Red Hat Hyperconverged Infrastructure for Cloud 13

Deployment Guide

Deploying a Red Hat Hyperconverged Infrastructure for Cloud Solution

Last Updated: 2018-07-13

Red Hat Hyperconverged Infrastructure for Cloud 13 Deployment Guide

Deploying a Red Hat Hyperconverged Infrastructure for Cloud Solution

Legal Notice

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:

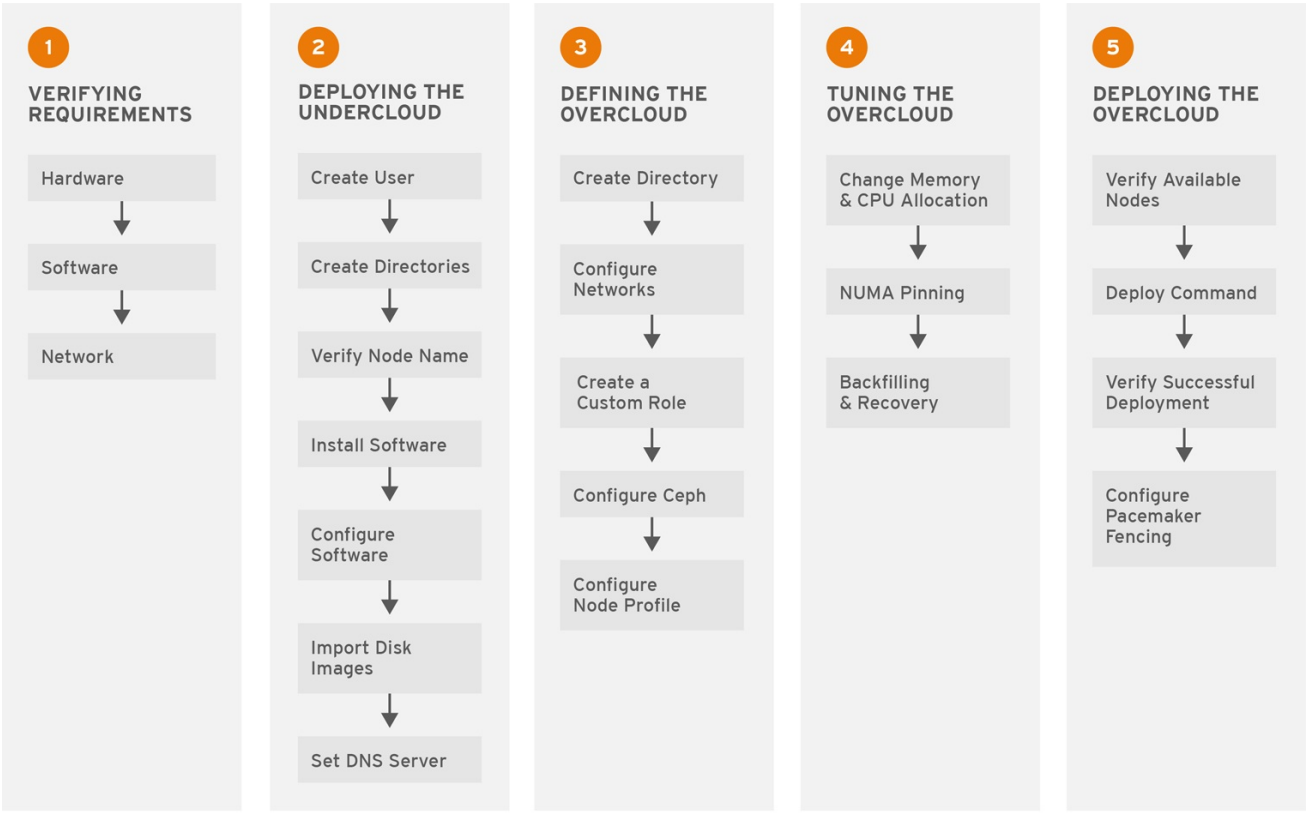
- Simplify the deployment of RHOSP and RHCS.
- Provide a more predictable performance experience.
- Achieve a lower cost of entry for RHOSP and RHCS by colocating their respective services on the same node.

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 Chapter 4, *Deploying Red Hat Hyperconverged Infrastructure for Cloud Using the Red Hat OpenStack Platform Director* for more details

OpenStack Platform Director for more details.

- See Chapter 5, *Deploying Red Hat Hyperconverged Infrastructure for Cloud Using the Command-line Interface* 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. VERIFYING 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 Section 2.4, “Verifying the Red Hat Hyperconverged Infrastructure for Cloud 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 42 nodes.

2.3. 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) Cloud solution requires specific software packages to be installed to function properly.

Prerequisites

- Have a valid Red Hat Hyperconverged Infrastructure for Cloud subscription.

Procedure

Do the following step on any node, as the **root** user.

1. Verify the available subscriptions:

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

Additional Resources

- See Appendix A, *Red Hat Hyperconverged Infrastructure for Cloud Required Repositories* for the required software repositories.

2.4. VERIFYING 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.

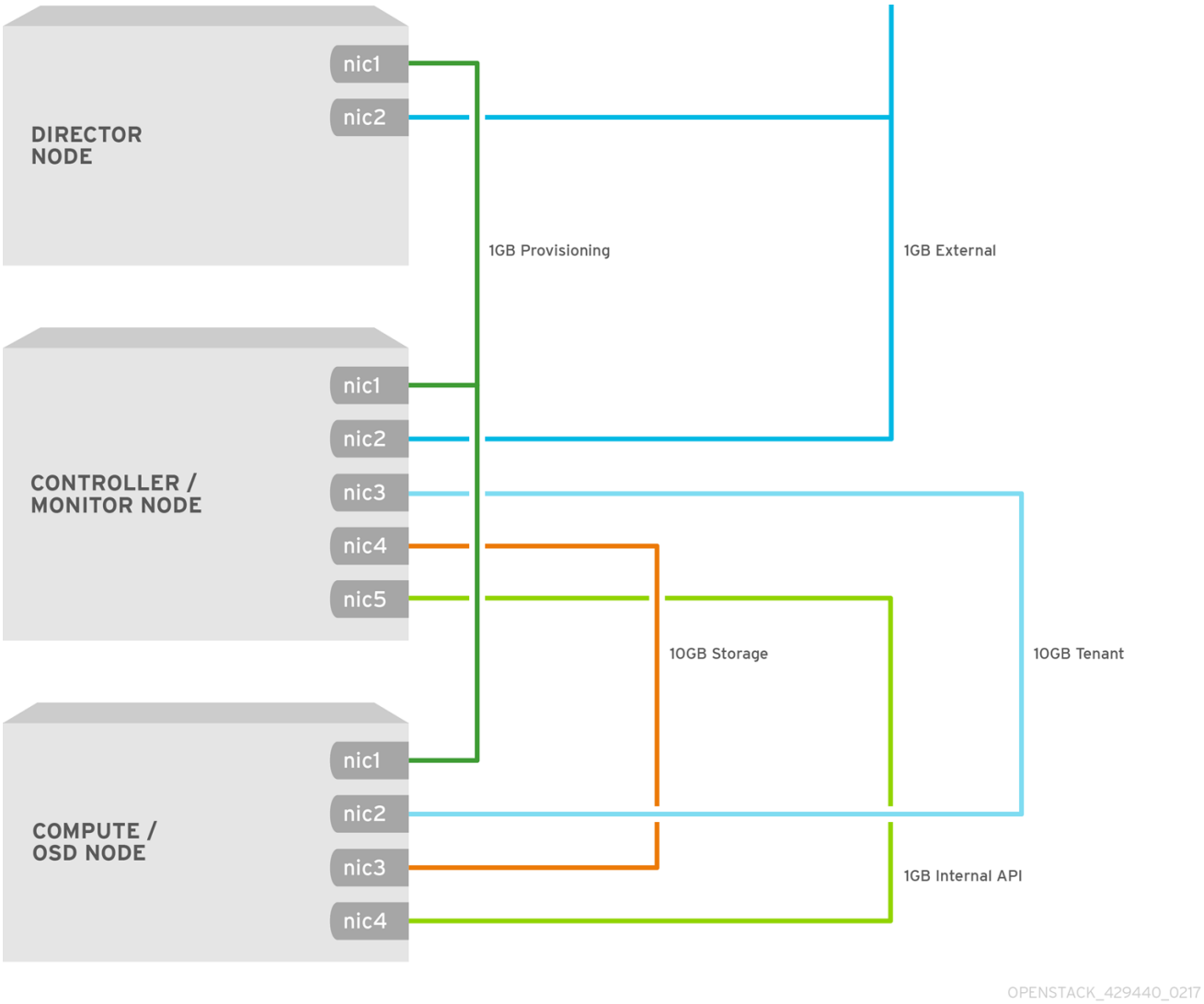
OpenStack Tenant Network

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.

2.5. 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

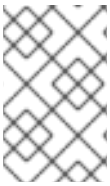
- Have a valid Red Hat Hyperconverged Infrastructure for Cloud subscription.
- Have access to Red Hat's software repositories through Red Hat's Content Delivery Network (CDN).

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. 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

- Install the undercloud

Procedure

1. Edit the **undercloud.conf** file, and add the following line:

```
clean_nodes = True
```

3.4. 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.

Prerequisites

- Have access to Red Hat's software repositories through Red Hat's Content Delivery Network (CDN).

Procedure

Summary of the 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

Do the following steps on the command-line interface of the RHOSP-d node:

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

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

   b. As **root**, set a password for **stack**. When prompted, enter the new password:

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

   c. As **root**, 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 ~/custom-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 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 Appendix B, *Red Hat Hyper-converged Infrastructure for Cloud 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 Section 3.2, "Understanding Ironic's Disk Cleaning Between Deployments" 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.

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, as a technician, 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.

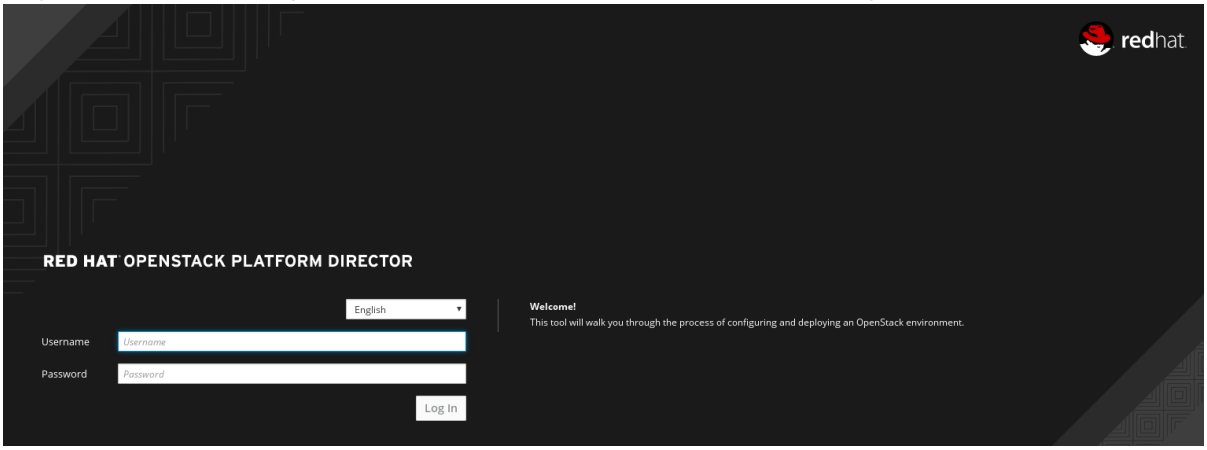
Procedure

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.

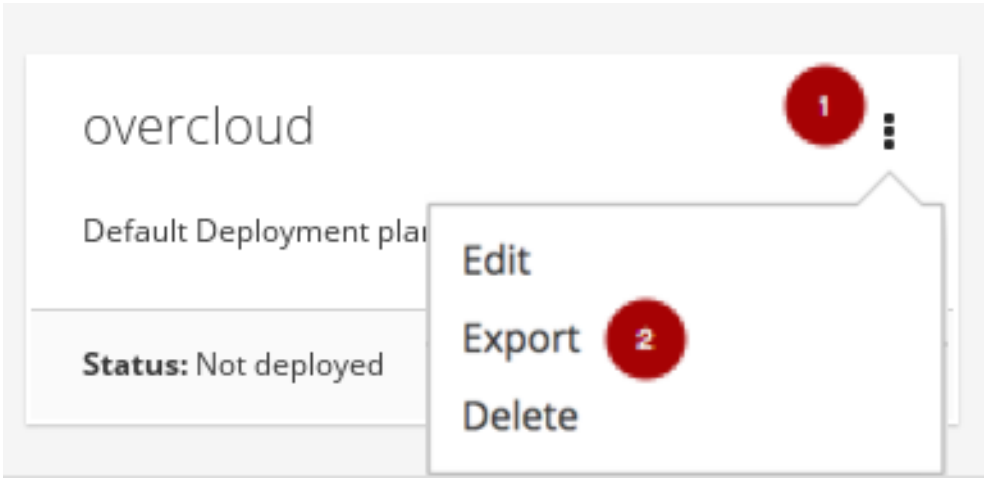


NOTE

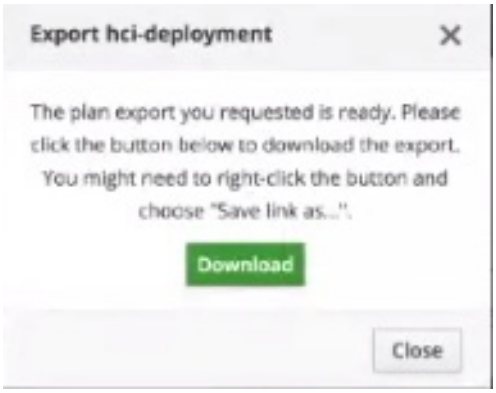
The default user name is **admin**. You can obtain the admin password by running the following command:

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

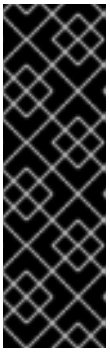
3. On the *Plans* tab, select the drop-down menu (1) from the *Overcloud* plan, and select *Export* (2).



4. Click on the *Download* button.



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

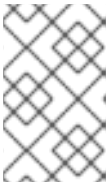


IMPORTANT

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.

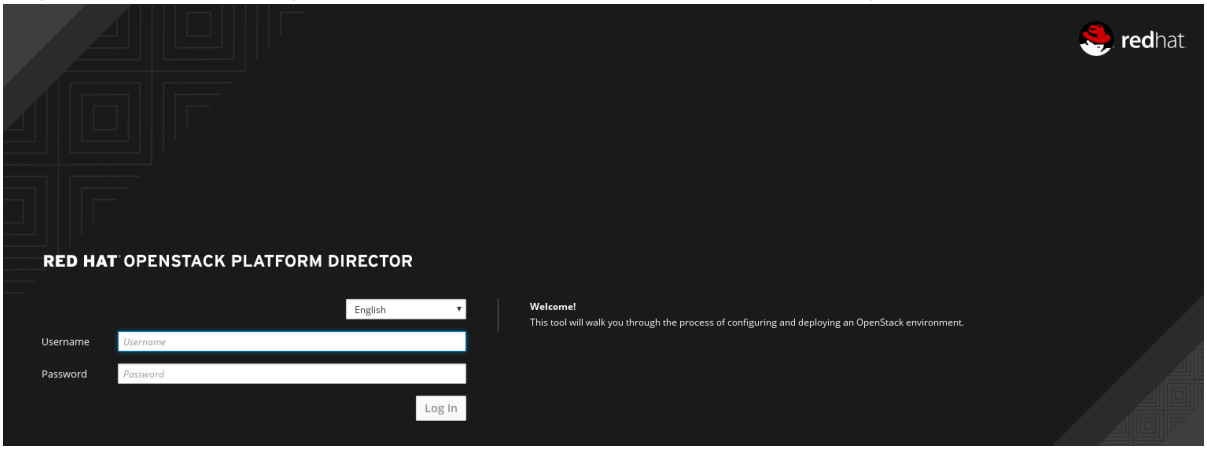
Procedure

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.



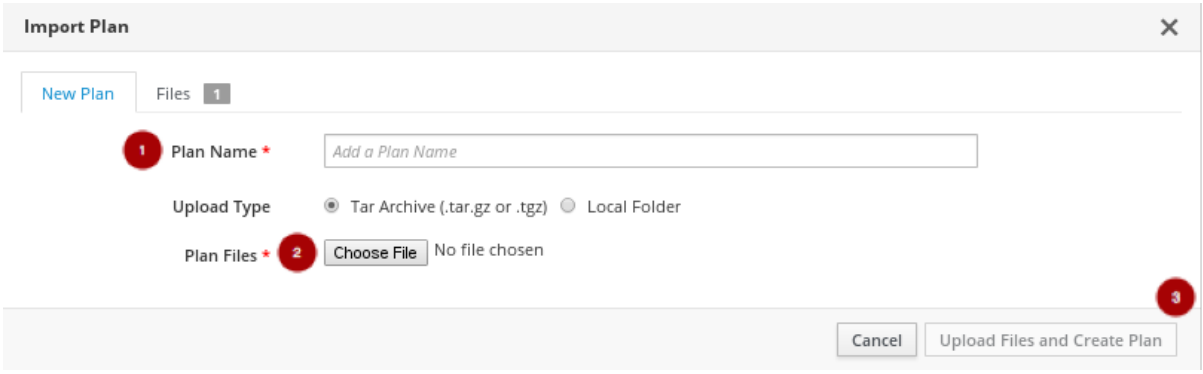
NOTE

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* (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).



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.

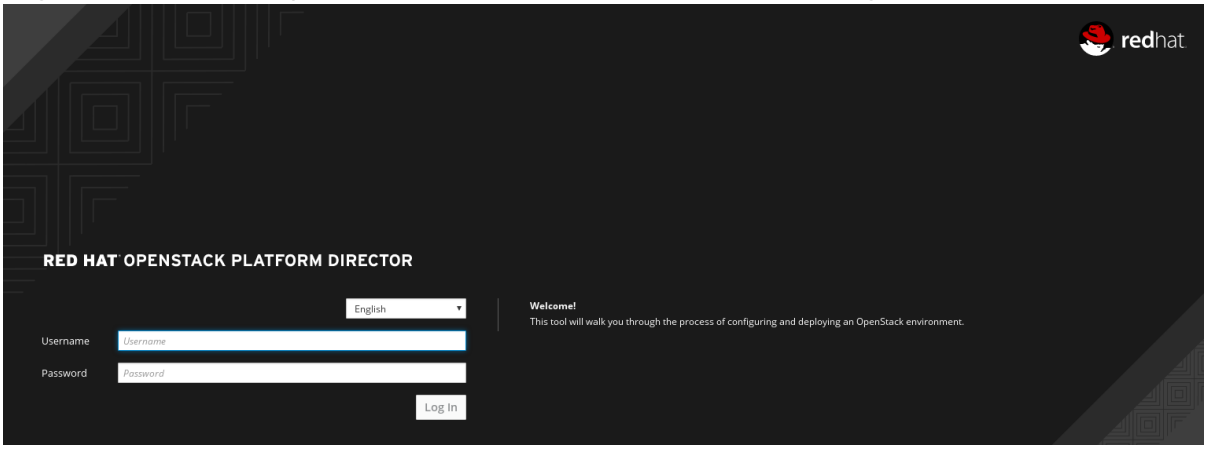
Procedure

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.

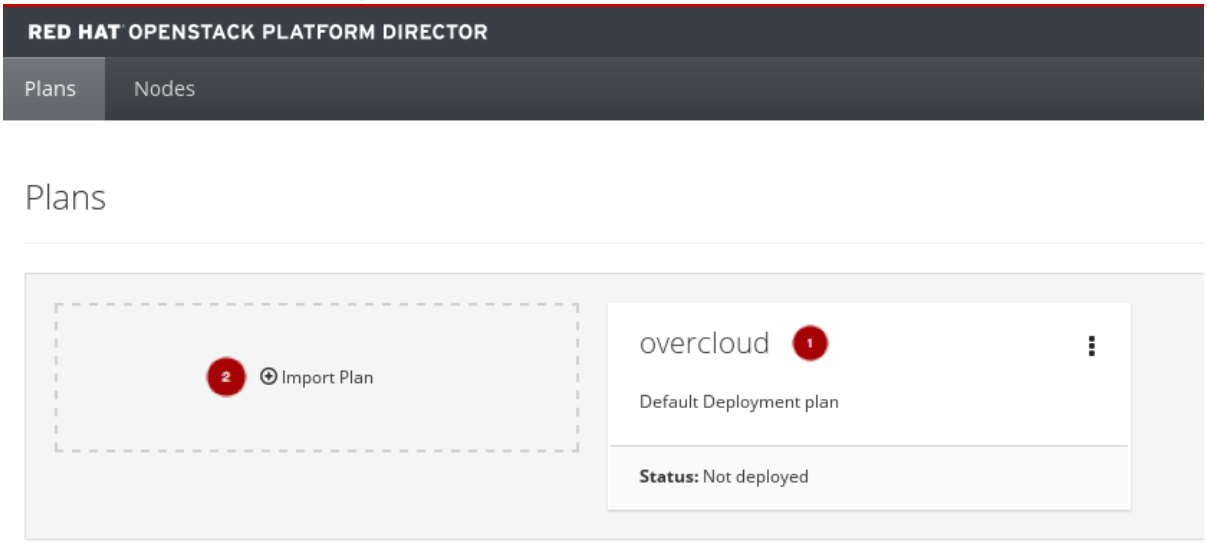


NOTE

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

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

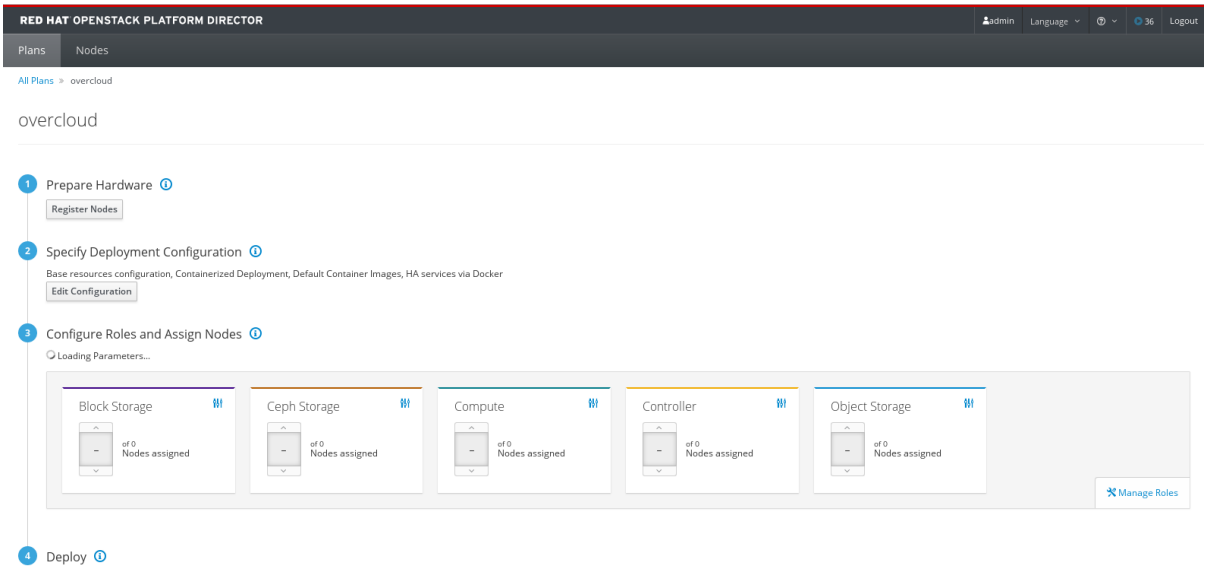
3. Select the default *overcloud* plan (1) or select the *Import Plan* (2).



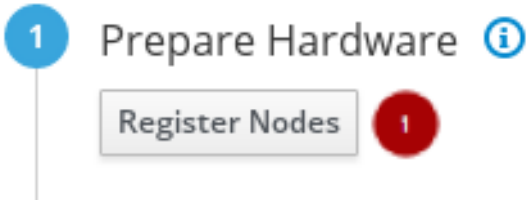
For more information on importing a plan, see Section 4.3, “Importing an Overcloud Plan Using the Red Hat OpenStack Platform Director”

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

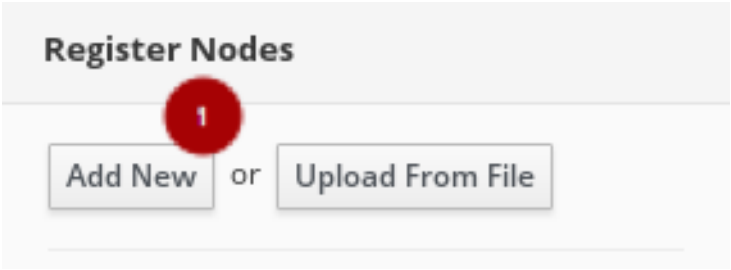
Figure 4.1. Example Plan Configuration Page



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



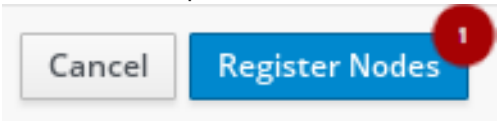
b. Click on the *Add New Node* button 1 .



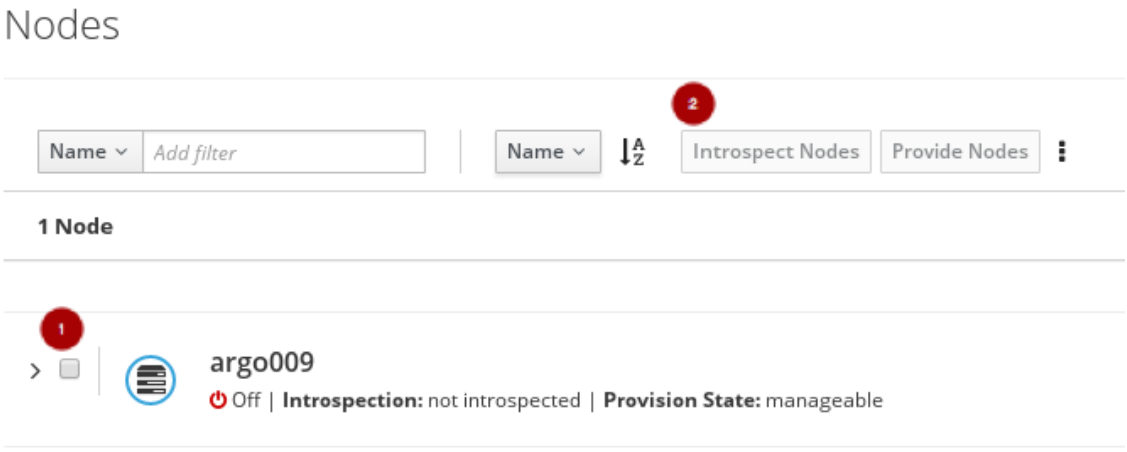
Alternatively, you can prepare the nodes by customizing the **`instackenv.json`** host definition file and uploading it. To create a custom **`instackenv.json`** host definition file, see Section 5.2.1, “Registering and Introspecting the Hardware” and Section 5.2.2, “Setting the Root Device” to prepare the nodes.

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

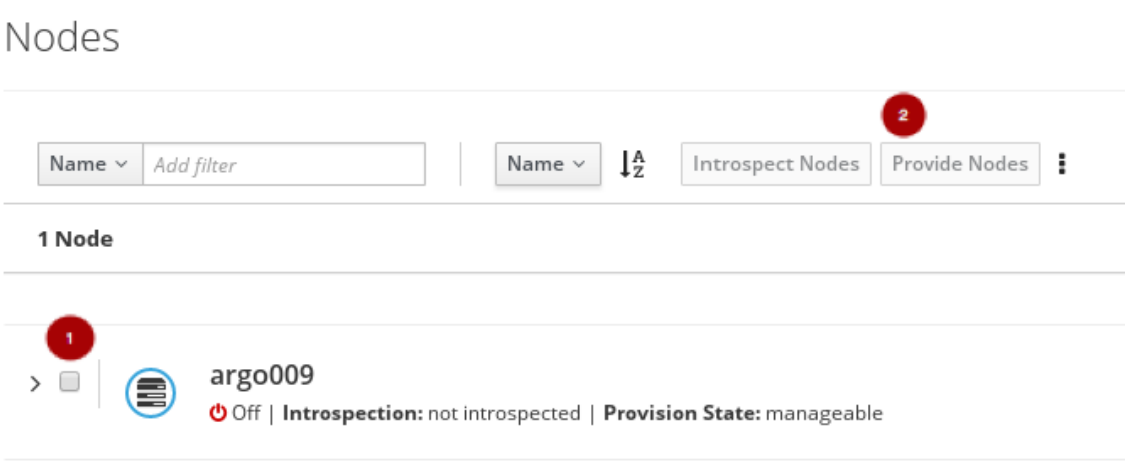
d. After all the required field are filled out, click on the *Register Node* button 1 .



e. Once the node is registered, select the node 1 , and click on the *Introspect Nodes* 2 button.

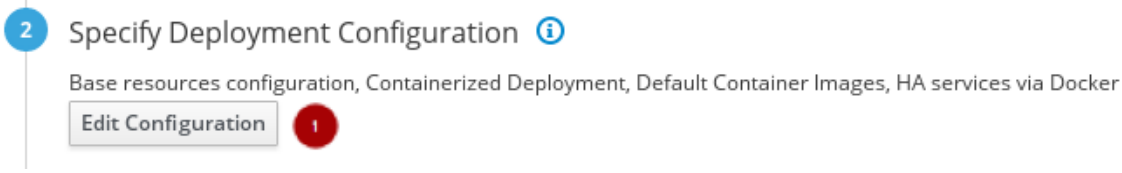


f. Once the introspection is done, select the node 1 , and click on the *Provide Nodes* 2 button.

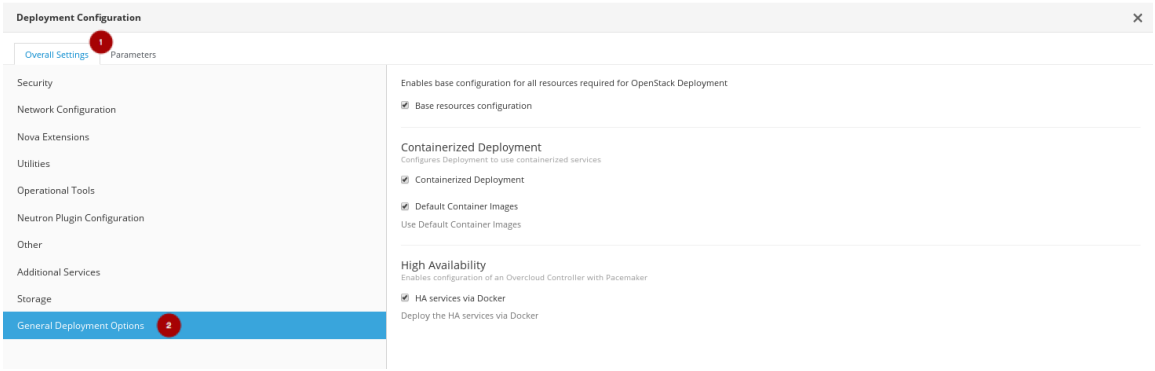


5. From the plan configuration page, edit the deployment configuration.

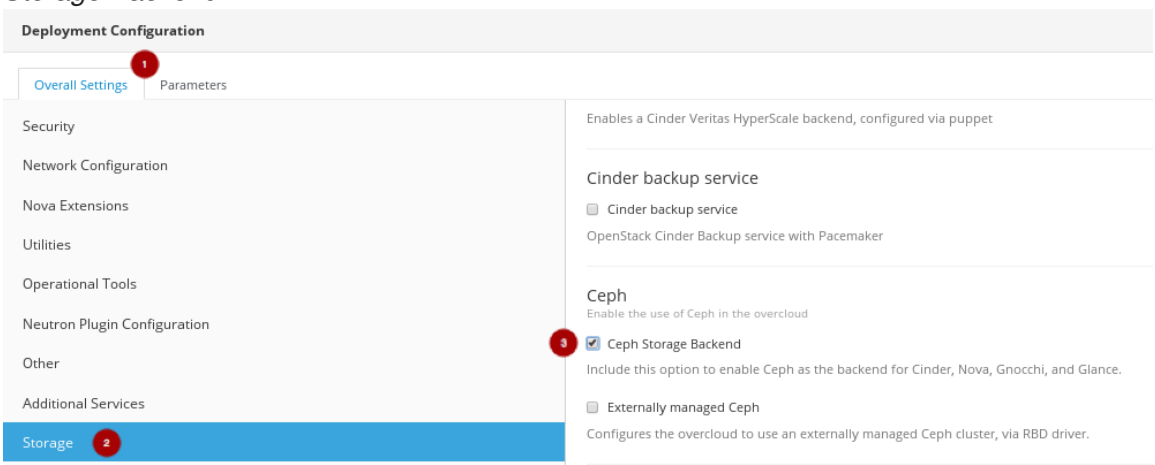
   a. Click on the *Edit Configuration* button 1 .

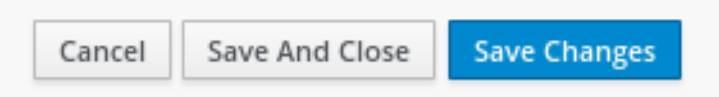
   b. On the *Overall Settings* tab 1 , click on the *General Deployment Options* section 2 , and enable the *HA services via Docker*, *Containerized Deployment*, and *Default Container Images*.

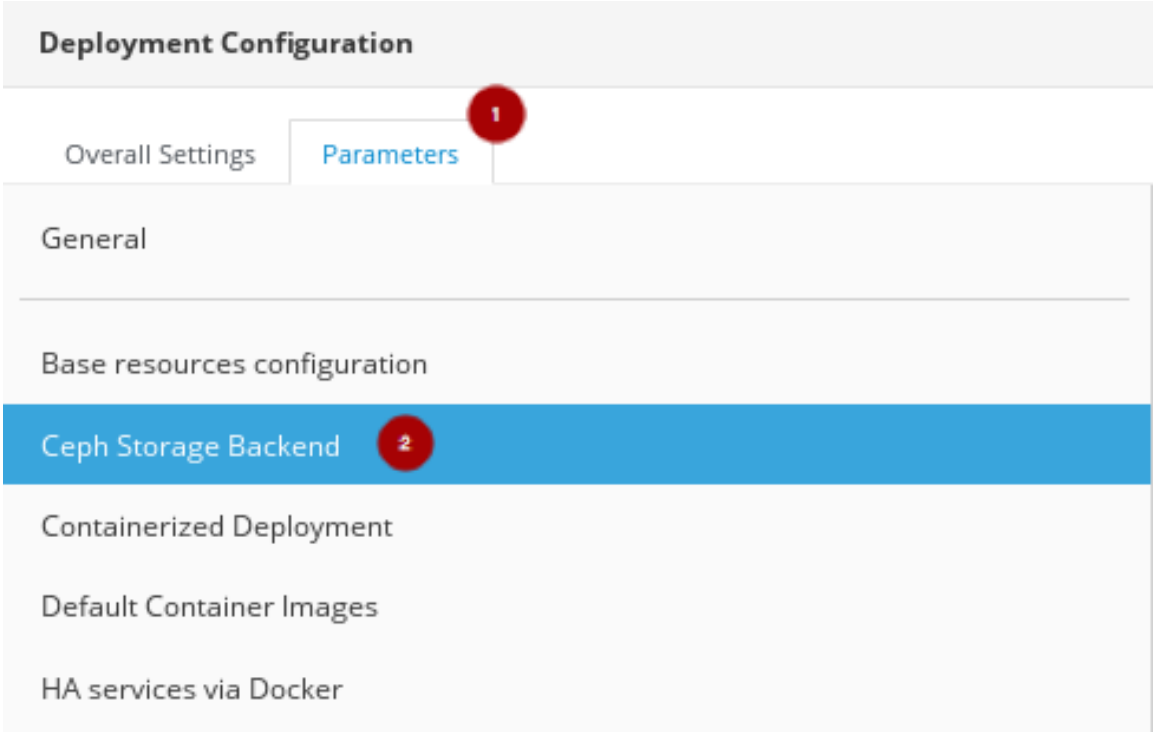
   c. On the *Overall Settings* tab 1 , click on the *Storage* section 2 , and enable the *Ceph Storage Backend* 3 .

Click on the *Save Changes* button.



d. Click on the *Parameters* tab (1), then click on the *Ceph Storage Backend* section (2) 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}
```

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

Update the *CephAnsibleDisksConfig* field with a disk list.

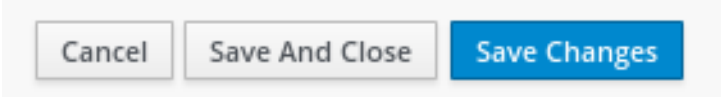
Example

```
{"devices":
["/dev/sda","/dev/sdb","/dev/sdc","/dev/sdd","/dev/sde","/dev/sdf
","/dev/sdg","/dev/sdh","/dev/sdi","/dev/sdj","/dev/sdk","/dev/sd
l"],"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/sd
o"],"journal_size":5120}
```

NOTE

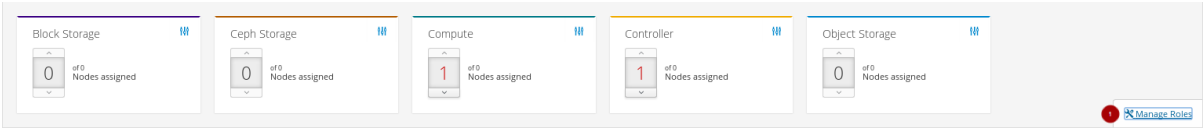
This disk listing is for block devices (**`devices`**) being used as OSDs, and the block devices dedicated (**`dedicated_devices`**) as OSD journals. See Section 5.4.5, "Setting the Red Hat Ceph Storage Parameters" for more information.

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

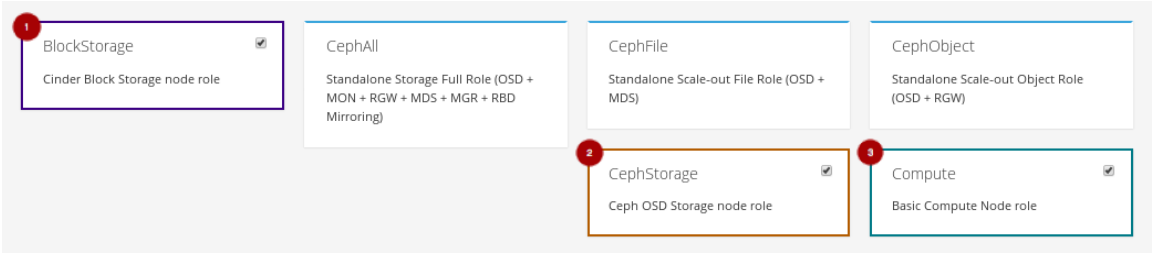


f. Back on the plan configuration page, the saved configuration changes will appear under the *Specify Deployment Configuration* step.

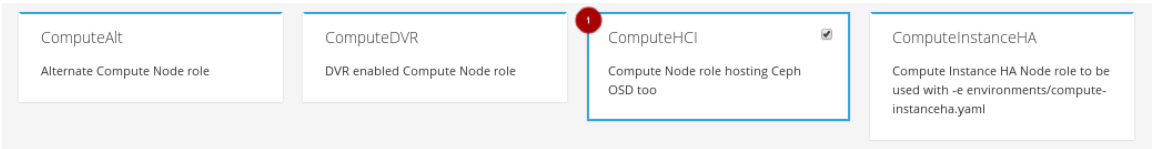
6. Configure the roles for the hyperconverged nodes by clicking on the *Manage Roles* link 1 .



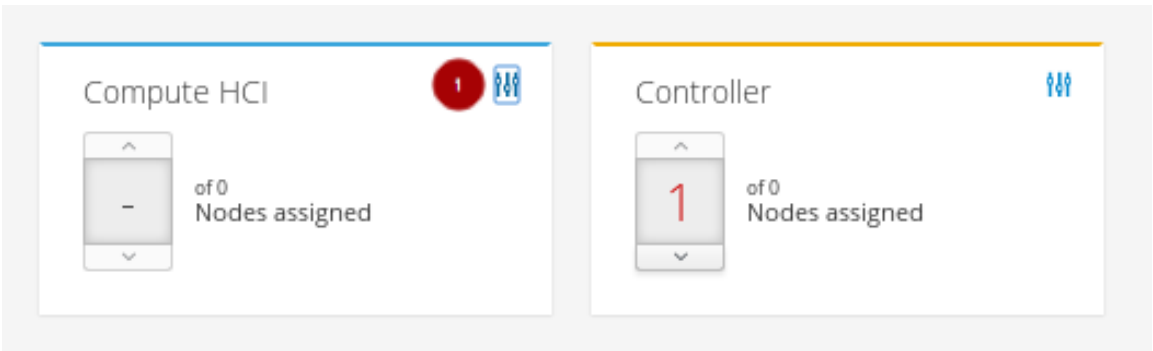
a. Unselect the *BlockStorage* 1 , *CephStorage* 2 , and *Compute* 3 roles by clicking on them.



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_ap
  i":["192.168.2.203","192.168.2.204","192.168.2.205"]}
  ```

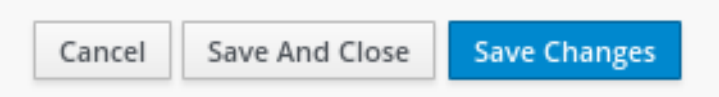
- The *OvercloudComputeHCIFlavor* field with the following value:

  ```
  osd-compute
  ```

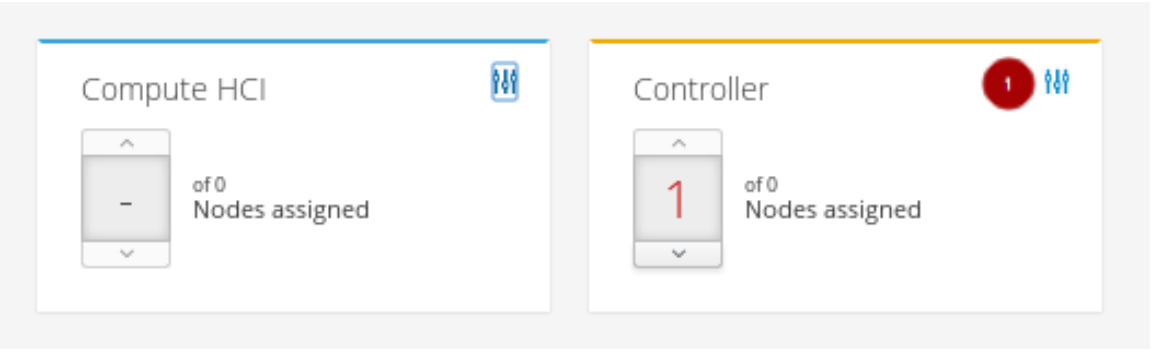
- The *ComputeHCISchedulerHints* field with the following value:

  ```
  {"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  .

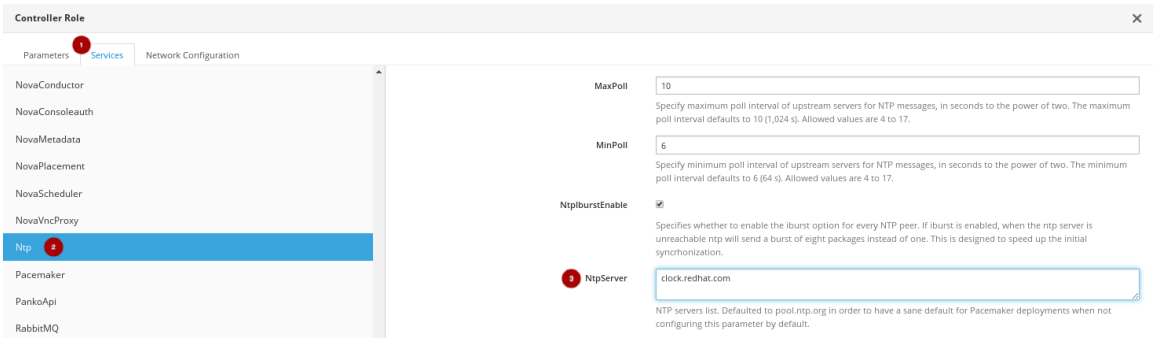


g. On the *Parameters* tab  , update the *ControllerIPs* field with the relevant IP addresses.

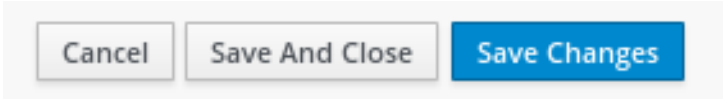
Example

```
{"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.

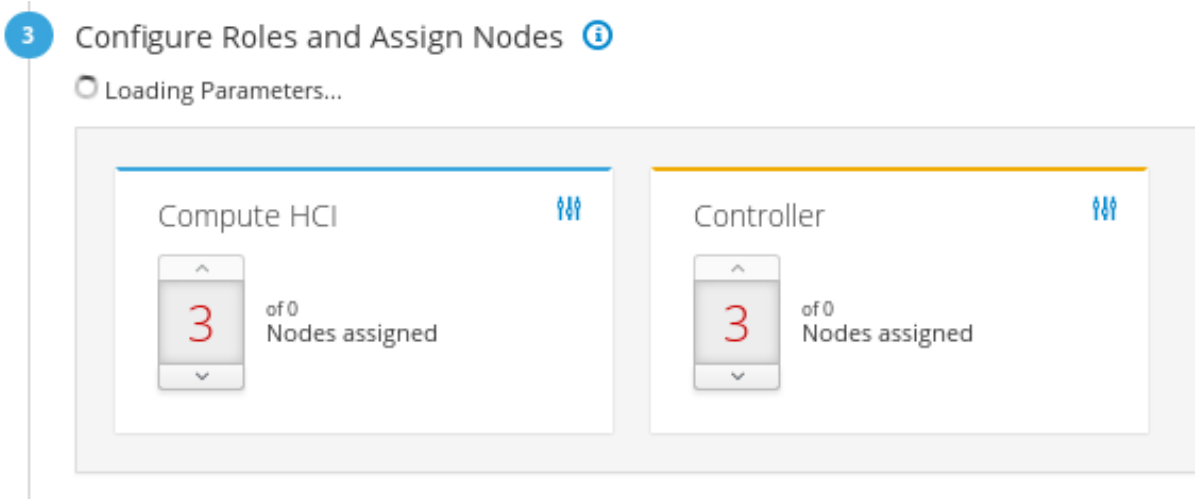


i. Click on the *Save And Close* button.

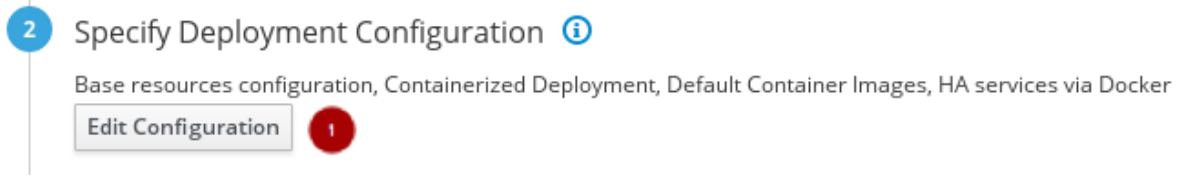


7. Assign the number of nodes needed in the environment for each role.

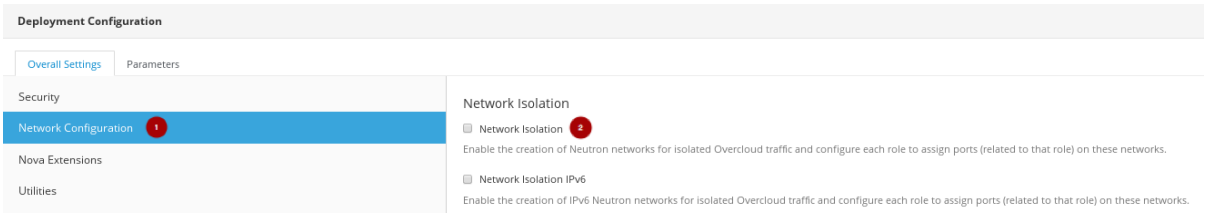
Figure 4.2. Example



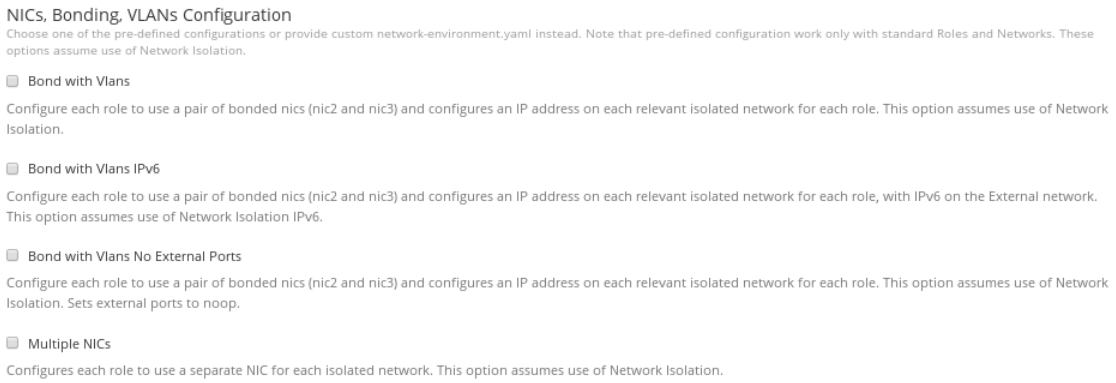
8. From the plan configuration page, click on the *Edit Configuration* button (1).



Edit the network configuration by clicking on the *Network Configuration* section (1), and select *Network Isolation* (2).



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

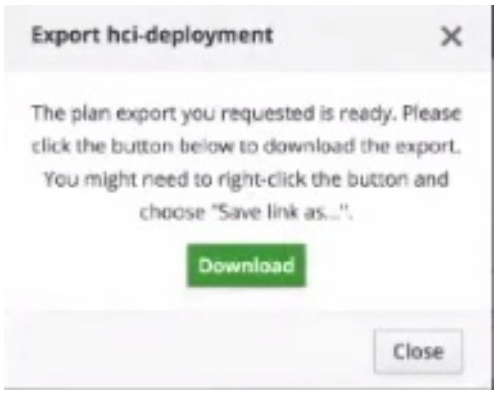


To customize the NICs in the environment, first you need to export the plan.

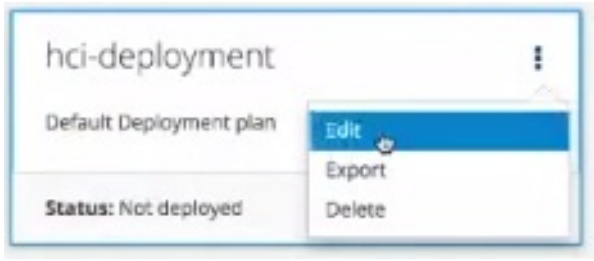
See Section 4.2, “Exporting an Overcloud Plan Using the Red Hat OpenStack Platform Director” on how to export a plan.

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

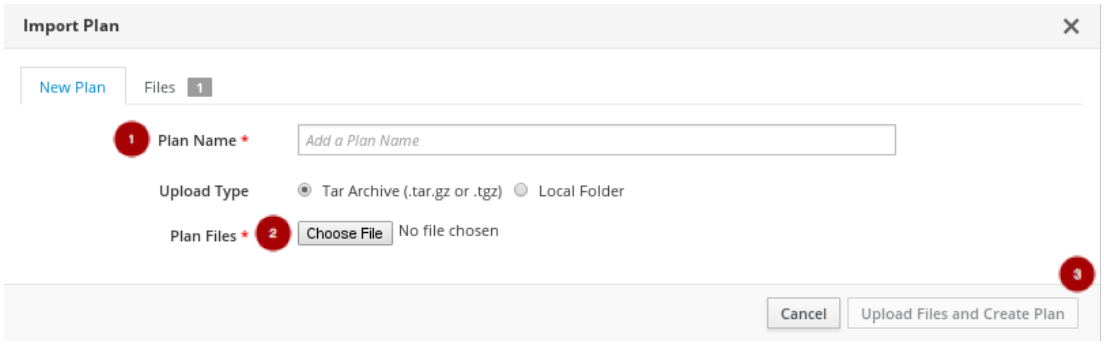
Example



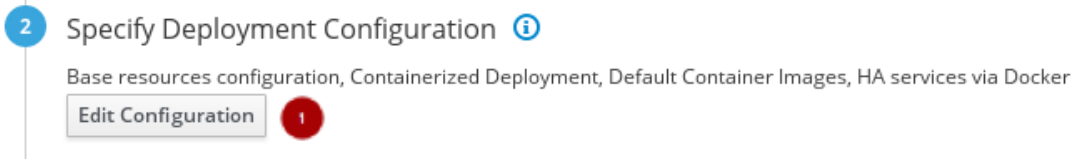
ii. After updating the plan tarball file, click the drop down menu and select *Edit*.



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.

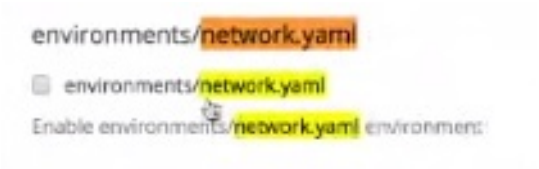


v. On the *Overall Settings* tab 1 , click on the *Other* section 2 .

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

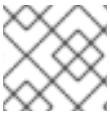
As a technician, you can deploy and manage the Red Hat Hyperconverged Infrastructure for Cloud solution using the command-line interface.

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.

Prerequisites

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

5.2.1. 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:

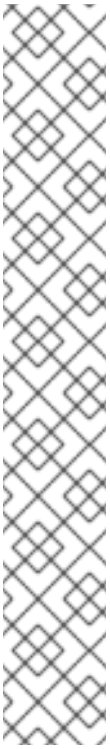
   ```
   {
     "pm_password": "$IPMI_USER_PASSWORD",
     "name": "$NODE_NAME",
     "pm_user": "$IPMI_USER_NAME",
     "pm_addr": "$IPMI_IP_ADDR",
     "pm_type": "pxe_ipmitool",
     "mac": [
            "$NIC_MAC_ADDR"
           ],
     "arch": "x86_64",
     "capabilities": "node:$NODE_ROLE-
   INSTANCE_NUM,boot_option:local"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

   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**.

     ```
     {
       "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.

5.2.2. 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.

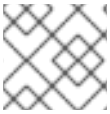
1. Configure the root disk device to use the **`smallest`** root device:

```
[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.

5.2.3. 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
```

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

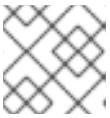
- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.
- Access to the Internet.

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/custom-
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

- See Section 5.3.2, “Including Additional Container Images for Red Hat OpenStack Platform Services” for more details.

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

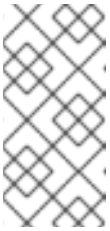
- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.
- Access to the Internet.

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
```

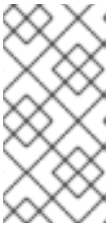
Use the IP address of the undercloud, which you previously set with the **local_ip** parameter in the **undercloud.conf** file. Alternatively, you can also obtain the full namespace with the following command:

```
(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/custom-
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 the 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 Section 5.3.2, “Including Additional Container Images for Red Hat OpenStack Platform Services” for more details.

Next Steps

- Prepare the overcloud for an upgrade.

Additional Resources

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

5.4. 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.

5.4.1. 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

5.4.2. 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 ~/custom-templates/nic-configs
```

5.4.3. 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 ~/custom-templates/ directory:

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

3. Copy the chosen template to the **~/custom-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 ~/custom-
templates/nic-configs/compute-hci.yaml
```

4. Add following definition, if it does note already exist, in the **parameters:** section of the **~/custom-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 **~/custom-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 ~/custom-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/custom-
   templates/nic-configs/controller-nics.yaml
   OS::TripleO::Compute::Net::SoftwareConfig: /home/stack/custom-
   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.

5.4.4. 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 Section 5.6, “Deploying the Overcloud Using the Command-line Interface” on using this custom role.

5.4.5. 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 the **~/custom-templates/ceph.yaml** file.
   a. Add all the Ceph OSD parameters to the **parameter_defaults** section:

Example

```
parameter_defaults:
    CephPoolDefaultSize: 3
    CephPoolDefaultPgNum: $NUM
    CephAnsibleDisksConfig:
      osd_scenario: non-collocated
      devices:
        - /dev/sda
        - /dev/sdb
        - /dev/sdc
        - /dev/sdd
        - /dev/sde
        - /dev/sdf
        - /dev/sdg
      dedicated_devices:
        - /dev/sdh
        - /dev/sdh
        - /dev/sdh
        - /dev/sdh
        - /dev/sdi
        - /dev/sdi
        - /dev/sdi
       CephAnsibleExtraConfig:
        osd_scenario: non-collocated
        osd_objectstore: filestore
        ceph_osd_docker_memory_limit: 5g
        ceph_osd_docker_cpu_limit: 1

       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.

b. For this step, the following Compute/OSD node disk configuration will be used as an example:

- **OSD** : 12 x 1TB SAS disks presented as **`/dev/[sda, sdb, …, sdg]`** block devices
- **OSD Journal** : 3 x 400GB SATA SSD disks presented as **`/dev/[sdh, sdi]`** block devices

Additional Resources

- For more details on tuning Ceph OSD parameters, see the Red Hat Ceph Storage Storage Strategies Guide.

5.4.6. 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 ~/custom-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`**.

5.4.7. Additional Resources

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

5.5. 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 in previous chapters.

5.5.1. Prerequisites

- Build the overcloud foundation by defining the overcloud.

5.5.2. 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**:

Defaut 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.

5.5.3. Applying Resource Isolation to the Ceph OSDs

Limiting the amount of CPU and memory for each Ceph OSD is important, so resources are free for the Nova Compute processes. The **ceph_osd_docker_memory_limit** option corresponds to the **docker run … -memory** command, and the **ceph_osd_docker_cpu_limit** option corresponds to the **docker run … -cpu-quota** command.

Example

```
CephAnsibleExtraConfig:
  ceph_osd_docker_memory_limit: 5g
  ceph_osd_docker_cpu_limit: 1
```

WARNING

Because these setting imposes a hard limit per OSD, it is possible that each OSD could run out of memory. Ceph OSDs consume extra memory during backfilling, so test the deployment with the expected workload. Include OSD removal in your testing to ensure the default settings of 5 GB of memory and 1 vCPU are appropriate for this deployment.

With hyper-converged nodes running the Nova Compute services and the Ceph OSD services, determinism is improved by pinning Ceph to one of the available NUMA nodes. The NUMA node with the network IRQ and storage controller IRQ is the NUMA node that Ceph must be pinned to. Building on the previous example, adding the **ceph_osd_docker_cpuset_cpus** and **ceph_osd_docker_cpuset_mems** options affects NUMA affinity for the OSD processes.

Example

```
CephAnsibleExtraConfig:
```

```
    ceph_osd_docker_memory_limit: 5g
    ceph_osd_docker_cpu_limit: 1
    ceph_osd_docker_cpuset_cpus: "0,2,4,6,8,10,12,14"
    ceph_osd_docker_cpuset_mems: "0"
```

The **ceph_osd_docker_cpuset_cpus** option corresponds to the **docker run … --cpuset-cpus** command and the **ceph_osd_docker_cpuset_mems** option corresponds to the **docker run … --cpuset-mems** command. These commands are ran when starting the OSDs.

WARNING

CephAnsibleExtraConfig is not additive. This means that you may only use it once in your Heat environment files.

5.5.4. 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.

5.5.5. 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.

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.

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

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

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

1. Generate the fencing Heat environment file:

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

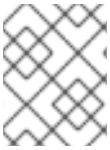
2. Include the **`fencing.yaml`** file with the **`openstack overcloud deploy`** command.

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

Procedure
Do the following step on the Red Hat OpenStack Platform director (RHOSP-d) node, as the **`stack`** user.

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/custom-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/custom-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 ~/custom-templates/network.yaml \
-e ~/custom-templates/ceph.yaml \
-e ~/custom-templates/layout.yaml
-e /home/stack/fencing.yaml
```

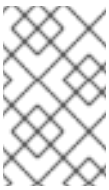
Command Details

- The **`time`** command is used to tell you how long the deployment takes.
- The **`openstack overcloud deploy`** command does the actual deployment.

- 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 Section 5.5.2, “Reserving CPU and Memory Resources for Hyperconverged Nodes” 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 **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 Section 5.4.3, “Configuring the Overcloud Networks”.
- The **ceph.yaml** file is explained in Section 5.4.5, “Setting the Red Hat Ceph Storage Parameters”.
- The **compute.yaml** file is explained in Appendix F, *Changing Nova Reserved Memory and CPU Allocation Manually*.
- The **layout.yaml** file is explained in Section 5.4.6, “Configuring the Overcloud Nodes Layout”.
- The **fencing.yaml** file is explained in Section 5.6.3, “Configuring the Controller for Pacemaker Fencing”.

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 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.

Procedure

Do the following steps on the Red Hat OpenStack Platform director node in a separate console sessiont, as the **stack** user.

1. Watch the deployment process and look for failures:

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

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

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

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
```

CHAPTER 6. UPGRADING THE RED HAT HYPERCONVERGED INFRASTRUCTURE FOR CLOUD SOLUTION

As a technician, you can upgrade the Red Hat Hyperconverged Infrastructure for Cloud solution by taking advantage of Red Hat OpenStack Platform's Fast-Forward Upgrade (FFU) feature. This document covers upgrades from Red Hat OpenStack Platform 10 (Newton) to 13 (Queens), and from Red Hat Ceph Storage 2 to 3.

Basic Upgrade Workflow

1. Prepare the environment
2. Upgrade the undercloud
3. Obtain updated container images
4. Prepare the overcloud
5. Perform the fast-forward upgrades
    a. Upgrade the hyperconverged Monitor/Controller nodes
    b. Upgrade the hyperconverged OSD/Compute nodes
    c. Upgrade Red Hat Ceph Storage
6. Finalize the upgrade

6.1. PREREQUISITES

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

6.2. INTRODUCING THE FAST-FORWARD UPGRADE PROCESS

The Fast-Forward Upgrade (FFU) feature provides an upgrade path spanning multiple Red Hat OpenStack Platform versions. This feature allows users to upgrade from the current long-life release, to the next long-life release of the Red Hat OpenStack Platform.

NOTE

Currently, the supported FFU path is from Red Hat OpenStack Platform 10 (Newton) to 13 (Queens).

Additional Resources

- See the Fast-Forward Upgrades Guide for more details.

6.3. PREPARING TO DO A RED HAT OPENSTACK PLATFORM UPGRADE

As a technician, you need to perform a series of tasks before proceeding with the upgrade process. This process involves the following basic steps:

1. Backing up the undercloud and overcloud.
2. Updating the undercloud to the latest minor version of Red Hat OpenStack Platform 10.
3. Rebooting the undercloud, if newer packages are installed.
4. Updating the overcloud images.
5. Updating the overcloud to the latest minor version of Red Hat OpenStack Platform 10.
6. Rebooting the overcloud nodes, if newer packages are installed.
7. Performing validation checks on both the undercloud and overcloud

Doing these steps ensure the existing Red Hat Hyperconverged Infrastructure for Cloud environment is in the best possible state before proceeding with an upgrade.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

6.3.1. Backing Up the Undercloud

A full undercloud backup includes the following databases and files:

- All MariaDB databases on the undercloud node
- MariaDB configuration file on the undercloud, for accurately restoring databases
- All swift data: **`/srv/node`**
- All data in the **`stack`** user's home directory: **`/home/stack`**
- The undercloud SSL certificates:
  - **`/etc/pki/ca-trust/source/anchors/ca.crt.pem`**
  - **`/etc/pki/instack-certs/undercloud.pem`**

NOTE

Confirm that you have sufficient disk space available before performing the backup process. The tarball can be expected to be at least 3.5 GB, but this is likely to be larger.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

Procedure

1. Log into the undercloud as the **`root`** user.
2. Back up the database:

```
[root@director ~]# mysqldump --opt --all-databases >
/root/undercloud-all-databases.sql
```

3. Archive the database backup and the configuration files:

```
[root@director ~]# tar --xattrs -czf undercloud-backup-`date
+%F`.tar.gz /root/undercloud-all-databases.sql
/etc/my.cnf.d/server.cnf /srv/node /home/stack /etc/pki/instack-
certs/undercloud.pem /etc/pki/ca-trust/source/anchors/ca.crt.pem
```

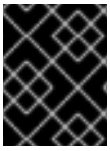
This creates a file named **undercloud-backup-[timestamp].tar.gz**.

Additional Resources

- If you need to restore the undercloud backup, see the Restore chapter in the *Back Up and Restore the Director Undercloud* guide.

6.3.2. Backing Up the Overcloud Control Plane Services

The following procedure creates a backup of the overcloud databases and configuration. While most of the overcloud configuration can be recreated using the **openstack overcloud deploy**, a backup of the overcloud database and services ensures you have a snapshot of a working environment. Having this snapshot helps, if you need to restore the overcloud to its original state in case of an operational failure.



IMPORTANT

This procedure only includes crucial control plane services. It does not include backups of Compute node workloads nor data on Red Hat Ceph Storage nodes.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

Procedure

1. Perform the database backup:
   a. Log into a Controller node. You can access the overcloud from the undercloud:

```
$ ssh heat-admin@192.0.2.100
```

   Change to the **root** user:

```
$ sudo -i
```

   b. Create a temporary directory to store the backups:

```
# mkdir -p /var/tmp/mysql_backup/
```

   c. Obtain the database password and store it in the **MYSQLDBPASS** environment variable. The password is stored in the **mysql::server::root_password** variable within the **/etc/puppet/hieradata/service_configs.json** file. Use the following command to store the password:

```
# MYSQLDBPASS=$(sudo hiera mysql::server::root_password)
```

d. Backup the database:

```
# mysql -uroot -p$MYSQLDBPASS -e "select distinct table_schema
from information_schema.tables where engine='innodb' and
table_schema != 'mysql';" \
  -s -N | xargs mysqldump -uroot -p$MYSQLDBPASS --single-
transaction --databases >
/var/tmp/mysql_backup/openstack_databases-`date +%F`-`date
+%T`.sql
```

This dumps a database backup called **/var/tmp/mysql_backup/openstack_databases-<date>.sql** where **<date>** is the system date and time.

e. Backup all the users and permissions information:

```
# mysql -uroot -p$MYSQLDBPASS -e "SELECT CONCAT('\"SHOW GRANTS
FOR ''',user,'''@''',host,''';\"') FROM mysql.user where
(length(user) > 0 and user NOT LIKE 'root')" \
  -s -N | xargs -n1 mysql -uroot -p$MYSQLDBPASS -s -N -e | sed
's/$/;/' > /var/tmp/mysql_backup/openstack_databases_grants-`date
+%F`-`date +%T`.sql
```

This will dump a database backup called **/var/tmp/mysql_backup/openstack_databases_grants-<date>.sql** where **<date>** is the system date and time.

2. Backup the OpenStack Telemetry database:

a. Connect to any controller and get the IP of the MongoDB primary instance:

```
# MONGOIP=$(sudo hiera mongodb::server::bind_ip)
```

b. Create the backup:

```
# mkdir -p /var/tmp/mongo_backup/
# mongodump --oplog --host $MONGOIP --out /var/tmp/mongo_backup/
```

3. Backup the Redis cluster:

a. Obtain the Redis endpoint from HAProxy:

```
# REDISIP=$(sudo hiera redis_vip)
```

b. Obtain the master password for the Redis custer:

```
# REDISPASS=$(sudo hiera redis::masterauth)
```

c. Check connectivity to the Redis cluster:

```
# redis-cli -a $REDISPASS -h $REDISIP ping
```

d. Dump the Redis database:

```
redis-cli -a $REDISPASS -h $REDISIP bgsave
```

This stores the database backup in the default **/var/lib/redis/** directory.

4. Backup the filesystem:

   a. Create a directory for the backup:

   ```
   # mkdir -p /var/tmp/filesystem_backup/
   ```

   b. Run the following **tar** command:

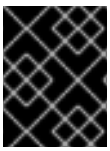
   ```
   # tar --ignore-failed-read \
       -zcvf /var/tmp/filesystem_backup/fs_backup-`date '+%Y-%m-%d-
   %H-%M-%S'`.tar.gz \
       /etc/nova \
       /var/log/nova \
       /var/lib/nova \
       --exclude /var/lib/nova/instances \
       /etc/glance \
       /var/log/glance \
       /var/lib/glance \
       /etc/keystone \
       /var/log/keystone \
       /var/lib/keystone \
       /etc/httpd \
       /etc/cinder \
       /var/log/cinder \
       /var/lib/cinder \
       /etc/heat \
       /var/log/heat \
       /var/lib/heat \
       /var/lib/heat-config \
       /var/lib/heat-cfntools \
       /etc/rabbitmq \
       /var/log/rabbitmq \
       /var/lib/rabbitmq \
       /etc/neutron \
       /var/log/neutron \
       /var/lib/neutron \
       /etc/corosync \
       /etc/haproxy \
       /etc/logrotate.d/haproxy \
       /var/lib/haproxy \
       /etc/openvswitch \
       /var/log/openvswitch \
       /var/lib/openvswitch \
       /etc/ceilometer \
       /var/lib/redis \
       /etc/sysconfig/memcached \
       /etc/gnocchi \
       /var/log/gnocchi \
       /etc/aodh \
       /var/log/aodh \
       /etc/panko \
   ```

```
    /var/log/panko \
    /etc/ceilometer \
    /var/log/ceilometer
```

The **--ignore-failed-read** option ignores any missing directories, which is useful if certain services are not used or separated on their own custom roles.

6.3.3. Updating the Current Undercloud Packages for OpenStack Platform 10.z

The director provides commands to update the packages on the undercloud node. This allows you to perform a minor update within the current version of the OpenStack Platform environment. This is a minor update within **OpenStack Platform 10**.



IMPORTANT

This procedure also updates the operating systems packages to the latest version of Red Hat Enterprise Linux.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

Procedure

1. Log into the undercloud as the **stack** user.
2. Stop the main OpenStack Platform services:

```
(undercloud) [stack@director ~]$ sudo systemctl stop 'openstack-*'
'neutron-*' httpd
```

NOTE

This causes a short period of downtime for the undercloud. The overcloud is still functional during the undercloud upgrade.

3. Update the **python-tripleoclient** package and its dependencies to ensure you have the latest scripts for the minor version update:

```
(undercloud) [stack@director ~]$ sudo yum update python-
tripleoclient
```

4. Run the **openstack undercloud upgrade** command:

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

Wait until this command completes its execution.

5. Reboot the undercloud to update the operating system's kernel and other system packages:

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

6. Wait until the node boots.

7. Log into the undercloud as the **stack** user.

6.3.4. Updating the Current Overcloud Images for Red Hat OpenStack Platform 10.z

In addition to undercloud package updates, Red Hat recommends to keep the overcloud images up to date and to keep the image configuration in sync with the latest **openstack-tripleo-heat-templates** package. This ensures successful deployment and scaling operations in between the current preparation stage and the actual fast-forward upgrade. The undercloud update process might download new image archives from the **rhosp-director-images** and **rhosp-director-images-ipa** packages. This process updates these images on the undercloud within **Red Hat OpenStack Platform 10**.

Prerequisites

- Update to the latest minor release of the current undercloud version.

Procedure

1. Check the **yum** log to determine if new image archives are available:

```
(undercloud) [stack@director ~]$ sudo grep "rhosp-director-images"
/var/log/yum.log
```

2. If new archives are available, replace the current images with new images. To install the new images, first remove any existing images from the **images** directory on the **stack** user's home (**/home/stack/images**):

```
(undercloud) [stack@director ~]$ rm -rf ~/images/*
```

3. Extract the archives:

```
(undercloud) [stack@director ~]$ cd ~/images
(undercloud) [stack@director ~]$ for i in /usr/share/rhosp-director-
images/overcloud-full-latest-10.0.tar /usr/share/rhosp-director-
images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

4. Import the latest images into the director and configure nodes to use the new images

```
(undercloud) [stack@director ~]$ cd ~
(undercloud) [stack@director ~]$ openstack overcloud image upload --
update-existing --image-path /home/stack/images/
(undercloud) [stack@director ~]$ openstack overcloud node configure
$(openstack baremetal node list -c UUID -f csv --quote none | sed
"1d" | paste -s -d " ")
```

5. To finalize the image update, verify the existence of the new images:

```
(undercloud) [stack@director ~]$ openstack image list
(undercloud) [stack@director ~]$ ls -l /httpboot
```

The director also retains the old images and renames them using the timestamp of when they were updated. If you no longer need these images, delete them. The director is now updated and using the latest images.

NOTE

You do not need to restart any services after the update.

6.3.5. Updating the Current Overcloud Packages for Red Hat OpenStack Platform 10.z

The director provides commands to update the packages on all overcloud nodes. This allows you to perform a minor update within the current version of the Red Hat OpenStack Platform environment. This is a minor update within **Red Hat OpenStack Platform 10**.

NOTE

The update process does not reboot any nodes in the Overcloud automatically.

IMPORTANT

This procedure also updates the operating systems packages to the latest version of Red Hat Enterprise Linux. Updates to the kernel and other system packages require a reboot.

Prerequisites

- Updated to the latest minor release of the current undercloud version.
- Performed a backup of the overcloud.

Procedure

1. Update the current plan using the original **`openstack overcloud deploy`** command and including the **`--update-plan-only`** option. For example:

```
(undercloud) [stack@director ~]$ openstack overcloud deploy --
update-plan-only \
  --templates  \
  -e /usr/share/openstack-tripleo-heat-
templates/environments/network-isolation.yaml \
  -e /home/stack/custom-templates/network-environment.yaml \
  -e /home/stack/custom-templates/storage-environment.yaml \
  -e /home/stack/custom-templates/rhel-registration/environment-
rhel-registration.yaml \
  [-e <environment_file>|...]
```

The **`--update-plan-only`** only updates the Overcloud plan stored in the director. Use the **`-e`** option to include environment files relevant to the Overcloud and its update path. The order of the environment files is important as the parameters and resources defined in subsequent environment files take precedence. Use the following list as an example of the environment file order:

- Any network isolation files, including the initialization file (**`environments/network-isolation.yaml`**) from the heat template collection and then the custom NIC configuration file.
- Any external load balancing environment files.
- Any storage environment files.

- Any environment files for Red Hat CDN or Satellite registration.
- Any other custom environment files.

2. Perform a package update on all nodes using the **openstack overcloud update** command. For example:

```
(undercloud) [stack@director ~]$ openstack overcloud update stack -i
overcloud
```

The **-i** runs an interactive mode to update each node. When the update process completes a node update, the script provides a breakpoint for you to confirm. Without the **-i** option, the update remains paused at the first breakpoint. Therefore, it is mandatory to include the **-i** option.

NOTE

Running an update on all nodes in parallel can cause problems. For example, an update of a package might involve restarting a service, which can disrupt other nodes. This is why the process updates each node using a set of breakpoints. This means nodes are updated one by one. When one node completes the package update, the update process moves to the next node.

3. The update process starts. During this process, the director reports an **IN_PROGRESS** status and periodically prompts you to clear breakpoints. For example:

```
not_started: [u'overcloud-controller-0', u'overcloud-controller-1',
u'overcloud-controller-2']
on_breakpoint: [u'overcloud-compute-0']
Breakpoint reached, continue? Regexp or Enter=proceed, no=cancel
update, C-c=quit interactive mode:
```

Press Enter to clear the breakpoint from last node on the **on_breakpoint** list. This begins the update for that node. You can also type a node name to clear a breakpoint on a specific node, or a Python-based regular expression to clear breakpoints on multiple nodes at once. However, it is not recommended to clear breakpoints on multiple controller nodes at once. Continue this process until all nodes have completed their update.

4. The update command reports a **COMPLETE** status when the update completes:

```
...
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
COMPLETE
update finished with status COMPLETE
```

5. If you configured fencing for the Controller nodes, the update process might disable it. When the update process completes, reenable fencing with the following command on one of the Controller nodes:

```
(undercloud) [stack@director ~]$ sudo pcs property set stonith-
enabled=true
```

6.3.6. Rebooting the Controller and Composable Nodes

The following procedure reboots controller nodes and standalone nodes based on composable roles. This excludes Compute nodes and Red Hat Ceph Storage nodes.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

Procedure

1. Select a node and login to it.
2. Reboot the node:

```
[heat-admin@overcloud-controller-0 ~]$ sudo reboot
```

3. Wait until the node boots.
4. Log into the node and check the services. For example:
   a. If the node uses Pacemaker services, check the node has rejoined the cluster:

```
[heat-admin@overcloud-controller-0 ~]$ sudo pcs status
```

   b. If the node uses Systemd services, check all services are enabled:

```
[heat-admin@overcloud-controller-0 ~]$ sudo systemctl status
```

6.3.7. Rebooting a Red Hat Ceph Storage Cluster

The following procedure reboots the Red Hat Ceph Storage nodes.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

Procedure

1. Log into a hyperconverged Controller/Ceph Monitor node and disable the storage cluster rebalancing feature temporarily:

```
[stack@cntr-mon ~]$ sudo ceph osd set noout
[stack@cntr-mon ~]$ sudo ceph osd set norebalance
```

2. Select the first Ceph Storage node to reboot and log into it.
3. Reboot the node:

```
[stack@cntr-mon ~]$ sudo reboot
```

4. Wait until the node boots.
5. Log into the node and check the cluster status:

```
[stack@cntr-mon ~]$ sudo ceph -s
```

Check that the **pgmap** reports all **pgs** as normal (**active+clean**).

6. Log out of the node, reboot the next node, and check its status. Repeat this process until you have rebooted all Ceph storage nodes.

7. When complete, log into a Ceph MON or Controller node and enable cluster rebalancing again:

```
[stack@cntr-mon ~]$ sudo ceph osd unset noout
[stack@cntr-mon ~]$ sudo ceph osd unset norebalance
```

8. Perform a final status check to verify the cluster reports **HEALTH_OK**:

```
[stack@cntr-mon ~]$ sudo ceph status
```

6.3.8. Rebooting the Compute Nodes

The following procedure reboots the Compute nodes. To ensure minimal downtime of instances in the Red Hat OpenStack Platform environment, this procedure also includes instructions on migrating instances from the chosen Compute node. This involves the following workflow:

- Select a Compute node to reboot and disable it so that it does not provision new instances
- Migrate the instances to another Compute node
- Reboot the empty Compute node and enable it

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

Procedure

1. Log into the undercloud as the **stack** user.

2. List all the compute nodes and their UUIDs:

```
[stack@director ~]$ source ~/stackrc
(undercloud) [stack@director ~]$ openstack server list --name
compute
```

Identify the UUID of the compute node you aim to reboot.

3. From the undercloud, select a compute node and disable it:

```
[stack@director ~]$ source ~/overcloudrc
(overcloud) [stack@director ~]$ openstack compute service list
(overcloud) [stack@director ~]$ openstack compute service set
[hostname] nova-compute --disable
```

4. List all instances on the compute node:

```
(overcloud) [stack@director ~]$ openstack server list --host
[hostname] --all-projects
```

5. Use one of the following commands to migrate the instances:

   a. Migrate the instance to a specific host of the choice:

   ```
   (overcloud) [stack@director ~]$ openstack server migrate
   [instance-id] --live [target-host]--wait
   ```

   b. Let **nova-scheduler** automatically select the target host:

   ```
   (overcloud) [stack@director ~]$ nova live-migration [instance-id]
   ```

   c. Live migrate all instances at once:

   ```
   [stack@director ~]$ nova host-evacuate-live [hostname]
   ```

   NOTE

   The **nova** command might cause some deprecation warnings, which are safe to ignore.

6. Wait until migration completes.

7. Confirm the migration was successful:

   ```
   (overcloud) [stack@director ~]$ openstack server list --host
   [hostname] --all-projects
   ```

8. Continue migrating instances until none remain on the chosen compute node.

9. Log into the compute node and reboot it:

   ```
   [heat-admin@overcloud-compute-0 ~]$ sudo reboot
   ```

10. Wait until the node boots.

11. Enable the compute node again:

    ```
    [stack@director ~]$ source ~/overcloudrc
    (overcloud) [stack@director ~]$ openstack compute service set
    [hostname] nova-compute --enable
    ```

12. Check whether the compute node is enabled:

    ```
    (overcloud) [stack@director ~]$ openstack compute service list
    ```

6.3.9. Verifying the System Packages Before Upgrading

Before the upgrade, all nodes should be using the latest versions of the following packages:

Package	Version

openvswitch	At least 2.9
qemu-img-rhev	At least 2.10
qemu-kvm-common-rhev	At least 2.10
qemu-kvm-rhev	At least 2.10
qemu-kvm-tools-rhev	At least 2.10

Prerequisites

- Access to all the nodes in the Red Hat Hyperconverged Infrastructure for Cloud environment.

Procedure

1. Log into a node.
2. Run **yum** to check the system packages:

```
[stack@director ~]$ sudo yum list qemu-img-rhev qemu-kvm-common-rhev
qemu-kvm-rhev qemu-kvm-tools-rhev openvswitch
```

3. Run **ovs-vsctl** to check the version currently running:

```
[stack@director ~]$ sudo ovs-vsctl --version
```

4. Repeat steps 1-3 for each node in the Red Hat Hyperconverged Infrastructure for Cloud environment.

6.3.10. Validating the Undercloud Before Upgrading

Follow this procedure to check the functionality of the Red Hat OpenStack Platform 10 undercloud before doing an upgrade.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

Procedure

1. Source the undercloud access details:

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

2. Check for failed Systemd services:

```
[stack@director ~]$ sudo systemctl list-units --state=failed
'openstack*' 'neutron*' 'httpd' 'docker'
```

3. Check the undercloud free space:

```
[stack@director ~]$ df -h
```

4. Check that clocks are synchronized on the undercloud:

```
[stack@director ~]$ sudo ntpstat
```

5. Check the undercloud network services:

```
[stack@director ~]$ openstack network agent list
```

All agents should be **Alive** and their state should be **UP**.

6. Check the undercloud compute services:

```
[stack@director ~]$ openstack compute service list
```

All agents' status should be **enabled** and their state should be **up**

7. Check the undercloud volume services:

```
[stack@director ~]$ openstack volume service list
```

All agents' status should be **enabled** and their state should be **up**.

Additional Resources

- See the Red Hat Knowledgebase article on how to remove deleted stack entries in the OpenStack Orchestration (heat) database: https://access.redhat.com/solutions/2215131

6.3.11. Validating the Overcloud Before Upgrading

Follow this procedure to check the functionality of the Red Hat OpenStack Platform 10 overcloud before an upgrade.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

Procedure

1. Source the undercloud access details:

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

2. Check the status of the bare metal nodes:

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

All nodes should have a valid power state (**on**) and maintenance mode should be **false**.

3. Check for failed Systemd services:

```
[stack@director ~]$ for NODE in $(openstack server list -f value -c
Networks | cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-
```

```
admin@$NODE "sudo systemctl list-units --state=failed 'openstack*'
'neutron*' 'httpd' 'docker' 'ceph*'" ; done
```

4. Check the HAProxy connection to all services. Obtain the Control Plane VIP address and authentication details for the **haproxy.stats** service:

```
[stack@director ~]$ NODE=$(openstack server list --name controller-0
-f value -c Networks | cut -d= -f2); ssh heat-admin@$NODE sudo 'grep
"listen haproxy.stats" -A 6 /etc/haproxy/haproxy.cfg'
```

   Use these details in the following cURL request:

```
[stack@director ~]$ curl -s -u admin:<PASSWORD> "http://<IP
ADDRESS>:1993/;csv" | egrep -vi "(frontend|backend)" | awk -F',' '{
print $1" "$2" "$18 }'
```

   Replace **<PASSWORD>** and **<IP ADDRESS>** details with the respective details from the **haproxy.stats** service. The resulting list shows the OpenStack Platform services on each node and their connection status.

5. Check overcloud database replication health:

```
[stack@director ~]$ for NODE in $(openstack server list --name
controller -f value -c Networks | cut -d= -f2); do echo "=== $NODE
===" ; ssh heat-admin@$NODE "sudo clustercheck" ; done
```

6. Check RabbitMQ cluster health:

```
[stack@director ~]$ for NODE in $(openstack server list --name
controller -f value -c Networks | cut -d= -f2); do echo "=== $NODE
===" ; ssh heat-admin@$NODE "sudo rabbitmqctl node_health_check" ;
done
```

7. Check Pacemaker resource health:

```
[stack@director ~]$ NODE=$(openstack server list --name controller-0
-f value -c Networks | cut -d= -f2); ssh heat-admin@$NODE "sudo pcs
status"
```

   Look for:

   - All cluster nodes **online**.
   - No resources **stopped** on any cluster nodes.
   - No **failed** pacemaker actions.

8. Check the disk space on each overcloud node:

```
[stack@director ~]$ for NODE in $(openstack server list -f value -c
Networks | cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-
admin@$NODE "sudo df -h --output=source,fstype,avail -x overlay -x
tmpfs -x devtmpfs" ; done
```

9. Check health of the Red Hat Ceph Storage cluster. The following command runs the **ceph** tool on a Controller node to check the cluster:

```
[stack@director ~]$ NODE=$(openstack server list --name controller-0
-f value -c Networks | cut -d= -f2); ssh heat-admin@$NODE "sudo ceph
-s"
```

10. Check the free space on the OSD. The following command runs the **ceph** tool on a Controller node to check the free space:

```
[stack@director ~]$ NODE=$(openstack server list --name controller-0
-f value -c Networks | cut -d= -f2); ssh heat-admin@$NODE "sudo ceph
df"
```

11. Check that clocks are synchronized on overcloud nodes

```
[stack@director ~]$ for NODE in $(openstack server list -f value -c
Networks | cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-
admin@$NODE "sudo ntpstat" ; done
```

12. Source the overcloud access details:

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

13. Check the overcloud network services:

```
[stack@director ~]$ openstack network agent list
```

All agents should be **Alive** and their state should be **UP**.

14. Check the overcloud compute services:

```
[stack@director ~]$ openstack compute service list
```

All agents' status should be **enabled** and their state should be **up**

15. Check the overcloud volume services:

```
[stack@director ~]$ openstack volume service list
```

All agents' status should be **enabled** and their state should be **up**.

Additional Resources

- Review the article "How can I verify my OpenStack environment is deployed with Red Hat recommended configurations?". This article provides some information on how to check the Red Hat OpenStack Platform environment and tune the configuration to Red Hat's recommendations.
- Review the article "Database Size Management for Red Hat Enterprise Linux OpenStack Platform" to check and clean unused database records for OpenStack Platform services on the overcloud.

Next Step

- Upgrading the undercloud from Red Hat OpenStack Platform 10 to 13

- Upgrading the undercloud from Red Hat OpenStack Platform 10 to 13.

6.4. UPGRADING THE UNDERCLOUD

The following procedures upgrades the undercloud and its overcloud images to Red Hat OpenStack Platform 13. You accomplish this by performing an upgrade through each sequential version of the undercloud from Red Hat OpenStack Platform 10 to Red Hat OpenStack Platform 13.

Prerequisites

- Preparing Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

6.4.1. Upgrading the Undercloud to Red Hat OpenStack Platform 11

This procedure upgrades the undercloud toolset and the core Heat template collection to the Red Hat OpenStack Platform 11 release.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

Procedure

1. Log into the undercloud as the **stack** user.
2. Disable the current OpenStack Platform repository:

```
[stack@director ~]$ sudo subscription-manager repos --disable=rhel-7-server-openstack-10-rpms
```

3. Enable the new OpenStack Platform repository:

```
[stack@director ~]$ sudo subscription-manager repos --enable=rhel-7-server-openstack-11-rpms
```

4. Stop the main OpenStack Platform services:

```
[stack@director ~]$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```

NOTE

This causes a short period of downtime for the undercloud. The overcloud is still functional during the undercloud upgrade.

5. The default Provisioning/Control Plane network has changed from **192.0.2.0/24** to **192.168.24.0/24**. If you used default network values in the previous **undercloud.conf** file, the Provisioning/Control Plane network is set to **192.0.2.0/24**. This means you need to set certain parameters in the **undercloud.conf** file to continue using the **192.0.2.0/24** network. These parameters are:
   - **local_ip**
   - **network_gateway**

- **undercloud_public_vip**
- **undercloud_admin_vip**
- **network_cidr**
- **masquerade_network**
- **dhcp_start**
- **dhcp_end**

Set the network values in **undercloud.conf** to ensure continued use of the **192.0.2.0/24** CIDR during future upgrades. Ensure the network configuration set correctly before running the **openstack undercloud upgrade** command.

6. Run **yum** to upgrade the director's main packages:

```
[stack@director ~]$ sudo yum update instack-undercloud openstack-
puppet-modules openstack-tripleo-common python-tripleoclient
```

7. Run the following command to upgrade the undercloud:

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

8. Wait until the undercloud upgrade process completes.

You have upgraded the undercloud to the Red Hat OpenStack Platform 11 release.

Additional Resources

- For more information about the new driver and migration instructions, see the Appendix "Virtual Baseboard Management Controller (VBMC)" in the *Director Installation and Usage Guide*.

6.4.2. Upgrading the Undercloud to Red Hat OpenStack Platform 12

This procedure upgrades the undercloud toolset and the core Heat template collection to the Red Hat OpenStack Platform 12 release.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

Procedure

1. Log into the undercloud as the **stack** user.
2. Disable the current OpenStack Platform repository:

```
[stack@director ~]$ sudo subscription-manager repos --disable=rhel-
7-server-openstack-11-rpms
```

3. Enable the new OpenStack Platform repository:

```
[stack@director ~]$ sudo subscription-manager repos --enable=rhel-7-
server-openstack-12-rpms
```

4. Install the **ceph-ansible** package:

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

5. Run **yum** to upgrade the director's main packages:

```
[stack@director ~]$ sudo yum update python-tripleoclient
```

6. Edit the **/home/stack/undercloud.conf** file and check that the **enabled_drivers** parameter does not contain the **pxe_ssh** driver. This driver is deprecated in favor of the Virtual Baseboard Metal Controller (VBMC) and removed from Red Hat OpenStack Platform.

7. Run the following command to upgrade the undercloud:

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

8. Wait until the undercloud upgrade process completes.

You have upgraded the undercloud to the Red Hat OpenStack Platform 12 release.

Additional Resources

- For more information about the new driver and migration instructions, see the Appendix "Virtual Baseboard Management Controller (VBMC)" in the *Director Installation and Usage Guide*.

6.4.3. Upgrading the Undercloud to Red Hat OpenStack Platform 13

This procedure upgrades the undercloud toolset and the core Heat template collection to the Red Hat OpenStack Platform 13 release.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

Procedure

1. Log into the undercloud as the **stack** user.

2. Disable the current OpenStack Platform repository:

```
[stack@director ~]$ sudo subscription-manager repos --disable=rhel-
7-server-openstack-12-rpms
```

3. Enable the new OpenStack Platform repository:

```
[stack@director ~]$ sudo subscription-manager repos --enable=rhel-7-
server-openstack-13-rpms
```

4. Run **yum** to upgrade the director's main packages:

```
[stack@director ~]$ sudo yum update python-tripleoclient
```

5. Run the following command to upgrade the undercloud:

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

6. Wait until the undercloud upgrade process completes.

7. Reboot the undercloud to update the operating system's kernel and other system packages:

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

8. Wait until the node boots.

You have upgraded the undercloud to the Red Hat OpenStack Platform 13 release.

Additional Resources

- For more information about the new driver and migration instructions, see the Appendix "Virtual Baseboard Management Controller (VBMC)" in the *Director Installation and Usage Guide*.

Next Step
Once the undercloud upgrade is complete, you can configure a source for the container images.

6.5. 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.

6.5.1. 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.

6.5.2. 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**.

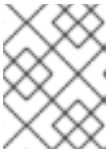
      --set ceph_tag

      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.

6.5.3. 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/custom-
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

- See Section 6.5.2, “Including Additional Container Images for Red Hat OpenStack Platform Services” for more details.

6.5.4. 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 10 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/custom-
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 the 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 Section 6.5.2, “Including Additional Container Images for Red Hat OpenStack Platform Services” for more details.

Next Steps

- Prepare the overcloud for an upgrade.

Additional Resources

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

6.6. PREPARING FOR THE OVERCLOUD UPGRADE

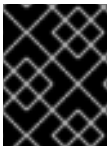
As a technician, you need to preparing the overcloud environment for upgrading the overcloud services.

6.6.1. Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.
- Upgraded the undercloud.
- Configured a container image source.

6.6.2. Preparing for Overcloud Upgrade Service Downtime

The overcloud upgrade process disables the main services at key points. This means you cannot use any overcloud services to create new resources during the upgrade duration. Workloads running in the overcloud remain active during this period, which means instances continue to run through the upgrade duration.



IMPORTANT

Plan a maintenance window to ensure no users can access the overcloud services for the duration of the upgrade.

Affected by overcloud upgrade

- OpenStack Platform services

Unaffected by overcloud upgrade

- Instances running during the upgrade

- Red Hat Ceph Storage OSDs
- Linux networking
- Open vSwitch networking
- Undercloud

6.6.3. Selecting a Hyperconverged OSD/Compute Node for Upgrade Testing

The overcloud upgrade process allows you to either:

- Upgrade all nodes in a role.
- Individual nodes separately.

To ensure a smooth overcloud upgrade process, it is useful to test the upgrade on one hyperconverged OSD/compute nodes in the environment before upgrading all the hyperconverged OSD/compute nodes. This ensures no major issues occur during the upgrade while maintaining minimal downtime to your workloads.

Use the following recommendations to help choose test nodes for the upgrade:

- Select a hyperconverged OSD/compute node for upgrade testing.
- Select a node without any critical instances running.
- If necessary, migrate critical instances from the selected test hyperconverged OSD/compute nodes to other hyperconverged OSD/compute nodes.

6.6.4. Using a New or an Existing Custom Roles Data

When using a custom roles file during the upgrade, there are two approaches to consider. You can create a new roles data file (**`roles_data_custom.yaml`**), see Section 6.6.5, “Generating a New Custom Roles Data File” for this procedure.

Or, you can use an existing roles data file from a previous deployment, for example, the **`custom-roles.yaml`** file. See Section 6.6.6, “New Composable Services”, Section 6.6.7, “Deprecated Composable Services”, Section 6.6.8, “Preparing for Composable Networks”, and Section 6.6.9, “Preparing for Deprecated Parameters” for more information on updating the custom roles data file accordingly.

Updating the roles data file ensures any new composable services will be added to the relevant roles in the environment.

6.6.5. Generating a New Custom Roles Data File

This procedure generates a new custom roles data file (**`roles_data_custom.yaml`**).

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

Procedure

1. As the **`stack`** user, source the **`stackrc`** file:

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

2. List the default role templates with the **`openstack overcloud role list`** command:

```
[stack@director ~]$ openstack overcloud role list
BlockStorage
CephStorage
Compute
ComputeHCI
ComputeOvsDpdk
Controller
...
```

3. View a role's YAML definition with the **`openstack overcloud role show`** command. For example:

```
[stack@director ~]$ openstack overcloud role show Compute
```

4. Generate a custom **`roles_data_custom.yaml`** file with the **`openstack overcloud roles generate`** command to join multiple predefined roles into a single file. For example, the following command joins the **`Controller`**, **`Compute`**, and **`CephStorage`** roles into a single file:

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

   The **`-o`** defines the name of the file to create.

5. This creates a new custom **`roles_data_custom.yaml`** file for the upgrade. If you need to add or remove composable services from any roles, then edit the file and make changes to suit the overcloud. If the **`OsdCompute`** role was used in a Red Hat Hyperconverged Infrastructure for Cloud v10 deployment, then you must replace **`ComputeHCI`** with **`OsdCompute`**.

Additional Resources

- For more information on custom **`roles_data_custom.yaml`** generation, see "Composable Services and Custom Roles" in the *Advanced Overcloud Customization* guide.

6.6.6. New Composable Services

Red Hat OpenStack Platform 13 contains new composable services. If you wish to manually edit an existing roles data file (**`roles_data.yaml`**), then use the following lists of new composable services for Red Hat OpenStack Platform roles. When generating a new custom roles data file (**`roles_data_custom.yaml`**) with their own roles, include these new compulsory services in their applicable roles.

NOTE

In a Red Hat Hyperconverged Infrastructure for Cloud v10 deployment, the custom role data used was the **`~/custom-templates/custom-roles.yaml`** file.

All Roles
The following new services apply to all roles.

`OS::TripleO::Services::MySQLClient`

Configures the MariaDB client on a node, which provides database configuration for other composable services. Add this service to all roles with standalone composable services.

`OS::TripleO::Services::Sshd`

Configures SSH access across all nodes. Used for instance migration.

`OS::TripleO::Services::CertmongerUser`

Allows the overcloud to require certificates from Certmonger. Only used if enabling TLS/SSL communication.

`OS::TripleO::Services::Docker`

Installs **`docker`** to manage containerized services.

`OS::TripleO::Services::ContainersLogrotateCrond`

Installs the **`logrotate`** service for container logs.

`OS::TripleO::Services::Securetty`

Allows configuration of **`securetty`** on nodes. Enabled with the **`environments/securetty.yaml`** environment file.

`OS::TripleO::Services::Tuned`

Enables and configures the Linux tuning daemon (**`tuned`**).

Specific Roles

The following new services apply to specific roles:

`OS::TripleO::Services::NovaPlacement`

Configures the OpenStack Compute (nova) Placement API. If using a standalone Nova API role in the current overcloud, add this service to the role. Otherwise, add the service to the Controller role.

`OS::TripleO::Services::PankoApi`

Configures the OpenStack Telemetry Event Storage (panko) service. If using a standalone Telemetry role in the current overcloud, add this service to the role. Otherwise, add the service to the Controller role.

`OS::TripleO::Services::Clustercheck`

Required on any role that also uses the **`OS::TripleO::Services::MySQL`** service, such as the **Controller** or standalone **Database** role.

`OS::TripleO::Services::Iscsid`

Configures the **`iscsid`** service on the **Controller**, **Compute**, and **BlockStorage** roles.

`OS::TripleO::Services::NovaMigrationTarget`

Configures the migration target service on **Compute** nodes.

Additional Resources

- For updated lists of services for specific custom roles, see the "Service Architecture: Standalone Roles" section in the *Advanced Overcloud Customization* guide.

6.6.7. Deprecated Composable Services

Always check for any deprecated services. When using a custom **`roles_data`** file, remove these services from their applicable roles.

`OS::TripleO::Services::Core`

This service acted as a core dependency for other Pacemaker services. This service has been removed to accommodate high availability composable services.

`OS::TripleO::Services::VipHosts`

This service configured the /etc/hosts file with node hostnames and IP addresses. This service is now integrated directly into the director's Heat templates.

Additional Resources

- For updated lists of services for specific custom roles, see the "Service Architecture: Standalone Roles" section in the *Advanced Overcloud Customization* guide.

6.6.8. Preparing for Composable Networks

This version of Red Hat OpenStack Platform introduces a new feature for composable networks. When using a custom **`roles_data`** file, edit the file to add the composable networks to each role. For example, the controller nodes:

```
- name: Controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
    - Tenant
```

Check the default **`/usr/share/openstack-tripleo-heat-templates/roles_data.yaml`** file for more examples of the syntax. Also, check the example role snippets in **`/usr/share/openstack-tripleo-heat-templates/roles`**.

Table 6.1. Mapping of composable networks to custom standalone roles

Role	Networks Required
Ceph Storage Monitor	**`Storage`**, **`StorageMgmt`**
Ceph Storage OSD	**`Storage`**, **`StorageMgmt`**
Ceph Storage RadosGW	**`Storage`**, **`StorageMgmt`**
Cinder API	**`InternalApi`**
Compute	**`InternalApi`**, **`Tenant`**, **`Storage`**
Controller	**`External`**, **`InternalApi`**, **`Storage`**, **`StorageMgmt`**, **`Tenant`**
Database	**`InternalApi`**
Glance	**`InternalApi`**
Heat	**`InternalApi`**

Horizon	**InternalApi**
Ironic	None required. Uses the Provisioning/Control Plane network for API.
Keystone	**InternalApi**
Load Balancer	**External**, **InternalApi**, **Storage**, **StorageMgmt**, **Tenant**
Manila	**InternalApi**
Message Bus	**InternalApi**
Networker	**InternalApi**, **Tenant**
Neutron API	**InternalApi**
Nova	**InternalApi**
OpenDaylight	**External**, **InternalApi**, **Tenant**
Redis	**InternalApi**
Sahara	**InternalApi**
Swift API	**Storage**
Swift Storage	**StorageMgmt**
Telemetry	**InternalApi**

6.6.9. Preparing for Deprecated Parameters

The following parameters are deprecated and have been replaced with role-specific parameters. If any of these deprecated parameters are being used, then update these parameters in the custom environment files accordingly.

Table 6.2. Deprecated Parameters

Old Parameter	New Parameter
controllerExtraConfig	**ControllerExtraConfig**
OvercloudControlFlavor	**OvercloudControllerFlavor**
controllerImage	**ControllerImage**
NovaImage	**ComputeImage**

NovaComputeExtraConfig	**ComputeExtraConfig**
NovaComputeServerMetadata	**ComputeServerMetadata**
NovaComputeSchedulerHints	**ComputeSchedulerHints**
NovaComputeIPs	**ComputeIPs**
SwiftStorageServerMetadata	**ObjectStorageServerMetadata**
SwiftStorageIPs	**ObjectStorageIPs**
SwiftStorageImage	**ObjectStorageImage**
OvercloudSwiftStorageFlavor	**OvercloudObjectStorageFlavor**

6.6.10. Software Repositories for Fast-Forward Upgrades

The fast-forward upgrade process uses a new method to switch software repositories. The director passes the following upstream codenames of each OpenStack Platform version to the script:

Codename	Version
ocata	OpenStack Platform 11
pike	OpenStack Platform 12
queens	OpenStack Platform 13

By default, the fast-forward upgrade process uses a script to change software repositories contained on Red Hat's Content Delivery Network (CDN) during each stage of the upgrade process. This script is included as part of the **OS::TripleO::Services::TripleoPackages** composable service (**puppet/services/tripleo-packages.yaml**) using the **FastForwardCustomRepoScriptContent** parameter.

You can also use a custom script by placing the commands underneath the **FastForwardCustomRepoScriptContent** parameter.

```
parameter_defaults:
  FastForwardCustomRepoScriptContent: |
    [INSERT UPGRADE SCRIPT HERE]
```

Example

```
parameter_defaults:
  FastForwardCustomRepoScriptContent: |
    set -e
    URL="satellite.example.com"
```

```
    case $1 in
      ocata)
        subscription-manager register --baseurl=https://$URL --force --
activationkey=rhosp11 --org=Default_Organization
        ;;
      pike)
        subscription-manager register --baseurl=https://$URL --force --
activationkey=rhosp12 --org=Default_Organization
        ;;
      queens)
        subscription-manager register --baseurl=https://$URL --force --
activationkey=rhosp13 --org=Default_Organization
        ;;
      *)
        echo "unknown release $1" >&2
        exit 1
    esac
```

Additional Resources

- See the Red Hat OpenStack Platform 13 Fast-Forward Upgrade guide for more information.

6.6.11. Preparing for the Red Hat Ceph Storage Upgrade

Due to the upgrade to containerized services, the method for installing and updating Ceph Storage nodes has changed. Ceph Storage configuration now uses a set of playbooks in the **`ceph-ansible`** package, which you install on the undercloud.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

Procedure

1. Check that you are using the latest resources and configuration in the storage environment file. This requires the following changes:

   a. The **`resource_registry`** uses containerized services from the **`docker/services/`** subdirectory of the core Heat template collection. For example:

   ```
   resource_registry:
     OS::TripleO::Services::CephMon: ../docker/services/ceph-
   ansible/ceph-mon.yaml
     OS::TripleO::Services::CephOSD: ../docker/services/ceph-
   ansible/ceph-osd.yaml
     OS::TripleO::Services::CephClient: ../docker/services/ceph-
   ansible/ceph-client.yaml
   ```

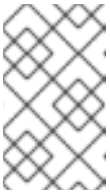
   b. Use the new **`CephAnsibleDisksConfig`** parameter to define how the disks are mapped. Previous versions of the Red Hat OpenStack Platform used the **`ceph::profile::params::osds`** hieradata to define the OSD layout. Convert this hieradata to the structure of the new **`CephAnsibleDisksConfig`** parameter. For example, if the hieradata contained the following:

   ```
   parameter_defaults:
     ExtraConfig:
   ```

```
    ceph::profile::params::osd_journal_size: 512
    ceph::profile::params::osds:
      '/dev/sdb': {}
      '/dev/sdc': {}
      '/dev/sdd': {}
```

Then the **CephAnsibleDisksConfig** would look like this:

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
    journal_size: 512
    osd_scenario: collocated
```



NOTE

For a full list of OSD disk layout options used with **ceph-ansible**, view the sample file in **/usr/share/ceph-ansible/group_vars/osds.yml.sample**.

6.6.12. Preparing Access to the Undercloud's Public API over SSL/TLS

The overcloud requires access to the undercloud's OpenStack Object Storage (swift) Public API during the upgrade. If the undercloud uses a self-signed certificate, then you need to add the undercloud's certificate authority to each overcloud node.

Prerequisites

- Using SSL/TLS for the undercloud public API.

Procedure

1. The undercloud's dynamic Ansible script has updated to the OpenStack Platform 12 version, which uses the **RoleNetHostnameMap** Heat parameter in the overcloud plan to define the inventory. However, the overcloud currently uses the OpenStack Platform 11 template versions, which do not have the **RoleNetHostnameMap** parameter. This means you need to create a temporary static inventory file, which you can generate with the following command:

```
[stack@director ~]$ openstack server list -c Networks -f value | cut
-d"=" -f2 > overcloud_hosts
```

2. Create an Ansible playbook (**undercloud-ca.yml**) that contains the following:

```
- name: Add undercloud CA to overcloud nodes
  hosts: all
  user: heat-admin
  become: true
  tasks:
    - name: Copy undercloud CA
      copy:
        src: /etc/pki/ca-trust/source/anchors/cm-local-ca.pem
```

```
        dest: /etc/pki/ca-trust/source/anchors/
    - name: Update trust
      command: "update-ca-trust extract"
    - name: Get the swift endpoint
      shell: |
        source ~stack/stackrc
        openstack endpoint list -c 'Service Name' -c 'URL' -c
Interface -f value | grep swift | grep public | awk -F/ '{print $3}'
      register: swift_endpoint
      delegate_to: 127.0.0.1
      become: yes
      become_user: stack
    - debug:
        var: swift_endpoint
    - name: Verify URL
      uri:
        url: https://{{ swift_endpoint.stdout }}/healthcheck
        return_content: yes
      register: verify
    - name: Report output
      debug:
        msg: "{{ ansible_hostname }} can access the undercloud's
Public API"
      when: verify.content == "OK"
```

This playbook contains multiple tasks that perform the following on each node:

- Copy the undercloud's certificate authority file (**`ca.crt.pem`**) to the overcloud node. The name of this file and its location might vary depending on the configuration. This example uses the name and location defined during the self-signed certificate procedure.
- Execute the command to update the certificate authority trust database on the overcloud node.
- Checks the undercloud's Object Storage Public API from the overcloud node and reports if successful.

3. Run the playbook with the following command:

```
[stack@director ~]$ ansible-playbook -i overcloud_hosts undercloud-
ca.yml
```

This uses the temporary inventory to provide Ansible with the overcloud nodes.

4. The resulting Ansible output should show a debug message for node. For example:

```
ok: [192.168.24.100] => {
    "msg": "overcloud-controller-0 can access the undercloud's
Public API"
}
```

Additional Resources

- For more information on running Ansible automation on the overcloud, see "Running Ansible Automation" in the *Director Installation and Usage* guide.

- For more information on configuring SSL/TLS, see "SSL/TLS Certificate Configuration" in the *Director Installation and Usage* guide.

6.6.13. Next Steps

- Performing an upgrade of the overcloud.

6.7. UPGRADING THE OVERCLOUD

As a technician, after you have prepared for the overcloud upgrade, now it is time to perform the actual overcloud upgrade.

6.7.1. Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.
- Upgraded the undercloud.
- Configured a container image source.
- Prepared for the overcloud upgrade.

6.7.2. Performing the Fast Forward Upgrade of the Overcloud

The fast forward upgrade requires running two commands that perform the following tasks:

- Updates the overcloud plan to OpenStack Platform 13.
- Prepares the nodes for the fast forward upgrade.
- Runs through upgrade steps of each subsequent version within the fast forward upgrade, including:
  - Performs version-specific tasks for each OpenStack Platform service.
  - Changes the repository to OpenStack Platform version within the fast forward upgrade.
  - Performs package and database upgrades for each subsequent version.
- Prepares the overcloud for the final upgrade to OpenStack Platform 13.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.
- Upgraded the undercloud.
- Configured a container image source.
- Prepared for the Overcloud Upgrade.

Procedure

1. As the **stack** user, source the **stackrc** file:

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

2. Run the fast forward upgrade preparation command:

   Example

   ```
   [stack@director ~]$ openstack overcloud ffwd-upgrade prepare --
   templates \
     -e $ALL_ENVIRONMENTS_USED_TO_DEPLOY \
     -e /home/stack/custom-templates/overcloud_images.yaml \
     -r /home/stack/custom-templates/roles_data_custom.yaml
   ```

   Include the following options relevant to the environment:

   - The **`$ALL_ENVIRONMENT_FILES_TO_DEPLOY`** represents all the environment files used during the initial deployment.
   - Path to custom configuration environment files (**`-e`**).
   - Path to the custom roles data file (**`-r`**):
     - Using a new **`roles_data_custom.yaml`** file.
     - Or, using an existing **`custom-roles.yaml`** file.
   - If used, a custom repository file.

3. Wait until the fast forward upgrade preparation completes.
4. Run the fast forward upgrade command:

   ```
   [stack@director ~]$ openstack overcloud ffwd-upgrade run
   ```

5. Wait until the fast forward upgrade completes.

6.7.3. Upgrading the Overcloud First Checkpoint

The following list is the state of the overcloud upgrade process at this first checkpoint:

- The overcloud packages and database have been upgraded to Red Hat OpenStack Platform 12 versions.
- All overcloud services are disabled.
- Ceph Storage nodes are at version 2.

The overcloud is now at a state to perform the standard upgrade steps to reach OpenStack Platform 13.

6.7.4. Upgrading the Controller Nodes

This process upgrades all the Controller nodes to OpenStack Platform 13. The process involves running the **`openstack overcloud upgrade run`** command and including the **`--roles Controller`** option to restrict operations to the Controller nodes only.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.

- Upgraded the undercloud.
- Configured a container image source.
- Prepared for the Overcloud Upgrade.

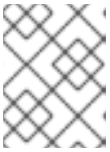
Procedure

1. As the **stack** user, source the **stackrc** file:

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

2. Run the upgrade command:

```
[stack@director ~]$ openstack overcloud upgrade run --roles
Controller --skip-tags validation
```



NOTE

The command uses **--skip-tags validation** because OpenStack Platform services are inactive on the overcloud and cannot be validated.

3. Wait until the Controller node upgrade completes.

6.7.5. Upgrading the Overcloud Second Checkpoint

The following list is the state of the overcloud upgrade process at this second checkpoint:

- The controller nodes and other nodes based on composable services have been upgraded to Red Hat OpenStack Platform 13 and all services are enabled.
- Upgrading the compute nodes has not started yet.
- The Red Hat Ceph Storage nodes are still at version 2 and have not been upgraded yet.

WARNING

Although the controller services are enabled, do not perform any workload operations while the compute node and the Red Hat Ceph Storage services are disabled. This can cause orphaned virtual machines. Wait until the entire environment is upgraded.

6.7.6. Upgrading the Test Hyperconverged OSD/Compute Node

This process upgrades the hyperconverged OSD/compute nodes selected for testing. The process involves running the **openstack overcloud upgrade run** command and including the **--nodes** option to restrict operations to the test nodes only. This procedure uses **--nodes compute-0** as an example in commands.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.
- Upgraded the undercloud.
- Configured a container image source.
- Prepared for the Overcloud Upgrade.

Procedure

1. Source the **stackrc** file:

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

2. Run the upgrade command:

Example

```
[stack@director ~]$ openstack overcloud upgrade run --nodes
osdcompute-0 --skip-tags validation
```

NOTE

This command uses the **--skip-tags validation** option, because the Red Hat OpenStack Platform services are inactive on the overcloud and cannot be validated.

3. Wait until the test node upgrade completes.

6.7.7. Upgrading the Hyperconverged OSD/Compute Nodes

This process upgrades all remaining Compute nodes to OpenStack Platform 13. The process involves running the **openstack overcloud upgrade run** command and including the **--roles OsdCompute** option to restrict operations to the Compute nodes only.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.
- Upgraded the undercloud.
- Configured a container image source.
- Prepared for the Overcloud Upgrade.

Procedure

1. As the **stack** user, source the **stackrc** file:

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

2. Run the upgrade command:

```
[stack@director ~]$ openstack overcloud upgrade run --roles
OsdCompute --skip-tags validation
```

NOTE

The command uses the **`--skip-tags validation`** option, because the Red Hat OpenStack Platform services are inactive on the overcloud and cannot be validated.

3. Wait until the hyperconverged OSD/compute node upgrade completes.

6.7.8. Upgrading the Overcloud Third Checkpoint

The following list is the state of the overcloud upgrade process at this third checkpoint:

- The controller nodes and other nodes based on composable services have been upgraded to Red Hat OpenStack Platform 13 and all services enabled.
- Compute nodes have been upgraded to Red Hat OpenStack Platform 13.
- The Red Hat Ceph Storage nodes are still at version 2 and have not been upgraded yet.

6.7.9. Upgrading Red Hat Ceph Storage

This process upgrades the Ceph Storage nodes. The process involves:

- Running the **`openstack overcloud ceph-upgrade run`** command to perform a rolling upgrade to a containerized Red Hat Ceph Storage 3 cluster.

Prerequisites

- A running Red Hat Hyperconverged Infrastructure for Cloud 10 environment.
- Upgraded the undercloud.
- Configured a container image source.
- The controller and compute nodes have been upgraded.

Procedure

1. As the **`stack`** user, source the **`stackrc`** file:

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

2. Run the Ceph Storage upgrade command. For example:

Example

```
[stack@director ~]$ openstack overcloud ceph-upgrade run --templates
\
    -e $ALL_ENVIRONMENT_FILES_TO_DEPLOY \
    -e /home/stack/custom-templates/overcloud_images.yaml \
    -r /home/stack/custom-templates/roles_data_custom.yaml \
    --ceph-ansible-playbook '/usr/share/ceph-ansible/infrastructure-
```

```
playbooks/switch-from-non-containerized-to-containerized-ceph-
daemons.yml,/usr/share/ceph-ansible/infrastructure-
playbooks/rolling_update.yml'
```

Include the following options relevant to the environment:

- The **$ALL_ENVIRONMENT_FILES_TO_DEPLOY** represents all the environment files used during the initial deployment.
- Path to custom configuration environment files (**-e**).
- Path to the custom roles data file (**-r**):
  - Using a new **roles_data_custom.yaml** file.
  - Or, using an existing **custom-roles.yaml** file.
- If used, a custom repository file.
- The following ansible playbooks:
  - **/usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-to-containerized-ceph-daemons.yml**
  - **/usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml**

3. Wait until the Ceph Storage node upgrade completes.

6.7.10. Upgrading the Overcloud Fourth Checkpoint

The following list is the state of the overcloud upgrade process at this fourth checkpoint:

- The controller nodes and other nodes based on composable services have been upgraded to Red Hat OpenStack Platform 13 and all services enabled.
- Compute nodes have been upgraded to Red Hat OpenStack Platform 13.
- The Red Hat Ceph Storage nodes have been upgraded to version 3.

Although the environment is now upgraded, you must perform one last step to finalize the upgrade.

6.7.11. Finalizing the Fast Forward Upgrade

The fast forward upgrade requires a final step to update the overcloud stack. This ensures the stack's resource structure aligns with a regular deployment of OpenStack Platform 13 and allows you to perform standard **openstack overcloud deploy** functions in the future.

Prerequisites

- An upgrade from Red Hat Hyperconverged Infrastructure for Cloud 10 to 13.

Procedure

1. As the **stack** user, source the **stackrc** file:

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

2. Run the fast forward upgrade finalization command:

Example

```
[stack@director ~]$ openstack overcloud ffwd-upgrade converge \
    -e $ALL_ENVIRONMENT_FILES_TO_DEPLOY
    -e /home/stack/custom-templates/overcloud_images.yaml \
    -r /home/stack/custom-templates/roles_data_custom.yaml
```

Include the following options relevant to your environment:

- The **$ALL_ENVIRONMENT_FILES_TO_DEPLOY** represents all the environment files used during the initial deployment.
- Path to custom configuration environment files (**-e**).
- Path to the custom roles data file (**-r**):
  - Using a new **roles_data_custom.yaml** file.
  - Or, using an existing **custom-roles.yaml** file.

3. Wait until the fast forward upgrade finalization completes.

6.7.12. Next Steps

- Post-upgrade steps for the overcloud configuration.

6.8. DOING THE POST UPGRADE STEPS

This procedure implements final steps after completing the fast forward upgrade process. This includes an overcloud reboot and any additional configuration steps or considerations after the fast forward upgrade process completes.

Also, you need to replace the current overcloud images with the new versions. The new images ensure that the undercloud can introspect and provision the nodes using the latest version of Red Hat OpenStack Platform software.

Prerequisites

- You have upgraded the undercloud to the latest version.

Procedure

1. Remove any existing images from the **images** directory on the **stack** user's home (**/home/stack/images**):

```
[stack@director ~]$ rm -rf ~/images/*
```

2. Extract the archives:

```
[stack@director ~]$ cd ~/images
```

```
[stack@director ~]$ for i 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 $i; done
[stack@director ~]$ cd ~
```

3. Import the latest images into the undercloud:

```
[stack@director ~]$ openstack overcloud image upload --update-
existing --image-path /home/stack/images/
```

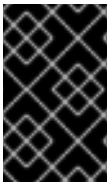
4. Configure the nodes to use the new images:

```
[stack@director ~]$ openstack overcloud node configure $(openstack
baremetal node list -c UUID -f value)
```

5. Verify the existence of the new images:

```
[stack@director ~]$ openstack image list
[stack@director ~]$ ls -l /httpboot
```

6. Reboot the overcloud nodes.



IMPORTANT

When deploying overcloud nodes, ensure the Overcloud image version corresponds to the respective Heat template version. For example, only use the Red Hat OpenStack Platform 13 images with the Red Hat OpenStack Platform 13 Heat templates.

6.9. ADDITIONAL RESOURCES

- See the Fast-Forward Upgrades Guide for more details.

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/custom-
templates/nic-configs/compute-nics.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: /home/stack/custom-
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 ~/custom-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:

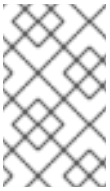
```
parameter_defaults:
  ExtraConfig:
    nova::compute::reserved_host_memory: $MEMORY_SIZE_IN_MB
    nova::cpu_allocation_ratio: $CPU_RATIO
```

   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.