Red Hat Hyperconverged Infrastructure for Cloud 13

Deployment Guide

Deploying a Red Hat Hyperconverged Infrastructure for Cloud Solution

Last Updated: 2020-06-04

Red Hat Hyperconverged Infrastructure for Cloud 13 Deployment Guide

Deploying a Red Hat Hyperconverged Infrastructure for Cloud Solution

Abstract

This document provides instructions for deploying the Red Hat Hyperconverged Infrastructure for Cloud solution, using Red Hat OpenStack Platform 13 and Red Hat Ceph Storage 3, all running on AMD64 and Intel 64 architectures.

Table of Contents

CHAPTER 1. INTRODUCING THE RED HAT HYPERCONVERGED INFRASTRUCTURE FOR CLOUD SOLUTION

The Red Hat Hyperconverged Infrastructure (RHHI) for Cloud solution is part of the broader software-defined RHHI solutions. The RHHI Cloud solution unifies Red Hat OpenStack Platform (RHOSP) 13 and Red Hat Ceph Storage (RHCS) 3 technologies into a single product to accomplish three goals:

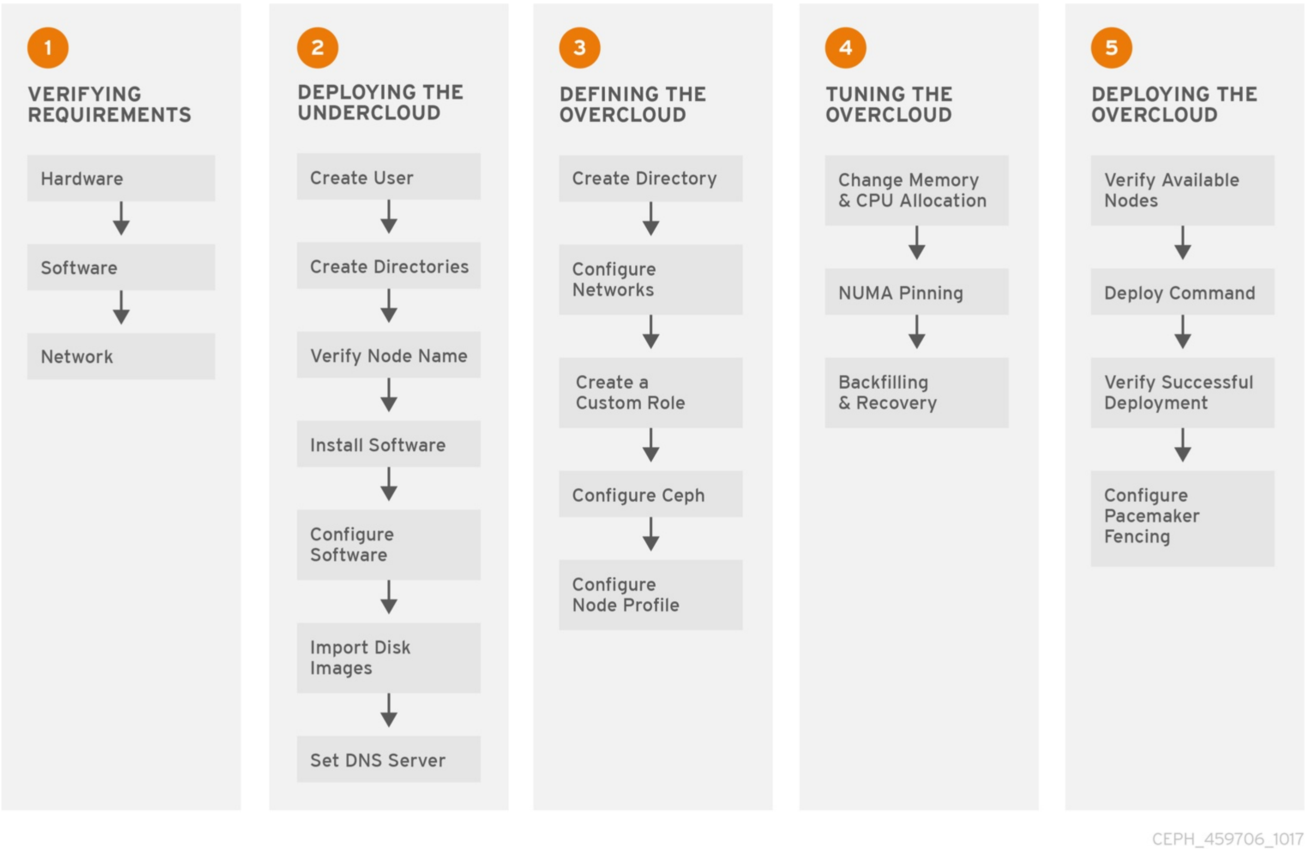
- Simplify the deployment of RHOSP and RHCS.
- Provide a more predictable performance experience.
- Achieve a lower cost of entry for RHOSP and RHCS by colocating their respective services on the same node.

The RHHI Cloud colocating scenarios are:

- The RHOSP Controller and the RHCS Monitor services on the same node.
- The RHOSP Nova Compute and the RHCS Object Storage Daemon (OSD) services on the same node.

Choosing a Deployment Workflow

You can choose to deploy the Red Hat Hyperconverged Infrastructure for Cloud by using either, the Red Hat OpenStack Platform Director web interface, or the command-line interface. This is the basic deployment workflow:



Additional Resources

- See the Deploying Red Hat Hyperconverged Infrastructure for Cloud using the Red Hat OpenStack Director section for more details.

- See the Deploying Red Hat Hyperconverged Infrastructure for Cloud using the command-line interface section for more details.

CHAPTER 2. VERIFYING THE RED HAT HYPERCONVERGED INFRASTRUCTURE FOR CLOUD REQUIREMENTS

As a technician, you need to verify three core requirements before deploying the Red Hat Hyperconverged Infrastructure for Cloud solution.

2.1. PREREQUISITES

- Verify the Hardware requirements.
- Verify the Software requirements.
- Verify the Network configuration.

2.2. THE RED HAT HYPERCONVERGED INFRASTRUCTURE FOR CLOUD HARDWARE REQUIREMENTS

Implementors of hyper-converged infrastructures will reflect a wide variety of hardware configurations. Red Hat recommends the following minimums when considering hardware:

CPU

For Controller/Monitor nodes, use dual-socket, 8-core CPUs. For Compute/OSD nodes, use dual-socket, 14-core CPUs for nodes with NVMe storage media, or dual-socket, 10-core CPUs for nodes with SAS/SATA SSDs.

RAM

Configure twice the RAM needed by the resident Nova virtual machine workloads.

OSD Disks

Use 7,200 RPM enterprise HDDs for general-purpose workloads or NVMe SSDs for IOPS-intensive workloads.

Journal Disks

Use SAS/SATA SSDs for general-purpose workloads or NVMe SSDs for IOPS-intensive workloads.

Network

Use two 10GbE NICs for Red Hat Ceph Storage (RHCS) nodes. Additionally, use dedicated NICs to meet the Nova virtual machine workload requirements. See the *network requirements* for more details.

Table 2.1. Minimum Node Quantity

Qty.	Role	Physical / Virtual
1	Red Hat OpenStack Platform director (RHOSP-d)	Either*
3	RHOSP Controller & RHCS Monitor	Physical
3	RHOSP Compute & RHCS OSD	Physical

NOTE

The RHOSP-d node can be virtualized for small deployments, that is less than 20TB in total capacity. If the solution deployment is larger than 20TB in capacity, then Red Hat recommends the RHOSP-d node be a physical node. Additional hyper-converged compute/storage nodes can be initially deployed or added at a later time.

IMPORTANT

Red Hat recommends using standalone compute and storage nodes for deployments spanning more than one datacenter rack, which is 30 nodes.

2.3. THE RED HAT HYPERCONVERGED INFRASTRUCTURE FOR CLOUD NETWORK REQUIREMENTS

Red Hat recommends using a minimum of five networks to serve various traffic roles:

Red Hat Ceph Storage

Ceph Monitor nodes use the public network. Ceph OSDs use the public network, if no private storage cluster network exists. Optionally, OSDs may use a private storage cluster network to handle traffic associated with replication, heartbeating and backfilling, leaving the public network exclusively for I/O. Red Hat recommends using a cluster network for larger deployments. The compute role needs access to this network.

External

Red Hat OpenStack Platform director (RHOSP-d) uses the External network to download software updates for the overcloud, and the overcloud operator uses it to access RHOSP-d to manage the overcloud. When tenant services establish connections via reserved floating IP addresses, the Controllers use the External network to route their traffic to the Internet. Overcloud users use the external network to access the overcloud.

OpenStack Internal API

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

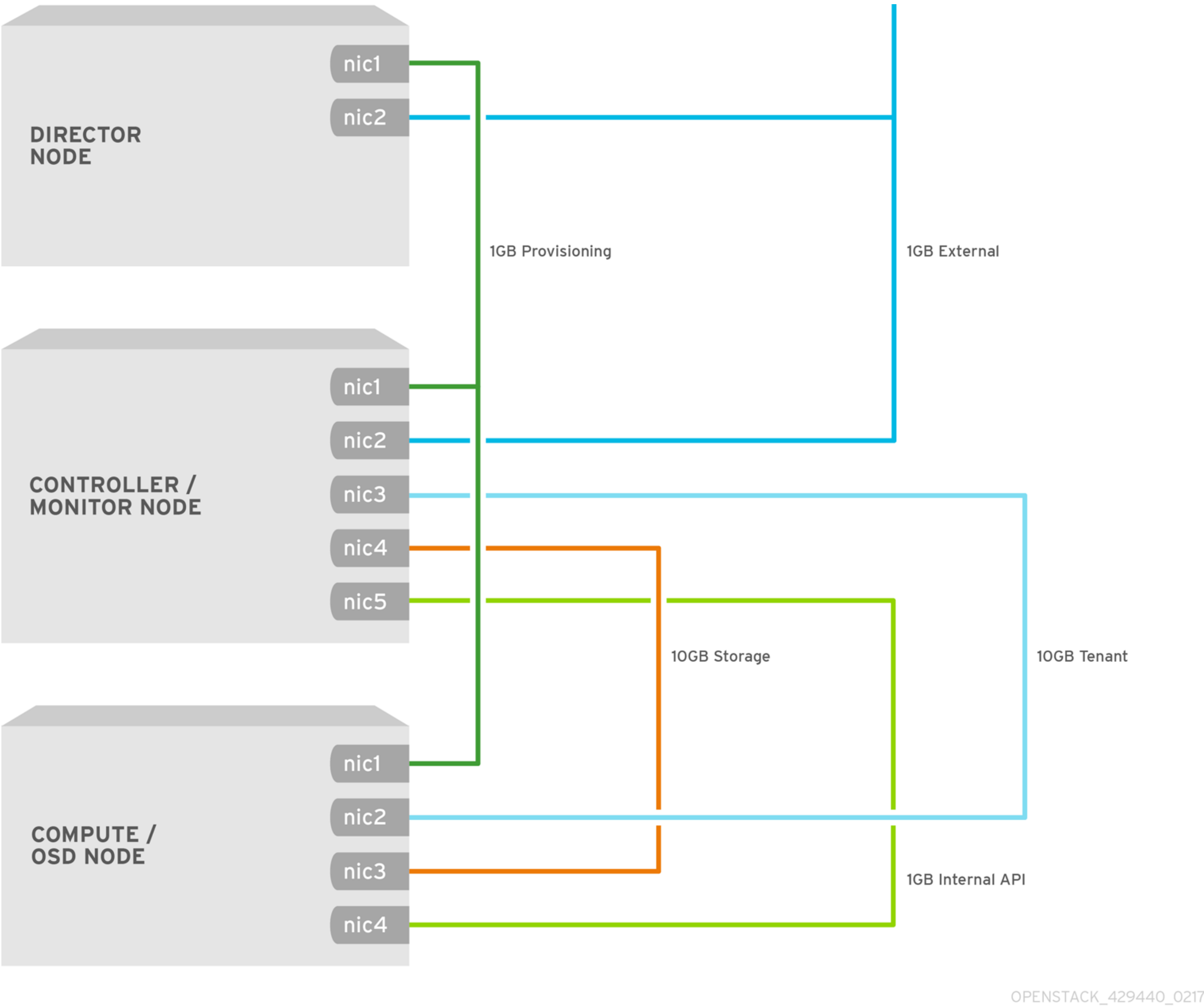
OpenStack Tenant Network

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

Red Hat OpenStack Platform Director Provisioning

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

Figure 2.1. Network Separation Diagram



NOTE

The NICs can be a logical bond of two physical NICs. It is not required to trunk each network to the same interface.

2.4. VERIFYING THE RED HAT HYPERCONVERGED INFRASTRUCTURE FOR CLOUD SOFTWARE REQUIREMENTS

Verify that the nodes have access to the necessary software repositories. The Red Hat Hyperconverged Infrastructure (RHHI) for Cloud solution requires specific software packages to be installed to function properly.

Prerequisites

- Have a valid Red Hat Hyperconverged Infrastructure for Cloud subscription.
- Root-level access to the nodes.

Procedure

1. On any node, verify the available subscriptions:

```
# subscription-manager list --available --all --matches="*OpenStack*"
```

Additional Resources

- See the Red Hat Hyperconverged Infrastructure for Cloud required repositories section for the required software repositories.

2.5. ADDITIONAL RESOURCES

- For more information, see the Red Hat Ceph Storage Hardware Guide.

CHAPTER 3. DEPLOYING THE UNDERCLOUD

As a technician, you can deploy an undercloud, which provides users with the ability to deploy and manage overclouds with the Red Hat OpenStack Platform Director interface.

3.1. PREREQUISITES

- Have a valid Red Hat Hyperconverged Infrastructure for Cloud subscription.
- Have access to Red Hat's software repositories through Red Hat's Content Delivery Network (CDN).

3.2. UNDERSTANDING IRONIC'S DISK CLEANING BETWEEN DEPLOYMENTS

Enabling Ironic's disk cleaning feature will permanently delete all data from all the disks on a node before that node becomes available again for deployment.

There are two facts that you should consider before enabling Ironic's disk cleaning feature:

- When director deploys Ceph it uses the ceph-disk command to prepare each OSD. Before ceph-disk prepares an OSD, it checks if the disk which will host the new OSD has data from an older OSD and if it does, then it will fail the disk preparation in order to not overwrite that data. It does this as a safety feature so that data is not lost.
- If a deployment attempt with director fails and is then repeated after the overcloud is deleted, then by default the data from the previous deployment will still be on the server disks. This data may cause the repeated deployment to fail because of how the ceph-disk command behaves.

NOTE

If an overcloud node is accidentally deleted and disk cleaning is enabled, then the data will be removed and can only be put back into the environment by rebuilding the node with Red Hat OpenStack Platform Director.

3.3. INSTALLING THE UNDERCLOUD

Several steps must be completed to install the undercloud. This procedure is installing the Red Hat OpenStack Platform director (RHOSP-d) as the undercloud. Here is a summary of the installation steps:

1. Create an installation user.
2. Create directories for templates and images.
3. Verify/Set the RHOSP-d node name.
4. Register the RHOSP-d node.
5. Install the RHOSP-d software.
6. Configure the RHOSP-d software.
7. Obtain and import disk images for the overcloud.

8. Set a DNS server on the undercloud's subnet.

Prerequisites

- Have access to Red Hat's software repositories through Red Hat's Content Delivery Network (CDN).
- Having **root** access to the Red Hat OpenStack Platform director (RHOSP-d) node.

Procedure

1. The RHOSP-d installation requires a non-root user with **sudo** privileges to do the installation.
    a. Create a user named **stack**:

    ```
    [root@director ~]# useradd stack
    ```

    b. Set a password for **stack**. When prompted, enter the new password:

    ```
    [root@director ~]# passwd stack
    ```

    c. Configure **sudo** access for the **stack** user:

    ```
    [root@director ~]# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a
    /etc/sudoers.d/stack
    [root@director ~]# chmod 0440 /etc/sudoers.d/stack
    ```

    d. Switch to the **stack** user:

    ```
    [root@director ~]# su - stack
    ```

    The RHOSP-d installation will be done as the **stack** user.

2. Create two new directories in the **stack** user's home directory, one named **templates** and the other named **images**:

    ```
    [stack@director ~]$ mkdir ~/images
    [stack@director ~]$ mkdir ~/templates
    ```

    These directories will organize the system image files and Heat template files used to create the overcloud environment later.

3. The installing and configuring process requires a fully qualified domain name (FQDN), along with an entry in the **/etc/hosts** file.
    a. Verify the RHOSP-d node's host name:

    ```
    [stack@director ~]$ hostname -f
    ```

    b. If needed, set the host name:

    ```
    sudo hostnamectl set-hostname FQDN_HOST_NAME
    sudo hostnamectl set-hostname --transient FQDN_HOST_NAME
    ```

Replace...

- *FQDN_HOST_NAME* with the fully qualified domain name (FQDN) of the RHOSP-d node.

  Example

  ```
  [stack@director ~]$ sudo hostnamectl set-hostname director.example.com
  [stack@director ~]$ sudo hostnamectl set-hostname --transient
  director.example.com
  ```

c. Add an entry for the RHOSP-d node name to the **/etc/hosts** file. Add the following line to the **/etc/hosts** file:

```
sudo echo "127.0.0.1 FQDN_HOST_NAME SHORT_HOST_NAME localhost
localhost.localdomain localhost4 localhost4.localdomain4" >> /etc/hosts
```

Replace...

- *FQDN_HOST_NAME* with the full qualified domain name of the RHOSP-d node.
- *SHORT_HOST_NAME* with the short domain name of the RHOSP-d node.

  Example

  ```
  [stack@director ~]$ sudo echo "127.0.0.1 director.example.com director localhost
  localhost.localdomain localhost4 localhost4.localdomain4" >> /etc/hosts
  ```

4. Register the RHOSP-d node on the Red Hat Content Delivery Network (CDN), and enable the required Red Hat software repositories using the Red Hat Subscription Manager.

a. Register the RHOSP-d node:

```
[stack@director ~]$ sudo subscription-manager register
```

When prompted, enter an authorized Customer Portal user name and password.

b. Lookup the valid **Pool ID** for the RHOSP entitlement:

```
[stack@director ~]$ sudo subscription-manager list --available --all --
matches="*Hyperconverged*"
```

Example Output

```
Subscription Name:   Red Hat Hyperconverged Infrastructure for Cloud
Provides:            Red Hat OpenStack
                     Red Hat Ceph Storage
SKU:                 RS00160
Contract:            1111111
Pool ID:             a1b2c3d4e5f6g7h8i9
Provides Management: Yes
Available:           1
Suggested:           1
```

```
Service Level:      Self-Support
Service Type:       L1-L3
Subscription Type:  Standard
Ends:               05/27/2018
System Type:        Virtual
```

c. Using the **Pool ID** from the previous step, attach the RHOSP entitlement:

```
[stack@director ~]$ sudo subscription-manager attach --pool=POOL_ID
```

Replace...

- *POOL_ID* with the valid pool id from the previous step.

Example

```
[stack@director ~]$ sudo subscription-manager attach --
pool=a1b2c3d4e5f6g7h8i9
```

d. Disable the default software repositories, and enable the required software repositories:

```
[stack@director ~]$ sudo subscription-manager repos --disable=*
[stack@director ~]$ sudo subscription-manager repos --enable=rhel-7-server-rpms --
enable=rhel-7-server-extras-rpms --enable=rhel-7-server-rh-common-rpms --
enable=rhel-ha-for-rhel-7-server-rpms --enable=rhel-7-server-openstack-13-rpms
```

e. If needed, update the base system software to the latest package versions, and reboot the RHOSP-d node:

```
[stack@director ~]$ sudo yum update
[stack@director ~]$ sudo reboot
```

Wait for the node to be completely up and running before continuing to the next step.

5. Install all the RHOSP-d software packages:

```
[stack@director ~]$ sudo yum install python-tripleoclient ceph-ansible
```

6. Configure the RHOSP-d software.

a. Red Hat provides a basic undercloud configuration template to use. Copy the **undercloud.conf.sample** file to the **stack** user's home directory, named **undercloud.conf**:

```
[stack@director ~]$ cp /usr/share/instack-undercloud/undercloud.conf.sample
~/undercloud.conf
```

b. The undercloud configuration template contains two sections: **[DEFAULT]** and **[auth]**. Open the **undercloud.conf** file for editing. Edit the **undercloud_hostname** with the RHOSP-d node name. Uncomment the following parameters under the **[DEFAULT]** section in the **undercloud.conf** file by deleting the **#** before the parameter. Edit the parameter values with the appropriate values as required for this solution's network configuration:

Parameter	Network	Edit Value?	Example Value

local_ip	Provisioning	Yes	**192.0.2.1/24**
network_gateway	Provisioning	Yes	**192.0.2.1**
undercloud_public_vip	Provisioning	Yes	**192.0.2.2**
undercloud_admin_vip	Provisioning	Yes	**192.0.2.3**
local_interface	Provisioning	Yes	**eth1**
network_cidr	Provisioning	Yes	**192.0.2.0/24**
masquerade_network	Provisioning	Yes	**192.0.2.0/24**
dhcp_start	Provisioning	Yes	**192.0.2.5**
dhcp_end	Provisioning	Yes	**192.0.2.24**
inspection_interface	Provisioning	No	**br-ctlplane**
inspection_iprange	Provisioning	Yes	**192.0.2.100,192.0.2.120**
inspection_extras	N/A	Yes	**true**
inspection_runbench	N/A	Yes	**false**
inspection_enable_uefi	N/A	Yes	**true**

Save the changes after editing the **undercloud.conf** file. See the Undercloud configuration parameters for detailed descriptions of these configuration parameters.

NOTE

Consider enabling Ironic's disk cleaning feature, if overcloud nodes are going to be repurposed again. See the Understanding Ironic disk cleaning between deployments section for more details.

c. Run the RHOSP-d configuration script:

```
[stack@director ~]$ openstack undercloud install
```

> **NOTE**
>
> This script will take several minutes to complete. This script will install additional software packages and generates two files:
>
> **undercloud-passwords.conf**
>
> A list of all passwords for the director's services.
>
> **stackrc**
>
> A set of initialization variables to help you access the director's command line tools.

d. Verify that the configuration script started and enabled all of the RHOSP services:

```
[stack@director ~]$ sudo systemctl list-units openstack-*
```

e. The configuration script gives the **stack** user access to all the container management commands. Refresh the **stack** user's permissions:

```
[stack@director ~]$ exec su -l stack
```

f. Initialize the **stack** user's environment to use the RHOSP-d command-line tools:

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

The command-line prompt will change, which indicates that OpenStack commands will authenticate and execute against the undercloud:

Example

```
(undercloud) [stack@director ~]$
```

7. The RHOSP-d requires several disk images for provisioning the overcloud nodes.

a. Obtain these disk images by installing **rhosp-director-images** and **rhosp-director-images-ipa** software packages:

```
(undercloud) [stack@director ~]$ sudo yum install rhosp-director-images rhosp-director-images-ipa
```

b. Extract the archive files to the **images** directory in the **stack** user's home directory:

```
(undercloud) [stack@director ~]$ cd ~/images
(undercloud) [stack@director ~]$ for x in /usr/share/rhosp-director-images/overcloud-full-latest-13.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-13.0.tar ; do tar -xvf $x ; done
```

c. Import the disk images into the RHOSP-d:

```
(undercloud) [stack@director ~]$ openstack overcloud image upload --image-path /home/stack/images/
```

d. To view a list of imported disk images, execute the following command:

```
(undercloud) [stack@director ~]$ openstack image list
```

Image Name	Image Type	Image Description
bm-deploy-kernel	Deployment	Kernel file used for provisioning and deploying systems.
bm-deploy-ramdisk	Deployment	RAMdisk file used for provisioning and deploying systems.
overcloud-full-vmlinuz	Overcloud	Kernel file used for the base system, which is written to the node’s disk.
overcloud-full-initrd	Overcloud	RAMdisk file used for the base system, which is written to the node’s disk.
overcloud-full	Overcloud	The rest of the software needed for the base system, which is written to the node’s disk.

NOTE

The **openstack image list** command will not display the introspection PXE disk images. The introspection PXE disk images are copied to the /**httpboot**/ directory.

```
(undercloud) [stack@director images]$ ls -l /httpboot
total 341460
-rwxr-xr-x. 1 root              root              5153184 Mar 31 06:58
agent.kernel
-rw-r--r--. 1 root              root            344491465 Mar 31 06:59
agent.ramdisk
-rw-r--r--. 1 ironic-inspector  ironic-inspector       337 Mar 31 06:23
inspector.ipxe
```

8. Set the DNS server so that it resolves the overcloud node host names.

    a. List the subnets:

    ```
    (undercloud) [stack@director ~]$ openstack subnet list
    ```

    b. Define the name server using the undercloud’s **neutron** subnet:

    ```
    openstack subnet set --dns-nameserver DNS_NAMESERVER_IP
    SUBNET_NAME_or_ID
    ```

    Replace…

    - *DNS_NAMESERVER_IP* with the IP address of the DNS server.
    - *SUBNET_NAME_or_ID* with the **neutron** subnet name or id.

Example

```
(undercloud) [stack@director ~]$ openstack subnet set --dns-nameserver
192.0.2.4 local-subnet
```

> **NOTE**
>
> Reuse the **--dns-nameserver *DNS_NAMESERVER_IP*** option for each name server.

c. Verify the DNS server by viewing the subnet details:

```
(undercloud) [stack@director ~]$ openstack subnet show SUBNET_NAME_or_ID
```

Replace...

- *SUBNET_NAME_or_ID* with the **neutron** subnet name or id.

Example

```
(undercloud) [stack@director ~]$ openstack subnet show local-subnet
+-------------------+-----------------------------------------------+
| Field             | Value                                         |
+-------------------+-----------------------------------------------+
| ...               |                                               |
| dns_nameservers   | 192.0.2.4                                     |
| ...               |                                               |
+-------------------+-----------------------------------------------+
```

Additional Resources

- For more information on all the undercloud configuration parameters located in the **undercloud.conf** file, see the Configuring the Director section in the RHOSP Director Installation and Usage Guide.

3.4. CONFIGURING THE UNDERCLOUD TO CLEAN THE DISKS BEFORE DEPLOYING THE OVERCLOUD

Updating the undercloud configuration file to clean disks before deploying the overcloud.

> **WARNING**
>
> Enabling this feature will destroy all data on all disks before they are provisioned in the overcloud deployment.

Prerequisites

- Installation of the undercloud.

Procedure

1. There are two options, an automatic or manual way to cleaning the disks before deploying the overcloud:

    a. First option is automatically cleaning the disks by editing the **undercloud.conf** file, and add the following line:

    ```
    clean_nodes = True
    ```

    > **NOTE**
    >
    > The bare metal provisioning service runs a **wipefs --force --all** command to accomplish the cleaning.

> **WARNING**
>
> Enabling this feature will destroy all data on all disks before they are provisioned in the overcloud deployment. Also, this will do an additional power cycle after the first introspection and before each deployment.

    a. The second option is to keep automatic cleaning off and run the following commands for each Ceph node:

    ```
    [stack@director ~]$ openstack baremetal node manage NODE
    [stack@director ~]$ openstack baremetal node clean NODE --clean-steps '[{"interface":
    "deploy", "step": "erase_devices_metadata"}]'
    [stack@director ~]$ openstack baremetal node provide NODE
    ```

    Replace...

    - *NODE* with the Ceph host name.

CHAPTER 4. DEPLOYING RED HAT HYPERCONVERGED INFRASTRUCTURE FOR CLOUD USING THE RED HAT OPENSTACK PLATFORM DIRECTOR

As a technician, you can deploy and manage the Red Hat Hyperconverged Infrastructure for Cloud solution using the Red Hat OpenStack Platform Director interface. Also, you should have a basic understanding of resource isolation, so there is not resource contention between Red Hat OpenStack Platform and Red Hat Ceph Storage.

4.1. PREREQUISITES

- Verify that all the requirements are met.
- Installing the undercloud

4.2. EXPORTING AN OVERCLOUD PLAN USING THE RED HAT OPENSTACK PLATFORM DIRECTOR

This procedure is for exporting a deployment plan using the OpenStack Platform Director. The default deployment plan contains a common, and exportable overcloud configuration.

Prerequisites

- Verify that all the requirements are met.
- Installation of the undercloud.

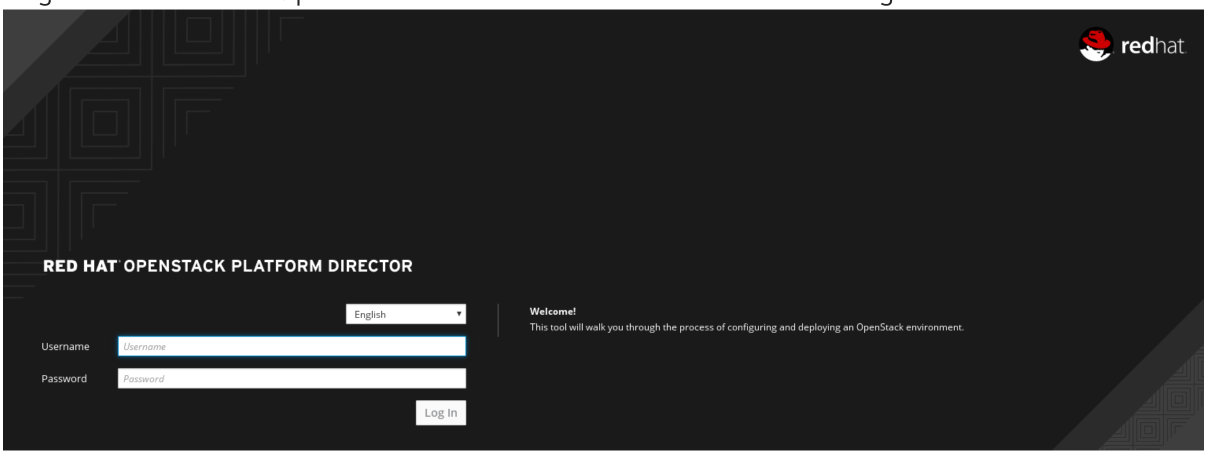
Procedure

1. Enter the IP address or host name of the undercloud into a web browser.

   NOTE

   If not using SSL, then the undercloud URL will need to use port 3000. For example: **http://192.168.0.4:3000**

2. Login to the Red Hat OpenStack Platform Director user interface using the correct credentials.

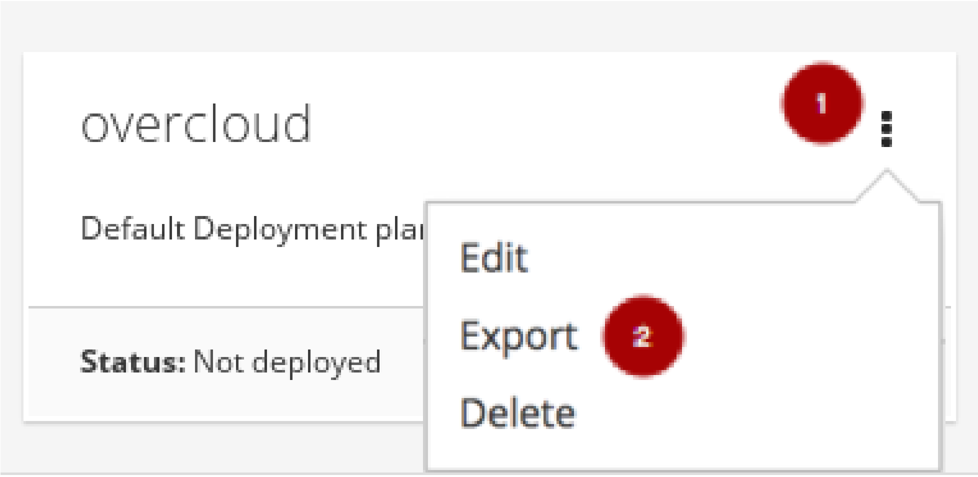


NOTE

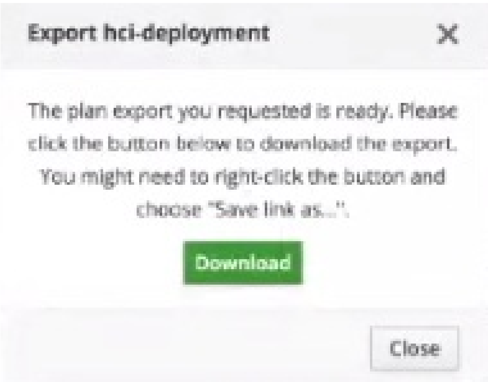
The default user name is **admin**. You can obtain the admin password by running the following command:

```
[stack@director ~]$ sudo hiera admin_password
```

3. On the *Plans* tab, select the drop-down menu 1 from the *Overcloud* plan, and select *Export* 2 .



4. Click on the *Download* button.



This will download a compressed tarball file to the local hard drive, which includes all the plan files.

IMPORTANT

If you need to add or modify the files contained within the tarball file, then before importing the tarball file you must recreate the tarball file, as follows:

Example

```
tar -czf my-deployment-plan.tar.gz -C my-deployment-plan-local-files/ .
```

NOTE

Currently, the OpenStack Platform Director interface does not support advance configuration of the plan, such as a custom network configuration. Advance configuration must be done manually by editing the files directly.

4.3. IMPORTING AN OVERCLOUD PLAN USING THE RED HAT OPENSTACK PLATFORM DIRECTOR

This procedure is for importing a deployment plan using the OpenStack Platform Director that has previously been exported.

Prerequisites

- Verify that all the requirements are met.
- Installation of the undercloud.

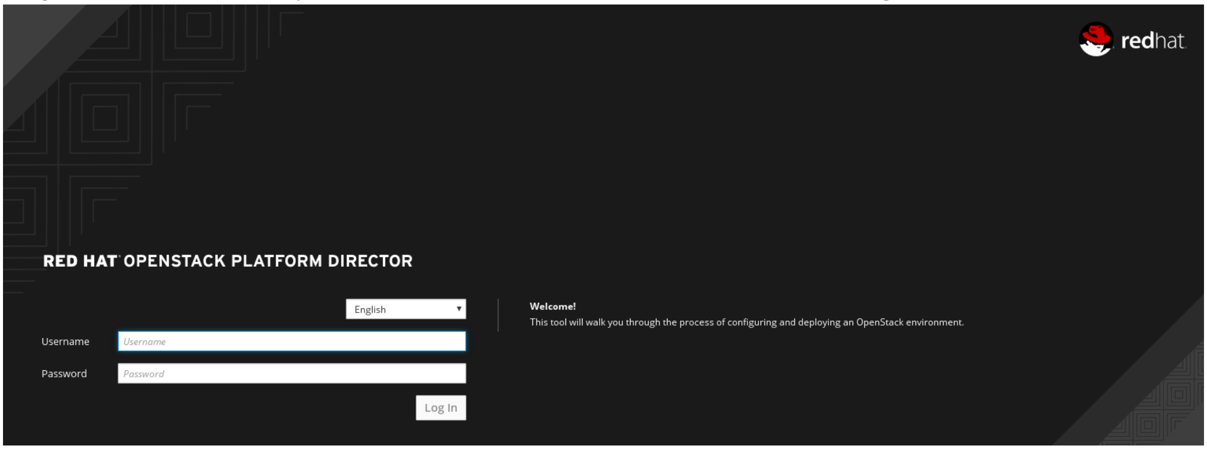
Procedure

1. Enter the IP address or host name of the undercloud into a web browser.

   NOTE

   If not using SSL, then the undercloud URL will need to use port 3000. For example: **http://192.168.0.4:3000**

2. Login to the Red Hat OpenStack Platform Director user interface using the correct credentials.



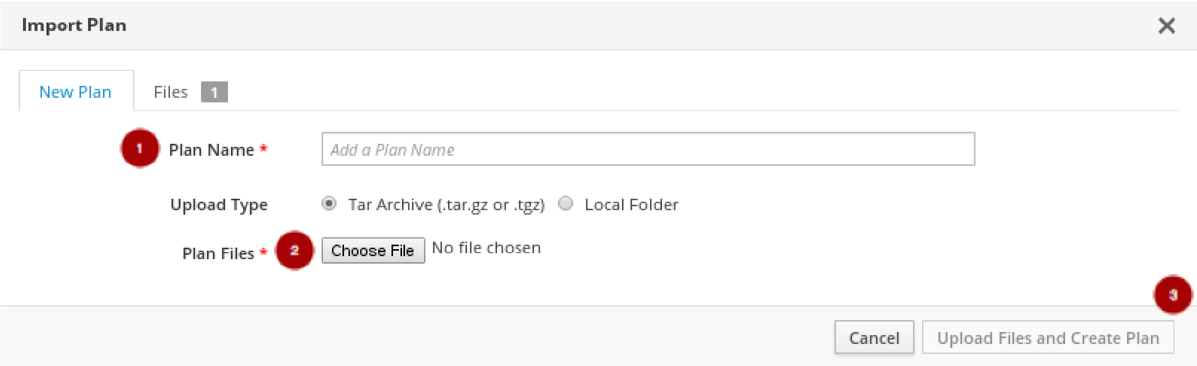
   NOTE

   The default user name is **admin**. You can obtain the admin password by running the following command:

```
[stack@director ~]$ sudo hiera admin_password
```

3. On the *Plans* tab, select the *Import Plan* button.

4. Enter *Plan Name* and click on the *Choose File* button . Browse to the location of the tarball file, and select it for import. Once the file is selected, click on the *Upload Files and Create Plan* button .



4.4. DEPLOYING THE OVERCLOUD USING THE RED HAT OPENSTACK PLATFORM DIRECTOR

This procedure deploys the overcloud using the Red Hat OpenStack Platform Director.

Prerequisites

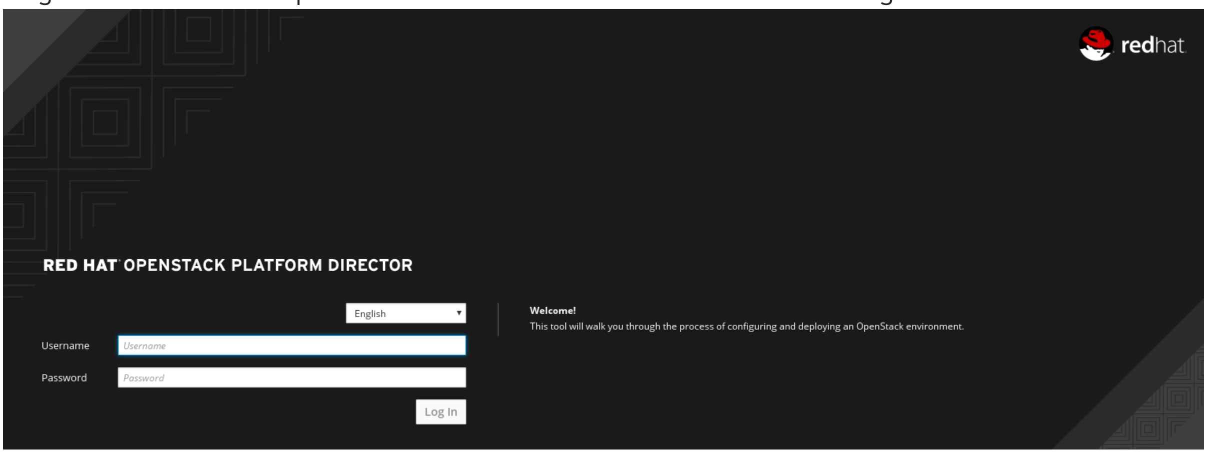
- Verify that all the requirements are met.
- Installation of the undercloud.

Procedure

1. Enter the IP address or host name of the undercloud into a web browser.

> **NOTE**
>
> If not using SSL, then the undercloud URL will need to include port 3000. For example: **http://192.168.0.4:3000**

2. Login to the Red Hat OpenStack Platform Director user interface using the correct credentials.

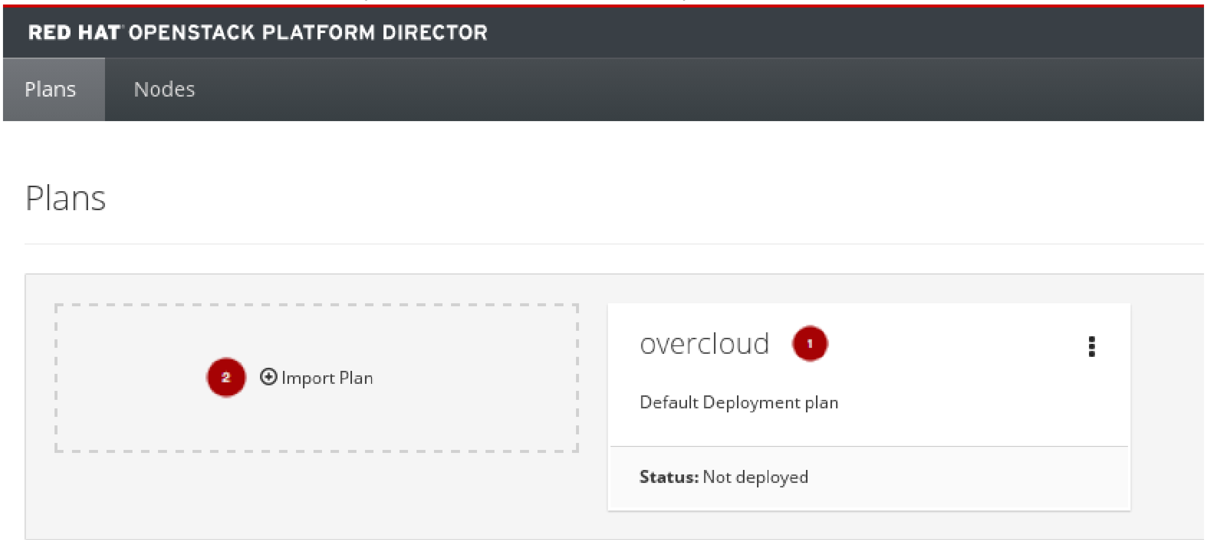


NOTE

The default user name is **admin**. You can obtain the admin password by running the following command:

```
[stack@director ~]$ sudo hiera admin_password
```

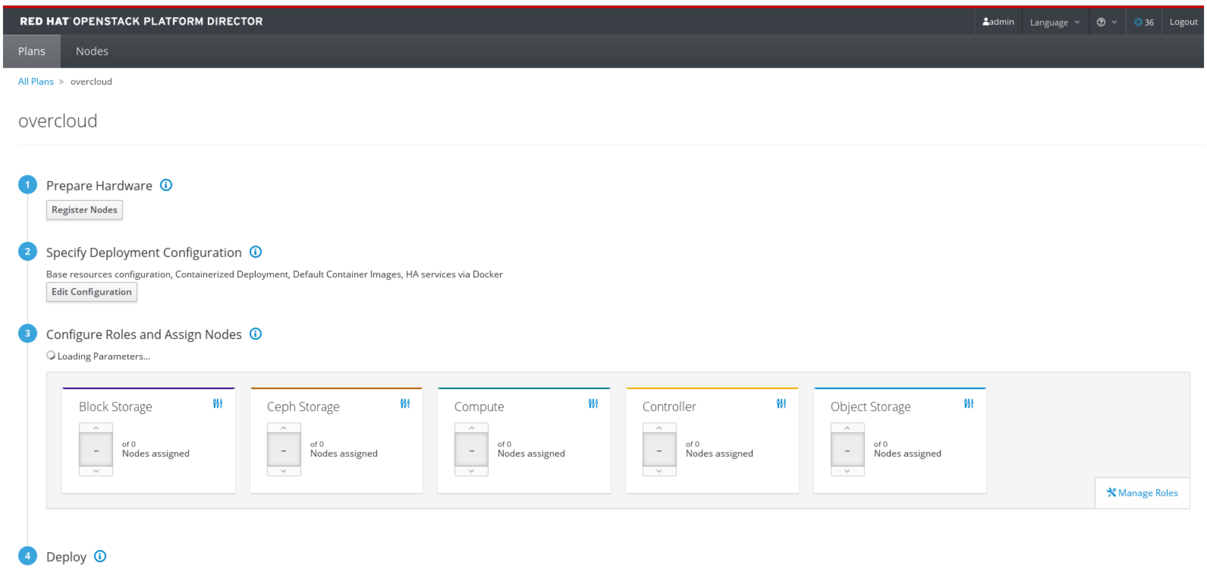
3. Select the default *overcloud* plan 1 or select the *Import Plan* 2 .



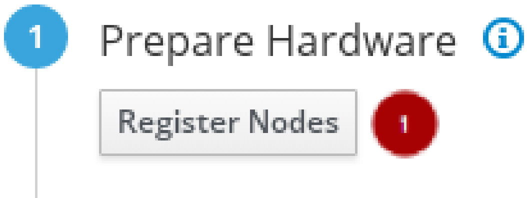
For more information on importing a plan, see Section 4.3, "Importing an overcloud plan using the Red Hat OpenStack Platform Director"

4. From the plan configuration page, prepare the hardware by adding registered nodes.

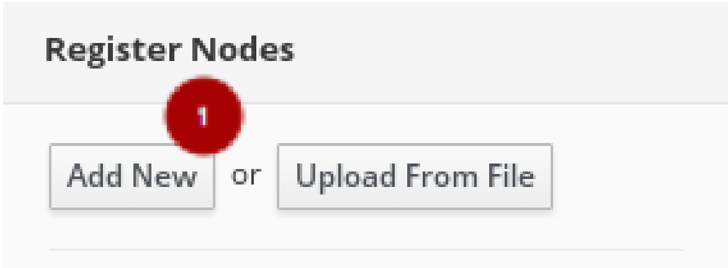
Figure 4.1. Example Plan Configuration Page



a. Click on the *Register Nodes* button 1 to registered the nodes.



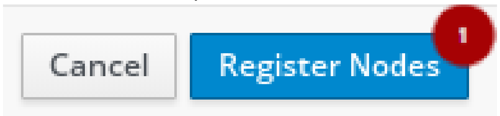
b. Click on the *Add New Node* button .



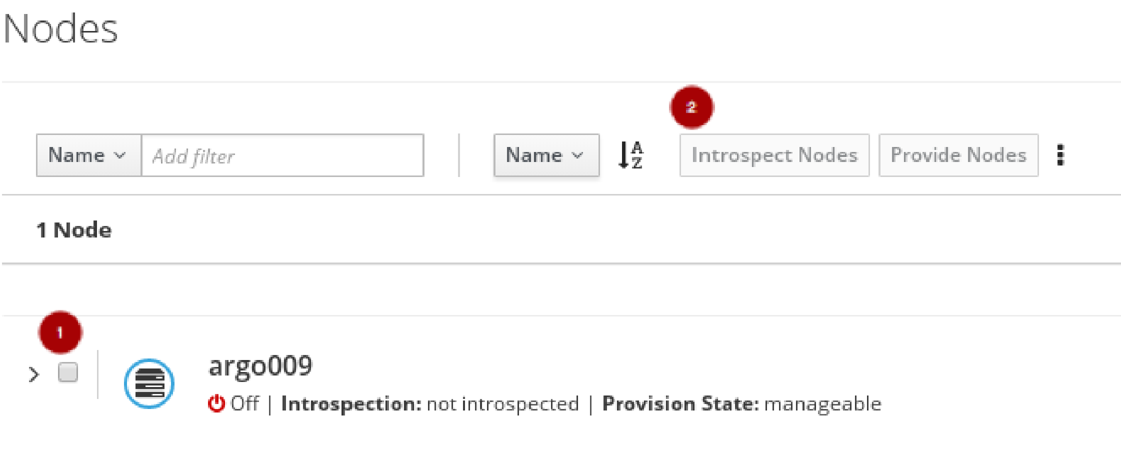
Alternatively, you can prepare the nodes by customizing the **instackenv.json** host definition file and uploading it. To create a custom **instackenv.json** host definition file, see Section 5.2.2, "Registering and introspecting the hardware" and Section 5.2.3, "Setting the root device" to prepare the nodes.

c. Fill out all the required fields, denoted by a small red asterisks, on the register node page.

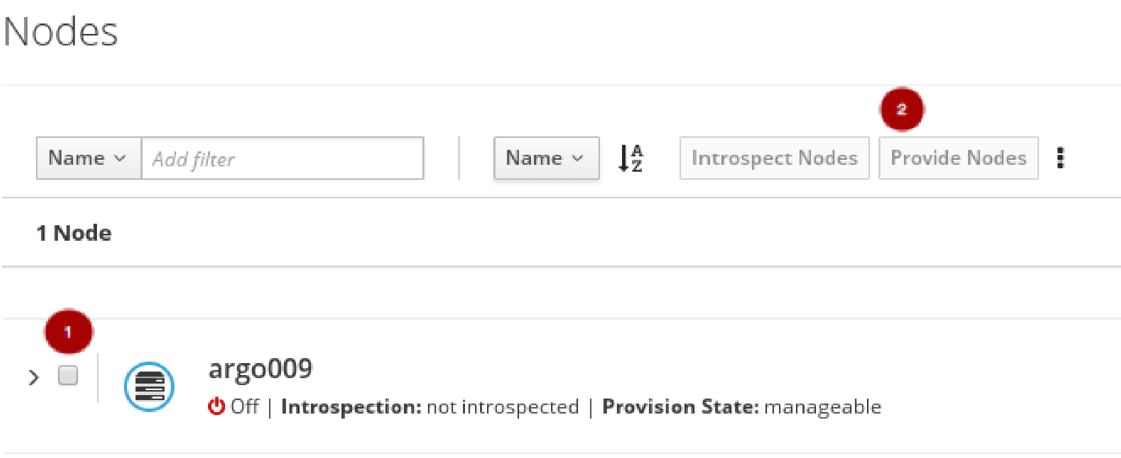
d. After all the required field are filled out, click on the *Register Node* button .



e. Once the node is registered, select the node , and click on the *Introspect Nodes* button.

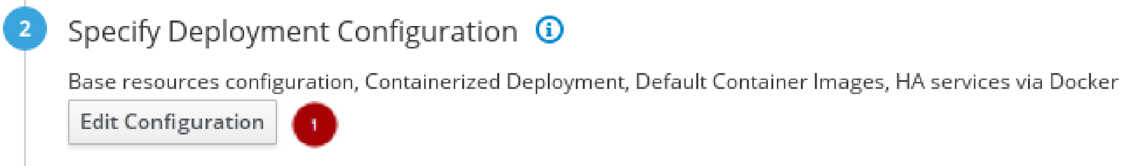


f. Once the introspection is done, select the node , and click on the *Provide Nodes* button.

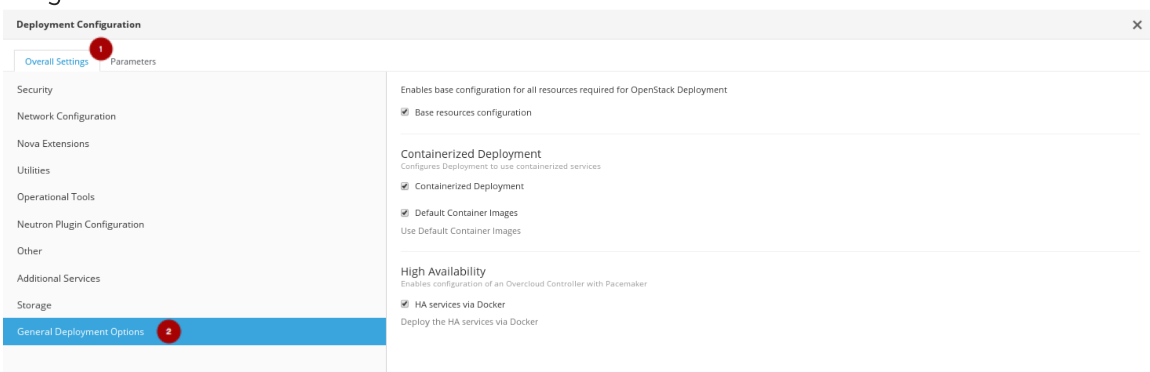


5. From the plan configuration page, edit the deployment configuration.

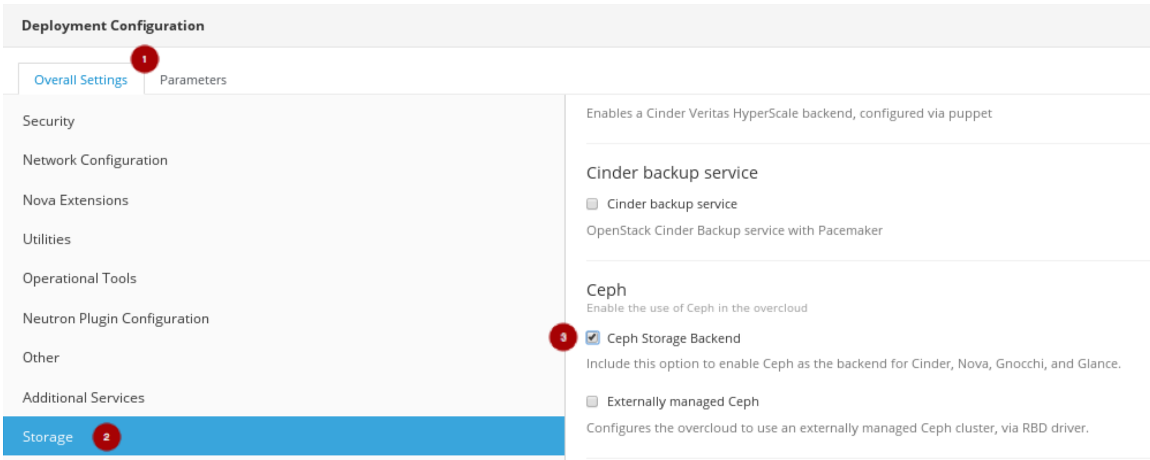
   a. Click on the *Edit Configuration* button .



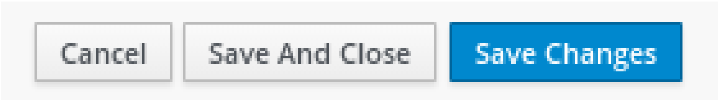
   b. On the *Overall Settings* tab , click on the *General Deployment Options* section , and enable the *HA services via Docker*, *Containerized Deployment*, and *Default Container Images*.



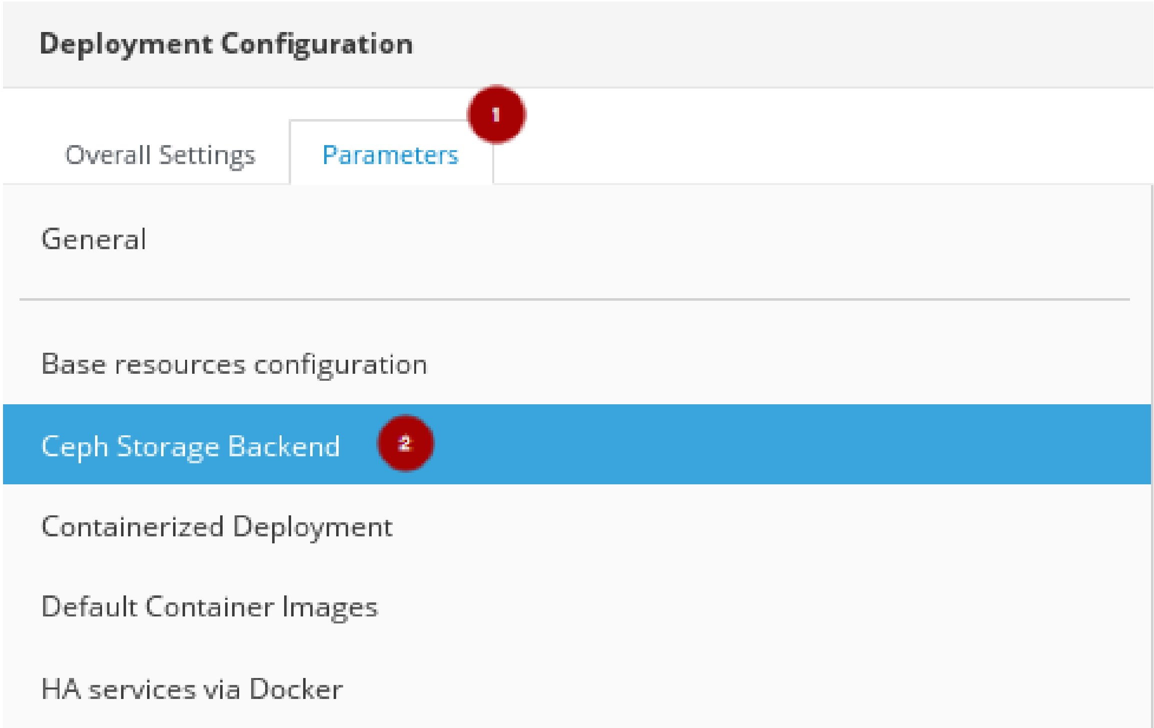
   c. On the *Overall Settings* tab , click on the *Storage* section , and enable the *Ceph Storage Backend* .



Click on the *Save Changes* button.



d. Click on the *Parameters* tab , then click on the *Ceph Storage Backend* section to edit additional Ceph parameters.



Update the *CephAnsibleExtraConfig* field with the following values:

```
{"ceph_osd_docker_memory_limit": "5g", "ceph_osd_docker_cpu_limit": 1,
"ceph_mds_docker_memory_limit": "4g", "ceph_mds_docker_cpu_limit": 1}
```

Update the *CephConfigOverrides* field with the following values.

```
{"osd_recovery_op_priority": 3, "osd_recovery_max_active": 3, "osd_max_backfills": 1}
```

Update the *CephConfigOverrides* field with the following values.

```
{"osd_recovery_op_priority": 3, "osd_recovery_max_active": 3, "osd_max_backfills": 1}
```

Set the *CephPoolDefaultSize* value to **3**.

Update the *CephAnsibleDisksConfig* field with a disk list.

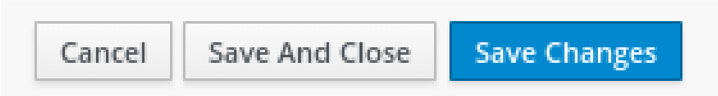
Example

```
{"devices":
["/dev/sda","/dev/sdb","/dev/sdc","/dev/sdd","/dev/sde","/dev/sdf","/dev/sdg","/dev/sdh","/dev
/sdi","/dev/sdj","/dev/sdk","/dev/sdl"],"dedicated_devices":
["/dev/sdm","/dev/sdm","/dev/sdm","/dev/sdm","/dev/sdn","/dev/sdn","/dev/sdn","/dev/sdn","/
dev/sdo","/dev/sdo","/dev/sdo","/dev/sdo"],"journal_size":5120}
```

NOTE

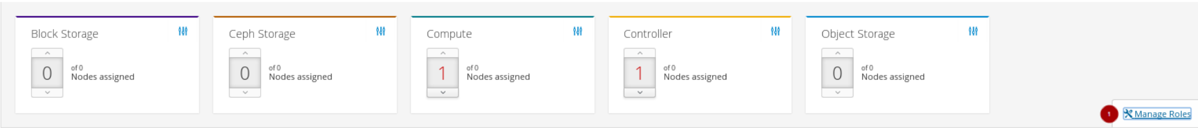
This disk listing is for block devices (**devices**) being used as OSDs, and the block devices dedicated (**dedicated_devices**) as OSD journals. See Section 5.5.5, "Setting the Red Hat Ceph Storage parameters" for more information.

e. Click on the *Save And Close* button.

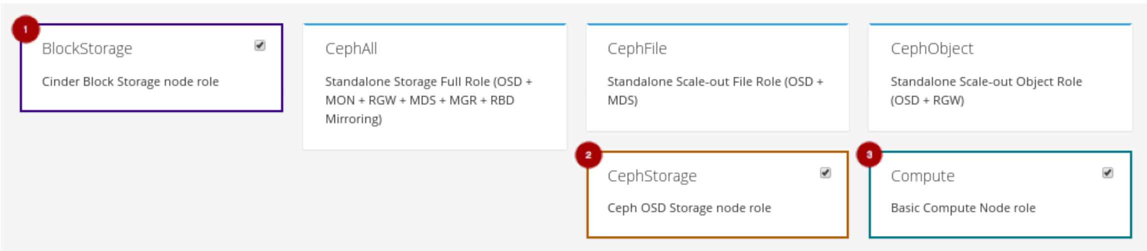


f. Back on the plan configuration page, the saved configuration changes will appear under the *Specify Deployment Configuration* step.

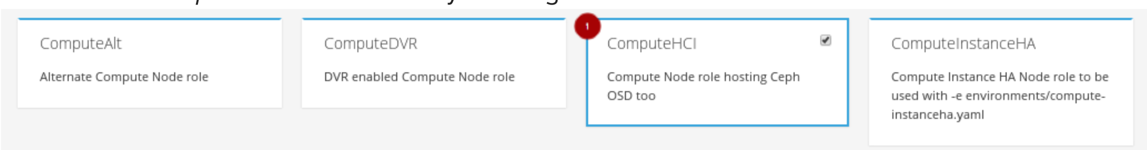
6. Configure the roles for the hyperconverged nodes by clicking on the *Manage Roles* link 1 .



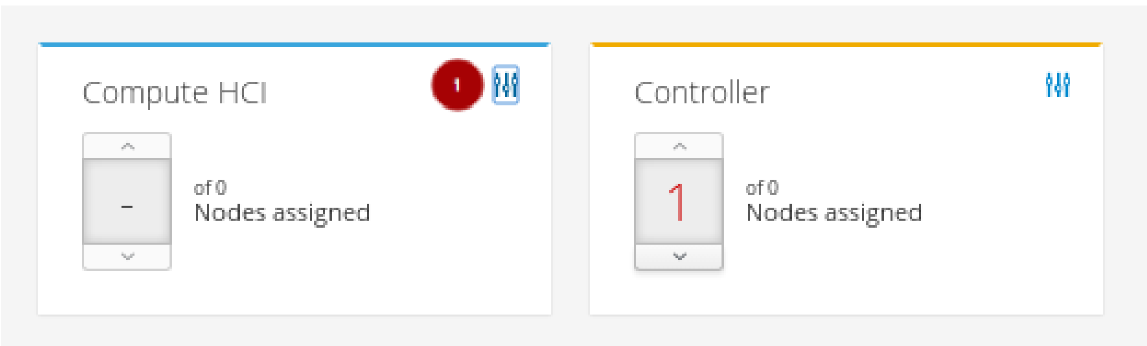
a. Unselect the *BlockStorage* 1 , *CephStorage* 2 , and *Compute* 3 roles by clicking on them.



b. Select the *ComputeHCI* 1 role by clicking on it.



c. Back on the plan configuration page, configure the *Compute HCI* role by clicking on the levers icon 1 .



d. On the *Parameters* tab, update the following parameters:

- The *ExtraConfig* field with the calculated resource allocation values. See Appendix E, *Tuning the Nova reserved memory and CPU allocation manually* for how to calculate the appropriate values.

- The *ComputeHCIIPs* field with all the relevant IP addresses for the environment.

  Example

  ```
  {"storage_mgmt":["172.16.2.203","172.16.2.204","172.16.2.205"],"storage":
  ["172.16.1.203","172.16.1.204","172.16.1.205"],"tenant":
  ["192.168.3.203","192.168.3.204","192.168.3.205"],"internal_api":
  ["192.168.2.203","192.168.2.204","192.168.2.205"]}
  ```

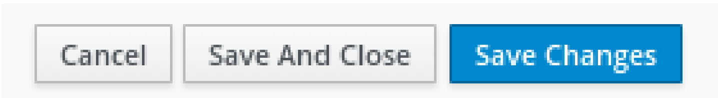
- The *OvercloudComputeHCIFlavor* field with the following value:

  ```
  osd-compute
  ```

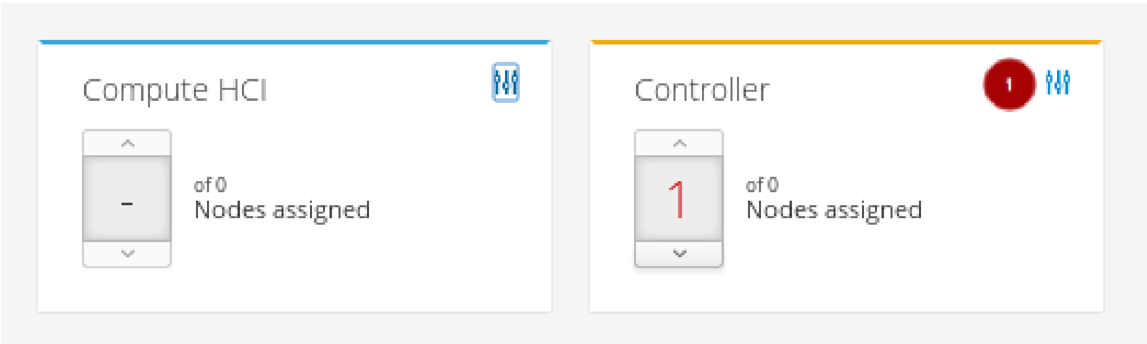
- The *ComputeHCISchedulerHints* field with the following value:

  ```
  {"capabilities:node":"hci-%index%"}
  ```

e. Click on the *Save And Close* button.



f. Back on the plan configuration page, configure the *Controller* role by clicking on the levers icon 1 .

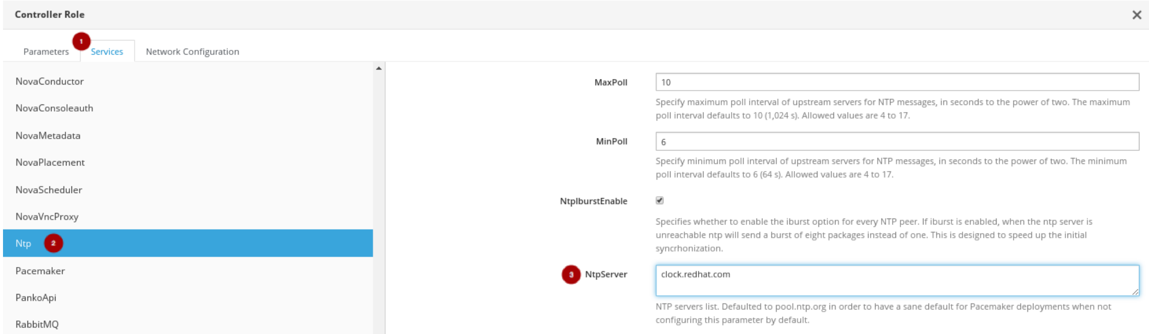


g. On the *Parameters* tab 1 , update the *ControllerIPs* field with the relevant IP addresses.

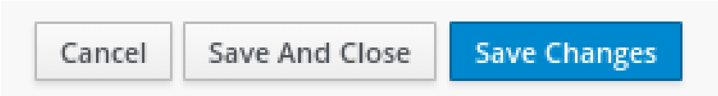
Example

```
{"storage_mgmt":["172.16.2.200","172.16.2.201","172.16.2.202"],"storage":
["172.16.1.200","172.16.1.201","172.16.1.202"],"tenant":
["192.168.3.200","192.168.3.201","192.168.3.202"],"internal_api":
["192.168.2.200","192.168.2.201","192.168.2.202"]}
```

h. On the *Services* tab 1 , in the *Ntp* section 2 , update the *NtpServer* field 3 with the relevant NTP server name.

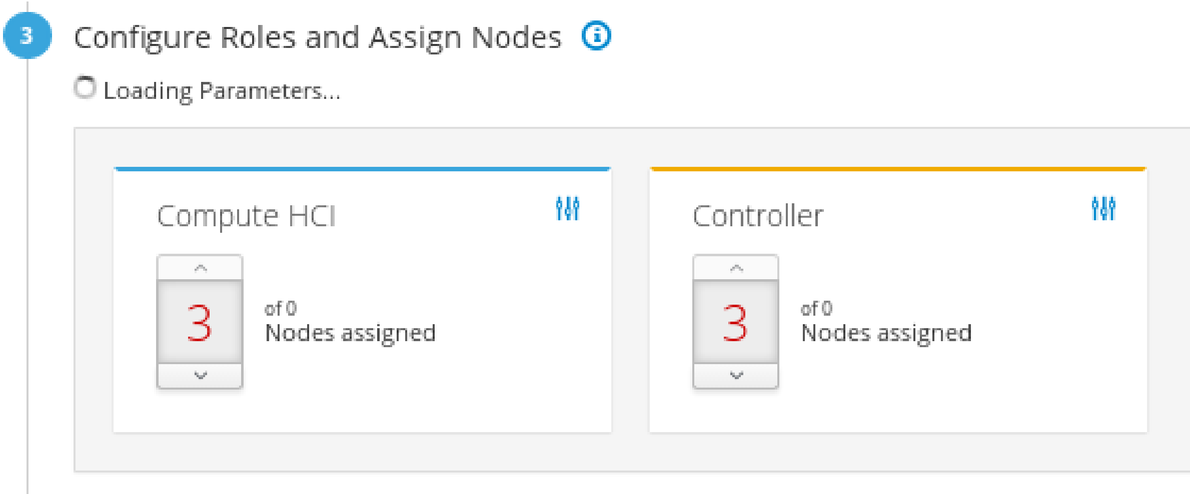


i. Click on the *Save And Close* button.

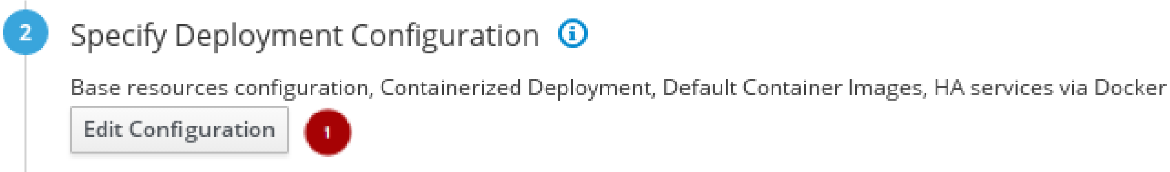


7. Assign the number of nodes needed in the environment for each role.

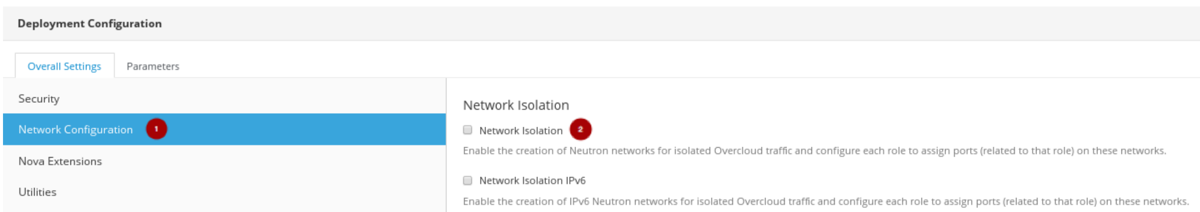
Figure 4.2. Example



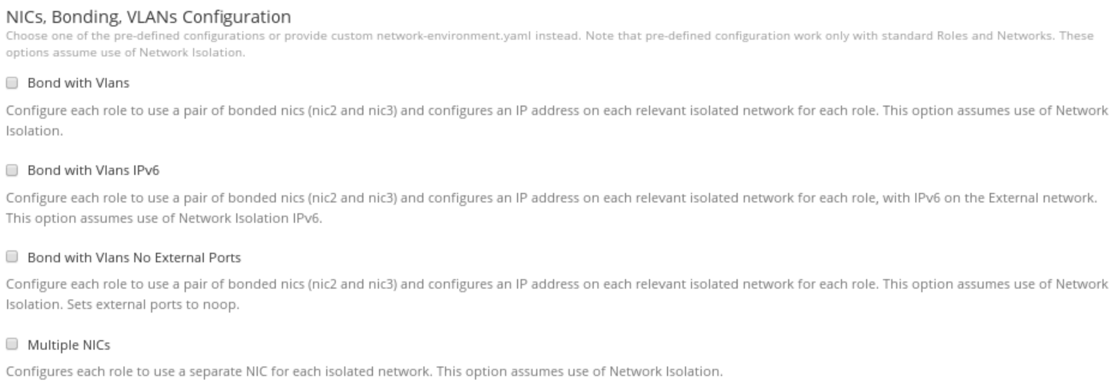
8. From the plan configuration page, click on the *Edit Configuration* button .



Edit the network configuration by clicking on the *Network Configuration* section , and select *Network Isolation* .



a. Select one of the NIC configuration templates or use a custom plan.

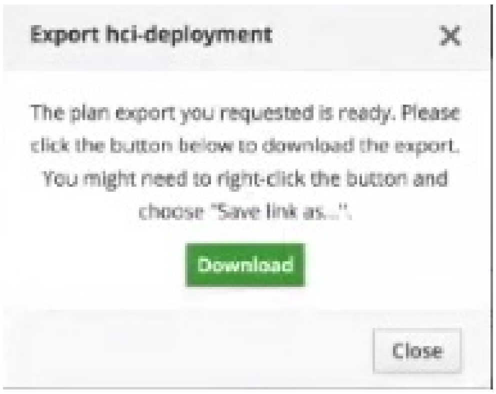


To customize the NICs in the environment, first you need to export the plan.

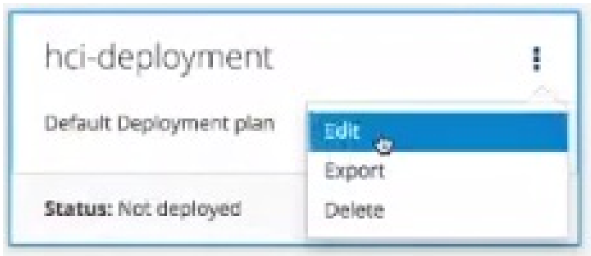
See Section 4.2, "Exporting an overcloud plan using the Red Hat OpenStack Platform Director" on how to export a plan.

i. Download the plan tarball file and make the necessary additions or modifications locally.

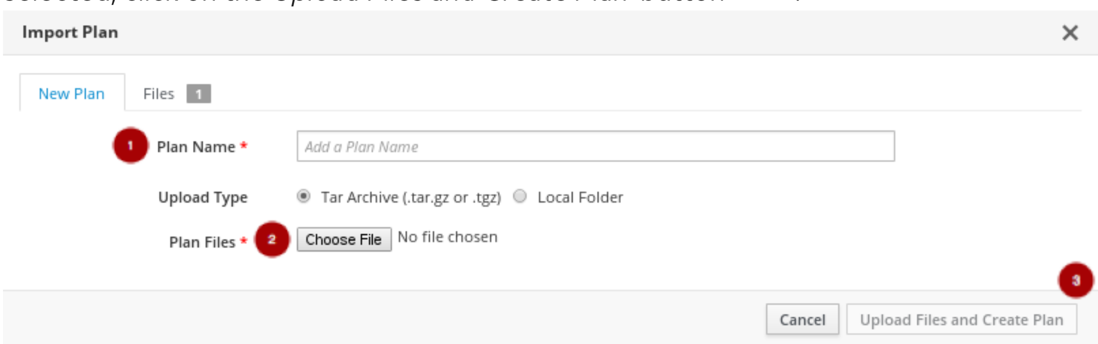
Example



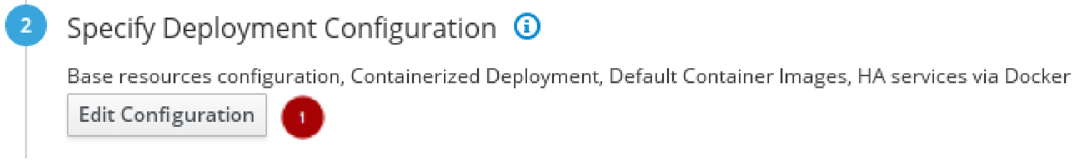
ii. After updating the plan tarball file, click the drop down menu and select *Edit*.



iii. Import the plan. Enter *Plan Name* 1 and click on the *Choose File* button 2 . Browse to the location of the tarball file, and select it for import. Once the file is selected, click on the *Upload Files and Create Plan* button 3 .



iv. Click on the *Edit Configuration* button.

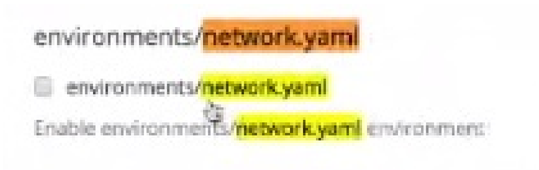


v. On the *Overall Settings* tab  , click on the *Other* section  .

vi. Select the *Others* section and include the custom templates.

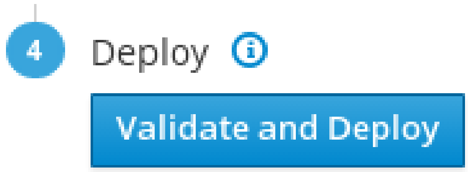
vii. Select any new or modified files from the file list.

Example



viii. Click on the *Parameters* tab and update any of the values accordingly.

9. Now, it is time to deploy the plan. From the plan configuration page, click on the *Validate and Deploy* button to deploy the overcloud plan.



10. Wait for the overcloud deployment to finish.

4.5. ADDITIONAL RESOURCES

- For more details on resource isolation, see Appendix E, *Tuning the Nova reserved memory and CPU allocation manually*.

CHAPTER 5. DEPLOYING RED HAT HYPERCONVERGED INFRASTRUCTURE FOR CLOUD USING THE COMMAND-LINE INTERFACE

As a technician, you can deploy and manage the Red Hat Hyperconverged Infrastructure for Cloud solution using the command-line interface.

5.1. PREREQUISITES

- Verify that all the requirements are met.
- Installing the undercloud

5.2. PREPARING THE NODES BEFORE DEPLOYING THE OVERCLOUD USING THE COMMAND-LINE INTERFACE

As a technician, before you can deploy the overcloud, the undercloud needs to understand the hardware being used in the environment.

NOTE

The Red Hat OpenStack Platform director (RHOSP-d) is also known as the undercloud.

5.2.1. Prerequisites

- Verify that all the requirements are met.
- Installing the undercloud

5.2.2. Registering and introspecting the hardware

The Red Hat OpenStack Platform director (RHOSP-d) runs an introspection process on each node and collects data about the node's hardware. This introspection data is stored on the RHOSP-d node, and is used for various purposes, such as benchmarking and root disk assignments.

Prerequisites

- Complete the software installation of the RHOSP-d node.
- The MAC addresses for the network interface cards (NICs).
- IPMI User name and password

Procedure

Do the following steps on the RHOSP-d node, as the **stack** user:

1. Create the **osd-compute** flavor:

```
[stack@director ~]$ openstack flavor create --id auto --ram 2048 --disk 40 --vcpus 2 osd-compute
[stack@director ~]$ openstack flavor set --property "capabilities:boot_option"="local" --
```

```
property "capabilities:profile"="osd-compute" osd-compute
```

2. Create and populate a host definition file for the Ironic service to manage the nodes.

   a. Create the **instackenv.json** host definition file:

   ```
   [stack@director ~]$ touch ~/instackenv.json
   ```

   b. Add a definition block for each node between the **nodes** stanza square brackets (**{"nodes": []}**), using this template:

   ```
   {
     "pm_password": "IPMI_USER_PASSWORD",
     "name": "NODE_NAME",
     "pm_user": "IPMI_USER_NAME",
     "pm_addr": "IPMI_IP_ADDR",
     "pm_type": "pxe_ipmitool",
     "mac": [
            "NIC_MAC_ADDR"
           ],
     "arch": "x86_64",
     "capabilities": "node:_NODE_ROLE-INSTANCE_NUM_,boot_option:local"
   },
   ```

   Replace...

   - *IPMI_USER_PASSWORD* with the IPMI password.
   - *NODE_NAME* with a descriptive name of the node. This is an optional parameter.
   - *IPMI_USER_NAME* with the IPMI user name that has access to power the node on or off.
   - *IPMI_IP_ADDR* with the IPMI IP address.
   - *NIC_MAC_ADDR* with the network card MAC address handling the PXE boot.
   - *NODE_ROLE-INSTANCE_NUM* with the node's role, along with a node number. This solution uses two roles: **control** and **osd-compute**.

     Example

     ```
     {
       "nodes": [
          {
             "pm_password": "AbC1234",
             "name": "m630_slot1",
             "pm_user": "ipmiadmin",
             "pm_addr": "10.19.143.61",
             "pm_type": "pxe_ipmitool",
             "mac": [
                "c8:1f:66:65:33:41"
             ],
             "arch": "x86_64",
              "capabilities": "node:control-0,boot_option:local"
          },
     ```

```
    {
      "pm_password": "AbC1234",
      "name": "m630_slot2",
      "pm_user": "ipmiadmin",
      "pm_addr": "10.19.143.62",
      "pm_type": "pxe_ipmitool",
      "mac": [
        "c8:1f:66:65:33:42"
      ],
      "arch": "x86_64",
       "capabilities": "node:osd-compute-0,boot_option:local"
    },
    ... Continue adding node definition blocks for each node in the initial
deployment here.
  ]
}
```

NOTE

The **osd-compute** role is a custom role that is created in a later step. To predictably control node placement, add these nodes in order. For example:

```
[stack@director ~]$ grep capabilities ~/instackenv.json
  "capabilities": "node:control-0,boot_option:local"
  "capabilities": "node:control-1,boot_option:local"
  "capabilities": "node:control-2,boot_option:local"
  "capabilities": "node:osd-compute-0,boot_option:local"
  "capabilities": "node:osd-compute-1,boot_option:local"
  "capabilities": "node:osd-compute-2,boot_option:local"
```

3. Import the nodes into the Ironic database:

```
[stack@director ~]$ openstack baremetal import ~/instackenv.json
```

   a. Verify that the **openstack baremetal import** command populated the Ironic database with all the nodes:

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

4. Assign the bare metal boot kernel and RAMdisk images to all the nodes:

```
[stack@director ~]$ openstack baremetal configure boot
```

5. To start the nodes, collect their hardware data and store the information in the Ironic database, execute the following:

```
[stack@director ~]$ openstack baremetal introspection bulk start
```

> **NOTE**
>
> Bulk introspection can take a long time to complete based on the number of nodes imported. Setting the **inspection_runbench** value to **false** in **~/undercloud.conf** file will speed up the bulk introspection process, but it will not collect the **sysbench** and **fio** benchmark data will not be collected, which can be useful data for the RHOSP-d.

a. Verify that the introspection process completes without errors for all the nodes:

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

Additional Resources

- For more information on assigning node identification parameters, see the Controlling Node Placement chapter of the RHOSP Advanced Overcloud Customization Guide.

5.2.3. Setting the root device

The Red Hat OpenStack Platform director (RHOSP-d) must identify the root disk to provision the nodes. By default Ironic will image the first block device, typically this block device is **/dev/sda**. Follow this procedure to change the root disk device according to the disk configuration of the Compute/OSD nodes.

This procedure will use the following Compute/OSD node disk configuration as an example:

- **OSD** : 12 x 1TB SAS disks presented as **/dev/[sda, sdb, …, sdl]** block devices
- **OSD Journal** : 3 x 400GB SATA SSD disks presented as **/dev/[sdm, sdn, sdo]** block devices
- **Operating System** : 2 x 250GB SAS disks configured in RAID1 presented as **/dev/sdp** block device

Since an OSD will use **/dev/sda**, Ironic will use **/dev/sdp**, the RAID 1 disk, as the root disk instead. During the hardware introspection process, Ironic stores the world-wide number (WWN) and size of each block device.

Prerequisites

- Complete the hardware introspection procedure.

Procedure

Run one of the following commands on the RHOSP-d node.

1. Configure the root disk device to use the **smallest** root device:

```
[stack@director ~]$ openstack baremetal configure boot --root-device=smallest
```

or

1. Configure the root disk device to use the disk's **by-path** name:

```
[stack@director ~]$ openstack baremetal configure boot --root-device=disk/by-path/pci-0000:00:1f.1-scsi-0:0:0:0
```

> **NOTE**
>
> Ironic will apply this root device directive to all nodes within Ironic's database.

1. Verify the correct root disk device was set:

```
openstack baremetal introspection data save NODE_NAME_or_UUID | jq .
```

Replace...

NODE_NAME_or_UUID with the host name or UUID of the node.

Additional Resources

- For more information on Defining the Root Disk for Nodes section in the RHOSP Director Installation and Usage Guide.

5.2.4. Verifying that Ironic's disk cleaning is working

To verify if Ironic's disk cleaning feature is working, you can toggle the node's state, then observe if the node's state goes into a cleaning state.

Prerequisites

- Installing the undercloud.

Procedure

1. Set the node's state to manage:

```
openstack baremetal node manage $NODE_NAME
```

Example

```
[stack@director ~]$ openstack baremetal node manage osdcompute-0
```

2. Set the node's state to provide:

```
openstack baremetal node provide NODE_NAME
```

Example

```
[stack@director ~]$ openstack baremetal node provide osdcompute-0
```

3. Check the node status:

```
openstack node list
```

5.2.5. Additional Resources

- For more information, see the RHOSP-d Installation and Usage Guide.

5.3. CONFIGURING A CONTAINER IMAGE SOURCE

As a technician, you can containerize the overcloud, but this first requires access to a registry with the required container images. Here you can find information on how to prepare the registry and the overcloud configuration to use container images for Red Hat OpenStack Platform.

There are several methods for configuring the overcloud to use a registry, based on the use case.

5.3.1. Registry methods

Red Hat Hyperconverged Infrastructure for Cloud supports the following registry types, choose one of the following methods:

Remote Registry

The overcloud pulls container images directly from **registry.access.redhat.com**. This method is the easiest for generating the initial configuration. However, each overcloud node pulls each image directly from the Red Hat Container Catalog, which can cause network congestion and slower deployment. In addition, all overcloud nodes require internet access to the Red Hat Container Catalog.

Local Registry

Create a local registry on the undercloud, synchronize the images from **registry.access.redhat.com**, and the overcloud pulls the container images from the undercloud. This method allows you to store a registry internally, which can speed up the deployment and decrease network congestion. However, the undercloud only acts as a basic registry and provides limited life cycle management for container images.

5.3.2. Including additional container images for Red Hat OpenStack Platform services

The Red Hat Hyperconverged Infrastructure for Cloud uses additional services besides the core Red Hat OpenStack Platform services. These additional services require additional container images, and you enable these services with their corresponding environment file. These environment files enable the composable containerized services in the overcloud and the director needs to know these services are enabled to prepare their images.

Prerequisites

- A running undercloud.

Procedure

1. As the **stack** user, on the undercloud node, using the **openstack overcloud container image prepare** command to include the additional services.
    a. Include the following environment file using the **-e** option:
        - Ceph Storage Cluster : **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml**
    b. Include the following **--set** options for Red Hat Ceph Storage:

        --set ceph_namespace
        Defines the namespace for the Red Hat Ceph Storage container image.

        --set ceph_image

Defines the name of the Red Hat Ceph Storage container image. Use image name: **rhceph-3-rhel7**.

--set ceph_tag

Defines the tag to use for the Red Hat Ceph Storage container image. When **--tag-from-label** is specified, the versioned tag is discovered starting from this tag.

2. Run the image prepare command:

Example

```
[stack@director ~]$ openstack overcloud container image prepare \
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  --set ceph_namespace=registry.access.redhat.com/rhceph \
  --set ceph_image=rhceph-3-rhel7 \
  --tag-from-label {version}-{release} \
  ...
```

NOTE

These options are passed in addition to any other options (**...**) that need to be passed to the **openstack overcloud container image prepare** command.

5.3.3. Using the Red Hat registry as a remote registry source

Red Hat hosts the overcloud container images on **registry.access.redhat.com**. Pulling the images from a remote registry is the simplest method because the registry is already setup and all you require is the URL and namespace of the image you aim to pull.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.
- Access to the Internet.

Procedure

1. To pull the images directly from **registry.access.redhat.com** in the overcloud deployment, an environment file is required to specify the image parameters. The following command automatically creates this environment file:

```
(undercloud) [stack@director ~]$ openstack overcloud container image prepare \
  --namespace=registry.access.redhat.com/rhosp13 \
  --prefix=openstack- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml
```

NOTE

Use the **-e** option to include any environment files for optional services.

2. This creates an **overcloud_images.yaml** environment file, which contains image locations, on the undercloud. Include this file with all future upgrade and deployment operations.

Additional Resources

- For more details, see the *Including additional container images for Red Hat OpenStack Platform services* section in the *Red Hat Hyperconverged Infrastructure for Cloud Deployment Guide* .

5.3.4. Using the undercloud as a local registry

You can configure a local registry on the undercloud to store overcloud container images. This method involves the following:

- The director pulls each image from the **registry.access.redhat.com**.
- The director creates the overcloud.
  - During the overcloud creation, the nodes pull the relevant images from the undercloud.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud environment.
- Access to the Internet.

Procedure

1. Create a template to pull the images to the local registry:

```
(undercloud) [stack@director ~]$ openstack overcloud container image prepare \
  --namespace=registry.access.redhat.com/rhosp13 \
  --prefix=openstack- \
  --tag-from-label {version}-{release} \
  --output-images-file /home/stack/local_registry_images.yaml
```

- Use the **-e** option to include any environment files for optional services.

NOTE

This version of the **openstack overcloud container image prepare** command targets the registry on the **registry.access.redhat.com** to generate an image list. It uses different values than the **openstack overcloud container image prepare** command used in a later step.

2. This creates a file called **local_registry_images.yaml** with the container image information. Pull the images using the **local_registry_images.yaml** file:

```
(undercloud) [stack@director ~]$ sudo openstack overcloud container image upload \
  --config-file  /home/stack/local_registry_images.yaml \
  --verbose
```

NOTE

The container images consume approximately 10 GB of disk space.

3. Find the namespace of the local images. The namespace uses the following pattern:

```
<REGISTRY_IP_ADDRESS>:8787/rhosp13
```

   Use the IP address of the undercloud, which you previously set with the **local_ip** parameter in the **undercloud.conf** file. Alternatively, you can also obtain the full namespace with the following command:

```
(undercloud) [stack@director ~]$ docker images | grep -v redhat.com | grep -o '^.*rhosp13' |
sort -u
```

4. Create a template for using the images in our local registry on the undercloud. For example:

```
(undercloud) [stack@director ~]$ openstack overcloud container image prepare \
  --namespace=192.168.24.1:8787/rhosp13 \
  --prefix=openstack- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml
```

   - Use the **-e** option to include any environment files for optional services.
   - If using Ceph Storage, include the additional parameters to define the Ceph Storage container image location: **--set ceph_namespace**, **--set ceph_image**, **--set ceph_tag**.

   NOTE

   This version of the **openstack overcloud container image prepare** command targets a Red Hat Satellite server. It uses different values than the **openstack overcloud container image prepare** command used in a previous step.

5. This creates an **overcloud_images.yaml** environment file, which contains image locations on the undercloud. Include this file with all future upgrade and deployment operations.

Additional Resources

- See the *Including additional container images for Red Hat OpenStack Platform services* section in the *Red Hat Hyperconverged Infrastructure for Cloud Deployment Guide* for more information.

Next Steps

- Prepare the overcloud for an upgrade.

5.3.5. Additional Resources

- See Section 4.2 in the *Red Hat OpenStack Platform Fast Forward Upgrades Guide* for more information.

5.4. ISOLATING RESOURCES AND TUNING THE OVERCLOUD USING THE COMMAND-LINE INTERFACE

Resource contention between Red Hat OpenStack Platform (RHOSP) and Red Hat Ceph Storage (RHCS) might cause a degradation of either service. Therefore, isolating system resources is important with the Red Hat Hyperconverged Infrastructure Cloud solution.

Likewise, tuning the overcloud is equally important for a more predictable performance outcome for a given workload.

To isolate resources and tune the overcloud, you will continue to refine the custom templates created previously.

5.4.1. Prerequisites

- Build the overcloud foundation by defining the overcloud.

5.4.2. Reserving CPU and memory resources for hyperconverged nodes

By default, the Nova Compute service parameters do not take into account the colocation of Ceph OSD services on the same node. Hyperconverged nodes need to be tuned in order to maintain stability and maximize the number of possible instances. Using a plan environment file allows you to set resource constraints for the Nova Compute service on hyperconverged nodes. Plan environment files define workflows, and the Red Hat OpenStack Platform director (RHOSP-d) executes the plan file with the OpenStack Workflow (Mistral) service.

The RHOSP-d also provides a default plan environment file specifically for configuring resource constraints on hyperconverged nodes:

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

Using the **-p** parameter invokes a plan environment file during the overcloud deployment.

This plan environment file will direct the OpenStack Workflow to:

1. Retrieve hardware introspection data.
2. Calculate optimal CPU and memory constraints for Compute on hyper-converged nodes based on that data.
3. Autogenerate the necessary parameters to configure those constraints.

In the **plan-environment-derived-params.yaml** plan environment file, the **hci_profile_config** option defines several CPU and memory allocation workload profiles. The **hci_profile** parameter sets which workload profile is enabled.

Here is the default **hci_profile**:

Default Example

```
hci_profile: default
  hci_profile_config:
    default:
      average_guest_memory_size_in_mb: 2048
      average_guest_cpu_utilization_percentage: 50
    many_small_vms:
      average_guest_memory_size_in_mb: 1024
      average_guest_cpu_utilization_percentage: 20
    few_large_vms:
```

```
      average_guest_memory_size_in_mb: 4096
      average_guest_cpu_utilization_percentage: 80
    nfv_default:
      average_guest_memory_size_in_mb: 8192
      average_guest_cpu_utilization_percentage: 90
```

In the above example, assumes that the average guest will use 2 GB of memory and 50% of their CPUs.

You can create a custom workload profile for the environment by adding a new profile to the **hci_profile_config** section. You can enable this custom workload profile by setting the **hci_profile** parameter to the profile's name.

Custom Example

```
hci_profile: my_workload
  hci_profile_config:
    default:
      average_guest_memory_size_in_mb: 2048
      average_guest_cpu_utilization_percentage: 50
    many_small_vms:
      average_guest_memory_size_in_mb: 1024
      average_guest_cpu_utilization_percentage: 20
    few_large_vms:
      average_guest_memory_size_in_mb: 4096
      average_guest_cpu_utilization_percentage: 80
    nfv_default:
      average_guest_memory_size_in_mb: 8192
      average_guest_cpu_utilization_percentage: 90
    my_workload:
      average_guest_memory_size_in_mb: 131072
      average_guest_cpu_utilization_percentage: 100
```

The **my_workload** profile assumes that the average guest will use 128 GB of RAM and 100% of the CPUs allocated to the guest.

Additional Resources

- See the Red Hat OpenStack Platform Hyper-converged Infrastructure Guide for more information.

5.4.3. Reserving CPU resources for Ceph

With hyperconverged deployments there can be contention between the Nova compute and Ceph processes for CPU resources. By default **ceph-ansible** will limit each OSD to one vCPU by using the **--cpu-quota** option on the **docker run** command. The **ceph_osd_docker_cpu_limit** option overrides this default limit, allowing you to use more vCPUs for each Ceph OSD process, for example:

```
CephAnsibleExtraConfig:
  ceph_osd_docker_cpu_limit: 2
```

Red Hat recommends setting the **ceph_osd_docker_cpu_limit** value to **2** as a starting point, and then adjust this value based on the hardware being used and workload being ran on this hyperconverged environment. This configuration option can be set in the **~/templates/ceph.yaml** file.

IMPORTANT

Always test the workload before running it in a production environment.

Additional Resources

- See the *Setting the Red Hat Ceph Storage parameters* section for more details on the **~/templates/ceph.yaml** file.
- See the *Recommended minimum hardware for containerized Ceph clusters* secton in the *Red Hat Ceph Storage Hardware Selection Guide* for more information.
- See the *Setting Dedicated Resources for Collocated Daemons* in the *Red Hat Ceph Storage Container Guide* for more information.

5.4.4. Reserving memory resources for Ceph

With hyperconverged deployments there can be contention between the Nova compute and Ceph processes for memory resources. Deployments of the Red Hat Hyperconverged Infrastructure for Cloud solution will use **ceph-ansible** to automatically tune Ceph's memory settings to reduce memory contention between collocated processes. The BlueStore object store is the recommended backend for hyperconverged deployments because of its better memory-handling features.

The **ceph_osd_docker_memory_limit** option is automatically set to the maximum memory size of the node as discovered by Ansible, regardless of the Ceph object store backend used, either FileStore or BlueStore.

WARNING

Red Hat recommends not overriding the **ceph_osd_docker_memory_limit** option.

The **osd_memory_target** option is the preferred way to reduce memory growth by the Ceph OSD processes. The **osd_memory_target** option is automatically set if the **is_hci** option is set to **true**, for example:

```
CephAnsibleExtraConfig:
  is_hci: true
```

These configuration options can be set in the **~/templates/ceph.yaml** file.

NOTE

The **osd_memory_target** option was introduced with the BlueStore object store feature starting with Red Hat Ceph Storage 3.2.

Additional Resources

- See the *Setting the Red Hat Ceph Storage parameters* section for more details on the **~/templates/ceph.yaml** file.

- See the *Recommended minimum hardware for containerized Ceph clusters* secton in the *Red Hat Ceph Storage Hardware Selection Guide* for more information.

- See the *Setting Dedicated Resources for Collocated Daemons* in the *Red Hat Ceph Storage Container Guide* for more information.

5.4.5. Tuning the backfilling and recovery operations for Ceph

Ceph uses a backfilling and recovery process to rebalance the storage cluster, whenever an OSD is removed. This is done to keep multiple copies of the data, according to the placement group policy. These two operations use system resources, so when a Ceph storage cluster is under load, then Ceph's performance will drop as Ceph diverts resources to the backfill and recovery process. To maintain acceptable performance of the Ceph storage when an OSD is removed, then reduce the priority of backfill and recovery operations. The trade off for reducing the priority is that there are less data replicas for a longer period of time, and putting the data at a slightly greater risk.

The three variables to modify are:

osd_recovery_max_active

The number of active recovery requests per OSD at one time. More requests will accelerate recovery, but the requests place an increased load on the cluster.

osd_max_backfills

The maximum number of backfills allowed to or from a single OSD.

osd_recovery_op_priority

The priority set for recovery operations. It is relative to osd client op priority.

Since the **osd_recovery_max_active** and **osd_max_backfills** parameters are set to the correct values already, there is no need to add them to the **ceph.yaml** file. If you want to overwrite the default values of **3** and **1** respectively, then add them to the **ceph.yaml** file.

Additional Resources

- For more information on the OSD configurable parameters, see the Red Hat Ceph Storage Configuration Guide.

5.4.6. Additional Resources

- See Table 5.2 Deployment Parameters in the Red Hat OpenStack Platform 10 Director Installation and Usage Guide for more information on the overcloud parameters.

- See Customizing Virtual Machine Settings for more information.

- See Section 5.6.4, "Running the deploy command" for details on running the **openstack overcloud deploy** command.

- For mapping Ceph OSDs to a disk layout on non-homogeneous nodes, see Mapping the Disk Layout to Non-Homogeneous Ceph Storage Nodes in the *Deploying an Overcloud with Containerized Red Hat Ceph* guide.

5.5. DEFINING THE OVERCLOUD USING THE COMMAND-LINE INTERFACE

As a technician, you can create a customizable set of TripleO Heat templates which defines the overcloud.

5.5.1. Prerequisites

- Verify that all the requirements are met.
- Deploy the Red Hat OpenStack Platform director, also known as the undercloud.

The high-level steps for defining the Red Hat Hyperconverged Infrastructure for Cloud overcloud:

1. Creating a Directory for Custom Templates
2. Configuring the Overcloud Networks
3. Creating the Controller and ComputeHCI Roles
4. Configuring Red Hat Ceph Storage for the overcloud
5. Configuring the Overcloud Node Profile Layouts

5.5.2. Creating a directory for the custom templates

The installation of the Red Hat OpenStack Platform director (RHOSP-d) creates a set of TripleO Heat templates. These TripleO Heat templates are located in the **/usr/share/openstack-tripleo-heat-templates/** directory. Red Hat recommends copying these templates before customizing them.

Prerequisites

- Deploy the undercloud.

Procedure

Do the following step on the command-line interface of the RHOSP-d node.

1. Create new directories for the custom templates:

```
[stack@director ~]$ mkdir -p ~/templates/nic-configs
```

5.5.3. Configuring the overcloud networks

This procedure will customize the network configuration files for isolated networks and assigning them to the Red Hat OpenStack Platform (RHOSP) services.

Prerequisites

- Verify that all the network requirements are met.

Procedure

Do the following steps on the RHOSP director node, as the **stack** user.

1. Choose the Compute NIC configuration template applicable to the 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**

> NOTE
>
> See the **README.md** in each template's respective directory for details about the NIC configuration.

2. Create a new directory within the ~/templates/ directory:

```
[stack@director ~]$ touch ~/templates/nic-configs
```

3. Copy the chosen template to the **~/templates/nic-configs/** directory and rename it to **compute-hci.yaml**:

   Example

```
[stack@director ~]$ cp /usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/compute.yaml ~/templates/nic-configs/compute-hci.yaml
```

4. Add following definition, if it does note already exist, in the **parameters:** section of the **~/templates/nic-configs/compute-hci.yaml** file:

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

5. Map **StorageMgmtNetworkVlanID** to a specific NIC on each node. For example, if you chose to trunk VLANs to a single NIC (**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
use_dhcp: false
vlan_id: {get_param: StorageMgmtNetworkVlanID}
addresses:
 -
  ip_netmask: {get_param: StorageMgmtIpSubnet}
```

IMPORTANT

Red Hat recommends setting the **mtu** to **9000**, when mapping a NIC to **StorageMgmtNetworkVlanID**. This MTU setting provides measurable performance improvement to the performance of Red Hat Ceph Storage. For more details, see Configuring Jumbo Frames in the Red Hat OpenStack Platform Advanced Overcloud Customization guide.

6. Create a new file in the custom templates directory:

```
[stack@director ~]$ touch ~/templates/network.yaml
```

7. Open and edit the **network.yaml** file.

   a. Add the **resource_registry** section:

   ```
   resource_registry:
   ```

   b. Add the following two lines under the **resource_registry:** section:

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

   These two lines point the RHOSP services to the network configurations of the Controller/Monitor and Compute/OSD nodes respectively.

   c. Add the **parameter_defaults** section:

   ```
   parameter_defaults:
   ```

   d. Add the following default parameters for the Neutron bridge mappings for the tenant network:

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

   This defines the bridge mappings assigned to the logical networks and enables the tenants to use **vxlan**.

   e. The two TripleO Heat templates referenced in step 2b requires parameters to define each network. Under the **parameter_defaults** section add the following lines:

   ```
   # Internal API used for private OpenStack Traffic
   InternalApiNetCidr: IP_ADDR_CIDR
   InternalApiAllocationPools: [{'start': 'IP_ADDR_START', 'end': 'IP_ADDR_END'}]
   InternalApiNetworkVlanID: VLAN_ID

   # Tenant Network Traffic - will be used for VXLAN over VLAN
   TenantNetCidr: IP_ADDR_CIDR
   TenantAllocationPools: [{'start': 'IP_ADDR_START', 'end': 'IP_ADDR_END'}]
   ```

```
TenantNetworkVlanID: VLAN_ID

# Public Storage Access - Nova/Glance <--> Ceph
StorageNetCidr: IP_ADDR_CIDR
StorageAllocationPools: [{'start': 'IP_ADDR_START', 'end': 'IP_ADDR_END'}]
StorageNetworkVlanID: VLAN_ID

# Private Storage Access - Ceph cluster/replication
StorageMgmtNetCidr: IP_ADDR_CIDR
StorageMgmtAllocationPools: [{'start': 'IP_ADDR_START', 'end': 'IP_ADDR_END'}]
StorageMgmtNetworkVlanID: VLAN_ID

# External Networking Access - Public API Access
ExternalNetCidr: IP_ADDR_CIDR

# Leave room for floating IPs in the External allocation pool (if required)
ExternalAllocationPools: [{'start': 'IP_ADDR_START', 'end': 'IP_ADDR_END'}]

# Set to the router gateway on the external network
ExternalInterfaceDefaultRoute: IP_ADDRESS

# Gateway router for the provisioning network (or undercloud IP)
ControlPlaneDefaultRoute: IP_ADDRESS

# The IP address of the EC2 metadata server, this is typically the IP of the undercloud
EC2MetadataIp: IP_ADDRESS

# Define the DNS servers (maximum 2) for the Overcloud nodes
DnsServers: ["DNS_SERVER_IP","DNS_SERVER_IP"]
```

Replace...

- *IP_ADDR_CIDR* with the appropriate IP address and net mask (CIDR).
- *IP_ADDR_START* with the appropriate starting IP address.
- *IP_ADDR_END* with the appropriate ending IP address.
- *IP_ADDRESS* with the appropriate IP address.
- *VLAN_ID* with the appropriate VLAN identification number for the corresponding network.
- *DNS_SERVER_IP* with the appropriate IP address for defining two DNS servers, separated by a comma (**,**).
  See the appendix for an example **network.yaml** file.

Additional Resources

- For more information on Isolating Networks, see the Red Hat OpenStack Platform Advance Overcloud Customization Guide.

5.5.4. Creating the *Controller* and *ComputeHCI* roles

The overcloud has five default roles: Controller, Compute, BlockStorage, ObjectStorage, and CephStorage. These roles contains a list of services. You can mix these services to create a custom deployable role.

Prerequisites

- Deploy the Red Hat OpenStack Platform director, also known as the undercloud.
- Create a Directory for Custom Templates.

Procedure

Do the following step on the Red Hat OpenStack Platform director node, as the **stack** user.

1. Generate a custom **roles_data_custom.yaml** file that includes the **Controller** and the **ComputeHCI**:

```
[stack@director ~]$ openstack overcloud roles generate -o ~/custom-templates/roles_data_custom.yaml Controller ComputeHCI
```

Additional Resources

- See the Deploying the overcloud using the command line in the *Red Hat Hyperconverged Infrastructure for Cloud Deployment Guide* for more information on using these custom roles.

5.5.5. Setting the Red Hat Ceph Storage parameters

This procedure defines what Red Hat Ceph Storage (RHCS) OSD parameters to use.

Prerequisites

- Deploy the Red Hat OpenStack Platform director, also known as the undercloud.
- Create a Directory for Custom Templates.

Procedure

Do the following steps on the Red Hat OpenStack Platform director node, as the **stack** user.

1. Open for editing the **~/templates/ceph.yaml** file.

   a. To use the BlueStore object store backend, update the following lines under the **CephAnsibleExtraConfig** section:

   ### Example

   ```
   CephAnsibleExtraConfig:
     osd_scenario: lvm
     osd_objectstore: bluestore
   ```

   b. Update the following options under the **parameter_defaults** section:

   ### Example

   ```
   parameter_defaults:
   ```

```
CephPoolDefaultSize: 3
CephPoolDefaultPgNum: NUM
CephAnsibleDisksConfig:
  osd_scenario: lvm
  osd_objectstore: bluestore
  devices:
    - /dev/sda
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
    - /dev/nvme0n1
    - /dev/sde
    - /dev/sdf
    - /dev/sdg
    - /dev/nvme1n1
CephAnsibleExtraConfig:
  osd_scenario: lvm
  osd_objectstore: bluestore
  ceph_osd_docker_cpu_limit: 2
  is_hci: true

CephConfigOverrides:
  osd_recovery_op_priority: 3
  osd_recovery_max_active: 3
  osd_max_backfills: 1
```

Replace...

NUM with the calculated values from the Ceph PG calculator.
For this example, the following Compute/OSD node disk configuration is being used:

- **OSD** : 12 x 1TB SAS disks presented as **/dev/[sda, sdb, ..., sdg]** block devices
- **OSD WAL and DB devices**: 2 x 400GB NVMe SSD disks presented as **/dev/[nvme0n1, nvme1n1]** block devices

Additional Resources

- For more details on tuning Ceph OSD parameters, see the Red Hat Ceph Storage Storage Strategies Guide.
- For more details on using the BlueStore object store, see the Red Hat Ceph Storage Administration Guide.
- For examples of the LVM scenario, see the *LVM simple* and *LVM advance* sections in the Red Hat Ceph Storage Installation Guide.

5.5.6. Configuring the overcloud nodes layout

The overcloud layout for the nodes defines, how many of these nodes to deploy based on the type, which pool of IP addresses to assign, and other parameters.

Prerequisites

- Deploy the Red Hat OpenStack Platform director, also known as the undercloud.

- Create a Directory for Custom Templates.

Procedure

Do the following steps on the Red Hat OpenStack Platform director node, as the **stack** user.

1. Create the **layout.yaml** file in the custom templates directory:

```
[stack@director ~]$ touch ~/templates/layout.yaml
```

2. Open the **layout.yaml** file for editing.

   a. Add the resource registry section by adding the following line:

```
resource_registry:
```

   b. Add the following lines under the **resource_registry** section for configuring the **Controller** and **ComputeHCI** roles to use a pool of IP addresses:

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

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

   c. Add a new section for the parameter defaults called **parameter_defaults** and include the following parameters underneath this section:

```
parameter_defaults:
  NtpServer: NTP_IP_ADDR
  ControllerHostnameFormat: 'controller-%index%'
  ComputeHCIHostnameFormat: 'compute-hci-%index%'
  ControllerCount: 3
  ComputeHCICount: 3
  OvercloudComputeFlavor: compute
  OvercloudComputeHCIFlavor: osd-compute
```

Replace...

NTP_IP_ADDR with the IP address of the NTP source. Time synchronization is very important!

Example

```
parameter_defaults:
  NtpServer: 10.5.26.10
  ControllerHostnameFormat: 'controller-%index%'
  ComputeHCIHostnameFormat: 'compute-hci-%index%'
  ControllerCount: 3
  ComputeHCICount: 3
  OvercloudComputeFlavor: compute
  OvercloudComputeHCIFlavor: osd-compute
```

The value of **3** for the **ControllerCount** and **ComputeHCICount** parameters means three Controller/Monitor nodes and three Compute/OSD nodes will be deployed.

d. Under the **parameter_defaults** section, add a two scheduler hints, one called **ControllerSchedulerHints** and the other called **ComputeHCISchedulerHints**. Under each scheduler hint, add the node name format for predictable node placement, as follows:

```
ControllerSchedulerHints:
  'capabilities:node': 'control-%index%'
ComputeHCISchedulerHints:
  'capabilities:node': 'osd-compute-%index%'
```

e. Under the **parameter_defaults** section, add the required IP addresses for each node profile, for example:

Example

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

ComputeHCIIPs:
  internal_api:
    - 192.168.2.203
    - 192.168.2.204
    - 192.168.2.205
  tenant:
    - 192.168.3.203
    - 192.168.3.204
    - 192.168.3.205
  storage:
```

```
    - 172.16.1.203
    - 172.16.1.204
    - 172.16.1.205
  storage_mgmt:
    - 172.16.2.203
    - 172.16.2.204
    - 172.16.2.205
```

From this example, node **control-0** would have the following IP addresses: **192.168.2.200**, **192.168.3.200**, **172.16.1.200**, and **172.16.2.200**.

5.5.7. Additional Resources

- The Red Hat OpenStack Platform Advanced Overcloud Customization Guide for more information.

5.6. DEPLOYING THE OVERCLOUD USING THE COMMAND-LINE INTERFACE

As a technician, you can deploy the overcloud nodes so the Nova Compute and the Ceph OSD services are colocated on the same node.

5.6.1. Prerequisites

- Deploy the undercloud.
- Define the overcloud.

5.6.2. Verifying the available nodes for Ironic

Before deploying the overcloud nodes, verify that the nodes are powered off and available.

> **WARNING**
>
> The nodes can not be in maintenance mode.

Prerequisites

- Having the **stack** user available on the Red Hat OpenStack Platform director node.

Procedure

1. Run the following command to verify all nodes are powered off, and available:

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

5.6.3. Configuring the controller for Pacemaker fencing

Isolating a node in a cluster so data corruption doesn't happen is called fencing. Fencing protects the integrity of cluster and cluster resources.

Prerequisites

- An IPMI user and password.
- Having the **stack** user available on the Red Hat OpenStack Platform director node.

Procedure

1. Generate the fencing Heat environment file:

```
[stack@director ~]$ openstack overcloud generate fencing --ipmi-lanplus instackenv.json --
output fencing.yaml
```

2. Include the **fencing.yaml** file with the **openstack overcloud deploy** command.

Additional Resources

- For more information, see the Deploying Red Hat Enterprise Linux OpenStack Platform 7 with Red Hat OpenStack Platform director.

5.6.4. Running the deploy command

After all the customization and tuning, it is time to deploy the overcloud.

> **NOTE**
>
> The deployment of the overcloud can take a long time to finish based on the sized of the deployment.

Prerequisites

- Preparing the nodes
- Configure a container image source
- Define the overcloud
- Isolating Resources and tuning
- Having the **stack** user available on the Red Hat OpenStack Platform director node.

Procedure

1. Run the following command:

```
[stack@director ~]$ time openstack overcloud deploy \
--templates /usr/share/openstack-tripleo-heat-templates \
--stack overcloud \
-p /usr/share/openstack-tripleo-heat-templates/plan-samples/plan-environment-derived-
params.yaml
-r /home/stack/templates/roles_data_custom.yaml \
```

```
-e /usr/share/openstack-tripleo-heat-templates/environments/docker.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/docker-ha.yaml \
-e /home/stack/templates/overcloud_images.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml
\
-e ~/templates/network.yaml \
-e ~/templates/ceph.yaml \
-e ~/templates/layout.yaml
-e /home/stack/fencing.yaml
```

Command Details

- The **time** command is used to tell you how long the deployment takes.
- The **openstack overcloud deploy** command does the actual deployment.
- Replace **$NTP_IP_ADDR** with the IP address of the NTP source. Time synchronization is very important!
- The **--templates** argument uses the default directory (**/usr/share/openstack-tripleo-heat-templates/**) containing the TripleO Heat templates to deploy.
- The **-p** argument points to the plan environment file for HCI deployments. See the Reserving CPU and memory resources for hyperconverged nodes section for more details.
- The **-r** argument points to the roles file and overrides the default **role_data.yaml** file.
- The **-e** argument points to an explicit template file to use during the deployment.
- The **puppet-pacemaker.yaml** file configures the controller node services in a highly available pacemaker cluster.
- The **storage-environment.yaml** file configures Ceph as a storage backend, whose **parameter_defaults** are passed by the custom template, **ceph.yaml**.
- The **network-isolation.yaml** file configures network isolation for different services, whose parameters are passed by the custom template, **network.yaml**. This file will be created automatically when the deployment starts.
- The **network.yaml** file is explained in Configuring the overcloud networks section for more details.
- The **ceph.yaml** file is explained in Setting the Red Hat Ceph Storage parameters section for more details.
- The **compute.yaml** file is explained in Changing Nova reserved memory and CPU allocations section for more details.
- The **layout.yaml** file is explained in Configuring the overcloud node profile layouts section for more details.
- The **fencing.yaml** file is explained in Configuring the controller for Pacemaker fencing section for more details.

IMPORTANT

The order of the arguments matters. The custom template files will override the default template files.

NOTE

Optionally, add the **--rhel-reg**, **--reg-method**, **--reg-org** options, if you want to use the Red Hat OpenStack Platform director (RHOSP-d) node as a software repository for package installations.

2. Wait for the overcloud deployment to finish.

Additional Resources

- See Table 5.2 Deployment Parameters in the Red Hat OpenStack Platform 13 Director Installation and Usage Guide for more information on the overcloud parameters.

5.6.5. Verifying a successful overcloud deployment

It is important to verify if the overcloud deployment was successful.

Prerequisites

- Having the **stack** user available on the Red Hat OpenStack Platform director node.

Procedure

1. Watch the deployment process and look for failures:

```
[stack@director ~]$ heat resource-list -n5 overcloud | egrep -i 'fail|progress'
```

Example output from a successful overcloud deployment:

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

 Stack overcloud CREATE_COMPLETE

Started Mistral Workflow. Execution ID: aeca4d71-56b4-4c72-a980-022623487c05
/home/stack/.ssh/known_hosts updated.
Original contents retained as /home/stack/.ssh/known_hosts.old
Overcloud Endpoint: http://10.19.139.46:5000/v2.0
Overcloud Deployed
```

2. After the deployment finishes, view the IP addresses for the overcloud nodes:

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

CHAPTER 6. UPDATING THE RED HAT HYPERCONVERGED INFRASTRUCTURE FOR CLOUD SOLUTION TO THE LATEST VERSIONS

As a technician, you can update the Red Hat Hyperconverged Infrastructure for Cloud solution to the latest versions of Red Hat OpenStack Platform 13, and Red Hat Ceph Storage 3. To update the Red Hat Hyperconverged Infrastructure for Cloud software sets to the latest versions, follow the instructions in the Red Hat OpenStack Platform 13 documentation:

- Keeping Red Hat OpenStack Platform Updated

APPENDIX A. RED HAT HYPERCONVERGED INFRASTRUCTURE FOR CLOUD REQUIRED REPOSITORIES

Table A.1. Required repositories

Name	Repository	Description of Requirement
Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rpms**	Base operating system repository.
Red Hat Enterprise Linux 7 Server - Extras (RPMs)	**rhel-7-server-extras-rpms**	Contains Red Hat OpenStack Platform dependencies.
Red Hat Enterprise Linux 7 Server - RH Common (RPMs)	**rhel-7-server-rh-common-rpms**	Contains tools for deploying and configuring Red Hat OpenStack Platform.
Red Hat Enterprise Linux High Availability (for RHEL 7 Server) (RPMs)	**rhel-ha-for-rhel-7-server-rpms**	High availability tools for Red Hat Enterprise Linux. Used for Controller node high availability.
Red Hat Enterprise Linux OpenStack Platform 13 for RHEL 7 (RPMs)	**rhel-7-server-openstack-13-rpms**	Core Red Hat OpenStack Platform repository. Also contains packages for Red Hat OpenStack Platform director.
Red Hat Ceph Storage 3 OSD for Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rhceph-3-osd-rpms**	Repository for RHCS Object Storage Daemons (OSDs). Enabled on Compute nodes.
Red Hat Ceph Storage 3 MON for Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rhceph-3-mon-rpms**	Repository for RHCS Monitor daemon. Enabled on Controller nodes.
Red Hat Ceph Storage 3 Tools for Red Hat Enterprise Linux 7 Workstation (RPMs)	**rhel-7-server-rhceph-3-tools-rpms**	Repository for RHCS tools and clients, such as the Ceph Object Gateway.

APPENDIX B. RED HAT HYPER-CONVERAGED INFRASTRUCTURE FOR CLOUD UNDERCLOUD CONFIGURATION PARAMETERS

local_ip

The IP address defined for the director's provisioning network. This is also the IP address the director uses for its DHCP and PXE boot services.

network_gateway

The gateway for the overcloud instances. This is the undercloud node, which forwards traffic to the external network.

undercloud_public_vip

The IP address defined for the director's Public API. Use an IP address on the provisioning network that does not conflict with any other IP addresses or address ranges. The director configuration attaches this IP address to its software bridge as a routed IP address, which uses the /32 netmask.

undercloud_admin_vip

The IP address defined for the director's Admin API. Use an IP address on the provisioning network that does not conflict with any other IP addresses or address ranges. The director configuration attaches this IP address to its software bridge as a routed IP address, which uses the /32 netmask.

local_interface

The chosen interface for the director's provisioning NIC. This is also the device the director uses for its DHCP and PXE boot services. The configuration script attaches this interface to a custom bridge defined with the **inspection_interface** parameter.

network_cidr

The network that the director uses to manage overcloud instances. This is the provisioning network, which the undercloud's neutron service manages.

masquerade_network

Defines the network that will masquerade for external access. This provides the provisioning network with a degree of network address translation (NAT), so that it has external access through the director.

dhcp_start

The start of the DHCP allocation range for overcloud nodes. Ensure this range contains enough IP addresses to allocate to all nodes.

dhcp_end

The end of the DHCP allocation range for overcloud nodes. Ensure this range contains enough IP addresses to allocate to all nodes.

inspection_interface

The bridge the director uses for node introspection. This is custom bridge that the director configuration creates. The **local_interface** attaches to this bridge. Leave this as the default, **br-ctlplane**.

inspection_iprange

A range of IP address that the director's introspection service uses during the PXE boot and provisioning process. Use comma-separated values to define the start and end of this range. Verify this range contains enough IP addresses for the nodes and does not conflict with the range for **dhcp_start** and **dhcp_end**.

inspection_extras

Defines whether to enable extra hardware collection during the inspection process. Requires **python-hardware** or **python-hardware-detect** package on the introspection image.

inspection_runbench

Runs a set of benchmarks during node introspection. Set to **true** to enable. This option is necessary if you intend to perform benchmark analysis when inspecting the hardware of registered nodes.

inspection_enable_uefi

Defines whether to support introspection of nodes with UEFI-only firmware.

APPENDIX C. RED HAT HYPERCONVERGED INFRASTRUCTURE FOR CLOUD - NOVA MEMORY AND CPU CALCULATOR SCRIPT SOURCE

This is the Python source code for the **nova_mem_cpu_calc.py** script.

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

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

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

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

if len(argv) < 5: usage("Too few command line params")
try:
  mem = int(argv[1])
  cores = int(argv[2])
  osds = int(argv[3])
  average_guest_size = int(argv[4])
  average_guest_util = float(argv[5])
```

```
except ValueError:
  usage("Non-integer input parameter")

average_guest_util_percent = 100 * average_guest_util

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

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

# display outputs including how to tune Nova reserved mem

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

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

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

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

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

APPENDIX D. RED HAT HYPERCONVERGED INFRASTRUCTURE FOR CLOUD - EXAMPLE NETWORK.YAML FILE

Example

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

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

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

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

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

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

  # External Networking Access - Public API Access

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

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

APPENDIX E. TUNING THE NOVA RESERVED MEMORY AND CPU ALLOCATION MANUALLY

Tuning the Nova environment for the planned workload can be a trial and error process. Red Hat recommends starting will a calculated base set of defaults and tune from there.

By tuning the **reserved_host_memory_mb** and **cpu_allocation_ratio** parameters, you can maximize the number of possible guests for the workload. Also, by fine tuning these values you can find the desired trade off between determinism and guest-hosting capacity for the workload.

Tuning Nova reserved memory

Nova's **reserved_host_memory_mb** parameter is the amount of memory, in megabytes (MB), to reserve for the node. Keep in mind, that on a hyper-converged Compute/OSD nodes, the memory must be shared between the two services, as not to starve either service of their required resources.

The following is an example of how to determine the **reserved_host_memory_mb** value for a hyper-converged node. Given a node with 256GB of RAM and 10 OSDs, assuming that each OSD consumes 3GB of RAM, that is 30GB of RAM for Ceph, and leaving 226GB of RAM for Nova Compute. If the average guest each uses 2GB of RAM, then the overall system could host 113 guest machines. However, there is the additional overhead for each guest machine running on the hypervisor that you must account for. Assuming this overhead is 500MB, the maximum number of 2GB guest machines that could be ran would be approximately 90.

Here is the mathematical formulas:

Approximate Number of Guest Machines = (*Memory Available for Nova in GB* / (*Memory per Guest Machine in GB* + *Hypervisor Memory Overhead in GB*))

Example

90.4 = (226 / (2 + .5))

Given the approximate number of guest machines and the number of OSDs, the amount of memory to reserve for Nova can be calculated.

Nova Reserved Memory in MB = 1000 * ((*OSD Memory Size in GB* * *Number of OSDs*) + (*Approximate Number of Guest Machines* * *Hypervisor Memory Overhead in GB*))

Example

75000 = 1000 * ((3 * 10) + (90 * .5))

Thus, **reserved_host_memory_mb** would equal **75000**. The parameter value must be in megabytes (MB).

Tuning CPU allocation ratio

Nova's **cpu_allocation_ratio** parameter is used by the Nova scheduler when choosing which compute nodes to run the guest machines. If the ratio of guest machines to compute nodes is 16:1, and the number of cores (vCPUs) on a node is 56, then the Nova scheduler may schedule enough guests to consume 896 cores, before it considers the node is unable to handle any more guest machines. The reason is because, the Nova scheduler does not take into account the CPU needs of the Ceph OSD services running on the same node as the Nova scheduler. Modifying the **cpu_allocation_ratio** parameter allows Ceph to have the CPU resources it needs to operate effectively without those CPU resources being given to Nova Compute.

The following is an example of how to determine the **cpu_allocation_ratio** value for a hyper-converged node. Given a node has 56 cores and 10 OSDs, and assuming that one core is used by each OSD, that leaves 46 cores for Nova. If each guest machine utilizes 100% of its core, then the number of available cores for guest machines is divided by the total number of cores on the node. In this scenario, the **cpu_allocation_ratio** value is **0.821429**.

However, because guest machines do not typically utilize 100% of their cores, the ratio must take into account an anticipated utilization percentage when determining the number of cores per guest machine. In a scenario, where you only anticipate on average, 10% core utilization per guest machine, the **cpu_allocation_ratio** value must be **8.214286**.

Here is the mathematical formulas:

1. *Number of Non Ceph Cores* = *Total Number of Cores on the Node* - (*Number of Cores per OSD* * *Number of OSDs*)
2. *Number of Guest Machine vCPUs* = *Number of Non Ceph Cores* / *Average Guest Machine CPU Utilization*
3. *CPU Allocation Ratio* = *Number of Guest Machine vCPUs* / *Total Number of Cores on the Node*

Example

1. 46 = 56 - (1 * 10)
2. 460 = 46 / .1
3. 8.214286 = 460 / 56

Nova memory and CPU calculator

Red Hat provides a calculator script to do all these calculations for you. The script name is **nova_mem_cpu_calc.py**, and takes 5 input parameters:

```
nova_mem_cpu_calc.py TOTAL_NODE_RAM_GB TOTAL_NODE_CORES
NUM_OSDs_PER_NODE AVG_GUEST_MEM_SIZE_GB AVG_GUEST_CPU_UTIL
```

Replace...

- *TOTAL_NODE_RAM_GB* with the total size of RAM in GB on the node.
- *TOTAL_NODE_CORES* with the total number of cores on the node.
- *NUM_OSDs_PER_NODE* with the number of Ceph OSDs per node.
- *AVG_GUEST_MEM_SIZE_GB* with the average memory size in GB for the guest machine.
- *AVG_GUEST_CPU_UTIL* with the average CPU utilization, expressed as a decimal, for the guest machine.

Example

```
[stack@director ~]$ ./nova_mem_cpu_calc.py 256 56 10 2 0.1
```

Additional Resources

- See the appendix for the full source code of the **nova_mem_cpu_calc.py** script.

APPENDIX F. CHANGING NOVA RESERVED MEMORY AND CPU ALLOCATION MANUALLY

Creating a custom template to overwrite the **reserved_host_memory_mb** and **cpu_allocation_ratio** default values.

Prerequisites

- Deploy the Red Hat OpenStack Platform director (RHOSP-d), also known as the undercloud.
- Create a Directory for Custom Templates.

Procedure

Do the following steps on the RHOSP-d node, as the **stack** user.

1. Create the **compute.yaml** file in the custom templates directory:

```
[stack@director ~]$ touch ~/templates/compute.yaml
```

2. Open the **compute.yaml** file for editing.

   a. Add the **reserved_host_memory** and **cpu_allocation_ratio** configuration parameters to the **ExtraConfig** section, under the parameter defaults section:

```
parameter_defaults:
  ExtraConfig:
    nova::compute::reserved_host_memory: MEMORY_SIZE_IN_MB
    nova::cpu_allocation_ratio: CPU_RATIO
```

Replace...

- *MEMORY_SIZE_IN_MB* with the memory size in megabytes (MB).
- *CPU_RATIO* with the ratio decimal value.

Example

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

NOTE

Red Hat OpenStack Platform director refers to the **reserved_host_memory_mb** parameter used by Nova as the **reserved_host_memory** parameter.