



Red Hat Reference Architecture Series

Hyper-converged Red Hat OpenStack Platform 10 and Red Hat Ceph Storage 2

John M. Fulton

Version 3.2, 2017-3-14

Table of Contents

1. Comments and Feedback	2
1.1. Staying In Touch	2
2. Executive Summary	3
3. Technical Summary	4
3.1. Using the RHsyseng HCI GitHub Repository	4
4. Hardware Recommendations	5
4.1. Required Servers	5
4.2. Recommended Networks	6
5. Prerequisites	9
5.1. Deploy the Undercloud	9
5.2. Register and Introspect Hardware	10
6. Define the Overcloud	14
6.1. Custom Environment Templates	14
6.2. Network Configuration	14
6.3. Hyper Converged Role Definition	19
6.4. Ceph Configuration	20
6.5. Overcloud Layout	27
7. Resource Isolation and Tuning	32
7.1. Nova Reserved Memory and CPU Allocation Ratio	32
7.2. Ceph NUMA Pinning	36
7.3. Reduce Ceph Backfill and Recovery Operations	42
7.4. Regarding tuned	43
8. Deployment	44
8.1. Verify Ironic Nodes are Available	44
8.2. Run the Deploy Command	45
8.3. Verify the Deployment Succeeded	46
8.4. Configure Controller Pacemaker Fencing	48
9. Operational Considerations	50
9.1. Configuration Updates	50
9.2. Adding Compute/Red Hat Ceph Storage Nodes	50
9.3. Removing Compute/Red Hat Ceph Storage Nodes	55
10. Conclusion	71
Appendix A: Contributors	72
Appendix B: References	73
Appendix C: Environment Details	74
C.1. Red Hat OpenStack Platform director	74

C.2. Overcloud Controller / Ceph Monitor	75
C.3. Overcloud Compute / Ceph OSD	75
C.4. Network Environment	76
Appendix D: Custom Heat Templates	78
Appendix E: Nova Memory and CPU Calculator	99
Appendix F: Example Fencing Script	102
Appendix G: GitHub Repository of Example Files	104

100 East Davie Street
Raleigh NC 27601 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

Linux is a registered trademark of Linus Torvalds. Red Hat, Red Hat Enterprise Linux and the Red Hat "Shadowman" logo are registered trademarks of Red Hat, Inc. in the United States and other countries.

Ceph is a registered trademark of Red Hat, Inc.

UNIX is a registered trademark of The Open Group.

Intel, the Intel logo and Xeon are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. All other trademarks referenced herein are the property of their respective owners.

© 2017 by Red Hat, Inc. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

The information contained herein is subject to change without notice. Red Hat, Inc. shall not be liable for technical or editorial errors or omissions contained herein.

Distribution of modified versions of this document is prohibited without the explicit permission of Red Hat Inc.

Distribution of this work or derivative of this work in any standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from Red Hat Inc.

The GPG fingerprint of the security@redhat.com key is: CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Send feedback to refarch-feedback@redhat.com

1. Comments and Feedback

In the spirit of open source, we invite anyone to provide feedback and comments on any reference architectures. Although we review our papers internally, sometimes issues or typographical errors are encountered. Feedback allows us to not only improve the quality of the papers we produce, but allows the reader to provide their thoughts on potential improvements and topic expansion to the papers. Feedback on the papers can be provided by emailing refarch-feedback@redhat.com. Please refer to the title within the email.

1.1. Staying In Touch

Join us on some of the popular social media sites where we keep our audience informed on new reference architectures as well as offer related information on things we find interesting.

1.1.1. Like us on Facebook:

<https://www.facebook.com/rhrefarch>

1.1.2. Follow us on Twitter:

<https://twitter.com/RedHatRefArch>

1.1.3. Plus us on Google+

<https://plus.google.com/u/0/b/114152126783830728030/>

2. Executive Summary

This reference architecture describes how to deploy Red Hat® OpenStack® Platform and Red Hat Ceph Storage in a way that both the OpenStack Nova Compute services and the Ceph Object Storage Daemon (OSD) services reside on the same node. A server which runs both compute and storage processes is known as a *hyper-converged node*. There is increasing interest in the field for hyper-convergence for cloud (NFVi and Enterprise) deployments. The reasons include smaller initial deployment foot prints, a lower cost of entry, and maximized capacity utilization.

The first section of this reference architecture provides a technical summary for an implementer to quickly deploy a hyper-converged overcloud by using templates and scripts from this reference architecture on GitHub.

The second section provides general hardware and network guidance.

The third section covers software prerequisites for the undercloud and managing hardware with Ironic.

The fourth section covers how to define a hyper-converged overcloud in Heat which is stored in a deployment plan.

The fifth section covers how to isolate resources in a hyper-converged overcloud to address contention between OpenStack and Ceph which could result in degradation of either service.

The sixth section covers how to deploy a hyper-converged overcloud using the deployment plan defined in the previous sections.

The seventh section covers some of the operational concerns of running a hyper-converged OpenStack and Ceph deployment. It covers configuration updates, adding new Compute/Red Hat Ceph Storage nodes, and removing running Compute/Red Hat Ceph Storage nodes.

This reference architecture has been completed with Red Hat Enterprise Linux 7.3, Red Hat OpenStack Platform 10, Red Hat OpenStack Platform director 10, and Red Hat Ceph Storage 2. All of the steps listed were performed by the Red Hat Systems Engineering team. The complete use case was deployed in the Systems Engineering lab on bare metal servers, except where otherwise noted.



Hyper-converged deployments in Red Hat OpenStack Platform 10 are Technology Previews. In order for the steps in this reference architecture to be supported, a support exception must be filed.

3. Technical Summary

This reference architecture describes the step-by-step procedures to deploy Red Hat OpenStack Platform and Red Hat Ceph Storage in a way that both the OpenStack Nova Compute services and the Ceph Object Storage Daemon (OSD) services reside on the same node.

In order to facilitate the implementation process of this reference architecture, all of the scripts and Heat templates may be accessed directly on GitHub. The following section shows an example of how an implementer may use the GitHub repository to make changes to implement a similar environment.

3.1. Using the RHsyseng HCI GitHub Repository

After installing the undercloud as described in [Deploy the Undercloud](#), [ssh](#) into the Red Hat OpenStack Platform director server as the stack user and clone the *RHsyseng HCI GitHub Repository*:

```
git clone https://github.com/RHsyseng/hci
```

Copy the *custom-templates* directory, provided in the repository, and customize the templates as described in [Define the Overcloud](#).

```
cp -r hci/custom-templates ~/
```

Copy the *nova_mem_cpu_calc.py* script as provided in the repository:

```
cp hci/scripts/nova_mem_cpu_calc.py ~/
```

Run *nova_mem_cpu_calc.py* to determine the appropriate resource isolation required for the HCI deployment as described in [Resource Isolation and Tuning](#) and then update the Heat environment templates as described in [Nova Memory and CPU Calculator](#). Ensure the overcloud will have access to NUMA related packages as described in [Ceph NUMA Pinning](#).

Copy the *deploy.sh* script and use the script to deploy the overcloud as described in [Deployment](#).

```
cp hci/scripts/deploy.sh ~/
~/deploy.sh
```



While the above steps provide a quick way to modify and potentially create a similar environment as this reference architecture, it is not meant to replace the comprehensiveness of this full document.

4. Hardware Recommendations

This reference architecture focuses on:

- Providing configuration instruction details
- Validating the interoperability of Red Hat OpenStack Platform Nova Compute instances and Red Hat Ceph Storage on the same physical servers.
- Providing automated methods to apply resource isolation to avoid contention between Nova Compute and Ceph OSD services.

Red Hat's experience with early hyper-converged adopters reflect a wide variety of hardware configurations. Baseline hardware performance and sizing recommendations for non-hyper-converged Ceph clusters can be found in the [Hardware Selection Guide for Ceph](#).

Additional considerations for hyper-converged Red Hat OpenStack Platform with Red Hat Ceph Storage server nodes include:

- **Network:** the recommendation is to configure 2x 10GbE NICs for Ceph. Additional NICs are recommended to meet Nova VM workload networking requirements that include bonding of NICs and trunking of VLANs.
- **RAM:** the recommendation is to configure 2x RAM needed by the resident Nova VM workloads.
- **OSD Media:** the recommendation is to configure 7,200 RPM enterprise HDDs for general-purpose workloads or NVMe SSDs for IOPS-intensive workloads. For workloads requiring large amounts of storage capacity, it may be better to configure separate storage and compute server pools (non hyper-converged).
- **Journal Media:** the recommendation is to configure SAS/SATA SSDs for general-purpose workloads or NVMe SSDs for IOPS-intensive workloads.
- **CPU:** the recommendation is to configure a minimum dual-socket 16-core CPUs for servers with NVMe storage media, or dual-socket 10-core CPUs for servers with SAS/SATA SSDs.

Details of the hardware configuration for this reference architecture can be found in [Appendix: Environment Details](#).

4.1. Required Servers

The minimum infrastructure requires at least six bare metal servers and either a seventh bare metal server or virtual machine hosted separately, not hosted on the six bare metal servers. These servers should be deployed in the following roles:

- 1 Red Hat OpenStack Platform director server (can be virtualized for small deployments)
- 3 Cloud Controllers/Ceph Monitors (Controller/Mon nodes)
- 3 Compute Hypervisors/Ceph storage servers (Compute/OSD nodes)

As part of this reference architecture, a fourth Compute/Ceph storage node is added to demonstrate scaling of an infrastructure.



Additional Compute/Ceph storage nodes may be initially deployed or added later. However, for deployments spanning more than one datacenter rack (42 nodes), Red Hat recommends the use of standalone storage and compute, and not a hyper-converged approach.

4.2. Recommended Networks

This reference architecture uses six networks to serve the roles described in this section. How these networks may be trunked as VLANs to connect to the servers is illustrated in [Figure 1. Network Separation Diagram](#). Further details of the Networks in this reference architecture are located in [Appendix: Environment Details](#).

4.2.1. Ceph Cluster Network

The Ceph OSDs use this network to balance data according to the replication policy. This private network only needs to be accessed by the OSDs. In a hyper-converged deployment, the compute role needs access to this network. Thus, [Define the Overcloud](#), describes how to modify *nic-configs/compute-nics.yaml* to ensure that compute nodes are deployed with a connection to this network.

The Heat Provider that Red Hat OpenStack Platform director uses to define this network can be referenced with the following:

```
OS::TripleO::Network::StorageMgmt: /usr/share/openstack-tripleo-heat-templates/network/storage_mgmt.yaml
```

4.2.2. Red Hat Ceph Storage

The Ceph monitor nodes are accessed via this network. The Heat Provider that Red Hat OpenStack Platform director uses to define this network can be referenced with the following:

```
OS::TripleO::Network::Storage: /usr/share/openstack-tripleo-heat-templates/network/storage.yaml
```

4.2.3. External

Red Hat OpenStack Platform director uses the external network to download software updates for the overcloud, and the cloud operator uses this network to access director to manage the overcloud.

The Controllers use the external network to route traffic to the Internet for tenant services that are externally connected via reserved floating IPs. Overcloud users use the external network to access the

overcloud.

The Compute nodes do not need to be directly connected to the external network, as their instances communicate via the Tenant network to the Controllers who then route external traffic on their behalf to the external network.

The Heat Provider that Red Hat OpenStack Platform director uses to define this network can be referenced with the following:

```
OS::TripleO::Network::External: /usr/share/openstack-tripleo-heat-templates/network/external.yaml
```

4.2.4. OpenStack Internal API

OpenStack provides both public facing and private API endpoints. This is an isolated network for the private endpoints.

The Heat Provider that Red Hat OpenStack Platform director uses to define this network can be referenced with the following:

```
OS::TripleO::Network::InternalApi: /usr/share/openstack-tripleo-heat-templates/network/internal_api.yaml
```

4.2.5. OpenStack Tenant Network

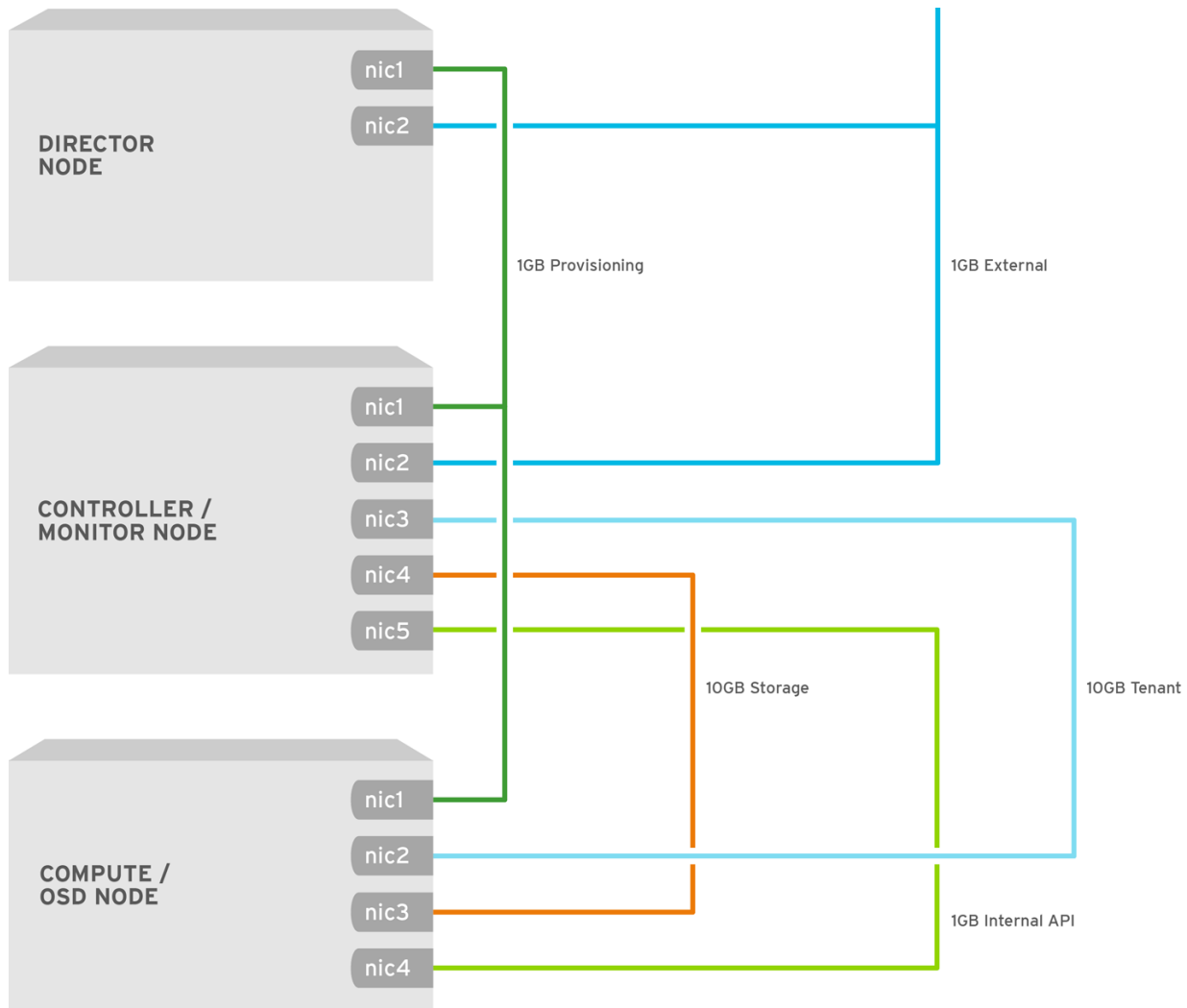
OpenStack tenants create private networks implemented by VLAN or VXLAN on this network.

The Heat Provider that Red Hat OpenStack Platform director uses to define this network can be referenced with the following:

```
OS::TripleO::Network::Tenant: /usr/share/openstack-tripleo-heat-templates/network/tenant.yaml
```

4.2.6. Red Hat OpenStack Platform director Provisioning

Red Hat OpenStack Platform director serves DHCP and PXE services from this network to install the operating system and other software on the overcloud nodes from baremetal. Red Hat OpenStack Platform director uses this network to manage the overcloud nodes, and the cloud operator uses it to access the overcloud nodes directly by `ssh` if necessary. The overcloud nodes must be configured to PXE boot from this network provisioning.



OPENSTACK_429440_0217

Figure 1. Network Separation Diagram



In [Figure 1. Network Separation Diagram](#), it is possible that the NICs could be a logical bond of two physical NICs and it is not required that each network be trunked to the same interface.

5. Prerequisites

Prior to deploying the overcloud, the undercloud needs to be deployed and the hardware to host the overcloud needs to be introspected by OpenStack's bare metal provisioning service, Ironic.

5.1. Deploy the Undercloud

To deploy Red Hat OpenStack Platform director, also known as the undercloud, complete Chapter 4, Installing the undercloud, of the Red Hat document [Director Installation and Usage](#). Be sure to complete the following sections of the referenced document before registering and introspecting hardware.

- 4.1. Creating a Director Installation User
- 4.2. Creating Directories for Templates and Images
- 4.3. Setting the Hostname for the System
- 4.4. Registering your System
- 4.5. Installing the Director Packages
- 4.6. Configuring the Director
- 4.7. Obtaining Images for Overcloud Nodes
- 4.8. Setting a Nameserver on the Undercloud's Neutron Subnet

This reference architecture used the following *undercloud.conf* when completing section 4.6 of the above.

```
[DEFAULT]
local_ip = 192.168.1.1/24
undercloud_public_vip = 192.168.1.10
undercloud_admin_vip = 192.168.1.11
local_interface = eth0
masquerade_network = 192.168.1.0/24
dhcp_start = 192.168.1.20
dhcp_end = 192.168.1.120
network_cidr = 192.168.1.0/24
network_gateway = 192.168.1.1

inspection_iprange = 192.168.1.150,192.168.1.180
inspection_interface = br-ctlplane
inspection_runbench = true
inspection_extras = false
inspection_enable_uefi = false
```

5.2. Register and Introspect Hardware

The registration and introspection of hardware requires a host definition file to provide the information that the OpenStack Ironic service needs to manage the hosts. The following host definition file, *instackenv.json*, provides an example of the servers being deployed in this reference architecture:

```
{
  "nodes": [
    {
      "pm_password": "PASSWORD",
      "name": "m630_slot14",
      "pm_user": "root",
      "pm_addr": "10.19.143.61",
      "pm_type": "pxe_ipmitool",
      "mac": [
        "c8:1f:66:65:33:44"
      ],
      "arch": "x86_64",
      "capabilities": "node:controller-0,boot_option:local"
    },
    ...
  ]
}
```

As shown in the example above, the **capabilities** entry contains both the server's role and server's number within that role, e.g. controller-0. This is done in order to predictably control node placement.

For this reference architecture, a custom role is created called **osd-compute** because servers in that role host both Ceph OSD and Nova Compute services. All servers used in the reference architecture are preassigned as a node in Ironic of either a controller or osd-compute. The host definition file contains the following capabilities entries:

```
$ grep capabilities instackenv.json
  "capabilities": "node:controller-0,boot_option:local"
  "capabilities": "node:controller-1,boot_option:local"
  "capabilities": "node:controller-2,boot_option:local"
  "capabilities": "node:osd-compute-0,boot_option:local"
  "capabilities": "node:osd-compute-1,boot_option:local"
  "capabilities": "node:osd-compute-2,boot_option:local"
$
```

For more information on assigning node specific identification, see section [8.1. Assigning Specific Node IDs](#) of the Red Hat document Advanced Openstack Customization.

As an optional parameter, a descriptive name of the server may be provided in the JSON file. The name shown in the following indicates that the server is in a blade chassis in slot 14.

```
"name": "m630_slot14",
```

To import the hosts described in `~/instackenv.json`, complete the following steps:

1. Populate the Ironic database with the file

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

2. Verify that the Ironic database was populated with all of the servers

```
openstack baremetal node list
```

3. Assign the kernel and ramdisk images to the imported servers

```
openstack baremetal configure boot
```

4. Via Ironic, use **IPMI** to turn the servers on, collect their properties, and record them in the Ironic database

```
openstack baremetal introspection bulk start
```



Bulk introspection time may vary based on node count and boot time. If `inspection_runbench = false` is set in `~/undercloud.conf`, then the introspection process shall not run and store the results of a **sysbench** and **fio** benchmark for each server. Though this makes the introspection take less time, e.g. less than five minutes for seven nodes in this reference implementation, Red Hat OpenStack Platform director will not capture additional hardware metrics that may be deemed useful.

5. Verify the nodes completed introspection without errors

```
[stack@hci-director ~]$ openstack baremetal introspection bulk status
```

Node UUID	Finished	Error
a94b75e3-369f-4b2d-b8cc-8ab272e23e89	True	None
7ace7b2b-b549-414f-b83e-5f90299b4af3	True	None
8be1d83c-19cb-4605-b91d-928df163b513	True	None
e8411659-bc2b-4178-b66f-87098a1e6920	True	None
04679897-12e9-4637-9998-af8bee30b414	True	None
48b4987d-e778-48e1-ba74-88a08edf7719	True	None

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

5.2.1. Set the Root Device

By default, Ironic images the first block device, identified as `/dev/sda`, with the operating system during deployment. This section covers how to change the block device to be imaged, known as the *root device*, by using *Root Device Hints*.

The Compute/OSD servers used for this reference architecture have the following hard disks with the following device file names as seen by the operating system:

- Twelve 1117GB SAS hard disks presented as `/dev/{sda, sdb, ..., sdl}`
- Three 400GB SATA SSD disks presented as `/dev/{sdm, sdn, sdo}`
- Two 277GB SAS hard disks configured in RAID1 presented as `/dev/sdp`

The RAID1 pair hosts the OS, while the twelve larger drives are configured as OSDs that journal to the SSDs. Since `/dev/sda` should be used for an OSD, Ironic needs to store which root device it should use instead of the default.

After introspection, Ironic stores the WWN and size of each server's block device. Since the RAID1 pair is both the smallest disk and the disk that should be used for the root device, the `openstack baremetal configure boot` command may be run a second time, after introspection, as below:

```
openstack baremetal configure boot --root-device=smallest
```

The above makes Ironic find the WWN of the smallest disk and then store a directive in its database to use that WWN for the root device when the server is deployed. Ironic does this for every server in its database. To verify that the directive was set for any particular server, run a command like the following:

```
[stack@hci-director ~]$ openstack baremetal node show r730xd_u33 | grep wwn  
| properties | {u'cpu_arch': u'x86_64', u'root_device': {u'wwn':  
u'0x614187704e9c7700'}, u'cpus': u'56', u'capabilities': u'node:osd-compute-  
2,cpu_hugepages:true,cpu_txt:true,boot_option:local,cpu_aes:true,cpu_vt:true,cpu_hugepage  
s_1g:true', u'memory_mb': u'262144', u'local_gb': 277}  
[stack@hci-director ~]$
```

In the above example `u'root_device': {u'wwn': u'0x614187704e9c7700'}` indicates that the root device is set to a specific WWN. The same command produces a similar result for each server. The server may be referred to by its name, as in the above example, but if the server does not have a name, then the UUID is used.

For the hardware used in this reference architecture, the size was a simple way to tell Ironic how to set the root device. For other hardware, other root device hints may be set using the vendor or model. If necessary, these values, in addition to the WWN and serial number, may be downloaded directly from Ironic's Swift container and be explicitly to set the root device for each node. An example of how to do this may be found in section [5.4. Defining the Root Disk for Nodes](#) of the Red Hat document [Director Installation and Usage](#). If the root device of each node needs to be set explicitly, then a script may be written to automate setting this value for a large deployment. Though in the example above, a simple root device hint abstracts this automation so that Ironic may handle it, even for a large number of nodes.

6. Define the Overcloud

The plan for the overcloud is defined in a set of Heat templates. This chapter covers the details of creating each Heat template to define the overcloud used in this reference architecture. As an alternative to creating new Heat templates, it is possible to directly download and modify the Heat templates used this reference architecture as described in the [Technical Summary](#).

6.1. Custom Environment Templates

The installation of the undercloud, covered in [Deploy the Undercloud](#), creates a directory of TripleO Heat Templates in `/usr/share/openstack-tripleo-heat-templates`. No direct customization of the TripleO Heat Templates shipped with Red Hat OpenStack Platform director is necessary; however, the creation of a separate directory called `~/custom-templates` shall be used to override default template values. Create the directory for the custom templates.

```
mkdir ~/custom-templates
```

The rest of this chapter consists of creating YAML files in the above directory to define the overcloud.

6.2. Network Configuration

In this section, the following three files are added to the `~/custom-templates` directory to define how the networks used in the reference architecture should be configured by Red Hat OpenStack Platform director:

- `~/custom-templates/network.yaml`
- `~/custom-templates/nic-configs/controller-nics.yaml`
- `~/custom-templates/nic-configs/compute-nics.yaml`

This section describes how to create new versions of the above files. It is possible to copy example files and then modify them based on the details of the environment. Complete copies of the above files, as used in this reference architecture, may be found in [Appendix: Custom Heat Templates](#). They may also be found online. See the [GitHub Repository of Example Files](#) Appendix for more details.

6.2.1. Assign OpenStack Services to Isolated Networks

Create a new file in `~/custom-templates` called `network.yaml` and add content to this file.

1. Add a `resource_registry` that includes two network templates

The `resource_registry` section contains references network configuration for the controller/monitor and compute/OSD nodes. The first three lines of `network.yaml` contains the following:

```
resource_registry:
  OS::TripleO::Controller::Net::SoftwareConfig: /home/stack/custom-templates/nic-
  configs/controller-nics.yaml
  OS::TripleO::Compute::Net::SoftwareConfig: /home/stack/custom-templates/nic-
  configs/compute-nics.yaml
```

The *controller-nics.yaml* and *compute-nics.yaml* files in the *nic-configs* directory above are created in the next section. Add the above to reference the empty files in the meantime.

2. Add the parameter defaults for the Neutron bridge mappings Tenant network

Within the *network.yaml*, add the following parameters:

```
parameter_defaults:
  NeutronBridgeMappings: 'datacentre:br-ex,tenant:br-tenant'
  NeutronNetworkType: 'vxlan'
  NeutronTunnelType: 'vxlan'
  NeutronExternalNetworkBridge: ''
```

The above defines the bridge mappings associated to the logical networks and enables tenants to use VXLAN.

3. Add parameter defaults based on the networks to be created

The Heat templates referenced in the `resource_registry` require parameters to define the specifics of each network. For example, the IP address range and VLAN ID for the storage network to be created. Under the `parameters_defaults` section to *network.yaml*, define the parameters for each network. The value of the parameters should be based on the networks that OpenStack deploys. For this reference architecture, the parameter values may be found in the Network section of [Appendix: Environment Details](#). These values are then supplied to the `parameters_defaults` section as follows:

```
# Internal API used for private OpenStack Traffic
InternalApiNetCidr: 192.168.2.0/24
InternalApiAllocationPools: [{'start': '192.168.2.10', 'end': '192.168.2.200'}]
InternalApiNetworkVlanID: 4049

# Tenant Network Traffic - will be used for VXLAN over VLAN
TenantNetCidr: 192.168.3.0/24
TenantAllocationPools: [{'start': '192.168.3.10', 'end': '192.168.3.200'}]
TenantNetworkVlanID: 4050

# Public Storage Access - e.g. Nova/Glance <--> Ceph
StorageNetCidr: 172.16.1.0/24
StorageAllocationPools: [{'start': '172.16.1.10', 'end': '172.16.1.200'}]
StorageNetworkVlanID: 4046

# Private Storage Access - i.e. Ceph background cluster/replication
StorageMgmtNetCidr: 172.16.2.0/24
StorageMgmtAllocationPools: [{'start': '172.16.2.10', 'end': '172.16.2.200'}]
StorageMgmtNetworkVlanID: 4047

# External Networking Access - Public API Access
ExternalNetCidr: 10.19.137.0/21
# Leave room for floating IPs in the External allocation pool (if required)
ExternalAllocationPools: [{'start': '10.19.139.37', 'end': '10.19.139.48'}]
# Set to the router gateway on the external network
ExternalInterfaceDefaultRoute: 10.19.143.254

# Gateway router for the provisioning network (or Undercloud IP)
ControlPlaneDefaultRoute: 192.168.1.1
# The IP address of the EC2 metadata server. Generally the IP of the Undercloud
EC2MetadataIp: 192.168.1.1
# Define the DNS servers (maximum 2) for the Overcloud nodes
DnsServers: ["10.19.143.247", "10.19.143.248"]
```

For more information on the above directives see section 6.2, Isolating Networks, of the Red Hat document [Advanced Overcloud Customization](#).

6.2.2. Define Server NIC Configurations

Within the *network.yaml* file, references were made to controller and compute Heat templates which need to be created in *~/custom-templates/nic-configs/*. Complete the following steps to create these files:

1. Create a *nic-configs* directory within the *~/custom-templates* directory

```
mkdir ~/custom-templates/nic-configs
```

2. Copy the appropriate sample network interface configurations

Red Hat OpenStack Platform director contains a directory of network interface configuration templates for the following four scenarios:

```
$ ls /usr/share/openstack-tripleo-heat-templates/network/config/  
bond-with-vlans  multiple-nics  single-nic-linux-bridge-vlans  single-nic-vlans  
$
```

In this reference architecture, VLANs are trunked onto a single NIC. The following is creates the *compute.yaml* and *controller.yaml* files:

```
cp /usr/share/openstack-tripleo-heat-templates/network/config/single-nic-  
vlans/compute.yaml ~/custom-templates/nic-configs/compute-nics.yaml
```

```
cp /usr/share/openstack-tripleo-heat-templates/network/config/single-nic-  
vlans/controller.yaml ~/custom-templates/nic-configs/controller-nics.yaml
```

3. Modify the Controller NICs template

Modify *~/custom-templates/nic-configs/controller-nics.yaml* based on the hardware that hosts the controller as described in [6.2. Creating a Network Environment File](#) of the Red Hat document *Advanced Overcloud Customization*.

4. Modify the Compute NICs template

Modify *~/custom-templates/nic-configs/compute-nics.yaml* as described in the previous step. Extend the provided template to include the *StorageMgmtIpSubnet* and *StorageMgmtNetworkVlanID* attributes of the storage management network. When defining the interface entries of the storage network, consider setting the MTU to 9000 (jumbo frames) for improved storage performance. An example of these additions to *compute-nics.yaml* includes the following:

```
-
  type: interface
  name: em2
  use_dhcp: false
  mtu: 9000
-
  type: vlan
  device: em2
  mtu: 9000
  use_dhcp: false
  vlan_id: {get_param: StorageMgmtNetworkVlanID}
  addresses:
    -
      ip_netmask: {get_param: StorageMgmtIpSubnet}
-
  type: vlan
  device: em2
  mtu: 9000
  use_dhcp: false
  vlan_id: {get_param: StorageNetworkVlanID}
  addresses:
    -
      ip_netmask: {get_param: StorageIpSubnet}
```



In order to prevent network misconfigurations from taking overcloud nodes out of production, network changes like MTU settings must be made during initial deployment, and may not yet be applied via Red Hat OpenStack Platform director retroactively to an existing deployment. Thus, if this setting is desired, then it should be set before the deployment.



All network switch ports between servers using the interface with the new MTU must be updated to support jumbo frames if the above setting is made. If this change is not made on the switch, then problems may manifest on the application layer that could cause the Ceph cluster to not reach quorum. If the setting above was made, and these types of problems are observed, then verify that all hosts using the network using jumbo frames can communicate at the desired MTU with a command like `ping -M do -s 8972 172.16.1.11`.

Complete versions of *compute-nics.yaml* and *controller-nics.yaml*, as used in this reference architecture, may be found in [Appendix: Custom Resources and Parameters](#). They may also be found online. See the [GitHub Repository of Example Files](#) Appendix for more details.

6.3. Hyper Converged Role Definition

This section covers how *composable roles* are used to create a new role called *OsdCompute*, that offers both Nova compute and Ceph OSD services.



Red Hat OpenStack Platform director 10 ships with the environment file `/usr/share/openstack-tripleo-heat-templates/environments/hyperconverged-ceph.yaml`, which merges the OSD service into the compute service for a hyperconverged compute role. This reference architecture does not use this template and instead composes a new role. This approach, of using composable roles, allows for overcloud deployments consisting of both compute nodes without OSDs mixed with compute nodes with OSDs. In addition, it allows for converged Compute/OSD nodes with differing OSD counts. It also allows the same deployment to contain OSD servers that do not run compute services. All of this is possible, provided that a new role is composed for each.

6.3.1. Composable Roles

By default the overcloud consists of five roles: Controller, Compute, BlockStorage, ObjectStorage, and CephStorage. Each role consists of a list of services. As of Red Hat OpenStack Platform 10, the services that are deployed per role may be seen in `/usr/share/openstack-tripleo-heat-templates/roles_data.yaml`.

It is possible to make a copy of `roles_data.yaml` and then define a new role within it consisting of any mix of available services found under other roles. This reference architecture follows this procedure to create a new role called *OsdCompute*. For more information about Composable Roles themselves, see the [Composable Roles](#) section of the Red Hat document *Advanced Overcloud Customization*.

6.3.2. Custom Template

Copy the roles data file to the custom templates directory for modification.

```
cp /usr/share/openstack-tripleo-heat-templates/roles_data.yaml ~/custom-templates/custom-roles.yaml
```

Edit `~/custom-templates/custom-roles.yaml` to add the following to the bottom of the file which defines a new role called *OsdCompute*.

```
- name: OsdCompute
  CountDefault: 0
  HostnameFormatDefault: '%stackname%-osd-compute-%index%'
  ServicesDefault:
    - OS::TripleO::Services::CephOSD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::NeutronSriovAgent
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::VipHosts
```

The list of services under `ServicesDefault` is identical to the list of services under the Compute role, except the CephOSD service has been added to the list. The `CountDefault` of 0 ensures that no nodes from this new role are deployed unless explicitly requested and the `HostnameFormatDefault` defines what each node should be called when deployed, e.g. overcloud-osd-compute-0, overcloud-osd-compute-1, etc.

In the [Deployment](#) section, the new `~/custom-templates/custom-roles.yaml`, which contains the above, is passed to the `openstack overcloud deploy` command.

6.4. Ceph Configuration

In this section, the following three files are added to the `~/custom-templates` directory to define how the Ceph configurations used in the reference architecture should be configured by Red Hat OpenStack Platform director:

- `~/custom-templates/ceph.yaml`
- `~/custom-templates/first-boot-template.yaml`

- `~/custom-templates/post-deploy-template.yaml`

This section describes how to create new versions of the above files. It is possible to copy example files and then modify them based on the details of the environment. Complete copies of the above files, as used in this reference architecture, may be found in [Appendix: Custom Heat Templates](#). They may also be found online. See the [GitHub Repository of Example Files](#) Appendix for more details.

Create a file in `~/custom-templates/` called `ceph.yaml`. In the next two subsections, content is added to this file.

6.4.1. Set the Resource Registry for Ceph

To set the resource registry for Ceph, add a `resource_registry` section to which includes a first-boot template and a post-deploy template to `ceph.yaml`.

```
resource_registry:
  OS::Triple0::NodeUserData: /home/stack/custom-templates/first-boot-template.yaml
  OS::Triple0::NodeExtraConfigPost: /home/stack/custom-templates/post-deploy-
template.yaml
```

The `first-boot-template.yaml` and `post-deploy-template.yaml` files above are used to configure Ceph during the deployment and are created in the next subsection.

Create the Firstboot Template

A Ceph deployment may fail to add an OSD or OSD journal disk under either of the following conditions:

1. The disk has an FSID from a previous Ceph install
2. The disk does not have a GPT disk label

The conditions above are avoided by preparing a disk with the following commands.

1. Erase all GPT and MBR data structures, including the FSID, with `sgdisk -Z $disk`
2. Convert an MBR or BSD disklabel disk to a GPT disk with `sgdisk -g $disk`

Red Hat OpenStack Platform director is configured to run the above commands on all disks, except the root disk, when initially deploying a server that hosts OSDs by using a firstboot Heat template.

Create a file called `~/custom-templates/first-boot-template.yaml` whose content is the following:


```
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}
```

Create a file called `~/custom-templates/wipe-disk.sh`, to be called by the above, whose content is the following:

```
#!/usr/bin/env bash
if [[ `hostname` = *"ceph"* ]] || [[ `hostname` = *"osd-compute"* ]]
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/${DEVICE}
        else
            echo "Root found in /dev/${DEVICE}"
        fi
    done
fi
```

Both *first-boot-template.yaml* and *wipe-disk.sh* are derivative works of the Red Hat document, [Red Hat Ceph Storage for the OpenStack section 2.9. Formatting Ceph Storage Node Disks to GPT](#). The *wipe-disk.sh* script has been modified to wipe all disks, except the one mounted by /, but only if hostname matches the pattern **ceph** or **osd-compute**.



The firstboot heat template, which is run by cloud-init when a node is first deployed, deletes data. If any data from a previous Ceph install is present, then it will be deleted. If this is not desired, then comment out the `OS::TripleO::NodeUserData` line with a `#` in the `~/custom-templates/ceph.yaml` file.

Create the Post Deploy Template

Post deploy scripts may used to run arbitrary shell scripts after the configuration built into TripleO, mostly implemented in Puppet, has run. Because some of the configuration done in [Resource Isolation and Tuning](#) is not presently configurable in the Puppet triggered by Red Hat OpenStack Platform

director, a Heat template to run a shell script will be put in place in this section and modified later.

Create a file called `~/custom-templates/post-deploy-template.yaml` whose content is the following:

```
heat_template_version: 2014-10-16

parameters:
  servers:
    type: json

resources:

  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      inputs:
        - name: OSD_NUMA_INTERFACE
      config: |
        #!/usr/bin/env bash
        {
          echo "TODO: pin OSDs to the NUMA node of $OSD_NUMA_INTERFACE"
        } 2>&1 > /root/post_deploy_heat_output.txt

  ExtraDeployments:
    type: OS::Heat::SoftwareDeployments
    properties:
      servers: {get_param: servers}
      config: {get_resource: ExtraConfig}
      input_values:
        OSD_NUMA_INTERFACE: 'em2'
      actions: ['CREATE']
```

When an overcloud is deployed with the above Heat environment template, the following would be found on each node.

```
[root@overcloud-osd-compute-2 ~]# cat /root/post_deploy_heat_output.txt
TODO: pin OSDs to the NUMA node of em2
[root@overcloud-osd-compute-2 ~]#
```

The `OSD_NUMA_INTERFACE` variable and embedded shell script will be modified in the [Ceph NUMA Pinning](#) section so that instead of logging that the OSDs *need to be* NUMA pinned, the systemd unit file for the OSD service will be modified by the script and restart the OSD services so that they *are* NUMA pinned.

6.4.2. Set the Parameter Defaults for Ceph

Add a parameter defaults section in `~/custom-templates/ceph.yaml` under the resource registry defined in the previous subsection.

Add parameter defaults for Ceph OSD tunables

As described in the [Red Hat Ceph Storage Strategies Guide](#), the following OSD values may be tuned to affect the performance of a Red Hat Ceph Storage cluster.

- Journal size (`journal_size`)
- Placement Groups (`pg_num`)
- Placement Group for placement purpose (`pgp_num`)
- Number of replicas for objects in the pool (`default_size`)
- Minimum number of written replicas for objects in a pool in order to acknowledge a write operation to the client (`default_min_size`)
- Recovery operations to be run in the event of OSD loss (`recovery_max_active` and `recovery_op_priority`)
- Backfill operations to be run in the event of OSD loss (`max_backfills`)

All of these values are set for the overcloud deployment by using the following in the `parameter_defaults` section of `~/custom-templates/ceph.yaml`. These parameters are passed as `ExtraConfig` when they benefit both the Ceph OSD and Monitor nodes, or passed as extra configuration only for the custom role, `OsdCompute`, by using `OsdComputeExtraConfig`.

```
parameter_defaults:
  ExtraConfig:
    ceph::profile::params::osd_pool_default_pg_num: 256
    ceph::profile::params::osd_pool_default_pgp_num: 256
    ceph::profile::params::osd_pool_default_size: 3
    ceph::profile::params::osd_pool_default_min_size: 2
    ceph::profile::params::osd_recovery_op_priority: 2
  OsdComputeExtraConfig:
    ceph::profile::params::osd_journal_size: 5120
```

The values provided above are reasonable example values for the size of the deployment in this reference architecture. See the [Red Hat Ceph Storage Strategies Guide](#) to determine the appropriate values for a larger number of OSDs. The recovery and backfill options above were chosen deliberately for a hyper-converged deployment, and details on these values are covered in [Reduce Ceph Backfill and Recovery Operations](#).

Add parameter defaults for Ceph OSD block devices

In this subsection, a list of block devices are defined. The list should be appended directly to the `OsdComputeExtraConfig` defined in the previous subsection.

The Compute/OSD servers used for this reference architecture have the following disks for Ceph:

- Twelve 1117GB SAS hard disks presented as `/dev/{sda, sdb, ..., sdl}` are used for OSDs
- Three 400GB SATA SSD disks presented as `/dev/{sdm, sdn, sdo}` are used for OSD journals

To configure Red Hat OpenStack Platform director to create four partitions on each SSD, to be used as a Ceph journal by each hard disk, the following list may be defined in Heat (it is not necessary to specify the partition number):

```
ceph::profile::params::osds:  
  '/dev/sda':  
    journal: '/dev/sdm'  
  '/dev/sdb':  
    journal: '/dev/sdm'  
  '/dev/sdc':  
    journal: '/dev/sdm'  
  '/dev/sdd':  
    journal: '/dev/sdm'  
  '/dev/sde':  
    journal: '/dev/sdn'  
  '/dev/sdf':  
    journal: '/dev/sdn'  
  '/dev/sdg':  
    journal: '/dev/sdn'  
  '/dev/sdh':  
    journal: '/dev/sdn'  
  '/dev/sdi':  
    journal: '/dev/sdo'  
  '/dev/sdj':  
    journal: '/dev/sdo'  
  '/dev/sdk':  
    journal: '/dev/sdo'  
  '/dev/sdl':  
    journal: '/dev/sdo'
```

The above should be added under the `parameter_defaults, OsdComputeExtraConfig` (under `ceph::profile::params::osd_journal_size: 5120`) in the `~/custom-templates/ceph.yaml` file. The complete file may be found in [Appendix: Custom Resources and Parameters](#). It may also be found online. See the [GitHub Repository of Example Files](#) Appendix for more details.

6.5. Overcloud Layout

This section creates a new custom template called *layout.yaml* to define the following properties:

- For each node type, how many of those nodes should be deployed?
- For each node, what specific server should be used?
- For each isolated network per node, which IP addresses should be assigned?
- Which isolated networks should the *OsdCompute* role use?

It also passes other parameters, which in prior versions of Red Hat OpenStack Platform, would be passed through the command line.

6.5.1. Configure the ports for both roles to use a pool of IPs

Create the file *~/custom-templates/layout.yaml* and add the following to it:

```
resource_registry:

    OS::TripleO::Controller::Ports::InternalApiPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/internal_api_from_pool.yaml
    OS::TripleO::Controller::Ports::TenantPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/tenant_from_pool.yaml
    OS::TripleO::Controller::Ports::StoragePort: /usr/share/openstack-tripleo-heat-
templates/network/ports/storage_from_pool.yaml
    OS::TripleO::Controller::Ports::StorageMgmtPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/storage_mgmt_from_pool.yaml

    OS::TripleO::OsdCompute::Ports::InternalApiPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/internal_api_from_pool.yaml
    OS::TripleO::OsdCompute::Ports::TenantPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/tenant_from_pool.yaml
    OS::TripleO::OsdCompute::Ports::StoragePort: /usr/share/openstack-tripleo-heat-
templates/network/ports/storage_from_pool.yaml
    OS::TripleO::OsdCompute::Ports::StorageMgmtPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/storage_mgmt_from_pool.yaml
```

The above defines the ports that are used by the *Controller* role and *OsdCompute* role. Each role has four lines of very similar template includes, but the *Controller* lines override what is already defined by the default for the *Controller* role, in order to get its IPs from a list defined later in this file. In contrast, the *OsdCompute* role has no default ports to override, as it is a new custom role. The ports that it uses are defined in this Heat template. In both cases the IPs that each port receives are defined in a list under *ControllerIPs* or *OsdComputeIPs* which will be added to the *layout.yaml* file in the next section.

6.5.2. Define the node counts and other parameters

This subsection defines for each node type, how many of which should be deployed. Under the `resource_registry` of `~/custom-templates/layout.yaml`, add the following:

```
parameter_defaults:
  NtpServer: 10.5.26.10

  ControllerCount: 3
  ComputeCount: 0
  CephStorageCount: 0
  OsdComputeCount: 3
```

In prior versions of Red Hat OpenStack Platform, the above parameters would have been passed through the command line, e.g. `--ntp-server` can now be passed in a Heat template as `NtpServer` as in the example above.

The above indicates that three *Controller* nodes are deployed and that three *OsdCompute* nodes are deployed. In [Adding Compute/Red Hat Ceph Storage Nodes](#) the value for `OsdComputeCount` is set to four to deploy an additional *OsdCompute* node. For this deployment, it is necessary to set the number of *Compute* nodes to zero because the `CountDefault` for a *Compute* node is 1, as can be verified by reading `/usr/share/openstack-tripleo-heat-templates/roles_data.yaml`. It is not necessary set the `CephStorageCount` to 0, because none are deployed by default. However, this parameter is included in this example to demonstrate how to add separate Ceph OSD servers that do not offer Nova compute services, and similarly the `ComputeCount` could be changed to offer Nova compute servers that do not offer Ceph OSD services. The process to mix hyper-converged Ceph/OSD nodes with non-hyper-converged Ceph or OSD nodes in a single deployment, is to increase these counts and define the properties of either standard role.

6.5.3. Configure scheduler hints to control node placement and IP assignment

In the [Register and Introspect Hardware](#) section, each node is assigned a *capabilities* profile of either *controller-X* or *osd-compute-Y*, where *X* or *Y* was the number of the physical node. These labels are used by the *scheduler hints* feature in Red Hat OpenStack Platform director to define specific node placement and ensure that a particular physical node is always be deployed to the overcloud with the same assignment; e.g. to ensure that the server in rack U33 is always *osd-compute-2*.

Append the following to `layout.yaml` under the `parameter_defaults` stanza to implement the predictable node placement described above.

```
ControllerSchedulerHints:
  'capabilities:node': 'controller-%index%'
NovaComputeSchedulerHints:
  'capabilities:node': 'compute-%index%'
CephStorageSchedulerHints:
  'capabilities:node': 'ceph-storage-%index%'
OsdComputeSchedulerHints:
  'capabilities:node': 'osd-compute-%index%'
```

Add the following to *layout.yaml* under the `parameter_defaults` stanza above to ensure that each node gets a specific IP.


```
ControllerIPs:
  internal_api:
    - 192.168.2.200
    - 192.168.2.201
    - 192.168.2.202
  tenant:
    - 192.168.3.200
    - 192.168.3.201
    - 192.168.3.202
  storage:
    - 172.16.1.200
    - 172.16.1.201
    - 172.16.1.202
  storage_mgmt:
    - 172.16.2.200
    - 172.16.2.201
    - 172.16.2.202
```

```
OsdComputeIPs:
  internal_api:
    - 192.168.2.203
    - 192.168.2.204
    - 192.168.2.205
  tenant:
    - 192.168.3.203
    - 192.168.3.204
    - 192.168.3.205
  storage:
    - 172.16.1.203
    - 172.16.1.204
    - 172.16.1.205
  storage_mgmt:
    - 172.16.2.203
    - 172.16.2.204
    - 172.16.2.205
```

The above specifies that the following predictable IP assignment will happen for each deploy:

- controller-0 will have the IPs
 - 192.168.2.200
 - 192.168.3.200
 - 172.16.1.200
 - 172.16.2.200
- controller-1 will have the IPs

- 192.168.2.201
- 192.168.3.201
- 172.16.1.201
- 172.16.2.201

and so on for the controller nodes and:

- osd-compute-0 will have the IPs
 - 192.168.2.203
 - 192.168.3.203
 - 172.16.1.203
 - 172.16.2.203
- osd-compute-1 will have the IPs
 - 192.168.2.204
 - 192.168.3.204
 - 172.16.1.204
 - 172.16.2.204

and so on for the osd-compute nodes.

For more information on assigning node specific identification, see section [7.1. Assigning Specific Node IDs](#) of the Red Hat document *Advanced Overcloud Customization*.

7. Resource Isolation and Tuning

This chapter is similar to the [Define the Overcloud](#) chapter covered previously in that it should result in changes made to the Heat environment files in the `~/custom-templates` directory. However, it differs in that the changes are made not to define the overcloud but to tune it in order to improve performance and isolate resources.

Isolating resources is important in a hyper-converged deployment because contention between Ceph and OpenStack could result in degradation of either service, and neither service is aware of the other's presence on the same physical host.

7.1. Nova Reserved Memory and CPU Allocation Ratio

In this section the reasoning behind how to tune the Nova settings for `reserved_host_memory_mb` and `cpu_allocation_ratio` is explained. A Python program is provided which takes as input properties of the hardware and planned workload and recommends the `reserved_host_memory_mb` and `cpu_allocation_ratio`. The settings provided favor making a hyper-converged deployment stable over maximizing the number of possible guests. Red Hat recommends starting with these defaults and testing specific workloads targeted at the OpenStack deployment. If necessary, these settings may be changed to find the desired trade off between determinism and guest-hosting capacity. The end of this section covers how to deploy the settings using Red Hat OpenStack Platform director.

7.1.1. Nova Reserved Memory

Nova's `reserved_host_memory_mb` is the amount of memory in MB to reserve for the host. If a node is dedicated only to offering compute services, then this value should be set to maximize the number of running guests. However, on a system that must also support Ceph OSDs, this value needs to be increased so that Ceph has access to the memory that it needs.

To determine the `reserved_host_memory_mb` for a hyper-converged node, assume that each OSD consumes 3GB of RAM. Given a node with 256GB of RAM and 10 OSDs, 30GB of RAM is used for Ceph and 226GB of RAM is available for Nova. If the average guest uses the `m1.small` flavor, which uses 2GB of RAM per guest, then the overall system could host 113 such guests. However there is an additional overhead to account for per guest for the hypervisor. Assume this overhead is a half GB. With this overhead taken into account, the maximum number of 2GB guests that could be run would be 226GB divided by 2.5GB of RAM, which is approximately 90 virtual guests.

Given this number of guests and the number of OSDs, the amount of memory to reserve that Nova cannot use would be the amount of guests times their overhead plus the amount of OSDs times the amount of RAM that each OSD should have. In other words, $(90 * 0.5) + (10 * 3)$, which is 75GB. Nova expects this value in MB and thus 75000 would be provided to the `nova.conf`.

These ideas may be expressed mathematically in the following Python code:

```
left_over_mem = mem - (GB_per_OSD * osds)
number_of_guests = int(left_over_mem /
                        (average_guest_size + GB_overhead_per_guest))
nova_reserved_mem_MB = MB_per_GB * (
    (GB_per_OSD * osds) +
    (number_of_guests * GB_overhead_per_guest))
```

The above is from the Nova Memory and CPU Calculator, which is covered in a future section of this paper.

7.1.2. Nova CPU Allocation Ratio

Nova's `cpu_allocation_ratio` is used by the Nova scheduler when choosing compute nodes to run guests. If the ratio has the default of 16:1 and the number of cores on a node, also known as vCPUs, is 56, then the Nova scheduler may schedule enough guests to consume 896 vCPUs before it considers the node unable to handle any more guests. Because the Nova scheduler does not take into account the CPU needs of Ceph OSD services running on the same node, the `cpu_allocation_ratio` should be modified so that Ceph has the CPU resources it needs to operate effectively without those CPU resources being given to Nova.

To determine the `cpu_allocation_ratio` for a hyper-converged node, assume that at least one core is used by each OSD (unless the workload is IO intensive). Given a node with 56 cores and 10 OSDs, that leaves 46 cores for Nova. If each guest uses 100% of the CPU that it is given, then the ratio should be the number of guest vCPUs divided by the number of cores; that is, 46 divided by 56, or 0.8. However, because guests don't usually consume 100% of their CPUs, the ratio should be raised by taking the anticipated percentage into account when determining the number of required guest vCPUs. So, if only 10%, or 0.1, of a vCPU is used by a guest, then the number of vCPUs for guests is 46 divided by 0.1, or 460. When this value is divided by the number of cores, 56, the ratio increases to approximately 8.

These ideas may be expressed mathematically in the following Python code:

```
cores_per_OSD = 1.0
average_guest_util = 0.1 # 10%
nonceph_cores = cores - (cores_per_OSD * osds)
guest_vCPUs = nonceph_cores / average_guest_util
cpu_allocation_ratio = guest_vCPUs / cores
```

The above is from the Nova Memory and CPU Calculator covered in the next section.

7.1.3. Nova Memory and CPU Calculator

The formulas covered above are in a script called *nova_mem_cpu_calc.py*, which is available in [Appendix: Nova Memory and CPU Calculator](#). It takes the following ordered parameters as input:

1. Total host RAM in GB
2. Total host cores
3. Ceph OSDs per server
4. Average guest size in GB
5. Average guest CPU utilization (0.0 to 1.0)

It prints as output a recommendation for how to set the `nova.conf` `reserved_host_memory_mb` and `cpu_allocation_ratio` to favor stability of a hyper-converged deployment. When the numbers from the example discussed in the previous section are provided to the script, it returns the following results.

```
$ ./nova_mem_cpu_calc.py 256 56 10 2 1.0
```

Inputs:

- Total host RAM in GB: 256
- Total host cores: 56
- Ceph OSDs per host: 10
- Average guest memory size in GB: 2
- Average guest CPU utilization: 100%

Results:

- number of guests allowed based on memory = 90
- number of guest vCPUs allowed = 46
- nova.conf reserved_host_memory = 75000 MB
- nova.conf cpu_allocation_ratio = 0.821429

Compare "guest vCPUs allowed" to "guests allowed based on memory" for actual guest count
\$

The amount of possible guests is bound by the limitations of either the CPU or the memory of the overcloud. In the example above, if each guest is using 100% of its CPU and there are only 46 vCPUs available, then it is not possible to launch 90 guests, even though there is enough memory to do so. If the anticipated guest CPU utilization decreases to only 10%, then the number of allowable vCPUs increases along with the `cpu_allocation_ratio`.

```
$ ./nova_mem_cpu_calc.py 256 56 10 2 0.1
```

Inputs:

- Total host RAM in GB: 256
- Total host cores: 56
- Ceph OSDs per host: 10
- Average guest memory size in GB: 2
- Average guest CPU utilization: 10%

Results:

- number of guests allowed based on memory = 90
- number of guest vCPUs allowed = 460
- nova.conf reserved_host_memory = 75000 MB
- nova.conf cpu_allocation_ratio = 8.214286

Compare "guest vCPUs allowed" to "guests allowed based on memory" for actual guest count
\$

After determining the desired values of the `reserved_host_memory_mb` and `cpu_allocation_ratio`, proceed to the next section to apply the new settings.

7.1.4. Change Nova Reserved Memory and CPU Allocation Ratio with Heat

Create the new file `~/custom-templates/compute.yaml` containing the following:

```
parameter_defaults:
  ExtraConfig:
    nova::compute::reserved_host_memory: 75000
    nova::cpu_allocation_ratio: 8.2
```

In the above example *ExtraConfig* is used to change the amount of memory that the Nova compute service reserves in order to protect both the Ceph OSD service and the host itself. Also, in the above example, *ExtraConfig* is used to change the Nova CPU allocation ratio of the Nova scheduler service so that it does not allocate any of the CPUs that the Ceph OSD service uses.



Red Hat OpenStack Platform director refers to the `reserved_host_memory_mb` variable used by Nova as `reserved_host_memory`.

To verify that the reserved host memory and CPU allocation ratio configuration changes were applied after [Deployment](#), `ssh` into any of the `OsdCompute` nodes and look for the configuration change in the `nova.conf`.

```
[root@overcloud-osd-compute-0 ~]# grep reserved_host_memory /etc/nova/nova.conf
reserved_host_memory_mb=75000
[root@overcloud-osd-compute-0 ~]#
```

```
[root@overcloud-osd-compute-0 ~]# grep cpu_allocation_ratio /etc/nova/nova.conf
cpu_allocation_ratio=8.2
[root@overcloud-osd-compute-0 ~]#
```

7.1.5. Updating the Nova Reserved Memory and CPU Allocation Ratio

The Overcloud workload may vary over time so it is likely that the `reserved_host_memory` and `cpu_allocation_ratio` will need to be changed. To do so after [Deployment](#), simply update the the values in `compute.yaml` and re-run the deployment command covered in [Deployment](#). More details on overcloud updates are in the [Configuration Updates](#) section.

7.2. Ceph NUMA Pinning

For systems which run both Ceph OSD and Nova Compute services, determinism can be improved by pinning Ceph to one of the available two NUMA nodes in a two socket x86 server. The socket to which Ceph should be pinned is the one that has the network IRQ and the storage controller. This choice is made because of a Ceph OSD's heavy use of network IO. The steps below describe how to create an Red Hat OpenStack Platform director post deployscript so that Ceph OSD daemons are NUMA pinned to a particular CPU socket when they are started.

7.2.1. Update the Post Deploy Script

In the [Ceph Configuration](#) section, a post-deploy-template was added to the resource registry of `ceph.yaml`. That post deploy template originally contained only the following:

```
heat_template_version: 2014-10-16

parameters:
  servers:
    type: json

resources:

  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      inputs:
        - name: OSD_NUMA_INTERFACE
      config: |
        #!/usr/bin/env bash
        {
          echo "TODO: pin OSDs to the NUMA node of $OSD_NUMA_INTERFACE"
        } 2>&1 > /root/post_deploy_heat_output.txt

  ExtraDeployments:
    type: OS::Heat::SoftwareDeployments
    properties:
      servers: {get_param: servers}
      config: {get_resource: ExtraConfig}
      input_values:
        OSD_NUMA_INTERFACE: 'em2'
      actions: ['CREATE']
```

The next two subsections will update the above file.

Set the Ceph Service Network Interface

The above Heat environment file has the following parameter:

```
OSD_NUMA_INTERFACE: 'em2'
```

Set the above to the name of the network device on which the Ceph services listen. In this reference architecture the device is `em2`, but the value may be determined for all deployments by either the `StorageNetwork` variable, or the `StorageMgmtNetwork` variable that was set in the [Network Configuration](#) section. Workloads that are read-heavy benefit from using the `StorageNetwork` variable, while workloads that are write-heavy benefit from using the `StorageMgmtNetwork` variable. In this reference architecture both networks are VLANs on the same interface.



If the Ceph OSD service uses a virtual network interface, like a bond, then use the name of the network devices that make up the bond, not the bond name itself. For example, if `bond1` uses `em2` and `em4`, then set `OSD_NUMA_INTERFACE` to either `em2` or `em4`, not `bond1`. If the `OSD_NUMA_INTERFACE` variable is set to a bond name, then the NUMA node will not be found and the Ceph OSD service will not be pinned to either NUMA node. This is because the `lstopo` command will not return virtual devices.

Modify the Shell Script

The following section of `custom-templates/post-deploy-template.yaml` contains a Heat config line and then embeds a shell script:

```
config: |
  #!/usr/bin/env bash
  {
    echo "TODO: pin OSDs to the NUMA node of $OSD_NUMA_INTERFACE"
  } 2>&1 > /root/post_deploy_heat_output.txt
```

Update the above so that rather than *embed* a simple shell script, it instead *includes* a more complex shell script in a separate file using Heat's `get_file` intrinsic function.

```
config: {get_file: numa-systemd-osd.sh}
```

The above change calls the script `numa-systemd-osd.sh`, which takes the network interface used for Ceph network traffic as an argument, and then uses `lstopo` to determine that interface's NUMA node. It then modifies the systemd unit file for the Ceph OSD service so that `numactl` is used to start the OSD service with a NUMA policy that prefers the NUMA node of the Ceph network's interface. It then restarts each Ceph OSD daemon sequentially so that the service runs with the new NUMA option.

When `numa-systemd-osd.sh` is run directly on a `osd_compute` node (with `OSD_NUMA_INTERFACE` set within the shell script), its output looks like the following:

```
[root@overcloud-osd-compute-0 ~]# ./numa-systemd-osd.sh
changed: --set /usr/lib/systemd/system/ceph-osd@.service Service ExecStart
'/usr/bin/numactl -N 0 --preferred=0 /usr/bin/ceph-osd -f --cluster ${CLUSTER} --id %i
--setuser ceph --setgroup ceph'

Status of OSD 1 before unit file update

* ceph-osd@1.service - Ceph object storage daemon
   Loaded: loaded (/usr/lib/systemd/system/ceph-osd@.service; disabled; vendor preset:
disabled)
   Active: active (running) since Fri 2016-12-16 02:50:02 UTC; 11min ago
   Main PID: 83488 (ceph-osd)
```

```

CGroup: /system.slice/system-ceph\x2dosd.slice/ceph-osd@1.service
└─83488 /usr/bin/ceph-osd -f --cluster ceph --id 1 --setuser ceph --setgroup
ceph

Dec 16 02:50:01 overcloud-osd-compute-0.localdomain systemd[1]: Starting Ceph object
storage daemon...
Dec 16 02:50:02 overcloud-osd-compute-0.localdomain ceph-osd-prestart.sh[83437]: create-
or-move update...
Dec 16 02:50:02 overcloud-osd-compute-0.localdomain systemd[1]: Started Ceph object
storage daemon.
Dec 16 02:50:02 overcloud-osd-compute-0.localdomain numactl[83488]: starting osd.1 at :/0
osd_data /v...l
Dec 16 02:50:02 overcloud-osd-compute-0.localdomain numactl[83488]: 2016-12-16
02:50:02.544592 7fecba...}
Dec 16 03:01:19 overcloud-osd-compute-0.localdomain systemd[1]:
[/usr/lib/systemd/system/ceph-osd@s...e'
Hint: Some lines were ellipsized, use -l to show in full.

Restarting OSD 1...

Status of OSD 1 after unit file update

* ceph-osd@1.service - Ceph object storage daemon
   Loaded: loaded (/usr/lib/systemd/system/ceph-osd@.service; disabled; vendor preset:
disabled)
   Active: active (running) since Fri 2016-12-16 03:01:21 UTC; 7ms ago
   Process: 89472 ExecStartPre=/usr/lib/ceph/ceph-osd-prestart.sh --cluster ${CLUSTER}
--id %i (code=exited, status=0/SUCCESS)
   Main PID: 89521 (numactl)
   CGroup: /system.slice/system-ceph\x2dosd.slice/ceph-osd@1.service
           └─89521 /usr/bin/numactl -N 0 --preferred=0 /usr/bin/ceph-osd -f --cluster
ceph --id 1 --se...

Dec 16 03:01:21 overcloud-osd-compute-0.localdomain systemd[1]: Starting Ceph object
storage daemon...
Dec 16 03:01:21 overcloud-osd-compute-0.localdomain ceph-osd-prestart.sh[89472]: create-
or-move update...
Dec 16 03:01:21 overcloud-osd-compute-0.localdomain systemd[1]: Started Ceph object
storage daemon.
Hint: Some lines were ellipsized, use -l to show in full.

Status of OSD 11 before unit file update
...
```

The logs of the node should indicate that `numactl` was used to start the OSD service.

```
[root@overcloud-osd-compute-0 ~]# journalctl | grep numa | grep starting
Dec 16 02:50:02 overcloud-osd-compute-0.localdomain numactl[83488]: starting osd.1 at :/0
osd_data /var/lib/ceph/osd/ceph-1 /var/lib/ceph/osd/ceph-1/journal
...
```

When the modified *post-deploy-template.yaml* and *numa-systemd-osd.sh* described above are run in [Deployment](#), the Ceph OSD daemons on each **osd-compute** node will be restarted under a NUMA policy. To verify that this has been completed after the deployment, check the log with **journalctl** as shown above or check the output of *numa-systemd-osd.sh* captured in */root/post_deploy_heat_output.txt*.

The full content of *post-deploy-template.yaml* and *numa-systemd-osd.sh* may be read in the Appendix on [Custom Heat Templates](#) and is also available online as described in the [GitHub Repository of Example Files](#) Appendix for more details.

Details on OSD systemd unit file update for NUMA

The *numa-systemd-osd.sh* script checks if the **hwloc** and **numactl** packages are installed, and if they are not installed, tries to install them with **yum**. To ensure these packages are available, consider either of the following options:

- Configure Red Hat OpenStack Platform director to register the overcloud to a yum repository containing the **numactl** and **hwloc** packages by using the **--rhel-reg**, **--reg-method**, **--reg-org** options described in [5.7. Setting Overcloud Parameters](#) of the Red Hat document *Director Installation and Usage*.
- Before uploading the overcloud images to the undercloud Glance service, install the **numactl**, **hwloc-libs**, and **hwloc** packages on the overcloud image with **virt-customize**, as described in [24.12 virt-customize: Customizing Virtual Machine Settings](#) from the *Virtualization Deployment and Administration Guide for Red Hat Enterprise Linux 7*.

The **numactl** package is necessary so that the Ceph OSD processes can be started with a NUMA policy. The **hwloc** package provides the **lstopo-no-graphics** command, which shows the CPU topology of the system. Rather than require the user to determine which NUMA socket Ceph should be pinned to, based on the IRQ of the `$OSD_NUMA_INTERFACE`, the following examines the system to determine the desired NUMA socket number. It uses the **lstopo-no-graphics** command, filters the output with **grep** and then loops through the output to determine which NUMA socket has the IRQ.

```
declare -A NUMASOCKET
while read TYPE SOCKET_NUM NIC ; do
    if [[ "$TYPE" == "NUMANode" ]]; then
        NUMASOCKET=$(echo $SOCKET_NUM | sed s/L//g);
    fi
    if [[ "$NIC" == "$OSD_NUMA_INTERFACE" ]]; then
        # because $NIC is the $OSD_NUMA_INTERFACE,
        # the NUMASOCKET has been set correctly above
        break # so stop looking
    fi
done < <(lstopo-no-graphics | tr -d [[:punct:]] | egrep "NUMANode|$OSD_NUMA_INTERFACE")
```

The `tr` command is used to trim away punctuation, as `lstopo-no-graphics` outputs the network interface in quotes. A regular expression passed to `egrep` shows only the lines containing the `NUMANode` or the `$OSD_NUMA_INTERFACE` defined earlier. A `while` loop with `read` is used to extract the three columns containing the desired strings. Each NUMA socket number is collected, without the preceding 'L' as per `sed`. The `$NUMASOCKET` is set for each iteration containing the `NUMANode` in case, during the next iteration, the `$OSD_NUMA_INTERFACE` is found. When the desired network interface is found, the loop exits with `break` before the `$NUMASOCKET` variable can be set to the next NUMA socket number. If no `$NUMASOCKET` is found, then the script exits.

The `crudini` command is used to save the `ExecStart` section of the default OSD unit file.

```
CMD=$(crudini --get $UNIT Service ExecStart)
```

A different `crudini` command is then used to put the same command back for the `ExecStart` command, but the command has a `numactl` call appended to its front.

```
crudini --verbose --set $UNIT Service ExecStart "$NUMA $CMD"
```

The `$NUMA` variable saves the `numactl` call to start the OSD daemon with a NUMA policy to only execute the command on the CPUs of the `$NUMASOCKET` identified previously. The `--preferred` option and not `--membind` is used. This is done because testing shows that hard pinning, with `--membind`, can cause swapping.

```
NUMA="/usr/bin/numactl -N $NUMASOCKET --preferred=$NUMASOCKET"
```

The last thing that `numa-systemd-osd.sh` does is to restart all of the OSD daemons on the server.

```
OSD_IDS=$(ls /var/lib/ceph/osd | awk 'BEGIN { FS = "-" } ; { print $2 }')
for OSD_ID in $OSD_IDS; do
    systemctl restart ceph-osd@$OSD_ID
done
```

A variation of the command above is used to show the status before and after the restart. This status is saved in `/root/post_deploy_heat_output.txt` on each `osd-compute` node.



Each time the above script is run, the OSD daemons are restarted sequentially on all Ceph OSD nodes. Thus, this script is only run on create, not on update, as per the `actions: ['CREATE']` line in `post-deploy-template.yaml`.

7.3. Reduce Ceph Backfill and Recovery Operations

When an OSD is removed, Ceph uses *backfill* and *recovery* operations to rebalance the cluster. This is done in order to keep multiple copies of data according to the placement group policy. These operations use system resources, so if a Ceph cluster is under load, then its performance will drop as it diverts resources to backfill and recovery. To keep the Ceph cluster performant when an OSD is removed, reduce the priority of backfill and recovery operations. The trade off of this tuning is that there are less data replicas for a longer time and thus, the data is at a slightly greater risk.

The three variables to modify for this setting have the following meanings as defined in the [Ceph Storage Cluster OSD Configuration Reference](#).

- *osd recovery max active*: The number of active recovery requests per OSD at one time. More requests will accelerate recovery, but the requests place an increased load on the cluster.
- *osd max backfills*: The maximum number of backfills allowed to or from a single OSD.
- *osd recovery op priority*: The priority set for recovery operations. It is relative to `osd client op` priority.

To have Red Hat OpenStack Platform director configure the Ceph cluster to favor performance during rebuild over recovery speed, configure a Heat environment file with the following values:

```
parameter_defaults:
  ExtraConfig:
    ceph::profile::params::osd_recovery_op_priority: 2
```

Red Hat Ceph Storage versions prior to 2 also require the following to be in the above file:

```
ceph::profile::params::osd_recovery_max_active: 3
ceph::profile::params::osd_max_backfills: 1
```

However, as these values are presently the defaults in version 2 and later, they do not need to be placed in the Heat environment file.

The above settings, were made to `~/custom-templates/ceph.yaml` in [Ceph Configuration](#). If they need to be updated, then the Heat template may be updated and the `openstack overcloud deploy` command, as covered in [Deployment](#), may be re-run and Red Hat OpenStack Platform director will update the configuration on the overcloud.

7.4. Regarding tuned

The default tuned profile for Red Hat Enterprise Linux 7 is *throughput-performance*. Though the *virtual-host* profile is recommended for Compute nodes, in the case of nodes which run both Ceph OSD and Nova Compute services, the *throughput-performance* profile is recommended in order to optimize for disk intensive workloads. This profile should already be enabled by default and may be checked, after [Deployment](#), by using a command like the following:

```
[stack@hci-director ~]$ for ip in $(nova list | grep compute | awk {'print $12'} | sed
s/ctlplane=//g); do ssh heat-admin@$ip "/sbin/tuned-adm active"; done
Current active profile: throughput-performance
Current active profile: throughput-performance
Current active profile: throughput-performance
Current active profile: throughput-performance
[stack@hci-director ~]$
```

8. Deployment

This section describes how to use Red Hat OpenStack Platform director to deploy OpenStack and Ceph so that Ceph OSDs and Nova Computes may cohabit the same server.

8.1. Verify Ironic Nodes are Available

The following command to verifies all Ironic nodes are powered off, available for provisioning, and not in maintenance mode:

```
[stack@hci-director ~]$ openstack baremetal node list
```

UUID	Name	Instance UUID	Power State	Provisioning State	Maintenance
d4f73b0b-c55a-4735-9	m630_slot13	None	power off	available	False
176-9cb063a08bc1					
b5cd14dd-c305-4ce2-9	m630_slot14	None	power off	available	False
f54-ef1e4e88f2f1					
706adf7a-b3ed-49b8-8	m630_slot15	None	power off	available	False
101-0b8f28a1b8ad					
c38b7728-63e4-4e6d-	r730xd_u29	None	power off	available	False
acbe-46d49aee049f					
7a2b3145-636b-4ed3	r730xd_u31	None	power off	available	False
-a0ff-f0b2c9f09df4					
5502a6a0-0738-4826-b	r730xd_u33	None	power off	available	False
b41-5ec4f03e7bfa					

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

8.2. Run the Deploy Command

The following command deploys the overcloud described in this reference architecture.

```
time openstack overcloud deploy --templates \  
-r ~/custom-templates/custom-roles.yaml \  
-e /usr/share/openstack-tripleo-heat-templates/environments/puppet-pacemaker.yaml \  
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \  
-e /usr/share/openstack-tripleo-heat-templates/environments/storage-environment.yaml \  
-e ~/custom-templates/network.yaml \  
-e ~/custom-templates/ceph.yaml \  
-e ~/custom-templates/compute.yaml \  
-e ~/custom-templates/layout.yaml
```

8.2.1. Deployment Command Details

There are many options passed in the command above. This subsection goes through each option in detail.

```
time openstack overcloud deploy --templates \  
-r ~/custom-templates/custom-roles.yaml
```

The above calls the `openstack overcloud deploy` command and uses the default location of the templates in `/usr/share/openstack-tripleo-heat-templates/`. The `time` command is used to time how long the deployment takes.

```
-r ~/custom-templates/custom-roles.yaml
```

The `-r`, or its longer extension `--roles-file`, overrides the default `roles_data.yaml` in the `--templates` directory. This is necessary because that file was copied, and the new `OsdCompute` role was created as described in [Hyper Converged Role Definition](#).

The next set of options passed is the following:

```
-e /usr/share/openstack-tripleo-heat-templates/environments/puppet-pacemaker.yaml \  
-e /usr/share/openstack-tripleo-heat-templates/environments/storage-environment.yaml \  
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml
```

Passing `--templates` makes the deployment use the Heat templates in `/usr/share/openstack-tripleo-heat-templates/` but the three environment files above, which reside in this directory, will not be used by the deployment by default. Thus, they need to be explicitly passed. Each of these Heat environment files perform the following functions:

- *puppet-pacemaker.yaml* - Configures controller node services in a highly available pacemaker cluster
- *storage-environment.yaml* - Configures Ceph as a storage backend, whose `parameter_defaults` are passed by the custom template *ceph.yaml*
- *network-isolation.yaml* - Configures network isolation for different services whose parameters are passed by the custom template *network.yaml*

The following includes the `~/custom-templates` defined in [Define the Overcloud](#) or in [Resource Isolation and Tuning](#)

```
-e ~/custom-templates/network.yaml \  
-e ~/custom-templates/ceph.yaml \  
-e ~/custom-templates/compute.yaml \  
-e ~/custom-templates/layout.yaml
```

The details of each environment file are covered in the following sections:

- *network.yaml* - is explained in [Network Configuration](#)
- *ceph.yaml* - is explained in [Ceph Configuration](#)
- *compute.yaml* - is explained in [Resource Isolation and Tuning](#)
- *layout.yaml* - is explained in [Overcloud Layout](#)

The order of the above arguments is necessary, since each environment file overrides the previous environment file.

8.3. Verify the Deployment Succeeded

1. Watch deployment progress and look for failures in a separate console window

```
heat resource-list -n5 overcloud | egrep -i 'fail|progress'
```

2. Run `openstack server list` to view IP addresses for the overcloud servers

```
[stack@hci-director ~]$ openstack server list
```

```
+-----+-----+-----+-----+
+-----+
| ID              | Name              | Status | Networks              |
Image Name      |
+-----+-----+-----+-----+
+-----+
| fc8686c1-a675-4c89-a508 | overcloud-controller-2 | ACTIVE | ctlplane=192.168.1.37 |
overcloud-full |
| -cc1b34d5d220          |                       |        |                        |
|                       |                       |        |                        |
| 7c6ae5f3-7e18-4aa2-a1f8 | overcloud-osd-compute-2 | ACTIVE | ctlplane=192.168.1.30 |
overcloud-full |
| -53145647a3de          |                       |        |                        |
|                       |                       |        |                        |
| 851f76db-427c-42b3      | overcloud-controller-0 | ACTIVE | ctlplane=192.168.1.33 |
overcloud-full |
| -8e0b-e8b4b19770f8     |                       |        |                        |
|                       |                       |        |                        |
| e2906507-6a06-4c4d-     | overcloud-controller-1 | ACTIVE | ctlplane=192.168.1.29 |
overcloud-full |
| bd15-9f7de455e91d      |                       |        |                        |
|                       |                       |        |                        |
| 0f93a712-b9eb-         | overcloud-osd-compute-0 | ACTIVE | ctlplane=192.168.1.32 |
overcloud-full |
| 4f42-bc05-f2c8c2edfd81 |                       |        |                        |
|                       |                       |        |                        |
| 8f266c17-ff39-422e-a935 | overcloud-osd-compute-1 | ACTIVE | ctlplane=192.168.1.24 |
overcloud-full |
| -effb219c7782          |                       |        |                        |
|                       |                       |        |                        |
+-----+-----+-----+-----+
+-----+
[stack@hci-director ~]$
```

- Wait for the overcloud deploy to complete. For this reference architecture, it took approximately 45 minutes.

```
2016-12-20 23:25:04Z [overcloud]: CREATE_COMPLETE Stack CREATE completed successfully

Stack overcloud CREATE_COMPLETE

Started Mistral Workflow. Execution ID: aeac4d71-56b4-4c72-a980-022623487c05
/home/stack/.ssh/known_hosts updated.
Original contents retained as /home/stack/.ssh/known_hosts.old
Overcloud Endpoint: http://10.19.139.46:5000/v2.0
Overcloud Deployed

real    44m24.800s
user    0m4.171s
sys     0m0.346s
[stack@hci-director ~]$
```

8.4. Configure Controller Pacemaker Fencing

Fencing is the process of isolating a node to protect a cluster and its resources. Without fencing, a faulty node can cause data corruption in a cluster. In Appendix, [Example Fencing Script](#), a script is provided to configure each controller node's IPMI as a fence device.

Prior to running *configure_fence.sh*, be sure to update it to replace **PASSWORD** with the actual IPMI password. For example, the following:

```
$SSH_CMD $i 'sudo pcs stonith create $(hostname -s)-ipmi fence_ipmilan
pcmk_host_list=$(hostname -s) ipaddr=$(sudo ipmitool lan print 1 | awk " /IP Address / {
print \$4 } ") login=root passwd=PASSWORD lanplus=1 cipher=1 op monitor interval=60sr'
```

would become:

```
$SSH_CMD $i 'sudo pcs stonith create $(hostname -s)-ipmi fence_ipmilan
pcmk_host_list=$(hostname -s) ipaddr=$(sudo ipmitool lan print 1 | awk " /IP Address / {
print \$4 } ") login=root passwd=p@55W0rd! lanplus=1 cipher=1 op monitor interval=60sr'
```

An example of running the *configure_fence.sh* script as the **stack** user on the undercloud is below:

1. Use *configure_fence.sh* to enable fencing

```
[stack@hci-director ~]$ ./configure_fence.sh enable
OS_PASSWORD=41485c25159ef92bc375e5dd9eea495e5f47dbd0
OS_AUTH_URL=http://192.168.1.1:5000/v2.0
OS_USERNAME=admin
OS_TENANT_NAME=admin
OS_NO_CACHE=True
192.168.1.34
192.168.1.32
192.168.1.31
Cluster Properties:
  cluster-infrastructure: corosync
  cluster-name: tripleo_cluster
  dc-version: 1.1.13-10.el7_2.2-44eb2dd
  have-watchdog: false
  maintenance-mode: false
  redis_REPL_INFO: overcloud-controller-2
  stonith-enabled: true
[stack@hci-director ~]$
```

2. Verify fence devices are configured with `pcs status`

```
[stack@hci-director ~]$ ssh heat-admin@192.168.1.34 "sudo pcs status | grep -i fence"
overcloud-controller-0-ipmi    (stonith:fence_ipmilan):      Started overcloud-
controller-2
overcloud-controller-1-ipmi    (stonith:fence_ipmilan):      Started overcloud-
controller-0
overcloud-controller-2-ipmi    (stonith:fence_ipmilan):      Started overcloud-
controller-0
[stack@hci-director ~]$
```

The `configure_fence.sh` script and steps above to configure it are from reference architecture [Deploying Red Hat Enterprise Linux OpenStack Platform 7 with RHEL-OSP Director 7.1](#).

9. Operational Considerations

9.1. Configuration Updates

The procedure to apply OpenStack configuration changes for the nodes described in this reference architecture does not differ from the procedure for non-hyper-converged nodes deployed by Red Hat OpenStack Platform director. Thus, to apply an OpenStack configuration, follow the procedure described in section [7.7. Modifying the Overcloud Environment of the Director Installation and Usage documentation](#). As stated in the documentation, the same Heat templates must be passed as arguments to the `openstack overcloud` command.

9.2. Adding Compute/Red Hat Ceph Storage Nodes

This section describes how to add additional hyper-converged nodes to an existing hyper-converged deployment that was configured as described earlier in this reference architecture.

9.2.1. Use Red Hat OpenStack Platform director to add a new Nova Compute / Ceph OSD Node

1. Create a new JSON file

Create a new JSON file describing the new nodes to be added. For example, if adding a server in a rack in slot U35, then a file like `u35.json` may contain the following:

```
{
  "nodes": [
    {
      "pm_password": "PASSWORD",
      "name": "r730xd_u35",
      "pm_user": "root",
      "pm_addr": "10.19.136.28",
      "pm_type": "pxe_ipmitool",
      "mac": [
        "ec:f4:bb:ed:6f:e4"
      ],
      "arch": "x86_64",
      "capabilities": "node:osd-compute-3,boot_option:local"
    }
  ]
}
```

2. Import the new JSON file into Ironic

```
openstack baremetal import u35.json
```

3. Observe that the new node was added

For example, the server in U35 was assigned the ID 7250678a-a575-4159-840a-e7214e697165.

```
[stack@hci-director scale]$ ironic node-list
```

UUID	Name	Instance UUID
Power State	Provision State	Maintenance
a94b75e3-369f-4b2d-b8cc-8ab272e23e89	None	629f3b1f-319f-4df7-8df1-0a9828f2f2f8
power on	active	False
7ace7b2b-b549-414f-b83e-5f90299b4af3	None	4b354355-336d-44f2-9def-27c54cbcc4f5
power on	active	False
8be1d83c-19cb-4605-b91d-928df163b513	None	29124fbb-ee1d-4322-a504-a1a190022f4e
power on	active	False
e8411659-bc2b-4178-b66f-87098a1e6920	None	93199972-51ff-4405-979c-3c4aabdee7ce
power on	active	False
04679897-12e9-4637-9998-af8bee30b414	None	e7578d80-0376-4df5-bbff-d4ac02eb1254
power on	active	False
48b4987d-e778-48e1-ba74-88a08edf7719	None	586a5ef3-d530-47de-8ec0-8c98b30f880c
power on	active	False
7250678a-a575-4159-840a-e7214e697165	None	None
None	available	False

```
[stack@hci-director scale]$
```

4. Set the new server in maintenance mode

Maintenance mode prevents the server from being used for another purpose, e.g. another cloud operator adding an additional node at the same time.

```
ironic node-set-maintenance 7250678a-a575-4159-840a-e7214e697165 true
```

5. Introspect the new hardware

```
openstack baremetal introspection start 7250678a-a575-4159-840a-e7214e697165 true
```

6. Verify that introspection is complete

The previous step takes time. The following command shows the status of the introspection.

```
[stack@hci-director ~]$ openstack baremetal introspection bulk status
```

Node UUID	Finished	Error
a94b75e3-369f-4b2d-b8cc-8ab272e23e89	True	None
7ace7b2b-b549-414f-b83e-5f90299b4af3	True	None
8be1d83c-19cb-4605-b91d-928df163b513	True	None
e8411659-bc2b-4178-b66f-87098a1e6920	True	None
04679897-12e9-4637-9998-af8bee30b414	True	None
48b4987d-e778-48e1-ba74-88a08edf7719	True	None
7250678a-a575-4159-840a-e7214e697165	True	None

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

7. Remove the new server from maintenance mode

This step is necessary in order for the Red Hat OpenStack Platform director Nova Scheduler to select the new node when scaling the number of computes.

```
ironic node-set-maintenance 7250678a-a575-4159-840a-e7214e697165 false
```

8. Assign the kernel and ramdisk of the full overcloud image to the new node

```
openstack baremetal configure boot
```

The IDs of the kernel and ramdisk that were assigned to the new node are seen with the following command:

```
[stack@hci-director ~]$ ironic node-show 7250678a-a575-4159-840a-e7214e697165 | grep
deploy_
| driver_info          | {'deploy_kernel': u'e03c5677-2216-4120-95ad-b4354554a590',
|
|                      | u'ipmi_password': u'*****', u'deploy_ramdisk': u'2c5957bd-
|
|                      | u'deploy_key': u'H301D1ETXCSSBDUMJY5YCCUFG12DJN0G',
u'configdrive': u'H4 |
[stack@hci-director ~]$
```

The `deploy_kernel` and `deploy_ramdisk` are checked against what is in Glance. In the following example, the names `bm-deploy-kernel` and `bm-deploy-ramdisk` were assigned from the Glance database.

```
[stack@hci-director ~]$ openstack image list
```

ID	Name	Status
f7dce3db-3bbf-4670-8296-fa59492276c5	bm-deploy-ramdisk	active
9b73446a-2c31-4672-a3e7-b189e105b2f9	bm-deploy-kernel	active
653f9c4c-8afc-4320-b185-5eb1f5ecb7aa	overcloud-full	active
714b5f55-e64b-4968-a307-ff609cbcce6c	overcloud-full-initrd	active
b9b62ec3-bfdb-43f7-887f-79fb79dcacc0	overcloud-full-vmlinuz	active

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

9. Update the appropriate Heat template to scale the OsdCompute node

Update `~/custom-templates/layout.yaml` change the `OsdComputeCount` from 3 to 4 and add a new IP in each isolated network for the `OsdCompute` node. For example, change the following:

```
OsdComputeIPs:
  internal_api:
    - 192.168.2.203
    - 192.168.2.204
    - 192.168.2.205
  tenant:
    - 192.168.3.203
    - 192.168.3.204
    - 192.168.3.205
  storage:
    - 172.16.1.203
    - 172.16.1.204
    - 172.16.1.205
  storage_mgmt:
    - 172.16.2.203
    - 172.16.2.204
    - 172.16.2.205
```

so that a .206 IP addresses is added as in the following:


```
OsdComputeIPs:
  internal_api:
    - 192.168.2.203
    - 192.168.2.204
    - 192.168.2.205
    - 192.168.2.206
  tenant:
    - 192.168.3.203
    - 192.168.3.204
    - 192.168.3.205
    - 192.168.3.206
  storage:
    - 172.16.1.203
    - 172.16.1.204
    - 172.16.1.205
    - 172.16.1.206
  storage_mgmt:
    - 172.16.2.203
    - 172.16.2.204
    - 172.16.2.205
    - 172.16.2.206
```

See [Configure scheduler hints to control node placement and IP assignment](#) for more information about the `~/custom-templates/layout.yaml` file.

10. Apply the overcloud update

Use the same command that was used to deploy the overcloud to update the overcloud so that the changes made in the previous step are applied.

```
openstack overcloud deploy --templates \
-r ~/custom-templates/custom-roles.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/puppet-pacemaker.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/storage-environment.yaml \
-e ~/custom-templates/network.yaml \
-e ~/custom-templates/ceph.yaml \
-e ~/custom-templates/layout.yaml
```

11. Verify that the new *OsdCompute* node was added correctly

Use `openstack server list` to verify that the new *OsdCompute* node was added and is available. In the example below the new node, `overcloud-osd-compute-3`, is listed as *ACTIVE*.

```
[stack@hci-director ~]$ openstack server list
```

ID	Name	Status	Networks
fc8686c1-a675-4c89-a508-cc1b34d5d220	overcloud-controller-2	ACTIVE	ctlplane=192.168.1.37 overcloud-full
7c6ae5f3-7e18-4aa2-a1f8-53145647a3de	overcloud-osd-compute-2	ACTIVE	ctlplane=192.168.1.30 overcloud-full
851f76db-427c-42b3-8e0b-e8b4b19770f8	overcloud-controller-0	ACTIVE	ctlplane=192.168.1.33 overcloud-full
e2906507-6a06-4c4d-bd15-9f7de455e91d	overcloud-controller-1	ACTIVE	ctlplane=192.168.1.29 overcloud-full
0f93a712-b9eb-4f42-bc05-f2c8c2edfd81	overcloud-osd-compute-0	ACTIVE	ctlplane=192.168.1.32 overcloud-full
8f266c17-ff39-422e-a935-efb219c7782	overcloud-osd-compute-1	ACTIVE	ctlplane=192.168.1.24 overcloud-full
5fa641cf-b290-4a2a-b15e-494ab9d10d8a	overcloud-osd-compute-3	ACTIVE	ctlplane=192.168.1.21 overcloud-full

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

The new Compute/Ceph Storage Node has been added the overcloud.

9.3. Removing Compute/Red Hat Ceph Storage Nodes

This section describes how to remove an *OsdCompute* node from an existing hyper-converged deployment that was configured as described earlier in this reference architecture.

Before reducing the compute and storage resources of a hyper-converged overcloud, verify that there will still be enough CPU and RAM to service the compute workloads, and migrate the compute workloads off the node to be removed. Verify that the Ceph cluster has the reserve storage capacity necessary to maintain a health status of **HEALTH_OK** without the Red Hat Ceph Storage node to be removed.

9.3.1. Remove the Ceph Storage Node

At this time of writing Red Hat OpenStack Platform director does not support the automated removal of a Red Hat Ceph Storage node, so steps in this section need to be done manually from one of the OpenStack Controller / Ceph Monitor nodes, unless otherwise indicated.

1. Verify that the **ceph health** command does not produce any "near full" warnings

```
[root@overcloud-controller-0 ~]# ceph health
HEALTH_OK
[root@overcloud-controller-0 ~]#
```



If the `ceph health` command reports that the cluster is *near full* as in the example below, then removing the OSD could result in exceeding or reaching the full ratio which could result in data loss. If this is the case, contact Red Hat before proceeding to discuss options to remove the Red Hat Ceph Storage node without data loss.

```
HEALTH_WARN 1 nearfull osds
osd.2 is near full at 85%
```

2. Determine the OSD numbers of the *OsdCompute* node to be removed

In the example below, overcloud-osd-compute-3 will be removed, and the `ceph osd tree` command shows that its OSD numbers are 0 through 44 counting by fours.

```
[root@overcloud-controller-0 ~]# ceph osd tree
ID WEIGHT  TYPE NAME                UP/DOWN REWEIGHT PRIMARY-AFFINITY
-1 52.37256 root default
-2 13.09314 host overcloud-osd-compute-3
  0 1.09109  osd.0                up 1.00000 1.00000
  4 1.09109  osd.4                up 1.00000 1.00000
  8 1.09109  osd.8                up 1.00000 1.00000
 12 1.09109  osd.12               up 1.00000 1.00000
 16 1.09109  osd.16               up 1.00000 1.00000
 20 1.09109  osd.20               up 1.00000 1.00000
 24 1.09109  osd.24               up 1.00000 1.00000
 28 1.09109  osd.28               up 1.00000 1.00000
 32 1.09109  osd.32               up 1.00000 1.00000
 36 1.09109  osd.36               up 1.00000 1.00000
 40 1.09109  osd.40               up 1.00000 1.00000
 44 1.09109  osd.44               up 1.00000 1.00000
...
```

3. Start a process to monitor the Ceph cluster

In a separate terminal, run the `ceph -w` command. This command is used to monitor the health of the Ceph cluster during OSD removal. The output of this command once started is similar to:

```
[root@overcloud-controller-0 ~]# ceph -w
cluster eb2bb192-b1c9-11e6-9205-525400330666
health HEALTH_OK
monmap e2: 3 mons at {overcloud-controller-0=172.16.1.200:6789/0,overcloud-
controller-1=172.16.1.201:6789/0,overcloud-controller-2=172.16.1.202:6789/0}
election epoch 8, quorum 0,1,2 overcloud-controller-0,overcloud-controller-
1,overcloud-controller-2
osdmap e139: 48 osds: 48 up, 48 in
flags sortbitwise
pgmap v106106: 1344 pgs, 6 pools, 11080 MB data, 4140 objects
35416 MB used, 53594 GB / 53628 GB avail
1344 active+clean

2016-11-29 02:13:17.058468 mon.0 [INF] pgmap v106106: 1344 pgs: 1344 active+clean; 11080
MB data, 35416 MB used, 53594 GB / 53628 GB avail
2016-11-29 02:15:03.674380 mon.0 [INF] pgmap v106107: 1344 pgs: 1344 active+clean; 11080
MB data, 35416 MB used, 53594 GB / 53628 GB avail
...
```

4. Mark OSDs of the node to be removed as *out*

Use the `ceph osd out <NUM>` command to remove all twelve OSDs from the `overcloud-osd-compute-3` node from the Ceph cluster. Allow for time between each OSD removal to ensure the cluster has time to complete the previous action before proceeding; this may be achieved by using a `sleep` statement. A script like the following, which uses `seq` to count from 0 to 44 by fours, may be used:

```
for i in $(seq 0 4 44); do
    ceph osd out $i;
    sleep 10;
done
```

Before running the above script, note the output of `ceph osd stat` with all OSDs *up* and *in*.

```
[root@overcloud-controller-0 ~]# ceph osd stat
osdmap e173: 48 osds: 48 up, 48 in
flags sortbitwise
[root@overcloud-controller-0 ~]#
```

The results of running the script above should look as follows:

```
[root@overcloud-controller-0 ~]# for i in $(seq 0 4 44); do ceph osd out $i; sleep 10;
done
marked out osd.0.
marked out osd.4.
marked out osd.8.
marked out osd.12.
marked out osd.16.
marked out osd.20.
marked out osd.24.
marked out osd.28.
marked out osd.32.
marked out osd.36.
marked out osd.40.
marked out osd.44.
[root@overcloud-controller-0 ~]#
```

After the OSDs are marked as *out*, the output of the `ceph osd stat` command should show that twelve of the OSDs are no longer *in* but still *up*.

```
[root@overcloud-controller-0 ~]# ceph osd stat
  osdmap e217: 48 osds: 48 up, 36 in
      flags sortbitwise
[root@overcloud-controller-0 ~]#
```

5. Wait for all of the placement groups to become active and clean

The removal of the OSDs will cause Ceph to rebalance the cluster by migrating placement groups to other OSDs. The `ceph -w` command started in step 3 should show the placement group states as they change from *active+clean* to *active*, *some degraded objects*, and finally *active+clean* when migration completes.

An example of the output of `ceph -w` command started in step 3 as it changes looks like the following:

```

2016-11-29 02:16:06.372846 mon.2 [INF] from='client.? 172.16.1.200:0/1977099347'
entity='client.admin' cmd=[{"prefix": "osd out", "ids": ["0"]}] : dispatch
...
2016-11-29 02:16:07.624668 mon.0 [INF] osdmap e141: 48 osds: 48 up, 47 in
2016-11-29 02:16:07.714072 mon.0 [INF] pgmap v106111: 1344 pgs: 8 remapped+peering, 1336
active+clean; 11080 MB data, 34629 MB used, 52477 GB / 52511 GB avail
2016-11-29 02:16:07.624952 osd.46 [INF] 1.8e starting backfill to osd.2 from (0'0,0'0]
MAX to 139'24162
2016-11-29 02:16:07.625000 osd.2 [INF] 1.ef starting backfill to osd.16 from (0'0,0'0]
MAX to 139'17958
2016-11-29 02:16:07.625226 osd.46 [INF] 1.76 starting backfill to osd.25 from (0'0,0'0]
MAX to 139'37918
2016-11-29 02:16:07.626074 osd.46 [INF] 1.8e starting backfill to osd.15 from (0'0,0'0]
MAX to 139'24162
2016-11-29 02:16:07.626550 osd.21 [INF] 1.ff starting backfill to osd.46 from (0'0,0'0]
MAX to 139'21304
2016-11-29 02:16:07.627698 osd.46 [INF] 1.32 starting backfill to osd.33 from (0'0,0'0]
MAX to 139'24962
2016-11-29 02:16:08.682724 osd.45 [INF] 1.60 starting backfill to osd.16 from (0'0,0'0]
MAX to 139'8346
2016-11-29 02:16:08.696306 mon.0 [INF] osdmap e142: 48 osds: 48 up, 47 in
2016-11-29 02:16:08.738872 mon.0 [INF] pgmap v106112: 1344 pgs: 6 peering, 9
remapped+peering, 1329 active+clean; 11080 MB data, 34629 MB used, 52477 GB / 52511 GB
avail
2016-11-29 02:16:09.850909 mon.0 [INF] osdmap e143: 48 osds: 48 up, 47 in
...
2016-11-29 02:18:10.838365 mon.0 [INF] pgmap v106256: 1344 pgs: 7 activating, 1
active+recovering+degraded, 7 activating+degraded, 9 active+degraded, 70 peering, 1223
active+clean, 8 active+remapped, 19 remapped+peering; 11080 MB data, 33187 MB used, 40189
GB / 40221 GB avail; 167/12590 objects degraded (1.326%); 80/12590 objects misplaced
(0.635%); 11031 kB/s, 249 objects/s recovering
...

```

Output like the above should continue as the Ceph cluster rebalances data, and eventually it returns to a health status of **HEALTH_OK**.

6. Verify that the cluster has returned to health status **HEALTH_OK**

```
[root@overcloud-controller-0 ~]# ceph -s
cluster eb2bb192-b1c9-11e6-9205-525400330666
health HEALTH_OK
monmap e2: 3 mons at {overcloud-controller-0=172.16.1.200:6789/0,overcloud-
controller-1=172.16.1.201:6789/0,overcloud-controller-2=172.16.1.202:6789/0}
election epoch 8, quorum 0,1,2 overcloud-controller-0,overcloud-controller-
1,overcloud-controller-2
osdmap e217: 48 osds: 48 up, 36 in
flags sortbitwise
pgmap v106587: 1344 pgs, 6 pools, 11080 MB data, 4140 objects
35093 MB used, 40187 GB / 40221 GB avail
1344 active+clean
[root@overcloud-controller-0 ~]#
```

7. Stop the OSD Daemons on the node being removed

From the Red Hat OpenStack Platform director server, **ssh** into the node that is being removed and run **systemctl stop ceph-osd.target** to stop all OSDs.

Note how the output of **ceph osd stat** changes after **systemctl** command is run; the number of *up* OSDs changes from 48 to 36.

```
[root@overcloud-osd-compute-3 ~]# ceph osd stat
osdmap e217: 48 osds: 48 up, 36 in
flags sortbitwise
[root@overcloud-osd-compute-3 ~]# systemctl stop ceph-osd.target
[root@overcloud-osd-compute-3 ~]# ceph osd stat
osdmap e218: 48 osds: 36 up, 36 in
flags sortbitwise
[root@overcloud-osd-compute-3 ~]#
```

Be sure to run **systemctl stop ceph-osd.target** on the same node which hosts the OSDs, e.g. in this case, the OSDs from overcloud-osd-compute-3 will be removed, so the command is run on overcloud-osd-compute-3.

8. Remove the OSDs

The script below does the following:

- Remove the OSD from the CRUSH map so that it no longer receives data
- Remove the OSD authentication key
- Remove the OSD

```
for i in $(seq 0 4 44); do
    ceph osd crush remove osd.$i
    sleep 10
    ceph auth del osd.$i
    sleep 10
    ceph osd rm $i
    sleep 10
done
```

Before removing the OSDs, note that they are in the CRUSH map for the Ceph storage node to be removed.

```
[root@overcloud-controller-0 ~]# ceph osd crush tree | grep overcloud-osd-compute-3 -A 20
    "name": "overcloud-osd-compute-3",
    "type": "host",
    "type_id": 1,
    "items": [
        {
            "id": 0,
            "name": "osd.0",
            "type": "osd",
            "type_id": 0,
            "crush_weight": 1.091095,
            "depth": 2
        },
        {
            "id": 4,
            "name": "osd.4",
            "type": "osd",
            "type_id": 0,
            "crush_weight": 1.091095,
            "depth": 2
        },
        {

```

```
[root@overcloud-controller-0 ~]#
```

When the script above is executed, it looks like the following:


```
[root@overcloud-osd-compute-3 ~]# for i in $(seq 0 4 44); do
>   ceph osd crush remove osd.$i
>   sleep 10
>   ceph auth del osd.$i
>   sleep 10
>   ceph osd rm $i
>   sleep 10
> done
removed item id 0 name 'osd.0' from crush map
updated
removed osd.0
removed item id 4 name 'osd.4' from crush map
updated
removed osd.4
removed item id 8 name 'osd.8' from crush map
updated
removed osd.8
removed item id 12 name 'osd.12' from crush map
updated
removed osd.12
removed item id 16 name 'osd.16' from crush map
updated
removed osd.16
removed item id 20 name 'osd.20' from crush map
updated
removed osd.20
removed item id 24 name 'osd.24' from crush map
updated
removed osd.24
removed item id 28 name 'osd.28' from crush map
updated
removed osd.28
removed item id 32 name 'osd.32' from crush map
updated
removed osd.32
removed item id 36 name 'osd.36' from crush map
updated
removed osd.36
removed item id 40 name 'osd.40' from crush map
updated
removed osd.40
removed item id 44 name 'osd.44' from crush map
updated
removed osd.44
[root@overcloud-osd-compute-3 ~]#
```

The `ceph osd stat` command should now report that there are only 36 OSDs.

```
[root@overcloud-controller-0 ~]# ceph osd stat
  osdmap e300: 36 osds: 36 up, 36 in
        flags sortbitwise
[root@overcloud-controller-0 ~]#
```

When an OSD is removed from the CRUSH map, CRUSH recomputes which OSDs get the placement groups, and data re-balances accordingly. The CRUSH map may be checked after the OSDs are removed to verify that the update completed.

Observe that overcloud-osd-compute-3 has no OSDs:

```
[root@overcloud-controller-0 ~]# ceph osd crush tree | grep overcloud-osd-compute-3 -A 5
    "name": "overcloud-osd-compute-3",
    "type": "host",
    "type_id": 1,
    "items": []
  },
  {
[root@overcloud-controller-0 ~]#
```

9.3.2. Remove the Node from the Overcloud

Though the OSDs on overcloud-osd-compute-3 are no longer a member of the Ceph cluster, its Nova compute services are still functioning and will be removed in this subsection. The hardware will be shut off, and the Overcloud Heat stack will no longer keep track of the node. All of the steps to do this should be carried out as the `stack` user on the Red Hat OpenStack Platform director system unless otherwise noted.

Before following this procedure, migrate any instances running on the compute node that will be removed to another compute node.

1. Authenticate to the overcloud

```
source ~/overcloudrc
```

2. Check the status of the compute node that is going to be removed

For example, overcloud-osd-compute-3 will be removed:

```
[stack@hci-director ~]$ nova service-list | grep compute-3
| 145 | nova-compute      | overcloud-osd-compute-3.localdomain | nova      | enabled | up
| 2016-11-29T03:40:32.000000 | - |
[stack@hci-director ~]$
```

3. Disable the compute node's service so that no new instances are scheduled on it

```
[stack@hci-director ~]$ nova service-disable overcloud-osd-compute-3.localdomain nova-compute
```

Host	Binary	Status
overcloud-osd-compute-3.localdomain	nova-compute	disabled

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

4. Authenticate to the undercloud

```
source ~/stackrc
```

5. Identify the Nova ID of the *OsdCompute* node to be removed

```
[stack@hci-director ~]$ openstack server list | grep osd-compute-3
| 6b2a2e71-f9c8-4d5b-aaf8-dada97c90821 | overcloud-osd-compute-3 | ACTIVE |
ctlplane=192.168.1.27 | overcloud-full |
[stack@hci-director ~]$
```

In the following example, the Nova ID is extracted with **awk** and **egrep** and set to the variable **\$nova_id**

```
[stack@hci-director ~]$ nova_id=$(openstack server list | grep compute-3 | awk {'print $2'} | egrep -vi 'id|^$')
[stack@hci-director ~]$ echo $nova_id
6b2a2e71-f9c8-4d5b-aaf8-dada97c90821
[stack@hci-director ~]$
```

6. Start a Mistral workflow to delete the node by UUID from the stack by name

```
[stack@hci-director ~]$ time openstack overcloud node delete --stack overcloud $nova_id
deleting nodes [u'6b2a2e71-f9c8-4d5b-aaf8-dada97c90821'] from stack overcloud
Started Mistral Workflow. Execution ID: 396f123d-df5b-4f37-b137-83d33969b52b

real    1m50.662s
user    0m0.563s
sys     0m0.099s
[stack@hci-director ~]$
```

In the above example, the stack to delete the node from needs to be identified by name, "overcloud", instead of by its UUID. However, it will be possible to supply either the UUID or name after [Red Hat Bugzilla 1399429](#) is resolved. It is no longer necessary when deleting a node to pass the Heat environment files with the `-e` option.

As shown by the `time` command output, the request to delete the node is accepted quickly. However, the Mistral workflow and Heat stack update will run in the background as it removes the compute node.

```
[stack@hci-director ~]$ heat stack-list
WARNING (shell) "heat stack-list" is deprecated, please use "openstack stack list"
instead
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| id                                | stack_name | stack_status      | creation_time
| updated_time                    |            |                    |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| 23e7c364-7303-4af6-b54d-cfbf1b737680 | overcloud  | UPDATE_IN_PROGRESS | 2016-11-
24T03:24:56Z | 2016-11-30T17:16:48Z |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
[stack@hci-director ~]$
```

Confirm that Heat has finished updating the overcloud.

```
[stack@hci-director ~]$ heat stack-list
WARNING (shell) "heat stack-list" is deprecated, please use "openstack stack list"
instead
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| id                                | stack_name | stack_status  | creation_time
| updated_time                    |            |               |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| 23e7c364-7303-4af6-b54d-cfbf1b737680 | overcloud  | UPDATE_COMPLETE | 2016-11-
24T03:24:56Z | 2016-11-30T17:16:48Z |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
[stack@hci-director ~]$
```

6. Observe that the node was deleted as desired.

In the example below, overcloud-osd-compute-3 is not included in the `openstack server list` output.

```
[stack@hci-director ~]$ openstack server list
```

ID	Name	Status	Networks
fc8686c1-a675-4c89-a508	overcloud-controller-2	ACTIVE	ctlplane=192.168.1.37
overcloud-full			
-cc1b34d5d220			
7c6ae5f3-7e18-4aa2-a1f8	overcloud-osd-compute-2	ACTIVE	ctlplane=192.168.1.30
overcloud-full			
-53145647a3de			
851f76db-427c-42b3	overcloud-controller-0	ACTIVE	ctlplane=192.168.1.33
overcloud-full			
-8e0b-e8b4b19770f8			
e2906507-6a06-4c4d-	overcloud-controller-1	ACTIVE	ctlplane=192.168.1.29
overcloud-full			
bd15-9f7de455e91d			
0f93a712-b9eb-	overcloud-osd-compute-0	ACTIVE	ctlplane=192.168.1.32
overcloud-full			
4f42-bc05-f2c8c2edfd81			
8f266c17-ff39-422e-a935	overcloud-osd-compute-1	ACTIVE	ctlplane=192.168.1.24
overcloud-full			
-effb219c7782			

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

7. Confirm that Ironic has turned off the hardware that ran the converted Compute/OSD services, and that it is available for other purposes.

```
[stack@hci-director ~]$ openstack baremetal node list
```

UUID	Name	Instance UUID	Power State	Provisioning State
c6498849-d8d8-404	m630_slot13	851f76db-427c-	power on	active

False					
2-aa1c-			42b3-8e0b-		
aa62ec2df17e			e8b4b19770f8		
a8b2e3b9-c62b-496	m630_slot14	e2906507-6a06	power on	active	
False					
5-8a3d-		-4c4d-			
c4e7743ae78b		bd15-9f7de455e91d			
f2d30a3a-8c74	m630_slot15	fc8686c1-a675-4c8	power on	active	
False					
-4fbf-afaa-		9-a508-cc1b34d5d2			
fb666af55dfc		20			
8357d7b0-bd62-4b7	r730xd_u29	0f93a712-b9eb-4f4	power on	active	
False					
9-91f9-52c2a50985		2-bc05-f2c8c2edfd			
d9		81			
fc6efdcb-ae5f-	r730xd_u31	8f266c17-ff39-422	power on	active	
False					
431d-		e-a935-effb219c77			
adf1-4dd034b4a0d3		82			
73d19120-6c93	r730xd_u33	7c6ae5f3-7e18-4aa	power on	active	
False					
-4f1b-ad1f-		2-a1f8-53145647a3			
4cce5913ba76		de			
a0b8b537-0975-406	r730xd_u35	None	power off	available	
False					
b-a346-e361464fd1					
e3					
+-----+-----+-----+-----+-----+					
+-----+					
[stack@hci-director ~]\$					

In the above, the server r730xd_u35 is powered off and available.

8. Check the status of the compute service that was removed in the overcloud

Authenticate back to the overcloud and observe the state of the nova-compute service offered by overcloud-osd-compute-3:

```
[stack@hci-director ~]$ source ~/overcloudrc
[stack@hci-director ~]$ nova service-list | grep osd-compute-3
| 145 | nova-compute      | overcloud-osd-compute-3.localdomain | nova      | disabled |
down | 2016-11-29T04:49:23.000000 | -                |
```

In the above example, the overcloud has a nova-compute service on the overcloud-osd-compute-3 host, but it is currently marked as disabled and down.

9. Remove the node's compute service from the overcloud Nova scheduler

Use `nova service-delete 135` to remove the nova-compute service offered by overcloud-osd-compute-3.

10. Update the Heat template to be consistent with the new overcloud (optional)

This step is optional and is done to make what is the Heat environment file reflect what is actually in the overcloud.

Update `~/custom-templates/layout.yaml` to change the `OsdComputeCount` from 4 to 3 and remove the IPs in each isolated network for the `OsdCompute` node. For example, change the following:


```
OsdComputeIPs:
  internal_api:
    - 192.168.2.203
    - 192.168.2.204
    - 192.168.2.205
    - 192.168.2.206
  tenant:
    - 192.168.3.203
    - 192.168.3.204
    - 192.168.3.205
    - 192.168.3.206
  storage:
    - 172.16.1.203
    - 172.16.1.204
    - 172.16.1.205
    - 172.16.1.206
  storage_mgmt:
    - 172.16.2.203
    - 172.16.2.204
    - 172.16.2.205
    - 172.16.2.206
```

so that it turns into the following:

```
OsdComputeIPs:
  internal_api:
    - 192.168.2.203
    - 192.168.2.204
    - 192.168.2.205
  tenant:
    - 192.168.3.203
    - 192.168.3.204
    - 192.168.3.205
  storage:
    - 172.16.1.203
    - 172.16.1.204
    - 172.16.1.205
  storage_mgmt:
    - 172.16.2.203
    - 172.16.2.204
    - 172.16.2.205
```

The Compute/Ceph Storage Node has been fully removed.

10. Conclusion

This reference architecture has covered how to use Red Hat OpenStack Platform director to deploy and manage Red Hat OpenStack Platform and Red Hat Ceph Storage in a way that both the OpenStack Nova Compute services and the Ceph Object Storage Daemon (OSD) services reside on the same node.

The [Deploy the Undercloud](#) section covered how to deploy an undercloud where the hardware covered in [Hardware Recommendations](#) was imported into Ironic as described in [Register and Introspect Hardware](#).

The [Define the Overcloud](#) section covered to use composable roles to define a hyper-converged overcloud in Heat as well as [Network Configuration](#) and [Ceph Configuration](#).

The [Resource Isolation and Tuning](#) covered how to isolate resources in a hyper-converged overcloud to address contention between OpenStack and Ceph which could result in degradation of either service. A [Nova Memory and CPU Calculator](#) was provided to tune Nova for a hyper-converged deployment based on workload and the [Ceph NUMA Pinning](#) section provided a post-deploy script to start Ceph storage services with a NUMA policy.

The final sections covered [Deployment](#) and operational considerations such as [Configuration Updates](#), [Adding Compute/Red Hat Ceph Storage Nodes](#), and [Removing Compute/Red Hat Ceph Storage Nodes](#).

Appendix A: Contributors

1. Brent Compton - content review
2. Ben England - content review
3. Roger Lopez - content review
4. Federico Lucifredi - content review

Appendix B: References

1. *Hardware Selection Guide for Red Hat Ceph Storage* - <https://www.redhat.com/en/resources/red-hat-ceph-storage-hardware-selection-guide>
2. *Red Hat OpenStack Platform director Installation and Usage* by Dan Macpherson et al - <https://access.redhat.com/documentation/en/red-hat-openstack-platform/10/single/director-installation-and-usage/>
3. *Advanced Overcloud Customization* - by Dan Macpherson et al - <https://access.redhat.com/documentation/en/red-hat-openstack-platform/10/single/advanced-overcloud-customization>
4. *Red Hat Ceph Storage for the Overcloud* - by Dan Macpherson et al - <https://access.redhat.com/documentation/en/red-hat-openstack-platform/10/single/red-hat-ceph-storage-for-the-overcloud>
5. *Deploying Red Hat Enterprise Linux OpenStack Platform 7 with RHEL-OSP Director 7.1* by Jacob Liberman - <https://access.redhat.com/articles/1610453>
6. *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide* - https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Virtualization_Deployment_and_Administration_Guide/

Appendix C: Environment Details

This appendix of the reference architecture describes the environment used to execute the use case in the Red Hat Systems Engineering lab.

The servers in this reference architecture are deployed in the following roles.

Table 1. Server hardware by role

Role	Count	Model
Red Hat OpenStack Platform director	1	Virtual Machine on Dell PowerEdge M630*
OpenStack Controller/Ceph MON	3	Dell PowerEdge M630
OpenStack Compute/Ceph OSD	4	Dell PowerEdge R730XD

* It is not possible to run the Red Hat OpenStack Platform director virtual machine on the same systems that host the OpenStack Controllers/Ceph MONs or OpenStack Computes/OSDs.

C.1. Red Hat OpenStack Platform director

The *undercloud* is a server used exclusively by the OpenStack operator to deploy, scale, manage, and life-cycle the *overcloud*, the cloud that provides services to users. Red Hat's undercloud product is Red Hat OpenStack Platform director.

The undercloud system hosting Red Hat OpenStack Platform director is a virtual machine running Red Hat Enterprise Linux 7.3 with the following specifications:

- 16 virtual CPUs
- 16GB of RAM
- 40GB of hard drive space
- Two virtual 1 Gigabit Ethernet (GbE) connections

The hypervisor which hosts this virtual machine is a Dell M630 with the following specifications:

- Two Intel E5-2630 v3 @ 2.40 GHz CPUs
- 128GB of RAM
- Two 558GB SAS hard disks configured in RAID1
- Two 1GbE connections
- Two 10GbE connections

The hypervisor runs Red Hat Enterprise Linux 7.3 and uses the KVM and Libvirt packages shipped with Red Hat Enterprise Linux to host virtual machines.

C.2. Overcloud Controller / Ceph Monitor

Controller nodes are responsible in order to provide endpoints for REST-based API queries to the majority of the OpenStack services. These include compute, image, identity, block, network, and data processing. The controller nodes also manage authentication and sends messaging to all the systems through a message queue and stores the state of the cloud in a database. In a production deployment, the controller nodes should be run as a highly available cluster.

Ceph Monitor nodes, which cohabitate with the controller nodes in this deployment, maintain the overall health of the Ceph cluster by keeping cluster map state including Monitor map, OSD map, Placement Group map, and CRUSH map. Monitors receive state information from other components to maintain maps and to circulate these maps to other Monitor and OSD nodes.

C.2.1. Overcloud Controller / Ceph Monitor Servers for this Reference Architecture

The servers which host the OpenStack Controller and Ceph Monitor services are three Dell M630s with the following specifications:

- Two Intel E5-2630 v3 @ 2.40 GHz CPUs
- 128GB of RAM
- Two 558GB SAS hard disks configured in RAID1
- Four 1GbE connections
- Four 10GbE connections

C.3. Overcloud Compute / Ceph OSD

Converged Compute/OSD nodes are responsible for running virtual machine instances after they are launched and for holding all data generated by OpenStack. They must support hardware virtualization and provide enough CPU cycles for the instances they host. They must also have enough memory to support the requirements of the virtual machine instances they host while reserving enough memory for each Ceph OSD. Ceph OSD nodes must also provide enough usable hard drive space for the data required by the cloud.

C.3.1. Overcloud Compute / Ceph OSD Servers for this Reference Architecture

The servers which host the OpenStack Compute and Ceph OSD services are four Dell R730XD with the following specifications:

- Two Intel E5-2683 v3 @ 2.00GHz CPUs
- 256GB of RAM
- Two 277GB SAS hard disks configured in RAID1

- Twelve 1117GB SAS hard disks
- Three 400GB SATA SSD disks
- Two 1GbE connections (only one is used)
- Two 10GbE connections

Aside from the RAID1 used for the operating system disks, none of the other disks are using RAID as per Ceph recommended practice.

C.4. Network Environment

C.4.1. Layer 1

The servers used in this reference architecture are physically connected as follows:

- Red Hat OpenStack Platform director:
 - 1GbE to Provisioning Network
 - 1GbE to External Network
- OpenStack Controller/Ceph Monitor:
 - 1GbE to Provisioning Network
 - 1GbE to External Network
 - 1GbE to Internal API Network
 - 10GbE to Cloud VLANs
 - 10GbE to Storage VLANs
- OpenStack Compute/Ceph OSD:
 - 1GbE to Provisioning Network
 - 1GbE to Internal API Network
 - 10GbE to Cloud VLANs
 - 10GbE to Storage VLANs

A diagram illustrating the above can be seen Section 5, [Figure 1 Network Separation Diagram](#).

C.4.2. Layers 2 and 3

The provisioning network is implemented with the following VLAN and range.

- VLAN 4048 (hci-pxe) 192.168.1.0/24

The internal API network is implemented with the following VLAN and range.

- VLAN 4049 (hci-api) 192.168.2.0/24

The Cloud VLAN networks are trunked into the first 10GbE interface of the OpenStack Controllers/Ceph Monitors and OpenStack Computes/Ceph OSDs. A trunk is used so that tenant VLANs networks may be added in the future, though the deployment's default allows VXLAN networks to run on top of the following network.

- VLAN 4050 (hci-tenant) 192.168.3.0/24

The Storage VLAN networks are trunked into the second 10GbE interface of the OpenStack Controllers/Ceph Monitors and OpenStack Computes/Ceph OSDs. The trunk contains the following two VLANs and their network ranges.

- VLAN 4046 (hci-storage-pub) 172.16.1.0/24
- VLAN 4047 (hci-storage-pri) 172.16.2.0/24

The external network is implemented upstream of the switches that implement the above.

Appendix D: Custom Heat Templates

The complete custom Heat templates used in this reference architecture are included in this appendix to be read and may accessed online. See the [GitHub Repository of Example Files](#) Appendix for more details. The `~/custom-templates/network.yaml` file contains the following:

```
resource_registry:
  OS::TripleO::OsdCompute::Net::SoftwareConfig: /home/stack/custom-templates/nic-
  configs/compute-nics.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: /home/stack/custom-templates/nic-
  configs/controller-nics.yaml

parameter_defaults:
  NeutronBridgeMappings: 'datacentre:br-ex,tenant:br-tenant'
  NeutronNetworkType: 'vxlan'
  NeutronTunnelType: 'vxlan'
  NeutronExternalNetworkBridge: ""

  # Internal API used for private OpenStack Traffic
  InternalApiNetCidr: 192.168.2.0/24
  InternalApiAllocationPools: [{'start': '192.168.2.10', 'end': '192.168.2.200'}]
  InternalApiNetworkVlanID: 4049

  # Tenant Network Traffic - will be used for VXLAN over VLAN
  TenantNetCidr: 192.168.3.0/24
  TenantAllocationPools: [{'start': '192.168.3.10', 'end': '192.168.3.200'}]
  TenantNetworkVlanID: 4050

  # Public Storage Access - e.g. Nova/Glance <--> Ceph
  StorageNetCidr: 172.16.1.0/24
  StorageAllocationPools: [{'start': '172.16.1.10', 'end': '172.16.1.200'}]
  StorageNetworkVlanID: 4046

  # Private Storage Access - i.e. Ceph background cluster/replication
  StorageMgmtNetCidr: 172.16.2.0/24
  StorageMgmtAllocationPools: [{'start': '172.16.2.10', 'end': '172.16.2.200'}]
  StorageMgmtNetworkVlanID: 4047

  # External Networking Access - Public API Access
```

```
ExternalNetCidr: 10.19.137.0/21
# Leave room for floating IPs in the External allocation pool (if required)
ExternalAllocationPools: [{'start': '10.19.139.37', 'end': '10.19.139.48'}]
# Set to the router gateway on the external network
ExternalInterfaceDefaultRoute: 10.19.143.254

# Gateway router for the provisioning network (or Undercloud IP)
ControlPlaneDefaultRoute: 192.168.1.1
# The IP address of the EC2 metadata server. Generally the IP of the Undercloud
EC2MetadataIp: 192.168.1.1
# Define the DNS servers (maximum 2) for the overcloud nodes
DnsServers: ["10.19.143.247", "10.19.143.248"]
```

The `~/custom-templates/nic-configs/compute-nics.yaml` file contains the following:

```
heat_template_version: 2015-04-30

description: >
  Software Config to drive os-net-config to configure VLANs for the
  compute and osd role (assumption is that compute and osd cohabitate)

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
ExternalNetworkVlanID:
  default: 10
  description: Vlan ID for the external network traffic.
  type: number
InternalApiNetworkVlanID:
  default: 20
  description: Vlan ID for the internal_api network traffic.
  type: number
StorageNetworkVlanID:
  default: 30
  description: Vlan ID for the storage network traffic.
  type: number
StorageMgmtNetworkVlanID:
  default: 40
  description: Vlan ID for the storage mgmt network traffic.
  type: number
TenantNetworkVlanID:
  default: 50
  description: Vlan ID for the tenant network traffic.
  type: number
ManagementNetworkVlanID:
  default: 60
  description: Vlan ID for the management network traffic.
  type: number
ExternalInterfaceDefaultRoute:
  default: '10.0.0.1'
  description: default route for the external network
  type: string
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 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: em3
              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: interface
              name: em2
              use_dhcp: false
              mtu: 9000
            -
              type: vlan
              device: em2
              mtu: 9000
              use_dhcp: false
              vlan_id: {get_param: StorageMgmtNetworkVlanID}
              addresses:
                -
                  ip_netmask: {get_param: StorageMgmtIpSubnet}
            -
              type: vlan
              device: em2
              mtu: 9000
              use_dhcp: false
              vlan_id: {get_param: StorageNetworkVlanID}
```

```
addresses:
  -
    ip_netmask: {get_param: StorageIpSubnet}
-
  type: interface
  name: em4
  use_dhcp: false
  addresses:
    -
      ip_netmask: {get_param: InternalApiIpSubnet}
-
  # VLAN for VXLAN tenant networking
  type: ovs_bridge
  name: br-tenant
  mtu: 1500
  use_dhcp: false
  members:
    -
      type: interface
      name: em1
      mtu: 1500
      use_dhcp: false
      # force the MAC address of the bridge to this interface
      primary: true
    -
      type: vlan
      mtu: 1500
      vlan_id: {get_param: TenantNetworkVlanID}
      addresses:
        -
          ip_netmask: {get_param: TenantIpSubnet}
# Uncomment when including environments/network-management.yaml
#-
#   type: interface
#   name: nic7
#   use_dhcp: false
#   addresses:
#     -
#       ip_netmask: {get_param: ManagementIpSubnet}
```

outputs:

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

The `~/custom-templates/nic-configs/controller-nics.yaml` file contains the following:

```
heat_template_version: 2015-04-30

description: >
  Software Config to drive os-net-config to configure VLANs 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
  ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
```

```
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
TenantNetworkVlanID:
    default: 50
    description: Vlan ID for the tenant network traffic.
    type: number
ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
ExternalInterfaceDefaultRoute:
    default: '10.0.0.1'
    description: default route for the external network
    type: string
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: p2p1
              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}
-
  type: ovs_bridge
  # Assuming you want to keep br-ex as external bridge name
  name: {get_input: bridge_name}
  use_dhcp: false
  addresses:
  -
    ip_netmask: {get_param: ExternalIpSubnet}
  routes:
  -
    ip_netmask: 0.0.0.0/0
    next_hop: {get_param: ExternalInterfaceDefaultRoute}
  members:
  -
    type: interface
    name: p2p2
    # force the MAC address of the bridge to this interface
    primary: true
-
  # Unused Interface
  type: interface
  name: em3
  use_dhcp: false
  defroute: false
-
  # Unused Interface
  type: interface
  name: em4
  use_dhcp: false
  defroute: false
-
  # Unused Interface
  type: interface
  name: p2p3
  use_dhcp: false
  defroute: false
-

```



```
type: interface
name: p2p4
use_dhcp: false
addresses:
-
  ip_netmask: {get_param: InternalApiIpSubnet}
-
type: interface
name: em2
use_dhcp: false
mtu: 9000
-
type: vlan
device: em2
mtu: 9000
use_dhcp: false
vlan_id: {get_param: StorageMgmtNetworkVlanID}
addresses:
-
  ip_netmask: {get_param: StorageMgmtIpSubnet}
-
type: vlan
device: em2
mtu: 9000
use_dhcp: false
vlan_id: {get_param: StorageNetworkVlanID}
addresses:
-
  ip_netmask: {get_param: StorageIpSubnet}
-
# VLAN for VXLAN tenant networking
type: ovs_bridge
name: br-tenant
mtu: 1500
use_dhcp: false
members:
-
  type: interface
  name: em1
  mtu: 1500
  use_dhcp: false
  # force the MAC address of the bridge to this interface
  primary: true
-
  type: vlan
  mtu: 1500
  vlan_id: {get_param: TenantNetworkVlanID}
  addresses:
```

```

-
  ip_netmask: {get_param: TenantIpSubnet}
# Uncomment when including environments/network-management.yaml
#-
# type: interface
# name: nic7
# use_dhcp: false
# addresses:
#   -
#     ip_netmask: {get_param: ManagementIpSubnet}

```

outputs:

```

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

```

The `~/custom-templates/ceph.yaml` file contains the following:

```

resource_registry:
  OS::Triple0::NodeUserData: /home/stack/custom-templates/first-boot-template.yaml
  OS::Triple0::NodeExtraConfigPost: /home/stack/custom-templates/post-deploy-
template.yaml

parameter_defaults:
  ExtraConfig:
    ceph::profile::params::fsid: eb2bb192-b1c9-11e6-9205-525400330666
    ceph::profile::params::osd_pool_default_pg_num: 256
    ceph::profile::params::osd_pool_default_pgp_num: 256
    ceph::profile::params::osd_pool_default_size: 3
    ceph::profile::params::osd_pool_default_min_size: 2
    ceph::profile::params::osd_recovery_max_active: 3
    ceph::profile::params::osd_max_backfills: 1
    ceph::profile::params::osd_recovery_op_priority: 2
  OsdComputeExtraConfig:
    ceph::profile::params::osd_journal_size: 5120
    ceph::profile::params::osds:
      '/dev/sda':
        journal: '/dev/sdm'
      '/dev/sdb':
        journal: '/dev/sdm'
      '/dev/sdc':
        journal: '/dev/sdm'
      '/dev/sdd':

```

```
    journal: '/dev/sdm'
  '/dev/sde':
    journal: '/dev/sdn'
  '/dev/sdf':
    journal: '/dev/sdn'
  '/dev/sdg':
    journal: '/dev/sdn'
  '/dev/sdh':
    journal: '/dev/sdn'
  '/dev/sdi':
    journal: '/dev/sdo'
  '/dev/sdj':
    journal: '/dev/sdo'
  '/dev/sdk':
    journal: '/dev/sdo'
  '/dev/sdl':
    journal: '/dev/sdo'
```

The `~/custom-templates/first-boot-template.yaml` file contains the following:

```
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}
```

The `~/custom-templates/wipe-disk.sh` file contains the following:

```
#!/usr/bin/env bash
if [[ `hostname` = *"ceph"* ]] || [[ `hostname` = *"osd-compute"* ]]
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/${DEVICE}
        else
            echo "Root found in /dev/${DEVICE}"
        fi
    done
fi
```

The `~/custom-templates/layout.yaml` file contains the following:

```
resource_registry:

    OS::TripleO::Controller::Ports::InternalApiPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/internal_api_from_pool.yaml
    OS::TripleO::Controller::Ports::TenantPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/tenant_from_pool.yaml
    OS::TripleO::Controller::Ports::StoragePort: /usr/share/openstack-tripleo-heat-
templates/network/ports/storage_from_pool.yaml
    OS::TripleO::Controller::Ports::StorageMgmtPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/storage_mgmt_from_pool.yaml

    OS::TripleO::OsdCompute::Ports::InternalApiPort: /usr/share/openstack-tripleo-heat-
```

```
templates/network/ports/internal_api_from_pool.yaml
OS::TripleO::OsdCompute::Ports::TenantPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/tenant_from_pool.yaml
OS::TripleO::OsdCompute::Ports::StoragePort: /usr/share/openstack-tripleo-heat-
templates/network/ports/storage_from_pool.yaml
OS::TripleO::OsdCompute::Ports::StorageMgmtPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/storage_mgmt_from_pool.yaml
```

```
parameter_defaults:
```

```
  NtpServer: 10.5.26.10
```

```
  ControllerCount: 3
```

```
  ComputeCount: 0
```

```
  CephStorageCount: 0
```

```
  OsdComputeCount: 3
```

```
  ControllerSchedulerHints:
```

```
    'capabilities:node': 'controller-%index%'
```

```
  NovaComputeSchedulerHints:
```

```
    'capabilities:node': 'compute-%index%'
```

```
  CephStorageSchedulerHints:
```

```
    'capabilities:node': 'ceph-storage-%index%'
```

```
  OsdComputeSchedulerHints:
```

```
    'capabilities:node': 'osd-compute-%index%'
```

```
  ControllerIPs:
```

```
    internal_api:
```

```
      - 192.168.2.200
```

```
      - 192.168.2.201
```

```
      - 192.168.2.202
```

```
    tenant:
```

```
      - 192.168.3.200
```

```
      - 192.168.3.201
```

```
      - 192.168.3.202
```

```
    storage:
```

```
      - 172.16.1.200
```

```
      - 172.16.1.201
```

```
      - 172.16.1.202
```

```
    storage_mgmt:
```

```
      - 172.16.2.200
```

```
      - 172.16.2.201
```

```
      - 172.16.2.202
```

```
  OsdComputeIPs:
```

```
    internal_api:
```

```
      - 192.168.2.203
```

```
      - 192.168.2.204
```

```

- 192.168.2.205
#- 192.168.2.206
tenant:
- 192.168.3.203
- 192.168.3.204
- 192.168.3.205
#- 192.168.3.206
storage:
- 172.16.1.203
- 172.16.1.204
- 172.16.1.205
#- 172.16.1.206
storage_mgmt:
- 172.16.2.203
- 172.16.2.204
- 172.16.2.205
#- 172.16.2.206

```

The `~/custom-templates/custom-roles.yaml` file contains the following:

```

# Specifies which roles (groups of nodes) will be deployed
# Note this is used as an input to the various *.j2.yaml
# jinja2 templates, so that they are converted into *.yaml
# during the plan creation (via a mistral action/workflow).
#
# The format is a list, with the following format:
#
# * name: (string) mandatory, name of the role, must be unique
#
# CountDefault: (number) optional, default number of nodes, defaults to 0
# sets the default for the {{role.name}}Count parameter in overcloud.yaml
#
# HostnameFormatDefault: (string) optional default format string for hostname
# defaults to '%stackname%-{{role.name.lower()}}-%index%'
# sets the default for {{role.name}}HostnameFormat parameter in overcloud.yaml
#
# ServicesDefault: (list) optional default list of services to be deployed
# on the role, defaults to an empty list. Sets the default for the
# {{role.name}}Services parameter in overcloud.yaml

- name: Controller
  CountDefault: 1
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephMon
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephRgw

```

- OS::Triple0::Services::CinderApi
- OS::Triple0::Services::CinderBackup
- OS::Triple0::Services::CinderScheduler
- OS::Triple0::Services::CinderVolume
- OS::Triple0::Services::Core
- OS::Triple0::Services::Kernel
- OS::Triple0::Services::Keystone
- OS::Triple0::Services::GlanceApi
- OS::Triple0::Services::GlanceRegistry
- OS::Triple0::Services::HeatApi
- OS::Triple0::Services::HeatApiCfn
- OS::Triple0::Services::HeatApiCloudwatch
- OS::Triple0::Services::HeatEngine
- OS::Triple0::Services::MySQL
- OS::Triple0::Services::NeutronDhcpAgent
- OS::Triple0::Services::NeutronL3Agent
- OS::Triple0::Services::NeutronMetadataAgent
- OS::Triple0::Services::NeutronApi
- OS::Triple0::Services::NeutronCorePlugin
- OS::Triple0::Services::NeutronOvsAgent
- OS::Triple0::Services::RabbitMQ
- OS::Triple0::Services::HAproxy
- OS::Triple0::Services::Keepalived
- OS::Triple0::Services::Memcached
- OS::Triple0::Services::Pacemaker
- OS::Triple0::Services::Redis
- OS::Triple0::Services::NovaConductor
- OS::Triple0::Services::MongoDb
- OS::Triple0::Services::NovaApi
- OS::Triple0::Services::NovaMetadata
- OS::Triple0::Services::NovaScheduler
- OS::Triple0::Services::NovaConsoleauth
- OS::Triple0::Services::NovaVncProxy
- OS::Triple0::Services::Ntp
- OS::Triple0::Services::SwiftProxy
- OS::Triple0::Services::SwiftStorage
- OS::Triple0::Services::SwiftRingBuilder
- OS::Triple0::Services::Snmp
- OS::Triple0::Services::Timezone
- OS::Triple0::Services::CeilometerApi
- OS::Triple0::Services::CeilometerCollector
- OS::Triple0::Services::CeilometerExpirer
- OS::Triple0::Services::CeilometerAgentCentral
- OS::Triple0::Services::CeilometerAgentNotification
- OS::Triple0::Services::Horizon
- OS::Triple0::Services::GnocchiApi
- OS::Triple0::Services::GnocchiMetricd
- OS::Triple0::Services::GnocchiStatsd

```

- OS::TripleO::Services::ManilaApi
- OS::TripleO::Services::ManilaScheduler
- OS::TripleO::Services::ManilaBackendGeneric
- OS::TripleO::Services::ManilaBackendNetapp
- OS::TripleO::Services::ManilaBackendCephFs
- OS::TripleO::Services::ManilaShare
- OS::TripleO::Services::AodhApi
- OS::TripleO::Services::AodhEvaluator
- OS::TripleO::Services::AodhNotifier
- OS::TripleO::Services::AodhListener
- OS::TripleO::Services::SaharaApi
- OS::TripleO::Services::SaharaEngine
- OS::TripleO::Services::IronicApi
- OS::TripleO::Services::IronicConductor
- OS::TripleO::Services::NovaIronic
- OS::TripleO::Services::TripleoPackages
- OS::TripleO::Services::TripleoFirewall
- OS::TripleO::Services::OpenDaylightApi
- OS::TripleO::Services::OpenDaylightOvs
- OS::TripleO::Services::SensuClient
- OS::TripleO::Services::FluentdClient
- OS::TripleO::Services::VipHosts

- name: Compute
  CountDefault: 1
  HostnameFormatDefault: '%stackname%-compute-%index%'
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::NeutronSriovAgent
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::VipHosts

```



```
- name: BlockStorage
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::BlockStorageCinderVolume
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::VipHosts

- name: ObjectStorage
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::SwiftStorage
    - OS::TripleO::Services::SwiftRingBuilder
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::VipHosts

- name: CephStorage
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephOSD
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::VipHosts

- name: OsdCompute
  CountDefault: 0
  HostnameFormatDefault: '%stackname%-osd-compute-%index%'
  ServicesDefault:
```

```
- OS::Triple0::Services::CephOSD
- OS::Triple0::Services::CACerts
- OS::Triple0::Services::CephClient
- OS::Triple0::Services::CephExternal
- OS::Triple0::Services::Timezone
- OS::Triple0::Services::Ntp
- OS::Triple0::Services::Snmp
- OS::Triple0::Services::NovaCompute
- OS::Triple0::Services::NovaLibvirt
- OS::Triple0::Services::Kernel
- OS::Triple0::Services::ComputeNeutronCorePlugin
- OS::Triple0::Services::ComputeNeutronOvsAgent
- OS::Triple0::Services::ComputeCeilometerAgent
- OS::Triple0::Services::ComputeNeutronL3Agent
- OS::Triple0::Services::ComputeNeutronMetadataAgent
- OS::Triple0::Services::TripleoPackages
- OS::Triple0::Services::TripleoFirewall
- OS::Triple0::Services::NeutronSriovAgent
- OS::Triple0::Services::OpenDaylightOvs
- OS::Triple0::Services::SensuClient
- OS::Triple0::Services::FluentdClient
- OS::Triple0::Services::VipHosts
```

The `~/custom-templates/compute.yaml` file contains the following:

```
parameter_defaults:
  ExtraConfig:
    nova::compute::reserved_host_memory: 75000
    nova::cpu_allocation_ratio: 8.2
```

The `~/custom-templates/post-deploy-template.yaml` file contains the following:

```
heat_template_version: 2014-10-16

parameters:
  servers:
    type: json

resources:

  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      inputs:
        - name: OSD_NUMA_INTERFACE
      config: {get_file: numa-systemd-osd.sh}

  ExtraDeployments:
    type: OS::Heat::SoftwareDeployments
    properties:
      servers: {get_param: servers}
      config: {get_resource: ExtraConfig}
      input_values:
        OSD_NUMA_INTERFACE: 'em2'
      actions: ['CREATE']
```

The `~/custom-templates/numa-systemd-osd.sh` file contains the following:

```
#!/usr/bin/env bash
{
if [[ `hostname` = *"ceph"* ]] || [[ `hostname` = *"osd-compute"* ]]; then

    # Verify the passed network interface exists
    if [[ ! $(ip add show $OSD_NUMA_INTERFACE) ]]; then
        exit 1
    fi

    # If NUMA related packages are missing, then install them
    # If packages are baked into image, no install attempted
    for PKG in numactl hwloc; do
        if [[ ! $(rpm -q $PKG) ]]; then
            yum install -y $PKG
            if [[ ! $? ]]; then
                echo "Unable to install $PKG with yum"
                exit 1
            fi
        fi
    done
fi
}
```

```

    fi
fi
done

# Find the NUMA socket of the $OSD_NUMA_INTERFACE
declare -A NUMASOCKET
while read TYPE SOCKET_NUM NIC ; do
    if [[ "$TYPE" == "NUMANode" ]]; then
        NUMASOCKET=$(echo $SOCKET_NUM | sed s/L//g);
    fi
    if [[ "$NIC" == "$OSD_NUMA_INTERFACE" ]]; then
        # because $NIC is the $OSD_NUMA_INTERFACE,
        # the NUMASOCKET has been set correctly above
        break # so stop looking
    fi
done < <(lstopo-no-graphics | tr -d [:punct:] | egrep "NUMANode|$OSD_NUMA_INTERFACE")

if [[ -z $NUMASOCKET ]]; then
    echo "No NUMANode found for $OSD_NUMA_INTERFACE. Exiting."
    exit 1
fi

UNIT='/usr/lib/systemd/system/ceph-osd@.service'
# Preserve the original ceph-osd start command
CMD=$(crudini --get $UNIT Service ExecStart)

if [[ $(echo $CMD | grep numactl) ]]; then
    echo "numactl already in $UNIT. No changes required."
    exit 0
fi

# NUMA control options to append in front of $CMD
NUMA="/usr/bin/numactl -N $NUMASOCKET --preferred=$NUMASOCKET"

# Update the unit file to start with numactl
# TODO: why doesn't a copy of $UNIT in /etc/systemd/system work with numactl?
crudini --verbose --set $UNIT Service ExecStart "$NUMA $CMD"

# Reload so updated file is used
systemctl daemon-reload

# Restart OSDs with NUMA policy (print results for log)
OSD_IDS=$(ls /var/lib/ceph/osd | awk 'BEGIN { FS = "-" } ; { print $2 }')
for OSD_ID in $OSD_IDS; do
    echo -e "\nStatus of OSD $OSD_ID before unit file update\n"
    systemctl status ceph-osd@$OSD_ID
    echo -e "\nRestarting OSD $OSD_ID..."
    systemctl restart ceph-osd@$OSD_ID
done

```

```
echo -e "\nStatus of OSD $OSD_ID after unit file update\n"
systemctl status ceph-osd@$OSD_ID
done
fi
} 2>&1 > /root/post_deploy_heat_output.txt
```

Appendix E: Nova Memory and CPU Calculator

The file referenced in this appendix may be found online. See the [GitHub Repository of Example Files](#) Appendix for more details.

```
#!/usr/bin/env python
# Filename:          nova_mem_cpu_calc.py
# Supported Language(s): Python 2.7.x
# Time-stamp:       <2017-03-10 20:31:18 jfulton>
# -----
# This program was originally written by Ben England
# -----
# Calculates cpu_allocation_ratio and reserved_host_memory
# for nova.conf based on the following inputs:
#
# input command line parameters:
# 1 - total host RAM in GB
# 2 - total host cores
# 3 - Ceph OSDs per server
# 4 - average guest size in GB
# 5 - average guest CPU utilization (0.0 to 1.0)
#
# It assumes that we want to allow 3 GB per OSD
# (based on prior Ceph Hammer testing)
# and that we want to allow an extra 1/2 GB per Nova (KVM guest)
# based on test observations that KVM guests' virtual memory footprint
# was actually significantly bigger than the declared guest memory size
# This is more of a factor for small guests than for large guests.
# -----
import sys
from sys import argv

NOTOK = 1 # process exit status signifying failure
MB_per_GB = 1000

GB_per_OSD = 3
GB_overhead_per_guest = 0.5 # based on measurement in test environment
cores_per_OSD = 1.0 # may be a little low in I/O intensive workloads

def usage(msg):
    print msg
    print(
        ("Usage: %s Total-host-RAM-GB Total-host-cores OSDs-per-server " +
         "Avg-guest-size-GB Avg-guest-CPU-util") % sys.argv[0])
```

```
sys.exit(NOTOK)

if len(argv) < 5: usage("Too few command line params")
try:
    mem = int(argv[1])
    cores = int(argv[2])
    osds = int(argv[3])
    average_guest_size = int(argv[4])
    average_guest_util = float(argv[5])
except ValueError:
    usage("Non-integer input parameter")

average_guest_util_percent = 100 * average_guest_util

# print inputs
print "Inputs:"
print "- Total host RAM in GB: %d" % mem
print "- Total host cores: %d" % cores
print "- Ceph OSDs per host: %d" % osds
print "- Average guest memory size in GB: %d" % average_guest_size
print "- Average guest CPU utilization: %.0f%" % average_guest_util_percent

# calculate operating parameters based on memory constraints only
left_over_mem = mem - (GB_per_OSD * osds)
number_of_guests = int(left_over_mem /
                        (average_guest_size + GB_overhead_per_guest))
nova_reserved_mem_MB = MB_per_GB * (
    (GB_per_OSD * osds) +
    (number_of_guests * GB_overhead_per_guest))
nonceph_cores = cores - (cores_per_OSD * osds)
guest_vCPUs = nonceph_cores / average_guest_util
cpu_allocation_ratio = guest_vCPUs / cores

# display outputs including how to tune Nova reserved mem

print "\nResults:"
print "- number of guests allowed based on memory = %d" % number_of_guests
print "- number of guest vCPUs allowed = %d" % int(guest_vCPUs)
print "- nova.conf reserved_host_memory = %d MB" % nova_reserved_mem_MB
print "- nova.conf cpu_allocation_ratio = %f" % cpu_allocation_ratio

if nova_reserved_mem_MB > (MB_per_GB * mem * 0.8):
    print "ERROR: you do not have enough memory to run hyperconverged!"
    sys.exit(NOTOK)

if cpu_allocation_ratio < 0.5:
    print "WARNING: you may not have enough CPU to run hyperconverged!"
```

```
if cpu_allocation_ratio > 16.0:
    print(
        "WARNING: do not increase VCPU overcommit ratio " +
        "beyond OSP8 default of 16:1")
    sys.exit(NOTOK)

print "\nCompare \"guest vCPUs allowed\" to \"guests allowed based on memory\" for actual
guest count"
```


Appendix F: Example Fencing Script

In [Configure Controller Pacemaker Fencing](#) the following script was used to configure **Pacemaker** fencing. The script comes directly from the reference architecture [Deploying Red Hat Enterprise Linux OpenStack Platform 7 with RHEL-OSP Director 7.1](#). The script, called *configure_fence.sh*, is available online. See the [GitHub Repository of Example Files](#) Appendix for more details.

```
#!/bin/bash

source ~/stackrc
env | grep OS_
SSH_CMD="ssh -l heat-admin"

function usage {
    echo "USAGE: $0 [enable|test]"
    exit 1
}

function enable_stonith {
    # for all controller nodes
    for i in $(nova list | awk ' /controller/ { print $12 } ' | cut -f2 -d=)
    do
        echo $i
        # create the fence device
        $SSH_CMD $i 'sudo pcs stonith create $(hostname -s)-ipmi fence_ipmilan
pcmk_host_list=$(hostname -s) ipaddr=$(sudo ipmitool lan print 1 | awk " /IP Address / {
print \$4 } ") login=root passwd=PASSWORD lanplus=1 cipher=1 op monitor interval=60sr'
        # avoid fencing yourself
        $SSH_CMD $i 'sudo pcs constraint location $(hostname -s)-ipmi avoids $(hostname
-s)'
    done

    # enable STONITH devices from any controller
    $SSH_CMD $i 'sudo pcs property set stonith-enabled=true'
    $SSH_CMD $i 'sudo pcs property show'
}

function test_fence {

    for i in $(nova list | awk ' /controller/ { print $12 } ' | cut -f2 -d= | head -n 1)
    do
        # get REDIS_IP
        REDIS_IP=$(SSH_CMD $i 'sudo grep -ri redis_vip /etc/puppet/hieradata/' | awk
'/vip_data.yaml/ { print $2 } ')
    done
}
```

```
# for all controller nodes
for i in $(nova list | awk ' /controller/ { print $12 } ' | cut -f2 -d=)
do
    if $SSH_CMD $i "sudo ip a" | grep -q $REDIS_IP
    then
        FENCE_DEVICE=$(($SSH_CMD $i 'sudo pcs stonith show $(hostname -s)-ipmi' | awk
' /Attributes/ { print $2 } ' | cut -f2 -d=)
        IUUUID=$(nova list | awk " /$i/ { print \$2 } ")
        UUID=$(ironic node-list | awk " /$IUUUID/ { print \$2 } ")
    else
        FENCER=$i
    fi
done 2>/dev/null

echo "REDIS_IP $REDIS_IP"
echo "FENCER $FENCER"
echo "FENCE_DEVICE $FENCE_DEVICE"
echo "UUID $UUID"
echo "IUUUID $IUUUID"

# stonith REDIS_IP owner
$SSH_CMD $FENCER sudo pcs stonith fence $FENCE_DEVICE

sleep 30

# fence REDIS_IP owner to keep ironic from powering it on
sudo ironic node-set-power-state $UUID off

sleep 60

# check REDIS_IP failover
$SSH_CMD $FENCER sudo pcs status | grep $REDIS_IP
}

if [ "$1" == "test" ]
then
    test_fence
elif [ "$1" == "enable" ]
then
    enable_stonith
else
    usage
fi
```

Appendix G: GitHub Repository of Example Files

The example custom templates and scripts provided in this reference implementation may be accessed online from <https://github.com/RHsyseng/hci>.



redhat.®