Red Hat OpenStack Platform 11
Hyper-Converged Infrastructure Guide

Understanding and configuring Hyper-Converged Infrastructure on the Red Hat OpenStack Platform overcloud

OpenStack Team

Red Hat OpenStack Platform 11 Hyper-Converged Infrastructure Guide

Understanding and configuring Hyper-Converged Infrastructure on the Red Hat OpenStack Platform overcloud

OpenStack Team
rhos-docs@redhat.com

Abstract

This document describes the Red Hat OpenStack Platform implementation of hyper-convergence, wherein Compute and Ceph Storage services are co-located on the same host. Red Hat OpenStack Platform supports both pure and mixed hyper-converged infrastructure types. This document also describes how to enable either one in your overcloud.

Table of Contents

CHAPTER 1. INTRODUCTION

The Red Hat OpenStack Platform implementation of hyper-converged infrastructures (HCI) uses Red Hat Ceph Storage as a storage provider. This infrastructure features hyper-converged nodes, where Compute and Ceph Storage services are co-located and configured for optimized resource usage. HCI supports either:

- Pure HCI: The Compute needs of an overcloud are provided entirely by hyper-converged nodes
- Mixed HCI: A mixture of hyper-converged and normal Compute nodes (mixed HCI)

This document describes how to deploy HCI of either type on an overcloud, in a way that allows integration with other features (for example, Network Function Virtualization). In addition, this document also covers how to ensure optimal performance of both Compute and Ceph Storage services on hyper-converged nodes.

1.1. ASSUMPTIONS

This document does not provide a complete deployment walkthrough for deploying HCI. Rather, it describes the settings required for deploying hyper-converged nodes on an overcloud. This allows you to integrate HCI seamlessly into your overcloud deployment plan.

The following sections also assume that:

1. The undercloud has already been deployed. For instructions on how to deploy the undercloud, see Director Installation and Usage.
2. Your environment can provision nodes that meet Compute and Ceph Storage requirements. See Overcloud Requirements (from Director Installation and Usage) for details.
3. All nodes in your environment have already been prepared. This means that the nodes have already been:

   a. Registered (as described in Registering the Nodes), and

   b. Tagged (as described in Manually Tagging the Nodes)

   For more information, see Red Hat Ceph Storage for the Overcloud.
4. The disks on nodes destined for Compute and Ceph OSD services must be cleaned, as described in Cleaning Ceph Storage Node Disks (from Red Hat Ceph Storage for the Overcloud).
5. You have an environment file prepared for registering your overcloud nodes, as described in Registering the Overcloud with an Environment File (from Advanced Overcloud Customization). The procedure in Configure Ceph NUMA Pinning includes a script that requires the installation of packages not provided by the director.

1.2. REFERENCES

This document is intended to be a supplement to several existing Red Hat OpenStack Platform documents. Refer to these documents for more detailed information on related concepts:

- Advanced Overcloud Customization: describes methods for configuring advanced OpenStack features through the director (for example, the use of custom roles).

- Director Installation and Usage: provides end-to-end deployment information for both undercloud and overcloud.
- Red Hat Ceph Storage for the Overcloud: describes how to deploy an overcloud that uses Red Hat Ceph Storage as a storage provider.
- Networking Guide: an advanced guide to Red Hat OpenStack Platform networking.
- Hyper-converged Red Hat OpenStack Platform 10 and Red Hat Ceph Storage 2 a reference architecture that describes how to deploy an environment featuring HCI on very specific hardware.

CHAPTER 2. PROCESS DESCRIPTION

Like most Red Hat OpenStack Platform features, hyper-convergence is best implemented through the director. This allows you to take advantage of existing Heat templates and environment files to orchestrate your deployment. In particular, the director also includes a default environment file for implementing a supported version of hyper-convergence.

On the other hand, the director’s infrastructure also provides a framework you can use to define your own Heat templates and environment files. This is useful when the existing ones do not cover your specific use case.

The following subsections briefly describe each step of the deployment process.

Choosing a Deployment Type

Red Hat OpenStack Platform allows you to deploy two types of HCIs: *pure* and *mixed*. For both types, Red Hat Ceph Storage is deployed as the primary storage provider for Red Hat OpenStack Platform services.

On pure HCI, all Compute nodes will be deployed with Ceph-OSD services co-located on them.

On mixed HCI, you can deploy normal Compute nodes alongside those with co-located Ceph-OSD services.

Before you start your deployment, determine which type of HCI you are deploying, as their respective implementations are quite different.

Configuring Resource Isolation

When you deploy HCI, the director does not configure Compute and Ceph Storage services to be aware of each other on hyper-converged nodes. Both services will consume resources as if they were on dedicated nodes. This can lead to resource contention, which can lead to performance degradation. You need to mitigate this manually.

Configure Networking

For both Pure and Mixed HCI deployments, you need to map the **`StorageMgmtNetwork`** ports to the right NICs. During this step, you can implement any other networking settings required in your environment.

Deployment

The deployment process for HCI (whether pure or mixed) involves specifying which environment files to include in the deployment. Pure HCI involves specifying an existing environment file that configures the services included in the Compute role. Mixed HCI involves defining a new flavor, tagging it to hyper-converged nodes, and invoking a custom roles file during deployment.

CHAPTER 3. CHOOSING A DEPLOYMENT TYPE

Deploying pure HCI is simpler, as it involves invoking an environment file that adds the Ceph-OSD services to the Compute role. Deploying mixed HCI involves defining a custom role, flavor, and networking settings for HCI nodes. As both HCI types offer contrasting deployment approaches, determine which type is more suitable for your environment.

The following sub-sections walk through each deployment type. For either deployment type, create a **`/home/stack/templates/`** directory to store any custom environment files and Heat templates.

3.1. PURE HCI

To deploy Red Hat OpenStack with pure, use the following environment file:

`/usr/share/openstack-tripleo-heat-templates/environments/hyperconverged-ceph.yaml`

This environment file redefines the Compute role by adding the **`CephOSD`** service to it. By default, the **`hyperconverged-ceph.yaml`** environment file assumes that you are using an isolated **`StorageMgmt`** network, and configures the Compute ports accordingly.

If you are not using an isolated **`StorageMgmt`** network, disable the **`StorageMgmtPort`** service. To do this:

1. Create a new environment file named **`hyperconverged-non-isolated.yaml`** in **`~/templates/`** containing the following:

   ```
   resource_registry:
     OS::TripleO::Compute::Ports::StorageMgmtPort::None
   ```

2. During the deployment process (Section 6.1, “Deploying Pure HCI”), invoke **`~/templates/hyperconverged-non-isolated.yaml`** when you run **`openstack overcloud deploy`**. This environment file must be invoked last to ensure that it overrides the others correctly.

Proceed to Chapter 4, *Configuring Resource Isolation on Hyper-converged Nodes* for instructions on mitigating resource contention between Compute and Ceph Storage services.

3.2. MIXED HCI

The Overcloud usually consists of nodes in predefined roles such as Controller nodes, Compute nodes, and different storage node types. Each of these default roles contains a set of services defined in the core Heat template collection on the director node. However, the architecture of the core Heat templates provides a method to:

- Create custom roles
- Add and remove services from each role

This allows us to define a new role both Compute and Ceph OSD services. This effectively co-locates both services, allowing you to deploy them together on the same *hyper-converged node*.

Note

For detailed information about custom roles, see Composable Services and Custom Roles from Advanced Overcloud Customization.

3.2.1. Creating a Custom Role for HCI Nodes

On the undercloud, default roles are defined in the following file:

`/usr/share/openstack-tripleo-heat-templates/roles_data.yaml`

Copy this file to the custom templates directory you created (namely, **`~/templates/`**):

```
$ cp usr/share/openstack-tripleo-heat-templates/roles_data.yaml
~/templates/roles_data_hci.yaml
```

To define a new role that co-locates both Compute and Ceph OSD, create a role that combines the services of both **`Compute`** and **`CephStorage`** entries. To do this, add a new role named **`OsdCompute`** in **`~/templates/roles_data_hci.yaml`**, copy the **`Compute`** role services to **`OsdCompute`**, and add the Ceph OSD service:

```
- name: OsdCompute # 1
  CountDefault: 1 # 2
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephOSD # 3
    - OS::TripleO::Services::Timezone
[...]
```

1

A role must have a unique name, defined here as **`OsdCompute`**.

2

The **`CountDefault:`** parameter how many nodes the director should apply the role to by default (in this case, **`1`**).

3

The **`OS::TripleO::Services::CephOSD`** entry is the only one on the **`CephStorage`** role that does not exist on the **`Compute`** role.

Red Hat OpenStack Platform supports deployments that feature both hyper-converged and non-hyper-converged Compute nodes. If you do not want to deploy any non-hyper-converged Compute nodes, set the **`CountDefault:`** parameter of the **`Compute`** role to **`0`**:

```
- name: Compute
  CountDefault: 0
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Ntp
[...]
```

You can use the CountDefault: parameter to define how many nodes to assign to each role. However, we recommend that you do so in a separate Heat template, as described later in Section 6.2, “Deploying Mixed HCI”.

3.2.2. Configuring Port Assignments for the Custom Role

The default Heat templates in **`/usr/share/openstack-tripleo-heat-templates/`** provide the necessary network settings for the default roles. These settings include how IP addresses and ports should be assigned for each service on each node.

Custom roles (like **`OsdCompute`** in Section 3.2.1, “Creating a Custom Role for HCI Nodes”) do not have the required port assignment Heat templates, so you need to define these yourself. To do so, create a new Heat template named **`ports.yaml`** in **`~/templates`** containing the following:

```
resource_registry:
  OS::TripleO::OsdCompute::Ports::ExternalPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/noop.yaml  # 1
  OS::TripleO::OsdCompute::Ports::InternalApiPort:
/usr/share/openstack-tripleo-heat-
templates/network/ports/internal_api.yaml
  OS::TripleO::OsdCompute::Ports::StoragePort: /usr/share/openstack-
tripleo-heat-templates/network/ports/storage.yaml
  OS::TripleO::OsdCompute::Ports::StorageMgmtPort:
/usr/share/openstack-tripleo-heat-
templates/network/ports/storage_mgmt_from_pool.yaml
  OS::TripleO::OsdCompute::Ports::TenantPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/tenant.yaml
  OS::TripleO::OsdCompute::Ports::StorageMgmtPort:
/usr/share/openstack-tripleo-heat-
templates/network/ports/storage_mgmt.yaml
```

1

If you are using DVR, replace this line with:

```
  OS::TripleO::OsdCompute::Ports::ExternalPort:
/usr/share/openstack-tripleo-heat-
templates/network/ports/external.yaml
```

See Configure Distributed Virtual Routing (DVR) from the Networking Guide for more details.

See Isolating Networks and Selecting Networks to Deploy (from Advanced Overcloud Customization) for related information.

3.2.3. Creating and Assigning a New Flavor

As mentioned in Section 1.1, “Assumptions”, you should have already registered and tagged each node with a corresponding flavor. However, since deploying mixed HCI involves defining a new OsdCompute role, you also need to create a new flavor for it:

1. To create a new role named **`osdcompute`**, run:

   ```
   $ openstack flavor create --id auto --ram 6144 --disk 40 --vcpus
   4 osdcompute
   ```

   Note

   For more details about this command, run **`openstack flavor create --help`**.

2. Map this flavor to a new profile, also named **`osdcompute`**:

   ```
   $ openstack flavor set --property "cpu_arch"="x86_64" --property
   "capabilities:boot_option"="local" --property
   "capabilities:profile"="osdcompute" osdcompute
   ```

   Note

   For more details about this command, run **`openstack flavor set --help`**.

3. Tag nodes into the new **`osdcompute`** profile:

   ```
   $ ironic node-update UUID add
   properties/capabilities='profile:osdcompute,boot_option:local'
   ```

   Note

   For more details about tagging nodes, see Manually Tagging the Nodes (from Red Hat Ceph Storage for the Overcloud).

See Manually Tagging the Nodes and Assigning Nodes and Flavors to Roles (from (from Red Hat Ceph Storage for the Overcloud) for related details.

CHAPTER 4. CONFIGURING RESOURCE ISOLATION ON HYPER-CONVERGED NODES

Whether through the default **`hyperconverged-ceph.yaml`** file (in Section 3.1, “Pure HCI”) or the custom **`OsdCompute`** role (in Section 3.2, “Mixed HCI”), the director creates hyper-converged nodes by co-locating Ceph OSD and Compute services. However, without any further tuning this co-location also risks *resource contention* between Ceph and Compute services, as neither are aware of each other’s presence on the same host. Resource contention can result in degradation of service. This, in turn, offsets any benefits provided by hyper-convergence.

To prevent contention, you need to configure resource isolation for both Ceph and Compute services. The following sub-sections describe how to do so.

4.1. RESERVE CPU AND MEMORY RESOURCES FOR COMPUTE

By default, the Compute service parameters do not take into account the co-location of Ceph OSD services on the same node. Hyper-converged nodes need to be tuned in order to address this to maintain stability and maximize the number of possible instances. Use the computations here as a guideline for an optimal baseline, then change the settings to find an acceptable trade-off between determinism and instance-hosting capacity. The examples provided in this document prioritize determinism and uptime.

The following Heat template parameters control how the Compute services consume memory and CPU resources on a node:

reserved_host_memory

This is the amount of memory (in MB) to reserve for the host node. To determine an appropriate value for hyper-converged nodes, assume that each OSD consumes 3GB of memory. Given a node with 256GB memory and 10 OSDs, you can allocate 30 GB of memory for Ceph, leaving 226 GB for Compute. With that much memory a node can host, for example, 113 instances using 2GB of memory each.

However, you still need to consider additional overhead per instance for the hypervisor. Assuming this overhead is 0.5 GB, the same node can only host 90 instances, which accounts for the 226GB divided by 2.5GB. The amount of memory to reserve for the host node (that is, memory the Compute service should not use) is:

`(In * Ov) + (Os * RA)`

Where:

- **`In`**: number of instances
- **`Ov`**: amount of overhead memory needed per instance
- **`Os`**: number of OSDs on the node
- **`RA`**: amount of RAM that each OSD should have

With 90 instances, this give us (90*0.5) + (10*3) = 75GB. The Compute service expects this value in MB, namely 75000.

The following Python code provides this computation:

```
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))
```

cpu_allocation_ratio

The Compute scheduler uses this when choosing which Compute nodes on which to deploy an instance. By default, this is 16.0 (as in, 16:1). This means if there are 56 cores on a node, the Compute scheduler will schedule enough instances to consume 896 vCPUs on a node before considering the node unable to host any more.

To determine a suitable cpu_allocation_ratio for a hyper-converged node, assume each Ceph OSD uses at least one core (unless the workload is I/O-intensive, and on a node with no SSD). On a node with 56 cores and 10 OSDs, this would leave 46 cores for Compute. If each instance uses 100 per cent of the CPU it receives, then the ratio would simply be the number of instance vCPUs divided by the number of cores; that is, 46 / 56 = 0.8. However, since instances do not normally consume 100 per cent of their allocated CPUs, you can raise the cpu_allocation_ratio by taking the anticipated percentage into account when determining the number of required guest vCPUs.

So, if we can predict that instances will only use 10 per cent (or 0.1) of their vCPU, then the number of vCPUs for instances can be expressed as 46 / 0.1 = 460. When this value is divided by the number of cores (56), the ratio increases to approximately 8.

The following Python code provides this computation:

```
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
```

Tip

You can also use the script in Section A.2.1, “Compute CPU and Memory Calculator” to compute baseline values for both **`reserved_host_memory`** and **`cpu_allocation_ratio`**.

After computing for the values you want to use, include them as defaults for HCI nodes. To do so, create a new environment file named compute.yaml in ~/templates containing your reserved_host_memory and cpu_allocation_ratio values. For pure HCI deployments, it should contain the following:

```
parameter_defaults:
  NovaComputeExtraConfig:  # 1
    nova::compute::reserved_host_memory: 181000
    nova::cpu_allocation_ratio: 8.2
```

1

The **NovaComputeExtraConfig** line applies all its nested parameters to all **Compute** roles. In a pure HCI deployment, all Compute nodes are also hyper-converged.

For mixed HCI, ~/templates/compute.yaml should contain:

```
parameter_defaults:
  OsdComputeExtraConfig:  # 1
    nova::compute::reserved_host_memory: 181000
    nova::cpu_allocation_ratio: 8.2
```

1

The **OsdComputeExtraConfig** line is a custom resource that applies all nested settings to the custom **OsdCompute** role, which we defined in Section 3.2, “Mixed HCI”.

4.2. CONFIGURE CEPH NUMA PINNING

When applying a hyper-converged role on a host that features NUMA, you can improve determinism by pinning the Ceph OSD services to one of the available NUMA sockets. When you do, pin the Ceph Storage services to the socket with the network IRQ and the storage controller. Doing this helps address the Ceph OSD’s heavy usage of network I/O.

You can orchestrate this through a simple shell script that takes a network interface as an argument and applies the necessary NUMA-related settings to the interface. This network interface will presumably be the one the Ceph OSD uses. Create this script (**numa-systemd-osd.sh**) in **~/templates**.

Important

See Section A.2.2, “Custom Script to Configure NUMA Pinning for Ceph OSD Services” for the contents of **numa-systemd-osd.sh**, including a more detailed description.

The **numa-systemd-osd.sh** script will also attempt to install NUMA configuration tools. As such, the overcloud nodes must also be registered with Red Hat, as described in Registering the Nodes (from Red Hat Ceph Storage for the Overcloud).

To run this script on the overcloud, first create a new Heat template named **ceph-numa-pinning-template.yaml** in **~/templates** with the following contents:

```
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} # 1

  ExtraDeployments:
    type: OS::Heat::SoftwareDeployments
    properties:
      servers: {get_param: servers}
      config: {get_resource: ExtraConfig}
      input_values:
        OSD_NUMA_INTERFACE: 'em2' # 2
      actions: ['CREATE'] # 3
```

1

The **get_file** function calls the **~/templates/numa-systemd-osd.sh**. This script should be able to take a network interface as an input (in this case, **OSD_NUMA_INTERFACE**) and perform the necessary NUMA-related configuration for it. See Section A.2.2, "Custom Script to Configure NUMA Pinning for Ceph OSD Services" for the contents of this script, along with a detailed description of how it works.

Important

On a Pure HCI deployment, you will need to edit the top-level **IF** statement in the **~/templates/numa-systemd-osd.sh** script. See Section A.2.2, "Custom Script to Configure NUMA Pinning for Ceph OSD Services" for details.

2

The **OSD_NUMA_INTERFACE** variable specifies the network interface that the Ceph OSD services should use (in this example, **em2**). The **~/templates/numa-systemd-osd.sh** script will apply the necessary NUMA settings to this interface.

3

As we only specify **CREATE** in **actions**, the script will only run during the initial deployment, and not during an update.

The interface used for **OSD_NUMA_INTERFACE** can be determined for all deployments by either the **StorageNetwork** variable, or the **StorageMgtmtNetwork** variable. Workloads that are read-heavy benefit from using the **StorageNetwork** interface, while write-heavy workloads benefit from using the **StorageMgtmtNetwork** one.

If the Ceph OSD service uses a virtual network interface (for example, a bond), use the name of the network devices that make up the bond, not the bond itself. For example, if **bond1** uses **em2** and **em4**, then set **OSD_NUMA_INTERFACE** to either **em2** or **em4** (not **bond1**). If the bond combines NICs which are not on the same NUMA node (as confirmed by **lstopo-no-graphics**), then do not use **numa-systemd-osd.sh**.

After creating the **`ceph-numa-pinning-template.yaml`** template, create an environment file named **`ceph-numa-pinning.yaml`** in **`~/templates`** with the following contents:

```
resource_registry:
  OS::TripleO::NodeExtraConfigPost: /home/stack/templates/ceph-numa-
pinning-template.yaml
```

This environment file will allow you to invoke the **`ceph-numa-pinning-template.yaml`** template later on in Chapter 6, *Deployment*.

4.3. REDUCE CEPH BACKFILL AND RECOVERY OPERATIONS

When a Ceph OSD is removed, Ceph uses *backfill* and *recovery* operations to rebalance the cluster. Ceph does this to keep multiple copies of data according to the placement group policy. These operations use system resources; as such, if a Ceph cluster is under load its performance will drop as it diverts resources to backfill and recovery.

To mitigate this performance effect during OSD removal, you can reduce the priority of backfill and recovery operations. Keep in mind that the trade off for this is that there are less data replicas for a longer time, which puts the data at a slightly greater risk.

To configure the priority of backfill and recovery operations, add an environment file named **`ceph-backfill-recovery.yaml`** to **`~/templates`** containing the following:

```
parameter_defaults:
  ExtraConfig:
    ceph::profile::params::osd_recovery_op_priority: 3 # 1
    ceph::profile::params::osd_recovery_max_active: 3 # 2
    ceph::profile::params::osd_max_backfills: 1 # 3
```

1

The **`osd_recovery_op_priority`** sets the priority for recovery operations, relative to the OSD client OP priority.

2

The **`osd_recovery_max_active`** sets 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. Set this to **`1`** if you want to reduce latency.

3

The **`osd_max_backfills`** sets the maximum number of backfills allowed to or from a single OSD.

Important

The values used in this sample are the current defaults. Unless you are planning to use **`ceph-backfill-recovery.yaml`** with different values, you do not need to add it to your deployment.

CHAPTER 5. FINALIZE NETWORKING SETTINGS

At this point, you should have completed the necessary settings to assign ports properly on HCI nodes for either Pure or HCI deployments. This configuration was described in either Section 3.1, “Pure HCI” or Section 3.2.2, “Configuring Port Assignments for the Custom Role”.

However, for both Pure and Mixed HCI deployments you still need to map the **`StorageMgmtPort`** to a physical NIC. To do this:

1. From the default Heat template collection, choose the Compute NIC configuration template suitable for your environment:

   - **`/usr/share/openstack-tripleo-heat-templates/network/config/single-nic-vlans/compute.yaml`**
   - **`/usr/share/openstack-tripleo-heat-templates/network/config/single-nic-linux-bridge-vlans/compute.yaml`**
   - **`/usr/share/openstack-tripleo-heat-templates/network/config/multiple-nics/compute.yaml`**
   - **`/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/compute.yaml`**

   See the **`README.md`** on each template’s respective directory for details about the NIC configuration.

2. Create a new directory within **`~/templates`** called **`nic-configs`** and then copy your chosen template to **`~/templates/nic-configs/`**.

3. Ensure the following definition is in the **`parameters:`** section of your new **`~/templates/nic-configs/compute.yaml`** template:

   ```
   StorageMgmtNetworkVlanID:
       default: 40
       description: Vlan ID for the storage mgmt network traffic.
       type: number
   ```

   Add the definition if it does not already exist (as is with **…`/single-nic-vlans/compute.yaml`**).

4. Map **`StorageMgmtNetworkVlanID`** to a specific NIC on each HCI node. For example, if you chose to trunk VLANs to a single NIC (that is, you copied **…`/single-nic-vlans/compute.yaml`**), then add the following entry to the **`network_config:`** section of **`~/templates/nic-configs/compute.yaml`**:

   ```
       -
                 type: vlan
                 device: em2
                 mtu: 9000  # 1
                 use_dhcp: false
   ```

```
                vlan_id: {get_param: StorageMgmtNetworkVlanID}
                addresses:
                  -
                    ip_netmask: {get_param: StorageMgmtIpSubnet}
```

1

When mapping a NIC to **`StorageMgmtNetworkVlanID`**, we recommend that you set the **`mtu`** to **`9000`** (jumbo frames). This MTU setting provides measurable performance improvement to the performance of Ceph. See Configure MTU Settings in Director (from the Networking Guide) and Configuring Jumbo Frames (from Advanced Overcloud Customization) for related details.

5. Create a networking environment file — namely, **`~/templates/network.yaml`**. This file should contain the following:

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

This file will be used later to invoke the customized Compute NIC template (**`~/templates/nic-configs/compute.yaml`**) during overcloud deployment (in Chapter 6, *Deployment*).

You can use **`~/templates/network.yaml`** to define any networking-related parameters or add any customized networking Heat templates. See Creating a Network Environment File from Advanced Overcloud Customization for more details.

CHAPTER 6. DEPLOYMENT

At this point, you should have already configured all the necessary settings to mitigate resource contention between co-located Compute and Ceph Storage services (as described in Chapter 4, *Configuring Resource Isolation on Hyper-converged Nodes*).

The following sub-sections correspond to the deployment process of either pure or mixed HCI. Both assume that:

1. You are using a separate base environment file (or set of files) for all other Ceph settings. Both sections assume that you are using the same **`/home/stack/templates/storage-environment.yaml`** file from Creating an Overcloud with Ceph Storage Nodes and Sample Environment File: Creating a Ceph Cluster (from Red Hat Ceph Storage for the Overcloud).

2. The same /home/stack/templates/storage-environment.yaml environment file also defines how many nodes you are assigning to each role. For related information on this, see Assigning Nodes and Flavors to Roles (also from Red Hat Ceph Storage for the Overcloud).

Proceed to the section that matches your environment:

- Section 6.1, “Deploying Pure HCI”
- Section 6.2, “Deploying Mixed HCI”

6.1. DEPLOYING PURE HCI

The creation of the overcloud requires additional arguments for the **`openstack overcloud deploy`** command. For example, assuming that you are deploying hyper-converged Compute nodes using a non-isolated **`StorageMgmt`** network:

```
$ openstack overcloud deploy --templates \
  -e /home/stack/templates/environment-rhel-registration.yaml
  -e /home/stack/templates/storage-environment.yaml \
  -e /usr/share/openstack-tripleo-heat-
templates/environments/hyperconverged-ceph.yaml \
  -e /home/stack/templates/compute.yaml \
  -e /home/stack/templates/network.yaml \
  -e /home/stack/templates/ceph-numa-pinning.yaml \
  -e /home/stack/templates/ceph-backfill-recovery.yaml \
  -e /home/stack/templates/hyperconverged-non-isolated.yaml \
  --ntp-server pool.ntp.org
```

Where:

- **--templates** - Creates the Overcloud from the default Heat template collection (namely, **`/usr/share/openstack-tripleo-heat-templates/`**).

- **-e /home/stack/templates/environment-rhel-registration.yaml** - Adds an environment file that registers overcloud nodes, as described in Registering the Overcloud with an Environment File (from Advanced Overcloud Customization). The procedure in Section 4.2, “Configure Ceph NUMA Pinning” includes a script that requires the installation of packages the director does not provide.

- **-e /home/stack/templates/storage-environment.yaml** - Adds a base environment file to the overcloud deployment that defines all other Ceph settings. For a detailed example of such a file, see Creating an Overcloud with Ceph Storage Nodes and Sample Environment File: Creating a Ceph Cluster (from Red Hat Ceph Storage for the Overcloud).

  Note

  In Sample Environment File: Creating a Ceph Cluster (from Red Hat Ceph Storage for the Overcloud), the **`/home/stack/templates/storage-environment.yaml`** file is also used to specify what flavors and how many nodes to assign per role. See Assigning Nodes and Flavors to Roles for details.

- **-e /usr/share/openstack-tripleo-heat-templates/environments/hyperconverged-ceph.yaml** - Adds the environment file that co-locates Ceph services on all Compute nodes, thereby making them hyper-converged. Update the path accordingly if you created a customized version of it elsewhere (for example, to **`~/templates/hyperconverged-ceph.yaml`** to account for non-isolated **`StorageMgmt`** network). See Section 3.1, "Pure HCI" for more information.

- **-e /home/stack/templates/compute.yaml** - Adds the environment file from Section 4.1, "Reserve CPU and Memory Resources for Compute".

- **-e /home/stack/templates/network.yaml** - Adds the environment file from Chapter 5, *Finalize Networking Settings*.

- **-e /home/stack/templates/ceph-numa-pinning.yaml** - Adds the environment file from Section 4.2, "Configure Ceph NUMA Pinning".

- **-e /home/stack/templates/ceph-backfill-recovery.yaml** - Adds the environment file from Section 4.3, "Reduce Ceph Backfill and Recovery Operations".

- **-e /home/stack/templates/hyperconverged-non-isolated.yaml** - Adds the environment file from Section 3.1, "Pure HCI", which disables the **`StorageMgtPort`** service. This is required when you are using a non-isolated **`StorageMgmt`** network. This environment file is invoked last to ensure that it overrides the settings in **`/usr/share/openstack-tripleo-heat-templates/environments/hyperconverged-ceph.yaml`**.

- **--ntp-server pool.ntp.org** - Sets our NTP server.

Use the **`-e`** flag to add environment files as needed for your planned overcloud deployment. For example, to also enable *Single-Root Input/Output Virtualization (SR-IOV)*, add its corresponding environment file:

```
  -e /usr/share/openstack-tripleo-heat-templates/environments/neutron-
sriov.yaml
```

To apply your SR-IOV network preferences, add an environment file defining them:

```
  -e /home/stack/templates/network-environment.yaml
```

Note

Currently, SR-IOV is the only Network Function Virtualization (NFV) implementation supported with HCI. See Configure SR-IOV Support for Virtual Networking (from the Network Functions Virtualization Configuration Guide) for more details.

For a full list of deployment options, run:

```
$ openstack help overcloud deploy
```

For more information, see Creating the Overcloud with the CLI Tools (from Director Installation and Usage).

Tip

You can also use an *answers file* to specify which environment files to include in your deployment. See Including Environment Files in Overcloud Creation (from Director Installation and Usage) for more details.

6.2. DEPLOYING MIXED HCI

The creation of the overcloud requires additional arguments for the **`openstack overcloud deploy command`**. For example:

```
$ openstack overcloud deploy --templates \
  -r /home/stack/templates/roles_data_hci.yaml \
  -e /home/stack/templates/ports.yaml
  -e /home/stack/templates/environment-rhel-registration.yaml
  -e /home/stack/templates/storage-environment.yaml \
  -e /home/stack/templates/compute.yaml \
  -e /home/stack/templates/network.yaml \
  -e /home/stack/templates/ceph-numa-pinning.yaml \
  -e /home/stack/templates/ceph-backfill-recovery.yaml \
  --ntp-server pool.ntp.org
```

Where:

- **--templates** - Creates the Overcloud from the default Heat template collection (namely, **`/usr/share/openstack-tripleo-heat-templates/`**).
- **-r /home/stack/templates/roles_data_hci.yaml** - Specifies the customized roles definition file from Section 3.2.1, “Creating a Custom Role for HCI Nodes”, which adds the custom **`OsdCompute`** role.
- **-e /home/stack/templates/ports.yaml** - Adds the environment file from Section 3.2.2, “Configuring Port Assignments for the Custom Role”, which configures the ports for the **`OsdCompute`** role.
- **-e /home/stack/templates/environment-rhel-registration.yaml** - Adds an environment file that registers overcloud nodes, as described in Registering the Overcloud with an Environment File (from Advanced Overcloud Customization). The procedure in Section 4.2, “Configure Ceph NUMA Pinning” includes a script that requires the installation of packages the director does not

provide.

- **-e /home/stack/templates/storage-environment.yaml** - Adds a base environment file to the overcloud deployment that defines all other Ceph settings. For a detailed example of such a file, see Creating an Overcloud with Ceph Storage Nodes and Sample Environment File: Creating a Ceph Cluster (from Red Hat Ceph Storage for the Overcloud).

  Note

  In Sample Environment File: Creating a Ceph Cluster (from Red Hat Ceph Storage for the Overcloud), the **`/home/stack/templates/storage-environment.yaml`** file is also used to specify what flavors and how many nodes to assign per role. See Assigning Nodes and Flavors to Roles for details.

- **-e /home/stack/templates/compute.yaml** - Adds the environment file from Section 4.1, “Reserve CPU and Memory Resources for Compute”.
- **-e /home/stack/templates/network.yaml** - Adds the environment file from Chapter 5, *Finalize Networking Settings*.
- **-e /home/stack/templates/ceph-numa-pinning.yaml** - Adds the environment file from Section 4.2, “Configure Ceph NUMA Pinning”.
- **-e /home/stack/templates/ceph-backfill-recovery.yaml** - Adds the environment file from Section 4.3, “Reduce Ceph Backfill and Recovery Operations”.
- **--ntp-server pool.ntp.org** - Sets our NTP server.

Use the **`-e`** flag to add environment files as needed for your planned overcloud deployment. For example, to also enable *Single-Root Input/Output Virtualization (SR-IOV)*, add its corresponding environment file:

```
  -e /usr/share/openstack-tripleo-heat-templates/environments/neutron-
sriov.yaml
```

To apply your SR-IOV network preferences, add an environment file defining them:

```
  -e /home/stack/templates/network-environment.yaml
```

Note

Currently, SR-IOV is the only Network Function Virtualization (NFV) implementation supported with HCI. See Configure SR-IOV Support for Virtual Networking (from the Network Functions Virtualization Configuration Guide) for more details.

For a full list of deployment options, run:

```
$ openstack help overcloud deploy
```

For more information, see Creating the Overcloud with the CLI Tools (from Director Installation and Usage).

Tip

You can also use an *answers file* to specify which environment files to include in your deployment. See Including Environment Files in Overcloud Creation (from Director Installation and Usage) for more details.

APPENDIX A. APPENDIX

A.1. SCALING

To scale HCI nodes up or down, the same principles (and for the most part, methods) for scaling Compute or Ceph Storage nodes apply. Be mindful of the following caveats:

A.1.1. Scaling Up

To scale up HCI nodes in a pure HCI environment, use the same methods for scaling up Compute nodes. See Adding Additional Nodes (from Director Installation and Usage) for details.

The same methods apply for scaling up HCI nodes in a mixed HCI environment. When you tag new nodes, remember to use the right flavor (in this case, **`osdcompute`**). See Section 3.2.3, "Creating and Assigning a New Flavor".

A.1.2. Scaling Down

The process for scaling down HCI nodes (in both pure and mixed HCI environments) can be summarized as follows:

1. Disable and rebalance the Ceph OSD services on the HCI node. This step is necessary because the director does not automatically rebalance the Red Hat Ceph Storage cluster when you remove HCI or Ceph Storage nodes.

   See Scaling Down and Replacing Ceph Storage Nodes (from Red Hat Ceph Storage for the Overcloud). Do not follow the steps here for removing the node, as you will need to migrate instances and disable the Compute services on the node first.

2. Migrate the instances from the HCI nodes. See Migrating Instances for instructions.

3. Disable the Compute services on the nodes to prevent them from being used to spawn new instances.

4. Remove the node from the overcloud.

For the third and fourth step (disabling Compute services and removing the node), see Removing Compute Nodes from Director Installation and Usage.

A.2. UPSTREAM TOOLS

Heat templates, environment files, scripts, and other resources relevant to hyper-convergence in OpenStack are available from the following upstream Github repository:

https://github.com/RHsyseng/hci/tree/master/custom-templates

This repository features the scripts in Section A.2.1, "Compute CPU and Memory Calculator" and Section A.2.2, "Custom Script to Configure NUMA Pinning for Ceph OSD Services".

To use these scripts, clone the repository directly to your undercloud:

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

A.2.1. Compute CPU and Memory Calculator

The Section 4.1, “Reserve CPU and Memory Resources for Compute” section explains how to determine suitable values for **`reserved_host_memory_mb`** and **`cpu_allocation_ratio`** on hyper-converged nodes. You can also use the following Python script (also available from the repository in Section A.2, “Upstream Tools”) to perform the computations for you:

```
#!/usr/bin/env python
# Filename:                nova_mem_cpu_calc.py
# Supported Langauge(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 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"
```

A.2.2. Custom Script to Configure NUMA Pinning for Ceph OSD Services

The Configure Ceph NUMA Pinning section describes the creation of a script that pins the Ceph OSD services to an available NUMA socket. This script, **`~/templates/numa-systemd-osd.sh`**, should:

- Take the network interface used for Ceph network traffic

- Use **lstopo** to determine that interface's NUMA socket
- Configure **numactl** to start the OSD service with a NUMA policy that prefers the NUMA node of the Ceph network's interface
- Restart each Ceph OSD daemon sequentially so that the service runs with the new NUMA option

Important

The **numa-systemd-osd.sh** script will also attempt to install NUMA configuration tools. As such, the overcloud nodes must also be registered with Red Hat, as described in Registering the Nodes (from Red Hat Ceph Storage for the Overcloud).

The following script does all of these for Mixed HCI deployments, assuming that the hostname of each hyper-converged node uses either **ceph** or **osd-compute**. Edit the top-level **IF** statement accordingly if you are customizing the hostnames of hyper-converged nodes:

```
#!/usr/bin/env bash
{
if [[ `hostname` = *"ceph"* ]] || [[ `hostname` = *"osd-compute"* ]];
then  # 1

    # 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

    if [[ ! $(lstopo-no-graphics | tr -d [:punct:] | egrep
"NUMANode|$OSD_NUMA_INTERFACE") ]];
    then
 echo "No NUMAnodes found. Exiting."
 exit 1
    fi

    # 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
 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
```

1

The top-level **IF** statement assumes that the hostname of each hyper-converged node contains either **ceph** or **osd-compute**. Edit the **IF** statement accordingly if you are customizing the hostnames of hyper-converged nodes.

Likewise, on a Pure HCI deployment all Compute nodes are hyper-converged. As such, on a Pure HCI deployment change the top-level **IF** statement to:

_

```
if [[ `hostname` = *"compute"* ]]; then
```