



# **Red Hat OpenStack Platform 12**

## **Hyper-Converged Infrastructure Guide**

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



# Red Hat OpenStack Platform 12 Hyper-Converged Infrastructure Guide

---

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

OpenStack Team  
rhos-docs@redhat.com

## Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## 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.

---

## Table of Contents

<b>CHAPTER 1. INTRODUCTION</b>	<b>3</b>
1.1. ASSUMPTIONS	3
1.2. REFERENCES	3
<b>CHAPTER 2. PROCESS DESCRIPTION</b>	<b>5</b>
<b>CHAPTER 3. PREPARE OVERCLOUD ROLE FOR HYPER-CONVERGED NODES</b>	<b>6</b>
3.1. CONFIGURING PORT ASSIGNMENTS FOR THE COMPUTEHCI ROLE	6
3.2. CREATING AND ASSIGNING A NEW FLAVOR	7
<b>CHAPTER 4. CONFIGURING RESOURCE ISOLATION ON HYPER-CONVERGED NODES</b>	<b>9</b>
4.1. RESERVE CPU AND MEMORY RESOURCES FOR COMPUTE	9
4.1.1. Override Calculated Settings for Memory or CPU Allocation	9
4.2. REDUCE CEPH BACKFILL AND RECOVERY OPERATIONS	10
<b>CHAPTER 5. FINALIZE NETWORKING SETTINGS</b>	<b>12</b>
<b>CHAPTER 6. DEPLOYMENT</b>	<b>14</b>
<b>APPENDIX A. APPENDIX</b>	<b>17</b>
A.1. SCALING	17
A.1.1. Scaling Up	17
A.1.2. Scaling Down	17
A.2. COMPUTE CPU AND MEMORY CALCULATOR	17
A.2.1. NovaReservedHostMemory	18
A.2.2. cpu_allocation_ratio	18



# 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 colocated and configured for optimized resource usage. You can deploy an overcloud with only hyper-converged nodes, or a mixture of hyper-converged nodes with normal Compute and Ceph Storage nodes.

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 [Deploying an Overcloud with Containerized Red Hat Ceph](#).

4. The disks on nodes destined for Compute and Ceph OSD services must be cleaned, as described in [Cleaning Ceph Storage Node Disks](#) (from [Deploying an Overcloud with Containerized Red Hat Ceph](#)).
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](#)).

## 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](#) guide describes methods for configuring advanced OpenStack features through the director (for example, the use of custom roles).
- [Director Installation and Usage](#) guide provides end-to-end deployment information for both undercloud and overcloud.
- [Deploying an Overcloud with Containerized Red Hat Ceph](#) guide describes how to deploy an overcloud that uses Red Hat Ceph Storage as a storage provider.

- [Networking Guide](#) provides an advanced guide to Red Hat OpenStack Platform networking.
- [Hyper-converged Red Hat OpenStack Platform 10 and Red Hat Ceph Storage 2](#) provides 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.

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.

### Prepare Overcloud Role for Hyper-Converged Nodes

To use hyper-converged nodes, you need to define a *role* for it. Red Hat OpenStack Platform provides default roles for normal overcloud nodes (for example, Controller, Compute, and Ceph Storage), as well as a predefined role for hyper-converged nodes, **ComputeHCI**. 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.

### Configuring Resource Isolation

When you deploy HCI, Compute and Ceph Storage services need to be aware of each other as hyper-converged nodes. Otherwise, both services will consume resources as if they were on dedicated nodes. This can lead to resource contention, which can lead to performance degradation.

### Configure Networking

When using hyper-converged nodes, 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 involves specifying which environment files to include in the deployment. This involves defining a new flavor for the **ComputeHCI** role, tagging it to hyper-converged nodes, and invoking the custom **roles\_data.yaml** file (from [Chapter 3, Prepare Overcloud Role for Hyper-Converged Nodes](#)) during deployment.

## CHAPTER 3. PREPARE OVERCLOUD ROLE FOR HYPER-CONVERGED NODES

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 with both Compute and Ceph object storage daemon (OSD) services. This effectively colocates both services, allowing you to deploy them together on the same *hyper-converged node*.

Roles used for the overcloud are defined in the **roles\_data.yaml** file. You can use the director to generate a customized version of this file, containing all the roles you intend to use for your overcloud. You can then invoke the custom version during [Chapter 6, Deployment](#).

Red Hat OpenStack Platform provides a predefined custom role specifically for hyper-converged nodes, named **ComputeHCI**. To use this role, you need to 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
```

This command will generate a custom **roles\_data.yaml** file in **/home/stack/roles\_data.yaml**. This custom file contains the **ComputeHCI** role, along with the **Controller**, **Compute**, and **CephStorage** roles. Add any other roles you intend to use in your overcloud to the command.



### NOTE

For detailed information about custom roles, see [Composable Services and Custom Roles](#) and [Examining the roles\\_data File](#) from [Advanced Overcloud Customization](#).

### 3.1. CONFIGURING PORT ASSIGNMENTS FOR THE COMPUTEHCI 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 **ComputeHCI** 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::ComputeHCI::Ports::ExternalPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/noop.yaml # 1
  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.yaml # 2
```

- 1 If you are using DVR, replace this line with:

```
OS::TripleO::ComputeHCI::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.

- 2 If you want the **ComputeHCI** role to select from a pool of IPs, replace this line with:

```
OS::TripleO::ComputeHCI::Ports::StorageMgmtPort:
/usr/share/openstack-tripleo-heat-
templates/network/ports/storage_mgmt_from_pool.yaml
```

If your environment uses IPv6 addresses, replace this line with:

```
OS::TripleO::ComputeHCI::Ports::StorageMgmtPort:
/usr/share/openstack-tripleo-heat-
templates/network/ports/storage_mgmt_v6.yaml
```

If you want the **ComputeHCI** role to select from a pool of IPv6 addresses, use:

```
OS::TripleO::ComputeHCI::Ports::StorageMgmtPort:
/usr/share/openstack-tripleo-heat-
templates/network/ports/storage_mgmt_from_pool_v6.yaml
```

For any other storage IP and port settings, review the other templates in `/usr/share/openstack-tripleo-heat-templates/network/ports/` for hints on customization.

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

## 3.2. 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 **ComputeHCI** role, you also need to create a new flavor for it:

1. To create a new flavor 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 [Deploying an Overcloud with Containerized Red Hat Ceph](#)).

See [Manually Tagging the Nodes](#) and [Assigning Nodes and Flavors to Roles](#) (from [Deploying an Overcloud with Containerized Red Hat Ceph](#)) for related details.

## CHAPTER 4. CONFIGURING RESOURCE ISOLATION ON HYPER-CONVERGED NODES

With the Red Hat OpenStack Platform implementation of HCI, the director creates hyper-converged nodes by colocating Ceph OSD and Compute services. However, without any further tuning this colocation 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 subsections 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 colocation 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. To do this, you need to set resource constraints for the Compute service on hyper-converged nodes. You can configure this through a *plan environment file*.

Plan environment files define *workflows*, which the director can execute through the OpenStack Workflow (Mistral) service. The director also provides a default plan environment file specifically for configuring resource constraints on hyper-converged nodes, namely:

**`/usr/share/openstack-tripleo-heat-templates/plan-samples/plan-environment-derived-params.yaml`**

Use the **-p** parameter to invoke this plan environment file during deployment (as in, to your **openstack overcloud deploy** command). This plan environment file will direct OpenStack Workflow to:

1. Retrieve hardware introspection data (collected during [Inspecting the Hardware of Nodes](#)),
2. Calculate optimal CPU and memory constraints for Compute on hyper-converged nodes based on that data, and
3. Autogenerate the necessary parameters to configure those constraints.

The **`~/plan-samples/plan-environment-derived-params.yaml`** plan environment file defines several CPU and memory allocation workload *profile* defined under **`hci_profile_config`**. The **`hci_profile`** parameter sets which workload profile is enabled; for example, if you are using NFV, set **`hci_profile: nfvddefault`**.

You can also define a custom profile in your own plan environment file using the same syntax. For example, to define a new profile named **`my_workload`**:

The **`average_guest_memory_size_in_mb`** and **`average_guest_cpu_utilization_percentage`** parameters in each workload profile will calculate values for the **`reserved_host_memory`** and **`cpu_allocation_ratio`** settings of Compute. These values are calculated based on Red Hat recommendations, and are similar to calculations made manually in previous releases (in particular, Reserve CPU and Memory Resources for Compute).

#### 4.1.1. Override Calculated Settings for Memory or CPU Allocation

You can override the Compute settings automatically defined by OpenStack Workflow through another environment file. This is useful if you want to only override either **reserved\_host\_memory** or **cpu\_allocation\_ratio** and let OpenStack Workflow define the other. Consider the following snippet:

```
parameter_defaults:
  ComputeHCIParameters:
    NovaReservedHostMemory: 181000 # 1
  ComputeHCIExtraConfig:
    nova::cpu_allocation_ratio: 8.2 # 2
```

1 The **NovaReservedHostMemory** parameter sets how much RAM should be reserved for the Ceph OSD services and per-guest instance overhead on hyper-converged nodes.

2 The **nova::cpu\_allocation\_ratio:** parameter sets the ratio that the Compute scheduler should use when choosing which Compute node to deploy an instance.

The **ComputeHCIParameters** and **ComputeHCIExtraConfig** hooks apply their nested parameters to all nodes that use the **ComputeHCI** role (namely, all hyper-converged nodes). For more information about manually determining optimal values for **NovaReservedHostMemory** and **nova::cpu\_allocation\_ratio:**, see [Section A.2, “Compute CPU and Memory Calculator”](#).

## 4.2. 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 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.

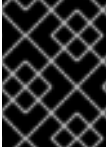
To configure the priority of backfill and recovery operations, add an environment file named **ceph-backfill-recovery.yaml** to **~/templates** containing the following:

```
parameter_defaults:
  CephAnsibleExtraConfig:
    osd_recovery_op_priority: 3 # 1
    osd_recovery_max_active: 3 # 2
    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. You do not need to add **ceph-backfill-recovery.yaml** to your deployment unless you plan to use different values.

===Reserving Memory and CPU Resources for Ceph On hyper-converged nodes, each containerized OSD should be limited in GB of RAM and vCPUs using **--memory** and **--cpu-quota** options of the docker run command. You can pass values to the options by modifying the **storage-container-config.yaml** file. The following example reserves 3 GB of RAM and 1 vCPUs per OSD:

```
parameter_defaults:
  CephAnsibleExtraConfig:
    ceph_osd_docker_memory_limit: 3g
    ceph_osd_docker_cpu_limit: 1
```

Save the above in **/home/stack/templates/storage-container-config.yaml**.

The values used in the example are good defaults for the average hyper-converged node. Depending on hardware and workload, other values may be appropriate. See the [Red Hat Ceph Storage Hardware Guide](#) for more details.

## CHAPTER 5. FINALIZE NETWORKING SETTINGS

At this point, you should have completed the necessary settings to assign ports properly on HCI nodes. However, on those nodes you still need to map the **StorageMgmtPort** to a physical NIC.

- 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.
- Create a new directory within `~/templates` called **nic-configs**. Copy your chosen template to `~/templates/nic-configs/` and rename it **compute-hci.yaml**.
- Ensure the following definition is in the **parameters:** section of your new `~/templates/nic-configs/compute-hci.yaml`:

```
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`).

- 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-hci.yaml`:

```
-
  type: vlan
  device: em2
  mtu: 9000 # 1
  use_dhcp: false
  vlan_id: {get_param: StorageMgmtNetworkVlanID}
  addresses:
    -
      ip_netmask: {get_param: StorageMgmtIpSubnet}
```

- 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, `~/templates/network.yaml`. This file should contain the following:

```
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 (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 the necessary settings to mitigate resource contention between colocated Compute and Ceph Storage services (as described in [Chapter 4, Configuring Resource Isolation on Hyper-Converged Nodes](#)).

Before you proceed, ensure 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-config.yaml` file from [Customizing the Storage Service](#) and [Sample Environment File: Creating a Ceph Cluster](#) (both sections from [Deploying an Overcloud with Containerized Red Hat Ceph](#)).
2. The same `/home/stack/templates/storage-config.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 [Deploying an Overcloud with Containerized Red Hat Ceph](#)).

To deploy your overcloud, run the following command:

```
$ 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 \
  --ntp-server pool.ntp.org
```

Where:

- **--templates** - Creates the Overcloud from the default Heat template collection (namely, `/usr/share/openstack-tripleo-heat-templates/`).
- **-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 hyper-converged deployment.
- **-r /home/stack/templates/roles\_data.yaml** - Specifies the customized roles definition file from [Chapter 3, Prepare Overcloud Role for Hyper-Converged Nodes](#), which includes the **ComputeHCI** role.
- **-e /home/stack/templates/ports.yaml** - Adds the environment file from [Section 3.1, “Configuring Port Assignments for the ComputeHCI Role”](#), 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 an Environment File](#) (from [Advanced Overcloud Customization](#)).

- **-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. See [Deploying an Overcloud with Containerized Red Hat Ceph](#) for more information.
- **-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](#) (from [Deploying an Overcloud with Containerized Red Hat Ceph](#)).



#### NOTE

In [Sample Environment File: Creating a Ceph Cluster](#) (from [Deploying an Overcloud with Containerized Red Hat Ceph](#)), the **/home/stack/templates/storage-config.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.

- **/home/stack/templates/storage-container-config.yaml** - Reserves CPU and Memory for each Ceph OSD storage container from Section 4.4, Reserving Memory and CPU Resources for Ceph.
- **-e /home/stack/templates/network.yaml** - Adds the environment file from [Chapter 5, Finalize Networking Settings](#).
- **-e /home/stack/templates/ceph-backfill-recovery.yaml** - Adds the environment file from [Section 4.2, “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 [SR-IOV Support for Virtual Networking](#) (from the [Networking 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 described below.

#### A.1.1. Scaling Up

To scale up HCI nodes in a pure HCI environment (as in, if all Compute nodes are hyper-converged nodes), 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 the overcloud features both hyper-converged and normal Compute nodes). When you tag new nodes, remember to use the right flavor (in this case, **osdcompute**). See [Section 3.2, “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 [Deploying an Overcloud with Containerized Red Hat Ceph](#)). 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 VMs from an Overcloud Compute Node](#) 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. COMPUTE CPU AND MEMORY CALCULATOR

With this release, you can use OpenStack Workflow to automatically set suitable CPU and memory allocation settings for hyper-converged nodes. However, in some instances you may only want to let OpenStack Workflow set either CPU and memory so you can set the other yourself. To do so, you can override them normally (as described in [Section 4.1.1, “Override Calculated Settings for Memory or CPU Allocation”](#)).

You can use the following script to calculate suitable baseline **NovaReservedHostMemory** and **cpu\_allocation\_ratio** values for your hyper-converged nodes.

[nova\\_mem\\_cpu\\_calc.py](#)

The following subsections describe both settings in greater detail.

### A.2.1. 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:

$$(\text{In} * \text{Ov}) + (\text{Os} * \text{RA})$$

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.2.2. 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
```

### TIP

You can also use the [nova\\_mem\\_cpu\\_calc.py](#) script to compute baseline values for both **reserved\_host\_memory** and **cpu\_allocation\_ratio**. See [Section A.2, “Compute CPU and Memory Calculator”](#) for more details.