Red Hat OpenStack Platform 16.1

Hyperconverged Infrastructure Guide

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

Last Updated: 2022-07-12

Red Hat OpenStack Platform 16.1 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

MAKING OPEN SOURCE MORE INCLUSIVE 3

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION 4

CHAPTER 1. CONFIGURING AND DEPLOYING A RED HAT OPENSTACK PLATFORM HYPERCONVERGED INFRASTRUCTURE 5
1.1. PREREQUISITES 5
1.2. PREPARING THE OVERCLOUD ROLE FOR HYPERCONVERGED NODES 6
1.2.1. Defining the root disk for multi-disk clusters 8
1.2.1.1. Properties that identify the root disk 9
1.3. CONFIGURING RESOURCE ISOLATION ON HYPERCONVERGED NODES 10
1.3.1. Process for autogenerating CPU and memory resources to reserve for the Compute service 11
1.3.2. Red Hat Ceph Storage backfill and recovery operations 13
1.4. PRE-DEPLOYMENT VALIDATIONS FOR CEPH STORAGE 13
1.4.1. Verifying the ceph-ansible package version 13
1.4.2. Verifying packages for pre-provisioned nodes 13
1.5. DEPLOYING THE HCI OVERCLOUD 14
1.5.1. Limiting the nodes on which ceph-ansible runs 15
1.6. OPENSTACK WORKFLOW COMPUTE CPU AND MEMORY CALCULATOR 16
1.7. ADDITIONAL RESOURCES 17

CHAPTER 2. SCALING HYPERCONVERGED NODES 18
2.1. SCALING UP HYPERCONVERGED NODES IN HCI ENVIRONMENTS 18
2.2. SCALING DOWN HYPERCONVERGED NODES IN HCI ENVIRONMENTS 18

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Tell us how we can make it better.

Using the Direct Documentation Feedback (DDF) function

Use the **Add Feedback** DDF function for direct comments on specific sentences, paragraphs, or code blocks.

1. View the documentation in the *Multi-page HTML* format.
2. Ensure that you see the **Feedback** button in the upper right corner of the document.
3. Highlight the part of text that you want to comment on.
4. Click **Add Feedback**.
5. Complete the **Add Feedback** field with your comments.
6. Optional: Add your email address so that the documentation team can contact you for clarification on your issue.
7. Click **Submit**.

CHAPTER 1. CONFIGURING AND DEPLOYING A 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.

To create and deploy HCI on an overcloud, integrate with other features in your overcloud such as Network Function Virtualization, and ensure optimal performance of both Compute and Red Hat Ceph Storage services on hyperconverged nodes, you must complete the following:

1. Prepare the predefined custom overcloud role for hyperconverged nodes, **ComputeHCI**.
2. Configure resource isolation.
3. Verify the available Red Hat Ceph Storage packages.
4. Deploy the HCI overcloud.

1.1. PREREQUISITES

- You have deployed the undercloud. For instructions on how to deploy the undercloud, see Director Installation and Usage.
- Your environment can provision nodes that meet RHOSP Compute and Red Hat Ceph Storage requirements. For more information, see Basic Overcloud Deployment.
- You have registered all nodes in your environment. For more information, see Registering Nodes.
- You have tagged all nodes in your environment. For more information, see Manually Tagging the Nodes.
- 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 .
- 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.

1.2. PREPARING THE OVERCLOUD ROLE FOR HYPERCONVERGED NODES

To designate nodes as hyperconverged, you need to define a hyperconverged role. 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.

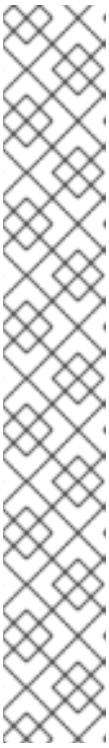
Procedure

1. Log in to the undercloud as the **stack** user.
2. Source the **stackrc** file:

```
[stack@director ~]$ source ~/stackrc
```

3. Generate a new custom roles data file that includes the **ComputeHCI** role, along with other roles you intend to use for the overcloud. The following example generates the roles data file **roles_data_hci.yaml** that includes the roles **Controller**, **ComputeHCI**, **Compute**, and **CephStorage**:

```
(undercloud)$ openstack overcloud roles \
 generate -o /home/stack/templates/roles_data_hci.yaml \
  Controller ComputeHCI Compute CephStorage
```



NOTE

The networks listed for the **ComputeHCI** role in the generated custom roles data file include the networks required for both Compute and Storage services, for example:

```
- name: ComputeHCI
  description: |
    Compute node role hosting Ceph OSD
  tags:
    - compute
  networks:
    InternalApi:
      subnet: internal_api_subnet
    Tenant:
      subnet: tenant_subnet
    Storage:
      subnet: storage_subnet
    StorageMgmt:
      subnet: storage_mgmt_subnet
```

4. Create a local copy of the **network_data.yaml** file to add a composable network to your overcloud. The **network_data.yaml** file interacts with the default network environment files, **/usr/share/openstack-tripleo-heat-templates/environments/***, to associate the networks you defined for your **ComputeHCI** role with the hyperconverged nodes. For more information, see Adding a composable network in the *Advanced Overcloud Customization* guide.
5. To improve the performance of Red Hat Ceph Storage, update the MTU setting for both the

Storage and **StorageMgmt** networks to **9000**, for jumbo frames, in your local copy of **network_data.yaml**. For more information, see Configuring MTU Settings in Director and Configuring jumbo frames.

6. Create the **computeHCI** overcloud flavor for hyperconverged nodes:

```
(undercloud)$ openstack flavor create --id auto \
 --ram <ram_size_mb> --disk <disk_size_gb> \
 --vcpus <no_vcpus> computeHCI
```

- Replace **<ram_size_mb>** with the RAM of the bare metal node, in MB.
- Replace **<disk_size_gb>** with the size of the disk on the bare metal node, in GB.
- Replace **<no_vcpus>** with the number of CPUs on the bare metal node.

NOTE

These properties are not used for scheduling instances. However, the Compute scheduler does use the disk size to determine the root partition size.

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

```
(undercloud)$ openstack baremetal node list
```

8. Tag each bare metal node that you want to designate as hyperconverged with a custom HCI resource class:

```
(undercloud)$ openstack baremetal node set \
 --resource-class baremetal.HCI <node>
```

Replace **<node>** with the ID of the bare metal node.

9. Associate the **computeHCI** flavor with the custom HCI resource class:

```
(undercloud)$ openstack flavor set \
--property resources:CUSTOM_BAREMETAL_HCI=1 \
computeHCI
```

To determine the name of a custom resource class that corresponds to a resource class of a Bare Metal service node, convert the resource class to uppercase, replace all punctuation with an underscore, and prefix with **CUSTOM_**.

NOTE

A flavor can request only one instance of a bare metal resource class.

10. Set the following flavor properties to prevent the Compute scheduler from using the bare metal flavor properties to schedule instances:

```
(undercloud)$ openstack flavor set \
 --property resources:VCPU=0 \
 --property resources:MEMORY_MB=0 \
```

```
--property resources:DISK_GB=0 computeHCI
```

11. Add the following parameters to the **node-info.yaml** file to specify the number of hyperconverged and Controller nodes, and the flavor to use for the hyperconverged and controller designated nodes:

```
parameter_defaults:
  OvercloudComputeHCIFlavor: computeHCI
  ComputeHCICount: 3
  OvercloudControlFlavor: baremetal
  ControllerCount: 3
```

Additional resources

- Composable Services and Custom Roles
- Examining the roles_data file
- Assigning Nodes and Flavors to Roles

1.2.1. Defining the root disk for multi-disk clusters

Most Ceph Storage nodes use multiple disks. When nodes use multiple disks, director must identify the root disk. By default, director writes the overcloud image to the root disk during the provisioning process.

Use this procedure to identify the root device by serial number. For more information about other properties you can use to identify the root disk, see Section 1.2.1.1, “Properties that identify the root disk”.

Procedure

1. Verify the disk information from the hardware introspection of each node. The following command to displays the disk information of a node:

```
(undercloud)$ openstack baremetal introspection data save 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 | jq ".inventory.disks"
```

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

```
[
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sda",
    "wwn_vendor_extension": "0x1ea4dcc412a9632b",
    "wwn_with_extension": "0x61866da04f3807001ea4dcc412a9632b",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380700",
    "serial": "61866da04f3807001ea4dcc412a9632b"
  }
  {
    "size": 299439751168,
```

```
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sdb",
    "wwn_vendor_extension": "0x1ea4e13c12e36ad6",
    "wwn_with_extension": "0x61866da04f380d001ea4e13c12e36ad6",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380d00",
    "serial": "61866da04f380d001ea4e13c12e36ad6"
  }
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sdc",
    "wwn_vendor_extension": "0x1ea4e31e121cfb45",
    "wwn_with_extension": "0x61866da04f37fc001ea4e31e121cfb45",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f37fc00",
    "serial": "61866da04f37fc001ea4e31e121cfb45"
  }
]
```


2. On the undercloud, set the root disk for a node. Include the most appropriate hardware attribute value to define the root disk.

```
(undercloud)$ openstack baremetal node set --property root_device='{"serial":"
<serial_number>"}' <node-uuid>
```

For example, to set the root device to disk 2, which has the serial number **61866da04f380d001ea4e13c12e36ad6**, enter the following command:

```
(undercloud)$ openstack baremetal node set --property root_device='{"serial":
"61866da04f380d001ea4e13c12e36ad6"}' 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0
```

NOTE

Configure the BIOS of each node to boot from the root disk that you choose. Configure the boot order to boot from the network first, then from the root disk.

Director identifies the specific disk to use as the root disk. When you run the **openstack overcloud deploy** command, director provisions and writes the overcloud image to the root disk.

1.2.1.1. Properties that identify the root disk

There are several properties that you can define to help director identify the root disk:

- **model** (String): Device identifier.
- **vendor** (String): Device vendor.
- **serial** (String): Disk serial number.
- **hctl** (String): Host:Channel:Target:Lun for SCSI.

- **size** (Integer): Size of the device in GB.
- **wwn** (String): Unique storage identifier.
- **wwn_with_extension** (String): Unique storage identifier with the vendor extension appended.
- **wwn_vendor_extension** (String): Unique vendor storage identifier.
- **rotational** (Boolean): True for a rotational device (HDD), otherwise false (SSD).
- **name** (String): The name of the device, for example: /dev/sdb1.

IMPORTANT

Use the **name** property only for devices with persistent names. Do not use **name** to set the root disk for any other devices because this value can change when the node boots.

1.3. CONFIGURING RESOURCE ISOLATION ON HYPERCONVERGED NODES

Colocating Ceph OSD and Compute services on hyperconverged nodes risks resource contention between Red Hat Ceph Storage 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.

You must configure resource isolation for both Ceph and Compute services to prevent contention.

Procedure

1. Optional: Override the autogenerated Compute settings by adding the following parameters to a Compute environment file:

```
parameter_defaults:
  ComputeHCIParameters:
    NovaReservedHostMemory: <ram>
    NovaCPUAllocationRatio: <ratio>
```

   - Replace **<ram>** with the amount of RAM to reserve for the Ceph OSD services and instance overhead on hyperconverged nodes, in MB.
   - Replace **<ratio>** with the ratio that the Compute scheduler should use when choosing which Compute node to deploy an instance on.
     For more information on the autogenerated Compute settings, see Process for autogenerating CPU and memory resources to reserve for the Compute service.

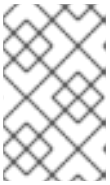
2. To reserve memory resources for Red Hat Ceph Storage, set the parameter **is_hci** to **true** in **/home/stack/templates/storage-container-config.yaml**:

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

   This allows **ceph-ansible** to reserve memory resources for Red Hat Ceph Storage, 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.

3. Optional: By default, **ceph-ansible** reserves one vCPU for each Ceph OSD. If you require more than one CPU per Ceph OSD, add the following configuration to **/home/stack/templates/storage-container-config.yaml**:

```
parameter_defaults:
  CephAnsibleExtraConfig:
    ceph_osd_docker_cpu_limit: <cpu_limit>
```

Replace **<cpu_limit>** with the number of CPUs to reserve for each Ceph OSD.

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

4. Optional: Reduce the priority of Red Hat Ceph Storage backfill and recovery operations when a Ceph OSD is removed by adding the following parameters to a Ceph environment file:

```
parameter_defaults:
  CephConfigOverrides:
    osd_recovery_op_priority: <priority_value>
    osd_recovery_max_active: <no_active_recovery_requests>
    osd_max_backfills: <max_no_backfills>
```

- Replace **<priority_value>** with the priority for recovery operations, relative to the OSD client OP priority.
- Replace **<no_active_recovery_requests>** with the number of active recovery requests per OSD, at one time.
- Replace **<max_no_backfills>** with the maximum number of backfills allowed to or from a single OSD.
  For more information on default Red Hat Ceph Storage backfill and recovery options, see Red Hat Ceph Storage backfill and recovery operations .

1.3.1. Process for autogenerating CPU and memory resources to reserve for the Compute service

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 inspection of the hardware nodes.
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.
cpu_allocation_ratio	parameter_defaults: ComputeHCIParameters: NovaCPUAllocationRatio: 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 **NovaCPUAllocationRatio**, see OpenStack Workflow Compute CPU and memory calculator.

TIP

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

nova_mem_cpu_calc.py

Additional resources

- Creating an inventory of the bare-metal node hardware

1.3.2. Red Hat Ceph Storage backfill and recovery operations

When a Ceph OSD is removed, Red Hat Ceph Storage uses backfill and recovery operations to rebalance the cluster. Red Hat Ceph Storage does this to keep multiple copies of data according to the placement group policy. These operations use system resources. If a Red Hat Ceph Storage 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

1.4. PRE-DEPLOYMENT VALIDATIONS FOR CEPH STORAGE

To help avoid overcloud deployment failures, verify that the required packages exist on your servers.

1.4.1. Verifying the ceph-ansible package version

The undercloud contains Ansible-based validations that you can run to identify potential problems before you deploy the overcloud. These validations can help you avoid overcloud deployment failures by identifying common problems before they happen.

Procedure

Verify that the correction version of the **ceph-ansible** package is installed:

```
$ ansible-playbook -i /usr/bin/tripleo-ansible-inventory /usr/share/ansible/validation-playbooks/ceph-ansible-installed.yaml
```

1.4.2. Verifying packages for pre-provisioned nodes

Ceph can service only overcloud nodes that have a certain set of packages. When you use pre-provisioned nodes, you can verify the presence of these packages.

For more information about pre-provisioned nodes, see Configuring a basic overcloud with pre-provisioned nodes.

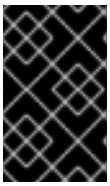
Procedure

Verify that the servers contained the required packages:

```
ansible-playbook -i /usr/bin/tripleo-ansible-inventory /usr/share/ansible/validation-playbooks/ceph-dependencies-installed.yaml
```

1.5. DEPLOYING THE HCI OVERCLOUD

You must deploy the overcloud after you complete the HCI configuration.



IMPORTANT

Do not enable Instance HA when you deploy a Red Hat OpenStack Platform (RHOSP) HCI environment. Contact your Red Hat representative if you want to use Instance HA with hyperconverged RHOSP deployments with Red Hat Ceph Storage.

Prerequisites

- You are using a separate base environment file, or set of files, for all other Red Hat Ceph Storage settings, for example, **/home/stack/templates/storage-config.yaml**. For more information, see Customizing the Storage service and Appendix A. Sample environment file: creating a Ceph Storage 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.

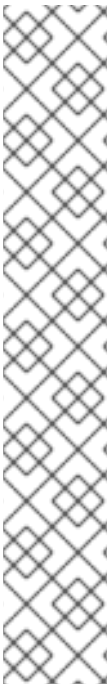
Procedure

- Add your new role and environment files to the stack with your other environment files and deploy your HCI overcloud:

```
(undercloud)$ openstack overcloud deploy --templates \
  -e [your environment files] \
  -r /home/stack/templates/roles_data_hci.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/net-single-nic-with-vlans.yaml
  -e /home/stack/templates/storage-config.yaml \
  -e /home/stack/templates/storage-container-config.yaml \
  -n /home/stack/templates/network_data.yaml \
  [-e /home/stack/templates/ceph-backfill-recovery.yaml \ ]
  --ntp-server pool.ntp.org
```

  Including **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-**

ansible.yaml in the deployment command adds the base environment file that deploys a containerized Red Hat Ceph cluster, with all default settings. For more information, see Deploying an Overcloud with Containerized Red Hat Ceph .



NOTE

Include the following options in the deployment command if your deployment uses single root input/output virtualization (SR-IOV).

If you use the ML2/OVS mechanism driver in your deployment, specify the following options:

```
-e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-sriov.yaml
-e /home/stack/templates/network-environment.yaml
```

If you use the ML2/OVN mechanism driver in your deployment, specify the following options:

```
-e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovn-sriov.yaml
-e /home/stack/templates/network-environment.yaml
```

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 an overcloud deployment in the *Director Installation and Usage* guide.

1.5.1. Limiting the nodes on which ceph-ansible runs

You can reduce deployment update time by limiting the nodes where **ceph-ansible** runs. When Red Hat OpenStack Platform (RHOSP) uses **config-download** to configure Ceph, you can use the **--limit** option to specify a list of nodes, instead of running **config-download** and **ceph-ansible** across your entire deployment. This feature is useful, for example, as part of scaling up your overcloud, or replacing a failed disk. In these scenarios, the deployment can run only on the new nodes that you add to the environment.

Example scenario that uses --limit in a failed disk replacement

In the following example procedure, the Ceph storage node **oc0-cephstorage-0** has a disk failure so it receives a new factory clean disk. Ansible needs to run on the **oc0-cephstorage-0** node so that the new disk can be used as an OSD but it does not need to run on all of the other Ceph storage nodes. Replace the example environment files and node names with those appropriate to your environment.

Procedure

1. Log in to the undercloud node as the **stack** user and source the **stackrc** credentials file:

```
# source stackrc
```

2. Complete one of the following steps so that the new disk is used to start the missing OSD.

   - Run a stack update and include the **--limit** option to specify the nodes where you want **ceph-ansible** to run:

```
$ openstack overcloud deploy --templates \
  -r /home/stack/roles_data.yaml \
  -n /usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e ~/my-ceph-settings.yaml \
  -e <other-environment_files> \
  --limit oc0-controller-0:oc0-controller-2:oc0-controller-1:oc0-cephstorage-0:undercloud
```

In this example, the Controllers are included because the Ceph mons need Ansible to change their OSD definitions.

- If **config-download** has generated an **ansible-playbook-command.sh** script, you can also run the script with the **--limit** option to pass the specified nodes to **ceph-ansible**:

```
./ansible-playbook-command.sh --limit oc0-controller-0:oc0-controller-2:oc0-controller-
1:oc0-cephstorage-0:undercloud
```

Warning

You must always include the undercloud in the limit list otherwise **ceph-ansible** cannot be executed when you use **--limit**. This is necessary because the **ceph-ansible** execution occurs through the **external_deploy_steps_tasks** playbook, which runs only on the undercloud.

1.6. OPENSTACK WORKFLOW COMPUTE CPU AND MEMORY CALCULATOR

The OpenStack Workflow calculates the optimal settings for CPU and memory and uses the results to populate the parameters **NovaReservedHostMemory** and **NovaCPUAllocationRatio**.

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))
```

NovaCPUAllocationRatio

The Compute scheduler uses **NovaCPUAllocationRatio** 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 **NovaCPUAllocationRatio** 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 **NovaCPUAllocationRatio** 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
```

1.7. ADDITIONAL RESOURCES

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.

CHAPTER 2. SCALING HYPERCONVERGED NODES

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

2.1. SCALING UP HYPERCONVERGED NODES IN HCI ENVIRONMENTS

To scale up hyperconverged nodes in HCI environments follow the same procedure for scaling up non-hyperconverged nodes. For more information, see 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 Red Hat Ceph Storage cluster, see Adding an OSD to a Ceph Storage node in *Deploying an Overcloud with Containerized Red Hat Ceph*.

2.2. SCALING DOWN HYPERCONVERGED NODES IN HCI ENVIRONMENTS

To scale down hyperconverged nodes in HCI environments you must rebalance the Ceph OSD services on the HCI node, migrate instances from the HCI nodes, and remove the Compute nodes from the overcloud.

Procedure

1. Disable and rebalance the Ceph OSD services on the HCI node. This step is necessary because director does not automatically rebalance the Red Hat Ceph Storage cluster when you remove HCI or Red Hat Ceph Storage nodes.
2. Migrate the instances from the HCI nodes. For more information, see Migrating virtual machines between Compute nodes in the *Configuring the Compute Service for Instance Creation* guide.
3. Remove the Compute nodes from the overcloud. For more information, see Removing Compute nodes.