Red Hat OpenStack Platform 16.0

Hyperconverged Infrastructure Guide

Understanding and configuring Hyperconverged Infrastructure on the Red Hat OpenStack Platform overcloud

Last Updated: 2020-08-18

Red Hat OpenStack Platform 16.0 Hyperconverged Infrastructure Guide

Understanding and configuring Hyperconverged Infrastructure on the Red Hat OpenStack Platform overcloud

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This document describes the Red Hat OpenStack Platform implementation of hyperconvergence, which colocates Compute and Ceph Storage services on the same host.

Table of Contents

CHAPTER 1. RED HAT OPENSTACK PLATFORM HYPERCONVERGED INFRASTRUCTURE

Red Hat OpenStack Platform (RHOSP) hyperconverged infrastructures (HCI) consist of hyperconverged nodes. Services are colocated on these hyperconverged nodes for optimized resource usage. In a RHOSP HCI, the Compute and storage services are colocated on hyperconverged nodes. You can deploy an overcloud with only hyperconverged nodes, or a mixture of hyperconverged nodes with normal Compute and Ceph Storage nodes.

NOTE

You must use Red Hat Ceph Storage as the storage provider.

TIP

- Use ceph-ansible 3.2 and later to automatically tune Ceph memory settings.
- Use BlueStore as the back end for HCI deployments, to make use of the BlueStore memory handling features.

This document describes how to deploy HCI on an overcloud, and integrate with other features in your overcloud, such as Network Function Virtualization. This document also covers how to ensure optimal performance of both Compute and Ceph Storage services on hyperconverged nodes.

1.1. PREREQUISITES

- You have deployed the undercloud. For instructions about how to deploy the undercloud, see the *Director Installation and Usage* guide.
- Your environment can provision nodes that meet Compute and Ceph Storage requirements. For more information, see Basic Overcloud Deployment in the *Director Installation and Usage* guide.
- You have registered all nodes in your environment. For more information, see Registering Nodes in the *Deploying an Overcloud with Containerized Red Hat Ceph* guide.
- You have tagged all nodes in your environment. For more information, see Manually Tagging the Nodes in the *Deploying an Overcloud with Containerized Red Hat Ceph* guide.
- You have cleaned the disks on nodes that you plan to use for Compute and Ceph OSD services. For more information, see Cleaning Ceph Storage Node Disks in the *Deploying an Overcloud with Containerized Red Hat Ceph* guide.
- You have prepared your overcloud nodes for registration with the Red Hat Content Delivery Network or a Red Hat Satellite server. For more information, see Ansible-based Overcloud Registration in the *Advanced Overcloud Customization* guide.

1.2. REFERENCES

For more detailed information about the Red Hat OpenStack Platform (RHOSP), see the following guides:

- *Director Installation and Usage*: This guide provides guidance on the end-to-end deployment of a RHOSP environment, both undercloud and overcloud.
- *Advanced Overcloud Customization*: This guide describes how to configure advanced RHOSP features through the director, such as how to use custom roles.
- *Deploying an Overcloud with Containerized Red Hat Ceph* : This guide describes how to deploy an overcloud that uses Red Hat Ceph Storage as a storage provider.
- *Networking Guide*: This guide provides details on RHOSP networking tasks.
- Hyper-converged Red Hat OpenStack Platform 10 and Red Hat Ceph Storage 2 : This guide provides a reference architecture that describes how to deploy an environment featuring HCI on very specific hardware.

CHAPTER 2. CONFIGURING AND DEPLOYING A RED HAT OPENSTACK PLATFORM HCI

The following procedure describes the high-level steps involved in configuring and deploying a Red Hat OpenStack Platform (RHOSP) HCI. Each step is expanded on in subsequent sections.

Procedure

1. Prepare the predefined custom overcloud role for hyperconverged nodes, **ComputeHCI**.
2. Configure resource isolation.
3. Map storage management network ports to NICs .
4. Deploy the overcloud.
5. (Optional) Scale the hyperconverged nodes.

CHAPTER 3. PREPARING THE OVERCLOUD ROLE FOR HYPERCONVERGED NODES

To use hyperconverged nodes, you need to define a role for it. Red Hat OpenStack Platform (RHOSP) provides the predefined role **ComputeHCI** for hyperconverged nodes. This role colocates the Compute and Ceph object storage daemon (OSD) services, allowing you to deploy them together on the same hyperconverged node. To use the **ComputeHCI** role, you need to generate a custom **roles_data.yaml** file that includes it, along with all the other roles you are using in your deployment.

The following procedure details how to use and configure this predefined role.

Procedure

1. Generate a custom **roles_data.yaml** file that includes **ComputeHCI**, along with other roles you intend to use for the overcloud:

```
$ openstack overcloud roles generate -o /home/stack/roles_data.yaml Controller
ComputeHCI Compute CephStorage
```

   For more information about custom roles, see Composable Services and Custom Roles and Examining the roles_data file.

2. Create a new heat template named **ports.yaml** in **~/templates**.

3. Configure port assignments for the **ComputeHCI** role by adding the following configuration to the **ports.yaml** file:

```
resource_registry:
  OS::TripleO::ComputeHCI::Ports::ExternalPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/<ext_port_file>.yaml
  OS::TripleO::ComputeHCI::Ports::InternalApiPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/internal_api.yaml
  OS::TripleO::ComputeHCI::Ports::StoragePort: /usr/share/openstack-tripleo-heat-
templates/network/ports/storage.yaml
  OS::TripleO::ComputeHCI::Ports::TenantPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/tenant.yaml
  OS::TripleO::ComputeHCI::Ports::StorageMgmtPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/<storage_mgmt_file>.yaml
```

   - Replace **<ext_port_file>** with the name of the external port file. Set to "external" if you are using DVR, otherwise set to "noop". For details on DVR, see Configure Distributed Virtual Routing (DVR).

   - Replace **<storage_mgmt_file>** with the name of the storage management file. Set to one of the following values:

Value	Description
storage_mgmt	Use if you do not want to select from a pool of IPs, and your environment does not use IPv6 addresses.
storage_mgmt_from_pool	Use if you want the ComputeHCI role to select from a pool of IPs.

Value	Description
storage_mgmt_v6	Use if your environment uses IPv6 addresses.
storage_mgmt_from_pool_v6	Use if you want the ComputeHCI role to select from a pool of IPv6 addresses

For more information, see Basic network isolation.

4. Create a flavor for the ComputeHCI role:

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

5. Configure the flavor properties:

```
$ openstack flavor set --property "cpu_arch"="x86_64" \
--property "capabilities:boot_option"="local" \
--property "resources:CUSTOM_BAREMETAL"="1" \
--property "resources:DISK_GB"="0" \
--property "resources:MEMORY_MB"="0" \
--property "resources:VCPU"="0" computeHCI
```

6. Map the flavor to a new profile:

```
$ openstack flavor set --property "capabilities:profile"="computeHCI" computeHCI
```

7. Retrieve a list of your nodes to identify their UUIDs:

```
$ openstack baremetal node list
```

8. Tag nodes into the new profile:

```
$ openstack baremetal node set --property
capabilities='profile:computeHCI,boot_option:local' <UUID>
```

For more information, see Manually Tagging the Nodes and Assigning Nodes and Flavors to Roles.

9. Add the following configuration to the **node-info.yaml** file to associate the **computeHCI** flavor with the ComputeHCI role:

```
parameter_defaults:
  OvercloudComputeHCIFlavor: computeHCI
  ComputeHCICount: 3
```

CHAPTER 4. CONFIGURING RESOURCE ISOLATION ON HYPERCONVERGED NODES

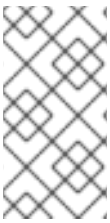
Colocating Ceph OSD and Compute services on hyperconverged nodes risks resource contention between Ceph and Compute services, as neither are aware of each other's presence on the same host. Resource contention can result in degradation of service, which offsets the benefits of hyperconvergence.

The following sections detail how resource isolation is configured for both Ceph and Compute services to prevent contention.

4.1. RESERVING CPU AND MEMORY RESOURCES FOR COMPUTE

The director provides a default plan environment file for configuring resource constraints on hyperconverged nodes during deployment. This plan environment file instructs the OpenStack Workflow to complete the following processes:

1. Retrieve the hardware introspection data collected during Inspecting the Hardware of Nodes in the *Director Installation and Usage* guide.
2. Calculate optimal CPU and memory allocation workload for Compute on hyperconverged nodes based on that data.
3. Autogenerate the parameters required to configure those constraints and reserve CPU and memory resources for Compute. These parameters are defined under the **hci_profile_config** section of the **plan-environment-derived-params.yaml** file.



NOTE

The **average_guest_memory_size_in_mb** and **average_guest_cpu_utilization_percentage** parameters in each workload profile are used to calculate values for the **reserved_host_memory** and **cpu_allocation_ratio** settings of Compute.

You can override the autogenerated Compute settings by adding the following parameters to your Compute environment file:

Autogenerated **nova.conf** parameter	Compute environment file override	Description
reserved_host_memory	parameter_defaults: ComputeHCIParameters: NovaReservedHostMemory: 181000	Sets how much RAM should be reserved for the Ceph OSD services and per-guest instance overhead on hyperconverged nodes.

Autogenerated **nova.conf** parameter	Compute environment file override	Description
cpu_allocation_ratio	parameter_defaults: ComputeHCIExtraConfig: nova::cpu_allocation_ratio: 8.2	Sets the ratio that the Compute scheduler should use when choosing which Compute node to deploy an instance on.

These overrides are applied to all nodes that use the ComputeHCI role, namely, all hyperconverged nodes. For more information about manually determining optimal values for **NovaReservedHostMemory** and **nova::cpu_allocation_ratio**, see Compute CPU and Memory Calculator.

TIP

You can use the following script to calculate suitable baseline **NovaReservedHostMemory** and **cpu_allocation_ratio** values for your hyperconverged nodes.

nova_mem_cpu_calc.py

4.2. RESERVING CPU AND MEMORY RESOURCES FOR CEPH

The following procedure details how to reserve CPU and memory resources for Ceph.

Procedure

1. Set the parameter **is_hci** to "true" in **/home/stack/templates/storage-container-config.yaml**:

```
parameter_defaults:
  CephAnsibleExtraConfig:
    is_hci: true
```

This allows **ceph-ansible** to reserve memory resources for Ceph, and reduce memory growth by Ceph OSDs, by automatically adjusting the **osd_memory_target** parameter setting for a HCI deployment.

WARNING

Red Hat does not recommend directly overriding the **ceph_osd_docker_memory_limit** parameter.

NOTE

As of ceph-ansible 3.2, the **ceph_osd_docker_memory_limit** is set automatically to the maximum memory of the host, as discovered by Ansible, regardless of whether the FileStore or BlueStore back end is used.

2. (Optional) By default, **ceph-ansible** reserves one vCPU for each Ceph OSD. If more than one CPU per Ceph OSD is required, add the following configuration to **/home/stack/templates/storage-container-config.yaml**, setting **ceph_osd_docker_cpu_limit** to the desired CPU limit:

```
parameter_defaults:
  CephAnsibleExtraConfig:
    ceph_osd_docker_cpu_limit: 2
```

For more information on how to tune CPU resources based on your hardware and workload, see Red Hat Ceph Storage Hardware Selection Guide .

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. If a Ceph cluster is under load, its performance drops 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. 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.

The parameters detailed in the following table are used to configure the priority of backfill and recovery operations.

Parameter	Description	Default value
osd_recovery_op_priority	Sets the priority for recovery operations, relative to the OSD client OP priority.	3
osd_recovery_max_active	Sets the number of active recovery requests per OSD, at one time. More requests accelerate recovery, but the requests place an increased load on the cluster. Set this to 1 if you want to reduce latency.	3
osd_max_backfills	Sets the maximum number of backfills allowed to or from a single OSD.	1

To change this default configuration, add an environment file named **ceph-backfill-recovery.yaml** to `~/**templates** that contains the following:

```
parameter_defaults:
  CephConfigOverrides:
    osd_recovery_op_priority: ${priority_value}
```

```
osd_recovery_max_active: ${no_active_recovery_requests}
osd_max_backfills: ${max_no_backfills}
```

CHAPTER 5. MAPPING STORAGE MANAGEMENT NETWORK PORTS TO NICS

The following procedure details how to map the storage management network ports to the physical NICs on your hyperconverged nodes.

Procedure

1. Copy the **compute.yaml** heat template file for your environment from the **/usr/share/openstack-tripleo-heat-templates/network/config** directory. The following options are available:
   - single-nic-vlans
   - single-nic-linux-bridge-vlans
   - multiple-nics
   - bond-with-vlans
     See the **README.md** in each template's directory for details on the NIC configuration.
2. Create a new directory within `~/**templates** called **nic-configs**.
3. Paste your copy of the **compute.yaml** template into **~/templates/nic-configs/** and rename it to **compute-hci.yaml**.
4. Check the **parameters:** section of **~/templates/nic-configs/compute-hci.yaml** for the following definition:

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

   Add the **StorageMgmtNetworkVlanID** definition if it is not already in the **compute-hci.yaml** file.
5. Map **StorageMgmtNetworkVlanID** to a specific NIC on each HCI node. For example, if you chose to trunk VLANs to a single NIC, then add the following entry to the **network_config:** section of **~/templates/nic-configs/compute-hci.yaml**:

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

NOTE

Set MTU to 9000 (jumbo frames) when mapping a NIC to **StorageMgmtNetworkVlanID** to improve the performance of Red Hat Ceph Storage. For more information, see Configure MTU Settings in Director and Configuring jumbo frames.

6. Create a networking environment file called **~/templates/network.yaml**.

7. Add the following configuration to the **network.yaml** file:

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

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

You can use **~/templates/network.yaml** to define any networking parameters, or add any customized networking heat templates. For more information, see Custom network environment file in the *Advanced Overcloud Customization* guide.

CHAPTER 6. DEPLOYING THE OVERCLOUD

Prerequisites

- You are using a separate base environment file, or set of files, for all other Ceph settings, for instance, **/home/stack/templates/storage-config.yaml**. For more information, see Customizing the Storage Service and Sample Environment File: Creating a Ceph Cluster .
- You have defined the number of nodes you are assigning to each role in the base environment file. For more information, see Assigning Nodes and Flavors to Roles .
- During undercloud installation, you set **generate_service_certificate=false** in the **undercloud.conf** file. Otherwise, you must inject a trust anchor when you deploy the overcloud, as described in Enabling SSL/TLS on Overcloud Public Endpoints.

IMPORTANT

Do not enable Instance HA when deploying a RHOSP HCI environment. Contact your Red Hat representative if you want to use Instance HA with hyperconverged RHOSP deployments with Ceph.

Procedure

Run the following command to deploy your HCI overcloud:

```
$ openstack overcloud deploy --templates \
  -p /usr/share/openstack-tripleo-heat-templates/plan-samples/plan-environment-derived-
params.yaml \
  -r /home/stack/templates/roles_data.yaml \
  -e /home/stack/templates/ports.yaml \
  -e /home/stack/templates/environment-rhel-registration.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  -e /home/stack/templates/storage-config.yaml \
  -e /home/stack/templates/storage-container-config.yaml \
  -e /home/stack/templates/network.yaml \
  [-e /home/stack/templates/ceph-backfill-recovery.yaml \ ]
  [-e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-sriov.yaml \]
  [-e /home/stack/templates/network-environment.yaml \ ]
  [-e <additional environment files for your planned overcloud deployment> \ ]
  --ntp-server pool.ntp.org
```

Where:

Argument	Description
--templates	Creates the overcloud from the default heat template collection: **/usr/share/openstack-tripleo-heat-templates/**).

Argument	Description
-p /usr/share/openstack-tripleo-heat-templates/plan-samples/plan-environment-derived-params.yaml	Specifies that the derived parameters workflow should be run during the deployment to calculate how much memory and CPU should be reserved for a hyperconverged deployment.
-r /home/stack/templates/roles_data.yaml	Specifies the customized roles definition file created in the Preparing the overcloud role for hyperconverged nodes procedure, which includes the ComputeHCI role.
-e /home/stack/templates/ports.yaml	Adds the environment file created in the Preparing the overcloud role for hyperconverged nodes procedure, which configures the ports for the ComputeHCI role.
-e /home/stack/templates/environment-rhel-registration.yaml	Adds an environment file that registers overcloud nodes, as described in Registering the overcloud with the rhsm composable service in the *Advanced Overcloud Customization* guide.
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml	Adds the base environment file that deploys a containerized Red Hat Ceph cluster, with all default settings. For more information, see the *Deploying an Overcloud with Containerized Red Hat Ceph* guide.
-e /home/stack/templates/storage-config.yaml	Adds a custom environment file that defines all other Ceph settings. For a detailed example of this, see Sample Environment File: Creating a Ceph Cluster in the *Deploying an Overcloud with Containerized Red Hat Ceph* guide. This sample environment file also specifies the flavors to use, and how many nodes to assign per role. For more information on this, see Assigning Nodes and Flavors to Roles in the *Deploying an Overcloud with Containerized Red Hat Ceph* guide.
-e /home/stack/templates/storage-container-config.yaml	Reserves CPU and memory for each Ceph OSD storage container, as described in Reserving CPU and memory resources for Ceph.
-e /home/stack/templates/network.yaml	Adds the environment file created in the Mapping storage management network ports to NICs procedure.

Argument	Description
-e /home/stack/templates/ceph-backfill-recovery.yaml	(Optional) Adds the environment file from Reduce Ceph Backfill and Recovery Operations.
-e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-sriov.yaml	(Optional) Adds the environment file for Single-Root Input/Output Virtualization (SR-IOV).
-e /home/stack/templates/network-environment.yaml	(Optional) Adds the environment file that applies your SR-IOV network preferences.
-e <environment file>	(Optional) Adds any additional environment files for your planned overcloud deployment.
--ntp-server pool.ntp.org	Sets our NTP server.

NOTE

Currently, SR-IOV is the only Network Function Virtualization (NFV) implementation supported with HCI.

For a full list of deployment options, run the following command:

```
$ openstack help overcloud deploy
```

For more details on deployment options, see Creating the Overcloud with the CLI Tools in the *Director Installation and Usage* guide.

TIP

You can also use an **answers** file to specify which environment files to include in your deployment. For more information, see Including Environment Files in Overcloud Creation in the *Director Installation and Usage* guide.

CHAPTER 7. SCALING HYPERCONVERGED NODES

To scale HCI nodes up or down, the same principles and methods for scaling Compute or Ceph Storage nodes apply.

7.1. SCALING UP

To scale up hyperconverged nodes in HCI environments follow the same procedure for scaling up non-hyperconverged nodes, as detailed in Adding nodes to the overcloud .

NOTE

When you tag new nodes, remember to use the right flavor.

For information about how to scale up HCI nodes by adding OSDs to a Ceph Storage cluster, see Adding an OSD to a Ceph Storage node in *Deploying an Overcloud with Containerized Red Hat Ceph* .

7.2. SCALING DOWN

Procedure

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.

1. Migrate the instances from the HCI nodes. See Migrating Virtual Machines Between Compute Nodes in the *Instances and Images* guide.

2. Disable the Compute services on the nodes to prevent new instances from being launched on the nodes.

3. Remove the node from the overcloud.

For steps 3 and 4, see Removing Compute nodes.

APPENDIX A. APPENDIX

A.1. COMPUTE CPU AND MEMORY CALCULATOR

The following subsections describe how the OpenStack Workflow calculates the optimal settings for CPU and memory.

A.1.1. NovaReservedHostMemory

The **NovaReservedHostMemory** parameter sets 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 3 GB of memory. Given a node with 256 GB 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 2 GB 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 226 GB divided by 2.5 GB. 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) = 75 GB. 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))
```

A.1.2. cpu_allocation_ratio

The Compute scheduler uses **cpu_allocation_ratio** 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
```