-59dad4f93839 |           | active          | compute         |
| f39bab79-c470-4290-a889-aabaf5b70773 |           | active          | compute         |
| 50aa713c-613e-4aac-a9f9-3115e3af0325 |           | active          | compute         |
| 2607dfe9-8c35-460e-9804-ff54fa4b7c57 |           | active          | compute         |
| 9511f044-a5e6-41fa-ac6d-dd48a841b7c0 |           | active          | compute         |
| fdb93df7-7811-40d6-b009-ae8d9b45b802 |           | active          | ceph-storage    |
| 9419e1ca-9a2e-40c2-bb70-a0634a9cee2e |           | active          | ceph-storage    |
| cb18332a-dc6b-4e88-a847-615ce0c94d80 |           | active          | ceph-storage    |
+--------------------------------------+-----------+-----------------+-----------------+
```

7.2.4 Define additional node properties

The director needs to identify the root disk during provisioning. This can be provided by serial disk number or disk size among other possible indicators. For example, identify the root disk by disk size using the following command:

```
ironic node-update server1 add properties/root_device='{"size": 465}'
```

As detailed in the Red Hat OpenStack Platform Director Installation and Usage Guide (https://access.redhat.com/documentation/en/red-hat-openstack-platform/9/paged/director-installation-and-usage/chapter-5-configuring-basic-overcloud-requirements), this information can be gathered after node introspection.

First, collect a copy of each node's hardware information that the director obtained from the introspection. This information is stored in the OpenStack Object Storage server (swift). Download this information to a new directory:

```
$ cd ~/
$ mkdir swift-data
$ cd swift-data
$ export IRONIC_DISCOVERD_PASSWORD=`sudo grep admin_password /etc/ironic-
inspector/inspector.conf | awk '! /^#/ {print $NF}' | awk -F'=' '{print
$2}'`
$ for node in $(ironic node-list | awk '!/UUID/ {print $2}'); do swift -U
service:ironic -K $IRONIC_DISCOVERD_PASSWORD download ironic-inspector
inspector_data-$node; done
```

This downloads the data from each `inspector_data` object from introspection. All objects use the node UUID as part of the object name:

```
[stack@director swift-data]$ ls -1
inspector_data-2607dfe9-8c35-460e-9804-ff54fa4b7c57
inspector_data-50aa713c-613e-4aac-a9f9-3115e3af0325
inspector_data-9419e1ca-9a2e-40c2-bb70-a0634a9cee2e
inspector_data-9511f044-a5e6-41fa-ac6d-dd48a841b7c0
inspector_data-96952265-672f-44ac-a7ad-3e021be7e401
inspector_data-b2bdb068-ac0d-43e7-a8f1-59dad4f93839
inspector_data-cb18332a-dc6b-4e88-a847-615ce0c94d80
inspector_data-ebe49216-b725-4221-a115-a2f799cd0a3e
```

```
inspector_data-ef94aeea-af61-4951-a128-bba5bd7a925e
inspector_data-f39bab79-c470-4290-a889-aabaf5b70773
inspector_data-fdb93df7-7811-40d6-b009-ae8d9b45b802
```

Check the disk information for each node. The following command displays each node ID and the disk information:

```
$ for node in $(ironic node-list | awk '!/UUID/ {print $2}'); do echo "NODE:
$node" ; cat inspector_data-$node | jq '.inventory.disks' ; echo "-----" ;
done
```

For example, the data for one node might show three disks:

```
NODE: cb18332a-dc6b-4e88-a847-615ce0c94d80
[
  {
    "size": 800166076416,
    "rotational": false,
    "vendor": "ATA",
    "name": "/dev/sda",
    "wwn_vendor_extension": null,
    "wwn_with_extension": "0x55cd2e404c20e513",
    "model": "INTEL SSDSC2BA80",
    "wwn": "0x55cd2e404c20e513",
    "serial": "BTHV6155092V800OGN"
  },
  {
    "size": 800166076416,
    "rotational": false,
    "vendor": "ATA",
    "name": "/dev/sdb",
    "wwn_vendor_extension": null,
    "wwn_with_extension": "0x55cd2e404c20e516",
    "model": "INTEL SSDSC2BA80",
    "wwn": "0x55cd2e404c20e516",
    "serial": "BTHV6155092X800OGN"
  },
  {
    "size": 1600321314816,
    "rotational": false,
    "vendor": null,
    "name": "/dev/nvme0n1",
    "wwn_vendor_extension": null,
    "wwn_with_extension": null,
    "model": "",
    "wwn": null,
    "serial": null
  }
]
```

Note: BIOS configuration of each node needs to indicate the selected root disk as booting disk.

To update node information with root disk serial number, perform the following command for all nodes:

```
ironic node-update ef94aeea-af61-4951-a128-bba5bd7a925e add
properties/root_device='{"serial": "CVLXXXXXXXXXXX"}'
```

Note: Because of a bug in cleaning ironic nodes (https://bugzilla.redhat.com/show_bug.cgi?id=1344004), do not switch `automated_clean` to true on in `/etc/ironic/ironic.conf`. At the time of the reference architecture deployment, this bug was still open, but Red Hat recently published an errata about it. Please refer to https://rhn.redhat.com/errata/RHEA-2016-2948.html for more information.

7.2.5 Setting up the overcloud

Before setup, it is necessary to understand and change the heat templates to fit the environment.

The purpose of a template, mostly in yaml format, is to define and create a "stack" which is a collection of resources and their configurations. Resources are objects in OpenStack and can include compute, network configuration, scaling rules, and others.

The director contains a core heat template collection for the overcloud. This collection is stored in `/usr/share/openstack-tripleo-heat-templates`. For network resources, for example, simply copy one of the following templates depending on the network configuration:

- `/usr/share/openstack-tripleo-heat-templates/network/config/single-nic-vlans` - Directory containing templates for single NIC with VLANs configuration on a per role basis.
- `/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans` - Directory containing templates for bonded NIC configuration on a per role basis.
- `/usr/share/openstack-tripleo-heat-templates/network/config/multiple-nics` - Directory containing templates for multiple NIC configuration using one NIC per role.
- `/usr/share/openstack-tripleo-heat-templates/network/config/single-nic-linux-bridge-vlans` - Directory containing templates for single NIC with VLANs configuration on a per role basis and using a Linux bridge instead of an Open vSwitch bridge.

This set up uses the `bond-with-vlans` configuration. Copy the version located at `/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans`.

```
$ cp -r /usr/share/openstack-tripleo-heat-templates/network/config/bond-
with-vlans ~/templates/nic-configs
```

Table 12. Customized Resource and Particular Order

Resource	Environment File	Resource Registry	Heat Template
Network	net-bond-with-vlans.yaml	OS::TripleO::Compute::Net::SoftwareConfig	compute.yaml
		OS::TripleO::Controller::Net::SoftwareConfig	controller.yaml
		OS::TripleO::CephStorage::Net::SoftwareConfig	ceph-storage.yaml
Storage	storage-environment.yaml	OS::TripleO::NodeUserData:	firstboot/wipe-disks.yaml
Custom	environment-rhel-registration.yaml	OS::TripleO::NodeExtraConfig:	rhel-registration.yaml rhel-registration-resource-registry.yaml

All templates described in the following sections are included in 53.

7.2.5.1 Configure network isolation (network environment file `net-bond-with-vlans.yaml`)

The network environment file `net-bond-with-vlans.yaml`, in this case, describes the network environment and includes the network interface configuration to use for overcloud nodes (`compute.yaml`, `controller.yaml`, `ceph-storage.yaml`) in the `resource_registry` section.

`Parameters_defaults` contains options for each network such as network CIDR, network pools, and VLANs. The control plane default route is the gateway router for the provisioning network and the undercloud IP. This matches with the `network_gateway` and `masquerade_network` in the `undercloud.conf` file. EC2Metadata IP is the undercloud IP.

According to an important notice in Red Hat documentation about bonding options, do not use LACP with OVS-based bonds. This configuration is problematic and unsupported due to Bug 1267291. An alternative to this is to use LACP with standard Linux kernel mode bonds and OVS on top. Please refer to https://github.com/novacain1/redhat-sepialab/blob/master/templates/nic-configs/compute.yaml#L101-L168 for more details.

For bonds options, we use `balance-slb` configured on a switch without LACP support.

```
bond_mode=balance-slb
```

The following is an example configuration for two interfaces, Eth 15 and 16, bond in one port channel on a switch:

```
Leaf-D2(config-if-Po16)#sh active
interface Port-Channel16
description Port-Channel 16 PCSD Et 15, 16
switchport trunk allowed vlan 203
switchport mode trunk
spanning-tree portfast
```

It is also recommended to add an optional parameter to change the time in milliseconds between rebalancing flows between bond members when OVS bonds are used.

```
other_config:bond-rebalance-interval=10000
```

All values need to be customized to match the local environment. For a full reference of these options, refer to the Network Environment Options and Open vswitch bonding options sections of the Red Hat OpenStack Platform Director Installation and Usage Guide.

7.2.5.2 Configure OpenStack components network interface configuration (NIC) for controller, compute, and Ceph* storage nodes (`compute.yaml`, `controller.yaml`, `ceph-storage.yaml`)

The controller node is configured as shown in Figure 11.

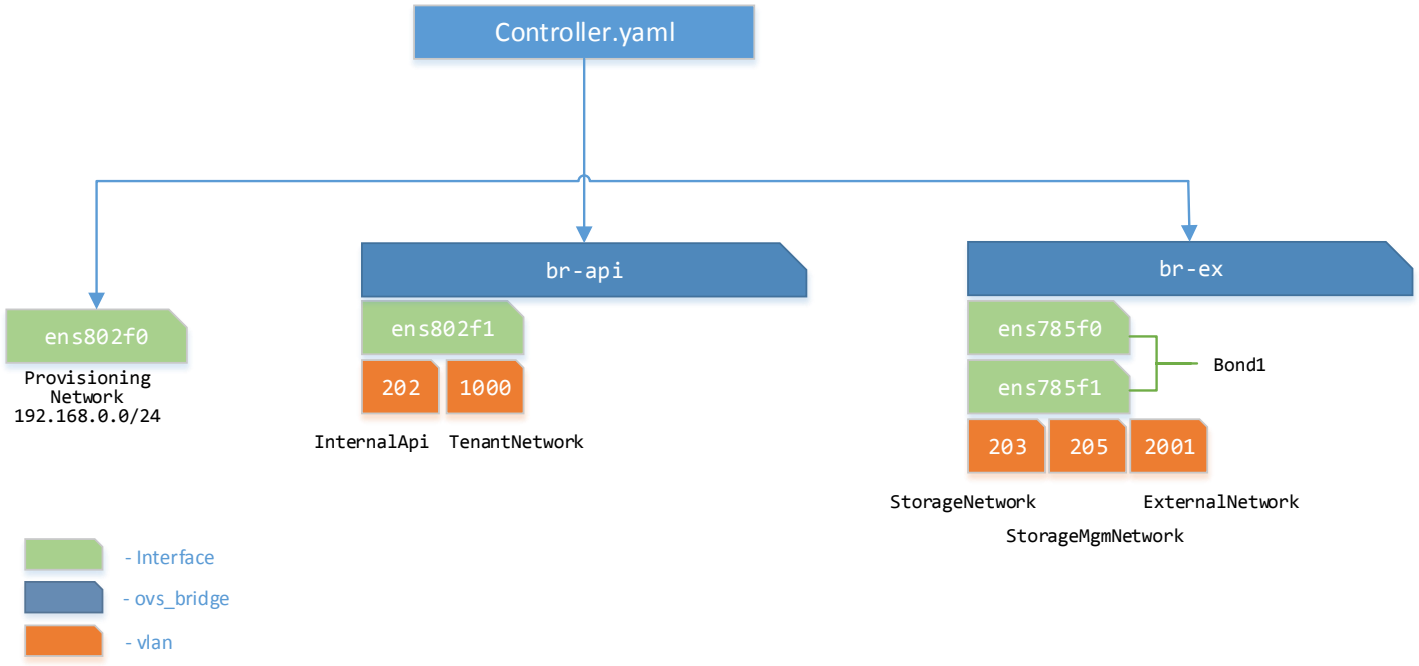


Figure 11. Network interface configuration for controller node

Table 13. Parameters for `controller.yaml`

ControlPlaneIP/EC2MetadataIP	**192.0.2.1**		
ExternalIPSubnet	100.64.0.0/20	ExternalNetworkVlanID	2001
InternalApiSubnet	192.168.0.0/24	InternalApiNetworkVlanID	202
StorageIpSubnet	192.168.1.0/24	StorageNetworkVlanID	203
StorageMgmtIpSubnet	192.168.2.0/24	StorageMgmtNetworkVlanID	205
TenantIPSubnet	172.16.0.0/16	TenantNetworkVlanID	1000

The default route for the external network is set to `100.64.0.1` and the static route for the nova metadata service of undercloud is set to `EC2Metadatalp`.

In the controller heat template, two OVS bridges are defined:

- `br-api` for `InternalApi` and `TenantNetwork`
- `br- ex` for `ExternalNetwork`, `StorageNetwork` and `StorageMgmtNetwork`

The compute node is configured as shown in Figure 12.

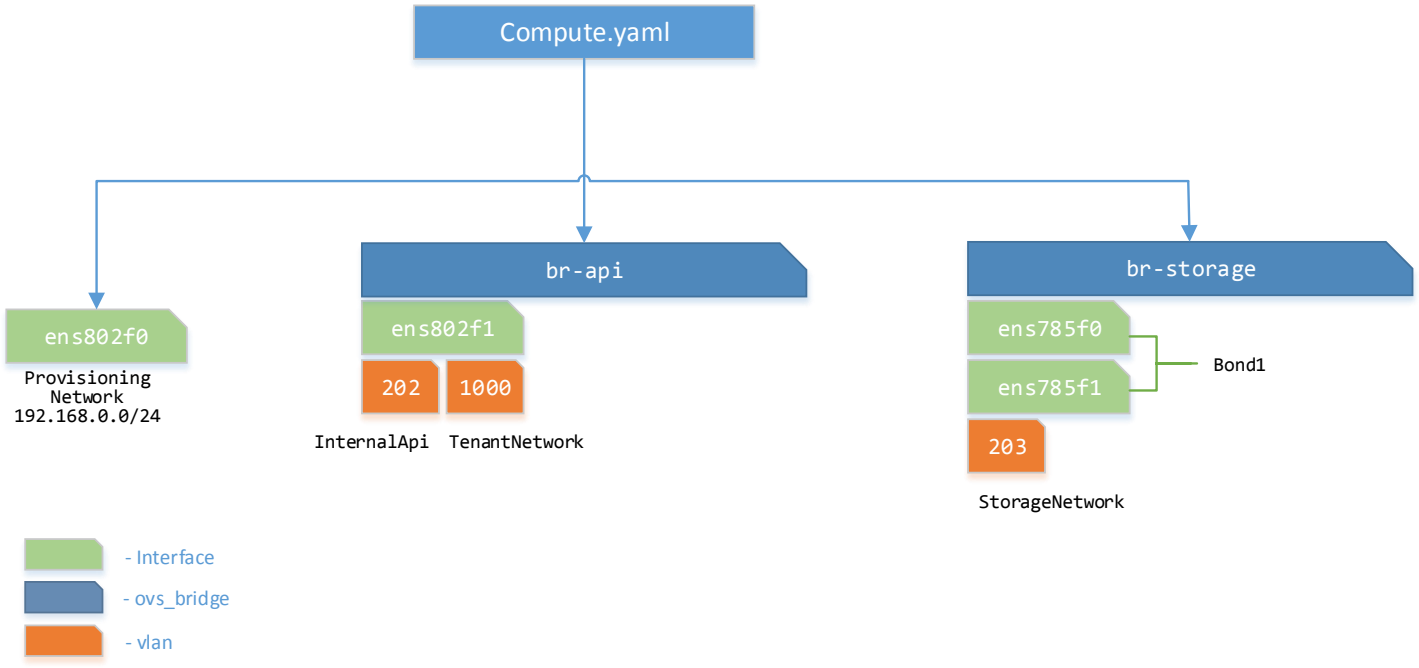


Figure 12. Network interface configuration for compute node

Table 14. Parameters for `compute.yaml`

ControlPlaneIP/ EC2MetadataIP	**192.0.2.1**		
InternalApiSubnet	192.168.0.0/24	InternalApiNetworkVlanID	202
StorageIpSubnet	192.168.1.0/24	StorageNetworkVlanID	203
TenantIPSubnet	172.16.0.0/16	TenantNetworkVlanID	1000

The default route is set to `ControlPlaneIP` and the static route for the nova metadata service of undercloud is set to `EC2Metadatalp`.

In compute heat, two Open vSwitches are defined:

- `br-api` for `InternalApi` and `TenantNetwork`
- `br- storage` for `StorageNetwork` with `mtu: 9000` option for `bond1`

The Ceph storage node is configured as shown in Figure 13.

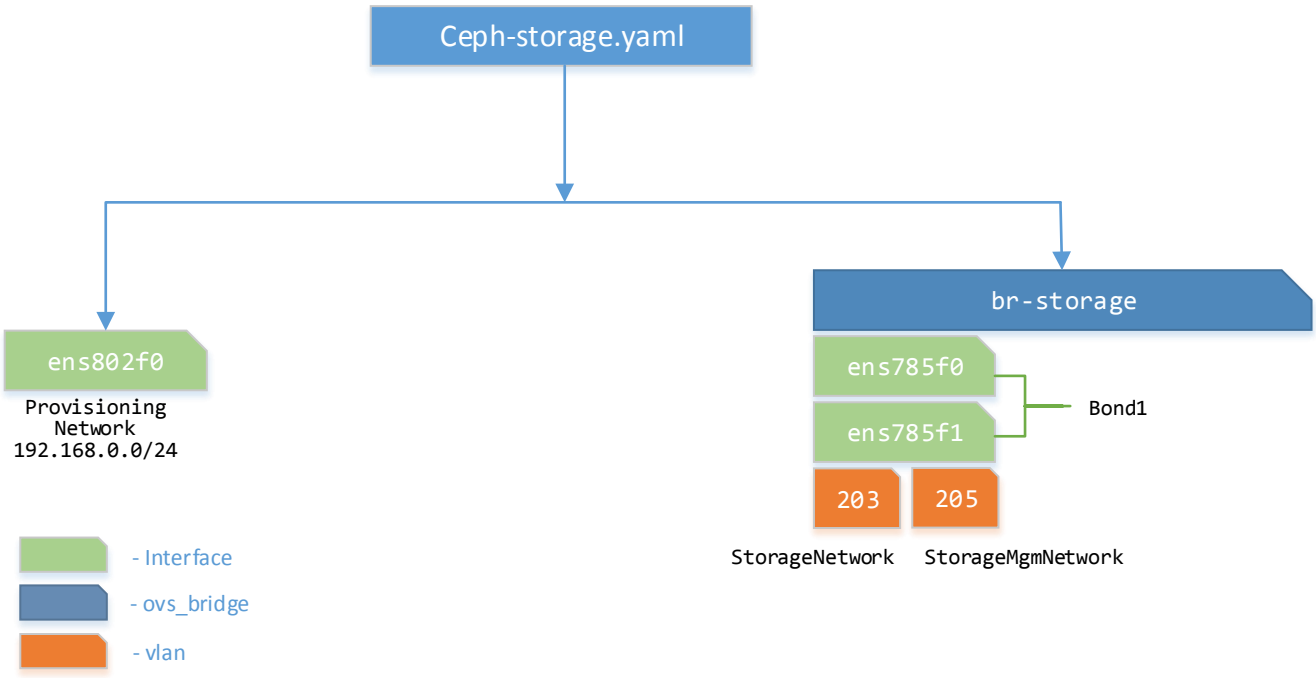


Figure 13. Network interface configuration for Ceph storage node

Table 15. Parameters for `ceph-storage.yaml`

ControlPlaneIP/ EC2MetadataIP	192.0.2.1		
StorageMgmtIpSubnet	192.168.2.0/24	StorageMgmtNetworkVlanID	205
StorageIpSubnet	192.168.1.0/24	StorageNetworkVlanID	203

The default route is set to `ControlPlaneIP` and the static route for the nova metadata service of undercloud is set to `EC2MetadataIp`.

In compute heat one Open vSwitch is defined:

- `br- storage` for `StorageNetwork` and `StorageMgmtNetwork` with `mtu: 9000` option for `bond1`

7.2.5.3 Configure the storage environment (storage-environment.yaml)

For Ceph storage node configuration refer to Appendix B. The heat sets Ceph as a back end for Cinder, Glance, and Nova ephemeral storage. The recommended Red Hat* Ceph Storage node configuration requires a disk layout similar to the following:

- `/dev/sda` - The root disk. The director copies the main overcloud image to the disk.
- `/dev/sdb` - The journal disk. This disk divides into partitions for Ceph OSD journals. For example, `/dev/sdb1`, `/dev/sdb2`, `/dev/sdb3`, and onward. The journal disk is usually a solid state drive (SSD) to aid with system performance.
- `/dev/sdc` and onward - The OSD disks. Use as many disks as necessary for your storage requirements.

It is important to erase all existing partitions on the disks targeted for journaling and OSDs before deploying the overcloud. In addition, the Ceph storage OSDs and journal disks require GUID Partition Table (GPT) disk labels. To meet this requirement according to RedHat documentation, an additional heat template

`/firstboot/wipe-disks.yaml` is used which wipes and converts all disks to GPT (except the disk containing the root file system) using the following bash script:

```
 #!/bin/bash
if [[ `hostname` = *"ceph"* ]]
then
  echo "Number of disks detected: $(lsblk -no NAME,TYPE,MOUNTPOINT | grep
"disk" | awk '{print $1}' | wc -l)"
  for DEVICE in `lsblk -no NAME,TYPE,MOUNTPOINT | grep "disk" | awk '{print
$1}'`
  do
    ROOTFOUND=0
    echo "Checking /dev/$DEVICE..."
    echo "Number of partitions on /dev/$DEVICE: $(expr $(lsblk -n
/dev/$DEVICE | awk '{print $7}' | wc -l) - 1)"
    for MOUNTS in `lsblk -n /dev/$DEVICE | awk '{print $7}'`
    do
      if [ "$MOUNTS" = "/" ]
      then
        ROOTFOUND=1
      fi
    done
    if [ $ROOTFOUND = 0 ]
    then
      echo "Root not found in /dev/${DEVICE}"
      echo "Wiping disk /dev/${DEVICE}"
      sgdisk -Z /dev/${DEVICE}
      sgdisk -g /dev/${DEVICE}

      { for disk in nvme0n1
        do
          ptype1=45b0969e-9b03-4f30-b4c6-b4b80ceff106
          sgdisk --new=1:0:+305100MiB  --change-name="1:ceph journal"  --
typecode="1:$ptype1" /dev/$disk
          sgdisk --new=2:305102MiB:+305100MiB  --change-name="2:ceph
journal"  --typecode="2:$ptype1" /dev/$disk
          sgdisk --new=3:610204MiB:+305100MiB  --change-name="3:ceph
journal"  --typecode="3:$ptype1" /dev/$disk
          sgdisk --new=4:915306MiB:+305100MiB  --change-name="4:ceph
journal"  --typecode="4:$ptype1" /dev/$disk
          sgdisk --new=5:1220408MiB:+305100MiB  --change-name="5:ceph
journal"  --typecode="5:$ptype1" /dev/$disk
        done } >> wipe-disk.txt

    else
      echo "Root found in /dev/${DEVICE}"
    fi
  done
fi
```

In this setup we use NVMe disks for journaling and the Linux NVMe kernel module enumerates devices as follows:

- `/dev/nvme0` - character device
- `/dev/nvme0n1` - whole block device
- `/dev/nvme0n1p1` - first partition
- `/dev/nvme0n1p2` - second partition

Where `ptype1=45b0969e-9b03-4f30-b4c6-b4b80ceff106` is GPT UUID type of partition for Ceph journal.

Refer to Ceph documentation (Ceph disk preparation and activation utility for OSD) for more information.

7.2.5.4 Register nodes to Red Hat* Content Delivery Network or Red Hat* Satellite server

The overcloud provides several methods to register nodes to the Red Hat Content Delivery Network, a Red Hat Satellite 5 server, or a Red Hat Satellite 6 server.

For this reference architecture, the environment files `environment-rhel-registration.yaml` and `rhel-registration-resource-registry.yaml` are used. For the full content of both of these environment files, see Appendix B. Copy the registration files from the heat template collection and customize with your own subscription details:

```
$ cp -r /usr/share/openstack-tripleo-heat-
templates/extraconfig/pre_deploy/rhel-registration ~/templates/
```

7.2.6 Deploying the overcloud

To deploy the overcloud, prepare a bash script with all needed settings:

```
#!/bin/bash
openstack overcloud deploy --templates \
  -e /usr/share/openstack-tripleo-heat-templates/overcloud-resource-
registry-puppet.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-
isolation.yaml \
  -e /home/stack/templates/rhel-registration/environment-rhel-
registration.yaml \
  -e /home/stack/templates/rhel-registration/rhel-registration-resource-
registry.yaml \
  -e /home/stack/templates/net-bond-with-vlans.yaml \
  -e /home/stack/templates/storage-environment.yaml \
  -e /home/stack/templates/limits.yaml \
  -t 150 \
  --control-scale 3 \
  --compute-scale 5 \
  --ceph-storage-scale 3 \
  --compute-flavor compute \
  --control-flavor control \
  --ceph-storage-flavor ceph-storage \
  --ntp-server 100.127.255.1 \
  --neutron-network-type vxlan \
  --neutron-bridge-mappings datacentre:br-ex,tenant:br-api \
  --neutron-network-vlan-ranges datacentre:2001:2001,tenant:1000:1010 \
  --neutron-tunnel-types vxlan \
  --verbose --debug --log-file overcloud_deploy.log
```

Some of the deployment parameters include:

- `control-scale` – Number of controller nodes.

- `compute-scale` – Number of compute nodes.
- `ceph-storage-scale` – Number of Ceph nodes.
- `ntp-server` – HA deployments that require an NTP server for time synchronization.
- `templates` – location of the heat templates directory.
- `network-isolation.yaml` –Enables the creation of neutron ports in the isolated networks.
- `net-bond-with-vlans.yaml` – The environment file built earlier.
- `t` – Deployment timeout in minutes.

For a full list of options, run:

```
$ openstack help overcloud deploy
```

When overcloud deployment finishes with success you should see the following. In Intel's deployment, this process took 15-20 minutes.

```
DEBUG: os_cloud_config.utils.clients Creating nova client.
Overcloud Endpoint: http://100.64.0.20:5000/v2.0/
Overcloud Deployed
DEBUG: openstackclient.shell clean_up DeployOvercloud
```

The director generates a script to act with the overcloud environment and save the file `overcloudrc` in the stack user's home directory. Run the following command to use this file:

```
$ source ~/overcloudrc
```

To see all deployed nodes, run command `nova list`:

```
[root@director stack]# nova list
+--------------------------------------+-------------------------+--------+-------+---------+---------------------+
| ID                                   | Name                    | Status | Task  | Power   | Networks            |
|                                      |                         |        | State | State   |                     |
+--------------------------------------+-------------------------+--------+-------+---------+---------------------+
| a3c3a1e3-a828-4968-8352-d381e71c8352 | overcloud-cephstorage-0 | ACTIVE | -     | Running | ctlplane=192.0.2.8  |
| 06208bdc-a7ae-4a67-b9fc-4e05f03920dc | overcloud-cephstorage-1 | ACTIVE | -     | Running | ctlplane=192.0.2.7  |
| 55f96a6b-04b2-44ba-a95f-9955ccd05314 | overcloud-cephstorage-2 | ACTIVE | -     | Running | ctlplane=192.0.2.10 |
| a19c7113-aff1-4291-af7b-30c96fdc91e1 | overcloud-compute-0     | ACTIVE | -     | Running | ctlplane=192.0.2.17 |
| ba84d186-8c9f-4c4b-95d3-2c7cff86539d | overcloud-compute-1     | ACTIVE | -     | Running | ctlplane=192.0.2.13 |
| c7cd58f4-39da-45ad-8dcf-8bf48cbc1cae | overcloud-compute-2     | ACTIVE | -     | Running | ctlplane=192.0.2.16 |
| e622081b-6a3a-4053-b318-588b6704e34c | overcloud-compute-3     | ACTIVE | -     | Running | ctlplane=192.0.2.18 |
| d6b8f8ea-2c1e-495a-a56e-f29bf1969efa | overcloud-compute-4     | ACTIVE | -     | Running | ctlplane=192.0.2.15 |
| b68e2fcf-127b-469d-991b-99c488e8c8fb | overcloud-controller-0  | ACTIVE | -     | Running | ctlplane=192.0.2.11 |
| 9ebd7e4e-8a82-4551-a682-25429bc095c1 | overcloud-controller-1  | ACTIVE | -     | Running | ctlplane=192.0.2.9  |
| 0e677cdc-7a47-42a4-a9da-1e1a2b7274bb | overcloud-controller-2  | ACTIVE | -     | Running | ctlplane=192.0.2.12 |
+--------------------------------------+-------------------------+--------+-------+---------+---------------------+
```

Next, log on to the nodes using `heat-admin` user using the control plane network to perform all post deployment tasks. For more details, see section 7.2.8. A login example is:

```
[root@director log]# ssh heat-admin@192.0.2.11
```

7.2.7 Debugging overcloud heat deployment

To monitor the overcloud creation:

- Check which heat resources have different states than complete. To see continuous progress, use this command with a conjunction of `watch -n` time:

  ```
  $ heat stack-list --show-nested | grep -vi complete

  $ watch -n 5 "heat stack-list --show-nested | grep -vi complete"
  ```

- Check the selected profile deployment:

  ```
  $ heat resource-show overcloud Controller
  ```

- Check the `os-collect-config` and `/var/log/messages` for deployment logs or errors:

```
[heat-admin@overcloud-controller-0 ~]$ sudo journalctl -fl -u os-collect-
config
[heat-admin@overcloud-controller-0 ~]$ sudo tail -f /var/log/messages
```

- If overcloud deployment failed, check the details of the failed resource using the command `heat resource-show overcloud [NAME_OF_FAILED_RESOURCE]`, connect on the failed node as `heat-admin` user, and see its logs:

  ```
  sudo journalctl -u os-collect-config
  ```

  The following is an example of a failure.

```
[stack@director ~]$ heat resource-list --nested-depth 5 overcloud | grep FAILED
WARNING (shell) "heat resource-list" is deprecated, please use "openstack stack resource
list" instead
| ControllerAllNodesValidationDeployment    | be2e69ba-2697-4971-8ac0-02fdad230eff
| OS::Heat::StructuredDeployments                  | CREATE_FAILED   | 2016-10-19T14:37:14
| overcloud                                                                          |
| 1                                         | 2eab1c89-f372-480b-98b1-03a875d64019
| OS::Heat::StructuredDeployment                   | CREATE_FAILED   | 2016-10-19T14:52:51
| overcloud-ControllerAllNodesValidationDeployment-iqib2c2olbwd                      |
| 2                                         | 1811fccd-4cd9-4efd-b208-272eee3ffcd7
| OS::Heat::StructuredDeployment                   | CREATE_FAILED   | 2016-10-19T14:52:51
| overcloud-ControllerAllNodesValidationDeployment-iqib2c2olbwd                      |
[stack@director ~]$  heat deployment-show 2eab1c89-f372-480b-98b1-03a875d64019
WARNING (shell) "heat deployment-show" is deprecated, please use "openstack software
deployment show" instead
{
  "status": "FAILED",
  "server_id": "ea1998d6-4c0e-42df-9478-e37968981dc3",
  "config_id": "b7fab1f2-81e7-46b0-9d9c-6f1624a43a14",
  "output_values": {
    "deploy_stdout": "Trying to ping 100.64.0.23 for local network
100.64.0.0/20...FAILURE\n",
    "deploy_stderr": "100.64.0.23 is not pingable. Local Network: 100.64.0.0/20\n",
    "deploy_status_code": 1
  },
```

In order to search all failed heats, run a script:

```
 for stack in $(heat stack-list | grep -i failed | awk '{ print $2 }'); do
  for nstack in $(heat stack-list --show-nested | grep $stack | grep -i
failed | awk '{ print $2 }'); do
    for resource in $( heat resource-list -n10 $nstack | grep -i failed |
awk '{ print $2 }'); do
      deployments=$(heat resource-list -n10 $nstack | grep -i failed | grep
-i $resource | grep Deployment | awk '{ print $4 }')
      if [ ! -z "$deployments"  ]; then
        for deployment in $deployments; do
          heat deployment-show $deployment
        done
      fi
    done
  done
done
```

7.2.7.1 Ceph issues

Ceph is a distributed storage system, so it depends upon networks to peer with OSDs, replicate objects, recover from faults, and check heartbeats. Networking issues can cause OSD latency and flapping OSDs.

Ensure that Ceph processes and Ceph-dependent processes are connected and/or listening.

```
$ netstat -a | grep ceph $ netstat -l | grep ceph $ sudo netstat -p | grep
ceph
```

Because of bug https://bugzilla.redhat.com/show_bug.cgi?id=1398236, the Ceph cluster status after successful deployment of the overcloud has errors because of no OSDs or active disks:

```
[root@overcloud-cephstorage-0 heat-admin]# ceph -s
    cluster 16eb2f62-ac3c-11e6-807b-001e67e2527d
     health HEALTH_ERR
            664 pgs stuck inactive
            664 pgs stuck unclean
            no osds
     monmap e1: 3 mons at {overcloud-controller-
0=192.168.1.18:6789/0,overcloud-controller-1=192.168.1.14:6789/0,overcloud-
controller-2=192.168.1.15:6789/0}
            election epoch 6, quorum 0,1,2 overcloud-controller-1,overcloud-
controller-2,overcloud-controller-0
     osdmap e4: 0 osds: 0 up, 0 in
      pgmap v5: 664 pgs, 4 pools, 0 bytes data, 0 objects
            0 kB used, 0 kB / 0 kB avail
                 664 creating
```

Drives listening after deployment:

```
[root@overcloud-cephstorage-1 ~]# ceph-disk list
 /dev/nvme0n1 :
 /dev/nvme0n1p1 ceph journal
 /dev/nvme0n1p2 ceph journal
 /dev/nvme0n1p3 ceph journal
 /dev/nvme0n1p4 ceph journal
 /dev/nvme0n1p5 ceph journal
/dev/sda other, unknown
/dev/sdb other, unknown
/dev/sdc other, unknown
/dev/sdd other, unknown
/dev/sde other, unknown
/dev/sdf other, unknown
 /dev/sdf1 other, iso9660
 /dev/sdf2 other, ext4, mounted on /
```

To fix this issue, use the Ceph-disk utility that prepares and activates a partition as a Ceph OSD. Perform this script on all Ceph nodes. After successfully finishing, activate all OSD disks with the command `ceph-disk activate-all`.

```
#!/bin/bash
clustered=48e6748c-acb8-11e6-b247-001e67e2527d
for i in $(nova list | grep "cephstorage" | awk '/ACTIVE/ {print $12}'| cut
-d  "=" -f2)
do
ssh -l heat-admin $i 'sudo ceph-disk prepare --cluster ceph --cluster-uuid
$clusterid /dev/sdb /dev/nvme0n1p1'
ssh -l heat-admin $i 'sudo ceph-disk prepare --cluster ceph --cluster-uuid
$clusterid /dev/sdc /dev/nvme0n1p2'
```

```
ssh -l heat-admin $i 'sudo ceph-disk prepare --cluster ceph --cluster-uuid
$clusterid /dev/sdd /dev/nvme0n1p3'
ssh -l heat-admin $i 'sudo ceph-disk prepare --cluster ceph --cluster-uuid
$clusterid /dev/sde /dev/nvme0n1p4'
ssh -l heat-admin $i 'sudo ceph-disk prepare --cluster ceph --cluster-uuid
$clusterid /dev/sdf /dev/nvme0n1p5'
ssh -l heat-admin $i 'sudo ceph-disk activate-all';
done
```

After successful OSD activation and peering process, the PG should become active and usable.

```
[root@overcloud-controller-0 ~]# ceph osd tree
ID WEIGHT  TYPE NAME                        UP/DOWN REWEIGHT PRIMARY-AFFINITY
-1 7.29996 root default
-2 3.64998     host overcloud-cephstorage-1
 0 0.73000         osd.0                         up  1.00000          1.00000
 1 0.73000         osd.1                         up  1.00000          1.00000
 2 0.73000         osd.2                         up  1.00000          1.00000
 3 0.73000         osd.3                         up  1.00000          1.00000
 5 0.73000         osd.5                         up  1.00000          1.00000
-3 3.64998     host overcloud-cephstorage-2
 4 0.73000         osd.4                         up  1.00000          1.00000
 6 0.73000         osd.6                         up  1.00000          1.00000
 7 0.73000         osd.7                         up  1.00000          1.00000
 9 0.73000         osd.9                         up  1.00000          1.00000
14 0.73000         osd.14                        up  1.00000          1.00000
-4       0     host overcloud-cephstorage-0
 8  0.73000          osd.8                       up  1.00000          1.00000
10  0.73000          osd.10                      up  1.00000          1.00000
11  0.73000          osd.11                      up  1.00000          1.00000
12  0.73000          osd.12                      up  1.00000          1.00000
13  0.73000          osd.13                      up  1.00000          1.00000
```

7.2.8 Post deployment

Perform the following steps after successful deployment.

- Check that the servers are registered with Red Hat* Subscription Management. Subscription manager status should reveal the status of this registration. If they are not registered, manually attach:

  ```
  # subscription-manager register --username admin-example --password secret -
  -auto-attach
  ```

- Verify you have a running cluster with all resources by `pcs status`:

  ```
  [root@overcloud-controller-2 ~]# pcs status
  Cluster name: tripleo_cluster
  Last updated: Thu Nov 24 12:30:55 2016          Last change: Thu Nov 24
  12:30:06 2016 by root via cibadmin on overcloud-controller-2
  Stack: corosync
  Current DC: overcloud-controller-1 (version 1.1.13-10.el7_2.4-44eb2dd) -
  partition with quorum
  3 nodes and 130 resources configured
  Online: [overcloud-controller-0 overcloud-controller-1 overcloud-controller-
  2 ]
  Full list of resources:
  . . .
  ```

```
PCSD Status:
  overcloud-controller-0: Online
  overcloud-controller-1: Online
  overcloud-controller-2: Online

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

- Correct any failed resource. The following is an entry with historic fails:

```
Failed Actions:
* openstack-glance-api_monitor_60000 on overcloud-controller-1 'not running'
(7): call=189, status=complete, exit reason='none',
    last-rc-change='Mon Nov 21 14:20:28 2016', queued=0ms, exec=0ms
```

  First, correct the failed resource and then reset the status of the resource and its fail count:

```
[root@overcloud-controller-1 ~]# pcs resource cleanup openstack-glance-api
Waiting for 3 replies from the CRMd... OK
Cleaning up openstack-glance-api:0 on overcloud-controller-0, removing fail-
count-openstack-glance-api
Cleaning up openstack-glance-api:0 on overcloud-controller-1, removing fail-
count-openstack-glance-api
Cleaning up openstack-glance-api:0 on overcloud-controller-2, removing fail-
count-openstack-glance-api
```

- Fence the controller nodes. When one of the controllers does not pass the health check, Pacemaker Designated Controller (DC) uses a Shoot-The-Other-Node-In-The-Head (stonith) service to fence off the faulty controller. Therefore, fencing the cluster helps ensure that a node cannot access one or more resources and reduces the risk of data corruption in a cluster. To allow Pacemaker to control the power management of each node in the cluster, configure the IPMI agent:

```
$ sudo pcs stonith create my-ipmilan-for-controller-0 fence_ipmilan
pcmk_host_list=overcloud-controller-0 ipaddr=192.168.42.30 login=root
passwd=r00tme lanplus=1 cipher=1 op monitor interval=60s

sudo pcs stonith create my-ipmilan-for-controller-1 fence_ipmilan
pcmk_host_list=overcloud-controller-1 ipaddr=192.168.42.31 login=root
passwd=r00tme lanplus=1 cipher=1 op monitor interval=60s


sudo pcs stonith create my-ipmilan-for-controller-2 fence_ipmilan
pcmk_host_list=overcloud-controller-2 ipaddr=192.168.42.32 login=root
passwd=r00tme lanplus=1 cipher=1 op monitor interval=60s
```

  Run the following command to see all stonith resources:

```
[root@overcloud-controller-1 ~]#   pcs stonith show
 my-ipmilan-for-controller-0    (stonith:fence_ipmilan):        Started
overcloud-controller-1
 my-ipmilan-for-controller-1    (stonith:fence_ipmilan):        Started
overcloud-controller-2
```

```
 my-ipmilan-for-controller-2    (stonith:fence_ipmilan):        Started
overcloud-controller-0
```

- Validate the OpenStack status using the command `openstack-status`.
- Create a tenant network:

```
$ source ~/overcloudrc
$ neutron net-create default
$ neutron subnet-create --name default --gateway 172.20.1.1 default
172.20.0.0/16
```

- Create an external network. In this configuration, we need to create the external network `nova` with VLAN 2001 as below:

```
neutron subnet-create --name nova --enable_dhcp=False --allocation-
pool=start=100.64.0.30,end=100.64.15.250 --gateway=100.64.0.1 nova
100.64.0.0/20
```

Check the result of the above command by issuing:

```
[root@overcloud-controller-1 ~]# neutron net-list
+--------------------------------------+--------------+----------------------------------+
| id                                   | name         | subnets                          |
+--------------------------------------+--------------+----------------------------------+
| d6b47c4c-2da3-4069-b50d-4f1cf1e57c37 | default      | 892fbaee-2a54-4910-aee1-e7c949760406
172.16.0.0/16    |
| 3a37dc1b-dde1-4cef-acbb-1dfcdb0d6f19 | nova         | c50bccc3-402d-4a6a-9700-574d09f56340
100.64.0.0/20    |
| 34706ecd-129e-469f-8a36-0dc6014faa32 | HA network tenant 7da8a71976564de7800b9aaaaa11e332 |
6caf2c25-bc86-4a0a-ae6d-11a2b2aa6950 169.254.192.0/18 |
+--------------------------------------+--------------+----------------------------------+
```

Create the router:

```
neutron net-create RouterMain --router:external --provider:network_type vlan --
provider:physical_network datacentre --provider:segmentation_id 2001

[root@overcloud-controller-1 ~]# neutron router-show RouterMain
+-------------------------+-------------------------------------------------------------------------
-----------------+
| Field                   | Value
|
+-------------------------+-------------------------------------------------------------------------
-----------------+
| admin_state_up          | True
|
| availability_zone_hints |
|
| availability_zones      | nova
|
| description             |
|
| distributed             | False
|
| external_gateway_info   | {"network_id": "3a37dc1b-dde1-4cef-acbb-1dfcdb0d6f19",
"enable_snat": true, "external_fixed_ips": [{"subnet_id": "c50bccc3-402d-4a6a-9700-
574d09f56340", "ip_address":                 |
|                         | "100.64.0.31"}]}
|
| ha                      | True
|
```

```
| id                        | 32f9142b-5edc-4402-92e5-dd0bee411856
|
| name                      | RouterMain
|
| routes                    |
|
| status                    | ACTIVE
|
| tenant_id                 | 7da8a71976564de7800b9aaaaa11e332
|
+---------------------------+-------------------------------------------------------------------
--------------------+
```

Appendix A. Drive Installation Instructions

NOTE: To maintain proper system cooling, all externally accessible drive bays must be populated with a drive carrier. Each drive carrier must have a hard disk drive (HDD), solid state device (SSD), or a supplied drive blank installed. The following instructions apply to 2.5" drives (compatible with the server chassis used for this reference architecture).

A. Remove the drive carrier from the chassis by pressing the green button and pulling open the lever.
B. Pull the carrier out of the drive bay.
C. Remove the four screws securing the plastic drive blank to the carrier.
D. Remove the drive blank from the carrier.

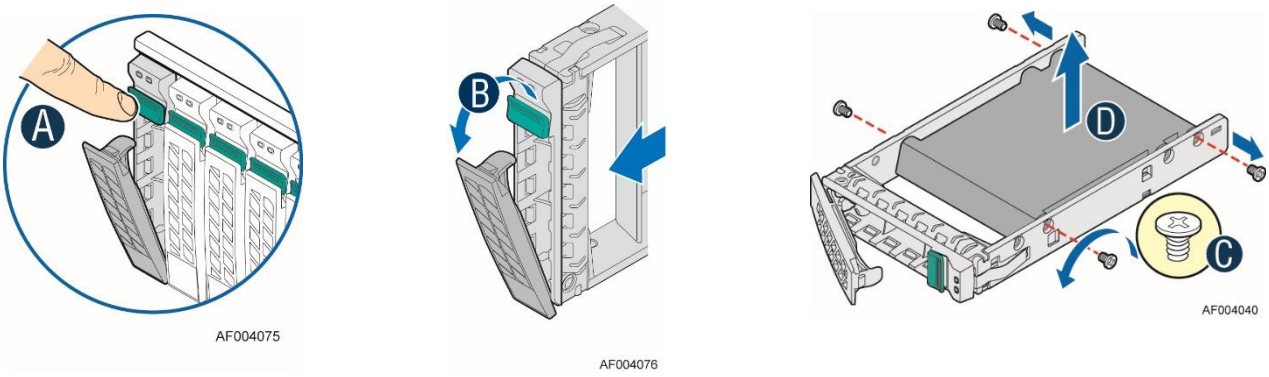


Figure 14. Extracting the drive carrier and removing the drive blank

E. Install the storage device into the carrier, verifying that the connector end of the drive is located towards the back of the carrier. Secure the drive to the carrier using the four screws.
F. With the lever open, insert the drive assembly into the chassis and push in the lever to lock it in place.

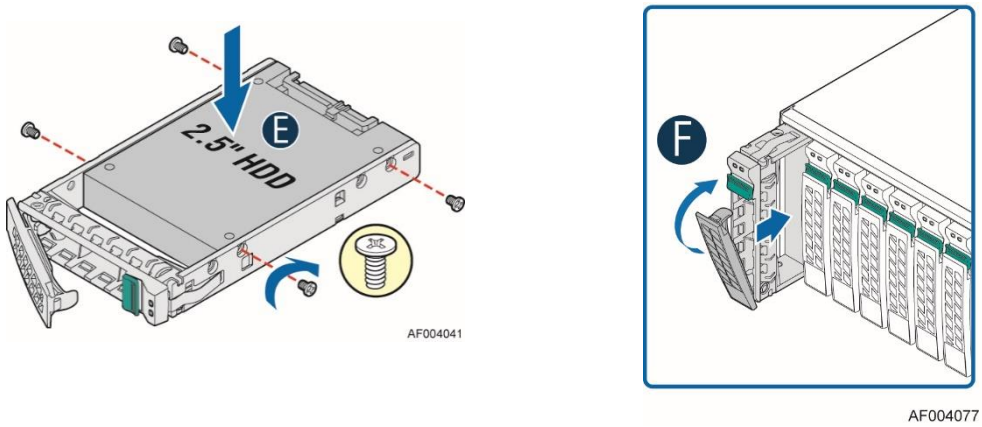


Figure 15. Installing the drive and inserting the drive assembly

Appendix B. Configuration Files

Note: Template files are sensitive to whitespaces and tabs. Please copy as is into a `.txt` file, rename the file to `.yaml`, and validate it in an online yaml parser.

B.1 undercloud.conf

```
[DEFAULT]

#
# From instack-undercloud
#

# Local file path to the necessary images. The path should be a
# directory readable by the current user that contains the full set of
# images. (string value)
#image_path = .

# Fully qualified hostname (including domain) to set on the
# Undercloud. If left unset, the current hostname will be used, but
# the user is responsible for configuring all system hostname settings
# appropriately.  If set, the undercloud install will configure all
# system hostname settings. (string value)
undercloud_hostname = director.pcsd.local

# IP information for the interface on the Undercloud that will be
# handling the PXE boots and DHCP for Overcloud instances.  The IP
# portion of the value will be assigned to the network interface
# defined by local_interface, with the netmask defined by the prefix
# portion of the value. (string value)
local_ip = 192.0.2.1/24

# Network gateway for the Neutron-managed network for Overcloud
# instances. This should match the local_ip above when using
# masquerading. (string value)
network_gateway = 192.0.2.1

# Virtual IP address to use for the public endpoints of Undercloud
# services.  Only used if undercloud_service_certficate is set.
# (string value)
undercloud_public_vip = 192.0.2.2

# Virtual IP address to use for the admin endpoints of Undercloud
# services.  Only used if undercloud_service_certficate is set.
# (string value)
undercloud_admin_vip = 192.0.2.3

# Certificate file to use for OpenStack service SSL connections.
# Setting this enables SSL for the OpenStack API endpoints, leaving it
# unset disables SSL. (string value)
undercloud_service_certificate = /etc/pki/instack-certs/undercloud.pem
```

```
# Network interface on the Undercloud that will be handling the PXE
# boots and DHCP for Overcloud instances. (string value)
local_interface = ens802f0

# Network CIDR for the Neutron-managed network for Overcloud
# instances. This should be the subnet used for PXE booting. (string
# value)
network_cidr = 192.0.2.0/24

# Network that will be masqueraded for external access, if required.
# This should be the subnet used for PXE booting. (string value)
masquerade_network = 192.0.2.0/24

# Start of DHCP allocation range for PXE and DHCP of Overcloud
# instances. (string value)
dhcp_start = 192.0.2.5

# End of DHCP allocation range for PXE and DHCP of Overcloud
# instances. (string value)
dhcp_end = 192.0.2.24

# Network interface on which inspection dnsmasq will listen.  If in
# doubt, use the default value. (string value)
# Deprecated group/name - [DEFAULT]/discovery_interface
inspection_interface = br-ctlplane

# Temporary IP range that will be given to nodes during the inspection
# process.  Should not overlap with the range defined by dhcp_start
# and dhcp_end, but should be in the same network. (string value)
# Deprecated group/name - [DEFAULT]/discovery_iprange
inspection_iprange = 192.0.2.100,192.0.2.120

# Whether to enable extra hardware collection during the inspection
# process. Requires python-hardware or python-hardware-detect package
# on the introspection image. (boolean value)
#inspection_extras = true

# Whether to run benchmarks when inspecting nodes. Requires
# inspection_extras set to True. (boolean value)
# Deprecated group/name - [DEFAULT]/discovery_runbench
#inspection_runbench = false

# Whether to support introspection of nodes that have UEFI-only
# firmware. (boolean value)
#inspection_enable_uefi = false

# Whether to enable the debug log level for Undercloud OpenStack
# services. (boolean value)
#undercloud_debug = true

# Whether to install Tempest in the Undercloud. (boolean value)
```

```
enable_tempest = true

# Whether to install Mistral services in the Undercloud. (boolean
# value)
#enable_mistral = false

# Whether to install Zaqar services in the Undercloud. (boolean value)
#enable_zaqar = false

# Whether to use iPXE for deploy by default. (boolean value)
#ipxe_deploy = true

# Whether to install Monitoring services in the Undercloud. (boolean
# value)
#enable_monitoring = false

# Whether to store events in the Undercloud Ceilometer. (boolean
# value)
#store_events = false

# Maximum number of attempts the scheduler will make when deploying
# the instance. You should keep it greater or equal to the number of
# bare metal nodes you expect to deploy at once to work around
# potential race condition when scheduling. (integer value)
# Minimum value: 1
#scheduler_max_attempts = 30


[auth]

#
# From instack-undercloud
#

# Password used for MySQL databases. If left unset, one will be
# automatically generated. (string value)
#undercloud_db_password = <None>

# Keystone admin token. If left unset, one will be automatically
# generated. (string value)
#undercloud_admin_token = <None>

# Keystone admin password. If left unset, one will be automatically
# generated. (string value)
#undercloud_admin_password = <None>

# Glance service password. If left unset, one will be automatically
# generated. (string value)
#undercloud_glance_password = <None>

# Heat db encryption key(must be 16, 24, or 32 characters. If left
# unset, one will be automatically generated. (string value)
```

```
#undercloud_heat_encryption_key = <None>

# Heat service password. If left unset, one will be automatically
# generated. (string value)
#undercloud_heat_password = <None>

# Neutron service password. If left unset, one will be automatically
# generated. (string value)
#undercloud_neutron_password = <None>

# Nova service password. If left unset, one will be automatically
# generated. (string value)
#undercloud_nova_password = <None>

# Ironic service password. If left unset, one will be automatically
# generated. (string value)
#undercloud_ironic_password = <None>

# Aodh service password. If left unset, one will be automatically
# generated. (string value)
#undercloud_aodh_password = <None>

# Ceilometer service password. If left unset, one will be
# automatically generated. (string value)
#undercloud_ceilometer_password = <None>

# Ceilometer metering secret. If left unset, one will be automatically
# generated. (string value)
#undercloud_ceilometer_metering_secret = <None>

# Ceilometer snmpd read-only user. If this value is changed from the
# default, the new value must be passed in the overcloud environment
# as the parameter SnmpdReadonlyUserName. This value must be between 1
# and 32 characters long. (string value)
#undercloud_ceilometer_snmpd_user = ro_snmp_user

# Ceilometer snmpd password. If left unset, one will be automatically
# generated. (string value)
#undercloud_ceilometer_snmpd_password = <None>

# Swift service password. If left unset, one will be automatically
# generated. (string value)
#undercloud_swift_password = <None>

# Mistral service password. If left unset, one will be automatically
# generated. (string value)
#undercloud_mistral_password = <None>

# Rabbitmq cookie. If left unset, one will be automatically generated.
# (string value)
#undercloud_rabbit_cookie = <None>
```

```
# Rabbitmq password. If left unset, one will be automatically
# generated. (string value)
#undercloud_rabbit_password = <None>

# Rabbitmq username. If left unset, one will be automatically
# generated. (string value)
#undercloud_rabbit_username = <None>

# Heat stack domain admin password. If left unset, one will be
# automatically generated. (string value)
#undercloud_heat_stack_domain_admin_password = <None>

# Swift hash suffix. If left unset, one will be automatically
# generated. (string value)
#undercloud_swift_hash_suffix = <None>

# Sensu service password. If left unset, one will be automatically
# generated. (string value)
#undercloud_sensu_password = <None>

# HAProxy stats password. If left unset, one will be automatically
# generated. (string value)
#undercloud_haproxy_stats_password = <None>
```

B.2 net-bond-with-vlans.conf

```
# This template configures each role to use a pair of bonded nics (nic2 and
# nic3) and configures an IP address on each relevant isolated network
# for each role. This template assumes use of network-isolation.yaml.
#
# FIXME: if/when we add functionality to heatclient to include heat
# environment files we should think about using it here to automatically
# include network-isolation.yaml.
resource_registry:
  OS::TripleO::Compute::Net::SoftwareConfig:
    /home/stack/templates/nic-configs/compute.yaml
  OS::TripleO::Controller::Net::SoftwareConfig:
    /home/stack/templates/nic-configs/controller.yaml
  OS::TripleO::CephStorage::Net::SoftwareConfig:
    /home/stack/templates/nic-configs/ceph-storage.yaml

parameter_defaults:
  # Customize all these values to match the local environment
  ControlPlaneIp: 192.0.2.1
  InternalApiNetCidr: 192.168.0.0/24
  StorageNetCidr: 192.168.1.0/24
  StorageMgmtNetCidr: 192.168.2.0/24
  TenantNetCidr: 172.16.0.0/16
  ExternalNetCidr: 100.64.0.0/20
  # CIDR subnet mask length for provisioning network
  ControlPlaneSubnetCidr: '20'
```

```
  InternalApiAllocationPools: [{'start': '192.168.0.11', 'end':
'192.168.0.200'}]
  StorageAllocationPools: [{'start': '192.168.1.10', 'end':
'192.168.1.100'}]
  StorageMgmtAllocationPools: [{'start': '192.168.2.10', 'end':
'192.168.2.100'}]
  TenantAllocationPools: [{'start': '172.16.0.10', 'end': '172.16.200.200'}]
  # Use an External allocation pool which will leave room for floating IPs
  ExternalAllocationPools: [{'start': '100.64.0.20', 'end':
'100.64.15.250'}]
  # Set to the router gateway on the external network
  ExternalInterfaceDefaultRoute: 100.64.0.1
  # Gateway router for the provisioning network (or Undercloud IP)
  ControlPlaneDefaultRoute: 192.0.2.1
  # Generally the IP of the Undercloud
  EC2MetadataIp: 192.0.2.1
  # Define the DNS servers (maximum 2) for the overcloud nodes
  DnsServers: ["8.8.8.8","8.8.4.4"]
  InternalApiNetworkVlanID: 202
  StorageNetworkVlanID: 203
  StorageMgmtNetworkVlanID: 205
  TenantNetworkVlanID: 1000
  ExternalNetworkVlanID: 2001
  # May set to br-ex if using floating IPs only on native VLAN on bridge br-
ex
  NeutronExternalNetworkBridge: "''"
  # Customize bonding options if required (ignored if bonds are not used)
  BondInterfaceOvsOptions:
    "bond_mode=balance-slb"
```

B.3 compute.yaml

```
heat_template_version: 2015-04-30

description: >
  Software Config to drive os-net-config with 2 bonded nics on a bridge
  with VLANs attached for the compute role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
  StorageIpSubnet:
```

```
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-
management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  BondInterfaceOvsOptions:
    default: 'bond_mode=balance-slb'
    description: The ovs_options string for the bond interface. Set things
like
                 lacp=active and/or bond_mode=balance-slb using this option.
    type: string
  InternalApiNetworkVlanID:
    default: 202
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 203
    description: Vlan ID for the storage network traffic.
    type: number
  TenantNetworkVlanID:
    default: 1000
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 202
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that
will be added to resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
```

```
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: ens802f0
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              addresses:
                -
                  ip_netmask:
                    list_join:
                      - '/'
                      - - {get_param: ControlPlaneIp}
                        - {get_param: ControlPlaneSubnetCidr}
              routes:
                -
                  ip_netmask: 169.254.169.254/32
                  next_hop: {get_param: EC2MetadataIp}
                -
                  default: true
                  next_hop: {get_param: ControlPlaneDefaultRoute}
            -
              type: ovs_bridge
              name: br-api
              members:
                -
                  type: interface
                  name: ens802f1
                  primary: true
                -
                  type: vlan
                  vlan_id: {get_param: InternalApiNetworkVlanID}
                  addresses:
                  -
                    ip_netmask: {get_param: InternalApiIpSubnet}
                -
                  type: vlan
                  vlan_id: {get_param: TenantNetworkVlanID}
                  addresses:
                  -
                    ip_netmask: {get_param: TenantIpSubnet}
            -
              type: ovs_bridge
              name: br-storage
```

```
            members:
              -
                type: ovs_bond
                name: bond1
                ovs_options: {get_param: BondInterfaceOvsOptions}
                members:
                  -
                    type: interface
                    name: ens785f0
                    primary: true
                  -
                    type: interface
                    name: ens785f1
              -
                type: vlan
                device: bond1
                mtu: 9000
                vlan_id: {get_param: StorageNetworkVlanID}
                addresses:
                  -
                    ip_netmask: {get_param: StorageIpSubnet}

outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value: {get_resource: OsNetConfigImpl}
```

B.4 controller.yaml

```
heat_template_version: 2015-04-30

description: >
  Software Config to drive os-net-config with 2 bonded nics on a bridge
  with VLANs attached for the controller role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
```

```
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-
management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  BondInterfaceOvsOptions:
    default: 'bond_mode=balance-slb'
    description: The ovs_options string for the bond interface. Set things
like
                 lacp=active and/or bond_mode=balance-slb using this option.
    type: string
  ExternalNetworkVlanID:
    default: 2001
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 202
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 203
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 205
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 1000
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 202
    description: Vlan ID for the management network traffic.
    type: number
  ExternalInterfaceDefaultRoute:
    default: '100.64.0.1'
    description: default route for the external network
    type: string
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '20'
    description: The subnet CIDR of the control plane network.
    type: string
  DnsServers: # Override this via parameter_defaults
```

```
    default: []
    description: A list of DNS servers (2 max for some implementations) that
will be added to resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: ens802f0
              use_dhcp: false
              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: ovs_bridge
              name: br-api
              use_dhcp: false
              members:
                -
                  type: interface
                  name: ens802f1
                  primary: true
                -
                  type: vlan
                  vlan_id: {get_param: InternalApiNetworkVlanID}
                  addresses:
                  -
                    ip_netmask: {get_param: InternalApiIpSubnet}
                -
                  type: vlan
                  vlan_id: {get_param: TenantNetworkVlanID}
                  addresses:
                  -
                    ip_netmask: {get_param: TenantIpSubnet}
```

```
            -
              type: ovs_bridge
              name: br-ex
              dns_servers: {get_param: DnsServers}
              members:
                -
                  type: ovs_bond
                  name: bond1
                  ovs_options: {get_param: BondInterfaceOvsOptions}
                  members:
                    -
                      type: interface
                      name: ens785f0
                      primary: true
                    -
                      type: interface
                      name: ens785f1
                -
                  type: vlan
                  device: bond1
                  vlan_id: {get_param: ExternalNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: ExternalIpSubnet}
                  routes:
                    -
                      default: true
                      next_hop: {get_param: ExternalInterfaceDefaultRoute}
                -
                  type: vlan
                  device: bond1
                  vlan_id: {get_param: StorageNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: StorageIpSubnet}
                -
                  type: vlan
                  device: bond1
                  vlan_id: {get_param: StorageMgmtNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: StorageMgmtIpSubnet}


outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value: {get_resource: OsNetConfigImpl}
```

B.5 ceph-storage.yaml

```
heat_template_version: 2015-04-30
```

```
description: >
  Software Config to drive os-net-config with 2 bonded nics on a bridge
  with VLANs attached for the ceph storage role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-
management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  BondInterfaceOvsOptions:
    default: 'bond_mode=balance-slb'
    description: The ovs_options string for the bond interface. Set things
like
                 lacp=active and/or bond_mode=balance-slb using this option.
    type: string
  InternalApiNetworkVlanID:
    default: 202
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 203
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 205
    description: Vlan ID for the storage mgmt network traffic.
```

```
    type: number
  ManagementNetworkVlanID:
    default: 202
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that
will be added to resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: ens802f0
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              addresses:
                -
                  ip_netmask:
                    list_join:
                      - '/'
                      - - {get_param: ControlPlaneIp}
                        - {get_param: ControlPlaneSubnetCidr}
              routes:
                -
                  ip_netmask: 169.254.169.254/32
                  next_hop: {get_param: EC2MetadataIp}
                -
                  default: true
                  next_hop: {get_param: ControlPlaneDefaultRoute}
            -
              type: ovs_bridge
              name: br-storage
              members:
```

```
                -
                  type: ovs_bond
                  name: bond1
                  ovs_options: {get_param: BondInterfaceOvsOptions}
                  members:
                    -
                      type: interface
                      name: ens785f0
                      primary: true
                    -
                      type: interface
                      name: ens785f1
                -
                  type: vlan
                  device: bond1
                  vlan_id: {get_param: StorageNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: StorageIpSubnet}
                -
                  type: vlan
                  device: bond1
                  mtu: 9000
                  vlan_id: {get_param: StorageMgmtNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: StorageMgmtIpSubnet}
                # Uncomment when including environments/network-
management.yaml
                #-
                #  type: vlan
                #  device: bond1
                #  vlan_id: {get_param: ManagementNetworkVlanID}
                #  addresses:
                #    -
                #      ip_netmask: {get_param: ManagementIpSubnet}

outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value: {get_resource: OsNetConfigImpl}
```

B.6 storage-environment.yaml

```
## A Heat environment file which can be used to set up storage
## backends. Defaults to Ceph used as a backend for Cinder, Glance and
## Nova ephemeral storage.
resource_registry:
  OS::TripleO::NodeUserData: /home/stack/templates/firstboot/wipe-disks.yaml

parameter_defaults:
  ExtraConfig:
```

```
  ceph::profile::params::osds:
  '/dev/sda':
    journal: '/dev/nvme0n1p1'
  '/dev/sdb':
    journal: '/dev/nvme0n1p2'
  '/dev/sdc':
    journal: '/dev/nvme0n1p3'
  '/dev/sdd':
    journal: '/dev/nvme0n1p4'
  '/dev/sde':
    journal: '/dev/nvme0n1p5'

  ceph::profile::params::osd_journal_size: 305100
  ceph::profile::params::osd_pool_default_pg_num: 200
  ceph::profile::params::osd_pool_default_pgp_num: 200
  ceph::profile::params::osd_pool_default_size: 3
  ceph::profile::params::osd_pool_default_min_size: 1
  ceph::profile::params::osd_op_threads: '4'

  ceph_pools:
  - volumes
  - vms
  - images

  ceph_classes: []

  ceph_osd_selinux_permissive: true

#### BACKEND SELECTION ####

## Whether to enable iscsi backend for Cinder.
CinderEnableIscsiBackend: false
## Whether to enable rbd (Ceph) backend for Cinder.
CinderEnableRbdBackend: true
## Whether to enable NFS backend for Cinder.
# CinderEnableNfsBackend: false
## Whether to enable rbd (Ceph) backend for Nova ephemeral storage.
NovaEnableRbdBackend: true
## Glance backend can be either 'rbd' (Ceph), 'swift' or 'file'.
GlanceBackend: rbd
## Gnocchi backend can be either 'rbd' (Ceph), 'swift' or 'file'.
GnocchiBackend: rbd


#### CINDER NFS SETTINGS ####

## NFS mount options
# CinderNfsMountOptions: ''
## NFS mount point, e.g. '192.168.122.1:/export/cinder'
# CinderNfsServers: ''
```

```
  #### GLANCE FILE BACKEND PACEMAKER SETTINGS (used for mounting NFS) ####

  ## Whether to make Glance 'file' backend a mount managed by Pacemaker
  # GlanceFilePcmkManage: false
  ## File system type of the mount
  # GlanceFilePcmkFstype: nfs
  ## Pacemaker mount point, e.g. '192.168.122.1:/export/glance' for NFS
  # GlanceFilePcmkDevice: ''
  ## Options for the mount managed by Pacemaker
  # GlanceFilePcmkOptions: ''


  #### CEPH SETTINGS ####

  ## Whether to deploy Ceph OSDs on the controller nodes. By default
  ## OSDs are deployed on dedicated ceph-storage nodes only.
  # ControllerEnableCephStorage: false

  ## When deploying Ceph Nodes through the oscplugin CLI, the following
  ## parameters are set automatically by the CLI. When deploying via
  ## heat stack-create or ceph on the controller nodes only,
  ## they need to be provided manually.

  ## Number of Ceph storage nodes to deploy
  # CephStorageCount: 0
  ## Ceph FSID, e.g. '4b5c8c0a-ff60-454b-a1b4-9747aa737d19'
  # CephClusterFSID: ''
  ## Ceph monitor key, e.g. 'AQC+Ox1VmEr3BxAALZejqeHj50Nj6wJDvs96OQ=='
  # CephMonKey: ''
  ## Ceph admin key, e.g. 'AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ=='
  # CephAdminKey: ''
```

B.7 wipe-disks.yaml

```
heat_template_version: 2014-10-16

description: >
  Wipe and convert all disks to GPT (except the disk containing the root
file system)

resources:
  userdata:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: wipe_disk}

  wipe_disk:
    type: OS::Heat::SoftwareConfig
    properties:
      config: {get_file: wipe-disk.sh}
```

```
outputs:
  OS::stack_id:
    value: {get_resource: userdata}
```

B.8 limits.yaml

```
parameters:
 MysqlMaxConnections: 8192
 RabbitFDLimit: 65436
```

B.9 environment-rhel-registration.yaml

```
# Note this can be specified either in the call
# to heat stack-create via an additional -e option
# or via the global environment on the seed in
# /etc/heat/environment.d/default.yaml
parameter_defaults:
  rhel_reg_activation_key: ""
  rhel_reg_auto_attach: "true"
  rhel_reg_base_url: ""
  rhel_reg_environment: ""
  rhel_reg_force: ""
  rhel_reg_machine_name: ""
  rhel_reg_org: ""
  rhel_reg_password: "your password"
  rhel_reg_pool_id: "your pool id"
  rhel_reg_release: ""
  rhel_reg_repos: "rhel-7-server-rpms rhel-7-server-extras-rpms rhel-7-
server-rh-common-rpms rhel-ha-for-rhel-7-server-rpms rhel-7-server-
openstack-8-director-rpms rhel-7-server-openstack-8-rpms rhel-7-server-
rhceph-1.3-osd-rpms rhel-7-server-rhceph-1.3-mon-rpms"
  rhel_reg_sat_url: ""
  rhel_reg_server_url: ""
  rhel_reg_service_level: ""
  rhel_reg_user: "your user name"
  rhel_reg_type: ""
  rhel_reg_method: "portal"
  rhel_reg_sat_repo: ""
```

B.10 rhel-registration-resource-registry.yaml

```
resource_registry:
  OS::TripleO::NodeExtraConfig: rhel-registration.yaml
```

B.11 rhel-registration.yaml

```
heat_template_version: 2014-10-16

description: >
  RHEL Registration and unregistration software deployments.

# Note extra parameters can be defined, then passed data via the
# environment parameter_defaults, without modifying the parent template
```

```
parameters:
  server:
    type: string
  # To be defined via a local or global environment in parameter_defaults
  rhel_reg_activation_key:
    type: string
  rhel_reg_auto_attach:
    type: string
  rhel_reg_base_url:
    type: string
  rhel_reg_environment:
    type: string
  rhel_reg_force:
    type: string
  rhel_reg_machine_name:
    type: string
  rhel_reg_org:
    type: string
  rhel_reg_password:
    type: string
  rhel_reg_pool_id:
    type: string
  rhel_reg_release:
    type: string
  rhel_reg_repos:
    type: string
  rhel_reg_sat_url:
    type: string
  rhel_reg_server_url:
    type: string
  rhel_reg_service_level:
    type: string
  rhel_reg_user:
    type: string
  rhel_reg_type:
    type: string
  rhel_reg_method:
    type: string
  rhel_reg_sat_repo:
    type: string

resources:

  RHELRegistration:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      inputs:
        - name: REG_ACTIVATION_KEY
        - name: REG_AUTO_ATTACH
        - name: REG_BASE_URL
        - name: REG_ENVIRONMENT
```

```
      - name: REG_FORCE
      - name: REG_MACHINE_NAME
      - name: REG_ORG
      - name: REG_PASSWORD
      - name: REG_POOL_ID
      - name: REG_RELEASE
      - name: REG_REPOS
      - name: REG_SAT_URL
      - name: REG_SERVER_URL
      - name: REG_SERVICE_LEVEL
      - name: REG_USER
      - name: REG_TYPE
      - name: REG_METHOD
      - name: REG_SAT_REPO
    config: {get_file: scripts/rhel-registration}

RHELRegistrationDeployment:
  type: OS::Heat::SoftwareDeployment
  properties:
    name: RHELRegistrationDeployment
    server:  {get_param: server}
    config: {get_resource: RHELRegistration}
    actions: ['CREATE'] # Only do this on CREATE
    input_values:
      REG_ACTIVATION_KEY: {get_param: rhel_reg_activation_key}
      REG_AUTO_ATTACH: {get_param: rhel_reg_auto_attach}
      REG_BASE_URL: {get_param: rhel_reg_base_url}
      REG_ENVIRONMENT: {get_param: rhel_reg_environment}
      REG_FORCE: {get_param: rhel_reg_force}
      REG_MACHINE_NAME: {get_param: rhel_reg_machine_name}
      REG_ORG: {get_param: rhel_reg_org}
      REG_PASSWORD: {get_param: rhel_reg_password}
      REG_POOL_ID: {get_param: rhel_reg_pool_id}
      REG_RELEASE: {get_param: rhel_reg_release}
      REG_REPOS: {get_param: rhel_reg_repos}
      REG_SAT_URL: {get_param: rhel_reg_sat_url}
      REG_SERVER_URL: {get_param: rhel_reg_server_url}
      REG_SERVICE_LEVEL: {get_param: rhel_reg_service_level}
      REG_USER: {get_param: rhel_reg_user}
      REG_TYPE: {get_param: rhel_reg_type}
      REG_METHOD: {get_param: rhel_reg_method}
      REG_SAT_REPO: {get_param: rhel_reg_sat_repo}

RHELUnregistration:
  type: OS::Heat::SoftwareConfig
  properties:
    group: script
    config: {get_file: scripts/rhel-unregistration}
    inputs:
      - name: REG_METHOD

RHELUnregistrationDeployment:
```

```
    type: OS::Heat::SoftwareDeployment
    properties:
      name: RHELUnregistrationDeployment
      server:  {get_param: server}
      config: {get_resource: RHELUnregistration}
      actions: ['DELETE'] # Only do this on DELETE
      input_values:
        REG_METHOD: {get_param: rhel_reg_method}

outputs:
  deploy_stdout:
    description: Deployment reference, used to trigger puppet apply on
changes
    value: {get_attr: [RHELRegistrationDeployment, deploy_stdout]}
```

Appendix C. References

For more information on Intel Data Blocks for Cloud, please visit http://www.intel.com/content/www/us/en/data-center-blocks/cloud/cloud-blocks.html.

For more details about Red Hat OpenStack Platform 9, please refer to:

- Red Hat OpenStack Platform 9 Product Guide
- Red Hat OpenStack Platform 9 Architecture Guide
- Nova Developer Documentation
- Heat Developer Documentation
- Red Hat OpenStack Platform 9 Director Installation and Usage Guide

For more information on Ceph storage, please refer to:

- Red Hat Ceph Storage 1.3 Storage Strategies Guide.
- Ceph Architecture
- Ceph PGs per Pool Calculator

Appendix D. Glossary

Term	Definition
ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
CA	Certificate Authority
Ceilometer	OpenStack* telemetry service
Ceph*	Object storage solution used in this reference architecture
CIDR	Classless Inter-Domain Routing
CIFS	Common Internet File System
Cinder	OpenStack* block storage service
CLI	Command Line Interface
CRUSH	Controlled Replication Under Scalable Hashing
Glance	OpenStack* image service
GPT	GUID Partition Table
HA	High Availability
HCI	Hyper-Converged Infrastructure
HDD	Hard Disk Drive
Heat	OpenStack* orchestration service
Horizon	OpenStack* dashboard service
IaaS	Infrastructure as a Service
IOPS	Input / Output Operations per Second
IPMI	Intelligent Platform Management Interface
iSCSI	Internet Small Computed Systems Interface
Ironic	OpenStack* bare metal provisioning service
Keystone	OpenStack* identity service
LACP	Link Aggregation Control Protocol
Neutron	OpenStack* networking service
NFS	Network File System
NIC	Network Interface Controller
Nova	OpenStack* compute service
NTP	Network Time Protocol
NVMe*	NVM Express (Non-Volatile Memory Host Controller Interface Specification (NVMHCI))
OOB	Out-of-Band
OSD	Object Storage Daemons
OVS	OpenVirtual Switch
Overcloud	Deployed cloud in environment using Red Hat* OpenStack* Platform director
PaaS	Platform as a Service
Pacemaker	Open source high availability resource manager software used on clusters
PCIe*	PCI Express
PG	Placement Group
PXE	Preboot Execution Environment
RADOS	Reliable Autonomic Distributed Object Store
SaaS	Software as a Service
SATA	Serial ATA
SDK	Software Development Kit
SSD	Solid State Drive

Term	Definition
SFP+	Enhanced Small Form Factor Pluggable Transceiver
Swift	OpenStack* object storage service
TOR	Top of Rack
TPM	Trusted Platform Module
TripleO	OpenStack*-on-OpenStack*
Undercloud	Director node in environment using Red Hat* OpenStack* Platform director
UUID	Universally Unique Identifier
VLAN	Virtual LAN
VM	Virtual Machine (also known as an instance)
VXLAN	Virtual Extensible LAN
XaaS	Everything as a Service