Red Hat OpenStack Platform 14

Upgrading Red Hat OpenStack Platform

Upgrading a Red Hat OpenStack Platform environment

Last Updated: 2020-04-14

Red Hat OpenStack Platform 14 Upgrading Red Hat OpenStack Platform

Upgrading a Red Hat OpenStack Platform environment

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This document lays out the different methods through which users can upgrade from Red Hat OpenStack Platform 13 (Queens) to 14 (Rocky). These methods assume that you will be upgrading to and from an OpenStack deployment installed on Red Hat Enterprise Linux 7.

Table of Contents

CHAPTER 1. INTRODUCTION

This document provides a workflow to help upgrade your Red Hat OpenStack Platform environment to the latest major version and keep it updated with minor releases of that version.

This guide provides an upgrade path through the following versions:

Old Overcloud Version	New Overcloud Version
Red Hat OpenStack Platform 13	Red Hat OpenStack Platform 14

1.1. HIGH LEVEL WORKFLOW

The following table provides an outline of the steps required for the upgrade process:

Step	Description
Preparing your environment	Perform a backup of the database and configuration of the undercloud and overcloud Controller nodes. Update to the latest minor release. Validate the environment.
Preparing container images	Create an environment file containing the parameters to prepare container images for OpenStack Platform 14 services.
Upgrading the undercloud	Upgrade the undercloud from OpenStack Platform 13 to OpenStack Platform 14.
Preparing the overcloud	Perform relevant steps to transition your overcloud configuration files to OpenStack Platform 14.
Upgrading your Controller nodes	Upgrade all Controller nodes simultaneously to OpenStack Platform 14.
Upgrading your Compute nodes	Test the upgrade on selected Compute nodes. If the test succeeds, upgrade all Compute nodes.
Upgrading your Ceph Storage nodes	Upgrade all Ceph Storage nodes. This includes an upgrade to containerized version of Red Hat Ceph Storage 3.
Finalize the upgrade	Run the convergence command to refresh your overcloud stack.

1.2. MAJOR CHANGES

The following is a high-level list of major changes that occur during the upgrade.

- The undercloud now uses containers to run services. The undercloud also uses the same architecture that configures the overcloud.
- The director now uses a container preparation method to automatically obtain container images for the undercloud and overcloud.
- The overcloud now uses an Ansible-based Red Hat Subscription Management method. This replaces the previous **rhel-registration** method.
- Composable networks now define a list of routes.
- The Telemetry API service is replaced by the OpenStack Telemetry Metrics (gnocchi) service and the OpenStack Telemetry Alarming (aodh) service APIs. The ceilometer-collector service is replaced by the ceilometer-notification-agent daemon. OpenStack Telemetry (ceilometer) as a whole has not been removed from OpenStack Platform 14.

1.3. BEFORE STARTING THE UPGRADE

Apply any firmware updates to your hardware before performing the upgrade.

CHAPTER 2. PREPARING FOR AN OPENSTACK PLATFORM UPGRADE

This process prepares your OpenStack Platform environment for a full update. This involves the following process:

- Update the undercloud packages and run the upgrade command
- Reboot the undercloud in case a newer kernel or newer system packages are installed
- Update the overcloud using the overcloud upgrade command
- Reboot the overcloud nodes in case a newer kernel or newer system packages are installed
- Perform a validation check on both the undercloud and overcloud

These procedures ensure your OpenStack Platform environment is in the best possible state before proceeding with the upgrade.

2.1. UPDATING THE CURRENT OPENSTACK PLATFORM VERSION

Before performing the upgrade to the next version of OpenStack Platform, it is recommended to perform a minor version update to your existing undercloud and overcloud. This means performing a minor version update for OpenStack Platform 13.

Follow the instructions in the Keeping Red Hat OpenStack Platform Updated guide for OpenStack Platform 13.

2.2. VALIDATING THE UNDERCLOUD

The following is a set of steps to check the functionality of your undercloud.

Procedure

1. Source the undercloud access details:

```
$ source ~/stackrc
```

2. Check for failed Systemd services:

```
(undercloud) $ sudo systemctl list-units --state=failed 'openstack*' 'neutron*' 'httpd' 'docker'
```

3. Check the undercloud free space:

```
(undercloud) $ df -h
```

   Use the "Undercloud Reqirements" as a basis to determine if you have adequate free space.

4. If you have NTP installed on the undercloud, check that clocks are synchronized:

```
(undercloud) $ sudo ntpstat
```

5. Check the undercloud network services:

```
(undercloud) $ openstack network agent list
```

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

6. Check the undercloud compute services:

```
(undercloud) $ openstack compute service list
```

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

Related Information

- The following solution article shows how to remove deleted stack entries in your OpenStack Orchestration (heat) database: https://access.redhat.com/solutions/2215131

2.3. VALIDATING A CONTAINERIZED OVERCLOUD

The following is a set of steps to check the functionality of your containerized overcloud.

Procedure

1. Source the undercloud access details:

```
$ source ~/stackrc
```

2. Check the status of your bare metal nodes:

```
(undercloud) $ 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:

```
(undercloud) $ 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 for failed containerized services:

```
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker ps -f 'exited=1' --all" ; done
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker ps -f 'status=dead' -f
'status=restarting'" ; done
```

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

```
(undercloud) $ 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 /var/lib/config-
data/puppet-generated/haproxy/etc/haproxy/haproxy.cfg'
```

Use these details in the following cURL request:

```
(undercloud) $ 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.

6. Check overcloud database replication health:

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

7. Check RabbitMQ cluster health:

```
(undercloud) $ for NODE in $(openstack server list --name controller -f value -c Networks | cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker exec $(ssh heat-admin@$NODE "sudo docker ps -f 'name=.*rabbitmq.*' -q") rabbitmqctl node_health_check" ; done
```

8. Check Pacemaker resource health:

```
(undercloud) $ 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.

9. Check the disk space on each overcloud node:

```
(undercloud) $ 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
```

10. Check overcloud Ceph Storage cluster health. The following command runs the **ceph** tool on a Controller node to check the cluster:

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

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

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

12. Check that clocks are synchronized on overcloud nodes

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

13. Source the overcloud access details:

```
(undercloud) $ source ~/overcloudrc
```

14. Check the overcloud network services:

```
(overcloud) $ openstack network agent list
```

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

15. Check the overcloud compute services:

```
(overcloud) $ openstack compute service list
```

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

16. Check the overcloud volume services:

```
(overcloud) $ openstack volume service list
```

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

Related Information

- 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 your Red Hat OpenStack Platform environment and tune the configuration to Red Hat's recommendations.

CHAPTER 3. UPGRADING THE UNDERCLOUD

This process upgrades the undercloud and its overcloud images to **Red Hat OpenStack Platform 14**.

3.1. CONVERTING TO NEXT GENERATION POWER MANAGEMENT DRIVERS

Red Hat OpenStack Platform now uses next generation drivers, also known as *hardware types*, that replace older drivers.

The following table shows an analogous comparison between older drivers with their next generation hardware type equivalent:

Old Driver	New Hardware Type
pxe_ipmitool	**ipmi**
pxe_drac	**idrac**
pxe_ilo	**ilo**
pxe_ucs	**cisco-ucs-managed**
pxe_irmc	**irmc**
VBMC (**pxe_ipmitool**)	**ipmi**
fake_pxe	**fake-hardware**

In OpenStack Platform 14, these older drivers have been removed and are no longer accessible. You must change to hardware types **before** upgrading to OpenStack Platform 14.

Procedure

1. Check the current list of hardware types enabled:

```
$ source ~/stackrc
$ openstack baremetal driver list --type dynamic
```

2. If you use a hardware type driver that is not enabled, enable the driver using the **enabled_hardware_types** parameter in the **undercloud.conf** file:

```
enabled_hardware_types = ipmi,redfish,idrac
```

3. Save the file and refresh the undercloud:

```
$ openstack undercloud install
```

4. Run the following commands, substituting the **OLDDRIVER** and **NEWDRIVER** variables for your power management type:

```
$ source ~/stackrc
$ OLDDRIVER="pxe_ipmitool"
$ NEWDRIVER="ipmi"
$ for NODE in $(openstack baremetal node list --driver $OLDDRIVER -c UUID -f value) ; do
openstack baremetal node set $NODE --driver $NEWDRIVER; done
```

3.2. UPGRADING THE DIRECTOR PACKAGES TO OPENSTACK PLATFORM 14

This procedure upgrades the director toolset and the core Heat template collection to the **OpenStack Platform 14** release.

Procedure

1. Log in to the director as the **stack** user.
2. Disable the current OpenStack Platform repository:

```
$ sudo subscription-manager repos --disable=rhel-7-server-openstack-13-rpms
```

3. Enable the new OpenStack Platform repository:

```
$ sudo subscription-manager repos --enable=rhel-7-server-openstack-14-rpms
```

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

```
$ sudo yum update -y python-tripleoclient* openstack-tripleo-common openstack-tripleo-
heat-templates
```

3.3. PREPARING CONTAINER IMAGES

The overcloud configuration requires initial registry configuration to determine where to obtain images and how to store them. Complete the following steps to generate and customize an environment file for preparing your container images.

Procedure

1. Log in to your undercloud host as the stack user.
2. Generate the default container image preparation file:

```
$ openstack tripleo container image prepare default \
  --local-push-destination \
  --output-env-file containers-prepare-parameter.yaml
```

This command includes the following additional options:

- **--local-push-destination** sets the registry on the undercloud as the location for container images. This means the director pulls the necessary images from the Red Hat Container

Catalog and pushes them to the registry on the undercloud. The director uses this registry as the container image source. To pull directly from the Red Hat Container Catalog, omit this option.

- **--output-env-file** is an environment file name. The contents of this file include the parameters for preparing your container images. In this case, the name of the file is **containers-prepare-parameter.yaml**.

NOTE

You can also use the same **containers-prepare-parameter.yaml** file to define a container image source for both the undercloud and the overcloud.

3. Edit the **containers-prepare-parameter.yaml** and make the modifications to suit your requirements.

3.4. CONTAINER IMAGE PREPARATION PARAMETERS

The default file for preparing your containers (**containers-prepare-parameter.yaml**) contains the **ContainerImagePrepare** Heat parameter. This parameter defines a list of strategies for preparing a set of images:

```
parameter_defaults:
  ContainerImagePrepare:
  - (strategy one)
  - (strategy two)
  - (strategy three)
  ...
```

Each strategy accepts a set of sub-parameters that define which images to use and what to do with them. The following table contains information about the sub-parameters you can use with each **ContainerImagePrepare** strategy:

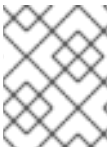
Parameter	Description
excludes	List of image name substrings to exclude from a strategy.
includes	List of image name substrings to include in a strategy. At least one image name must match an existing image. All **excludes** are ignored if **includes** is specified.
modify_append_tag	String to append to the tag for the destination image. For example, if you pull an image with the tag **14.0-89** and set the **modify_append_tag** to **-hotfix**, the director tags the final image as **14.0-89-hotfix**.
modify_only_with_labels	A dictionary of image labels that filter the images to modify. If an image matches the labels defined, the director includes the image in the modification process.

Parameter	Description
modify_role	String of ansible role names to run during upload but before pushing the image to the destination registry.
modify_vars	Dictionary of variables to pass to **modify_role**.
push_destination	The namespace of the registry to push images during the upload process. When you specify a namespace for this parameter, all image parameters use this namespace too. If set to **true**, the **push_destination** is set to the undercloud registry namespace. It is not recommended to set this parameters to **false** in production environments.
pull_source	The source registry from where to pull the original container images.
set	A dictionary of **key: value** definitions that define where to obtain the initial images.
tag_from_label	Defines the label pattern to tag the resulting images. Usually sets to **{version}-{release}**.

The **set** parameter accepts a set of **key: value** definitions. The following table contains information about the keys:

Key	Description
ceph_image	The name of the Ceph Storage container image.
ceph_namespace	The namespace of the Ceph Storage container image.
ceph_tag	The tag of the Ceph Storage container image.
name_prefix	A prefix for each OpenStack service image.
name_suffix	A suffix for each OpenStack service image.
namespace	The namespace for each OpenStack service image.
neutron_driver	The driver to use to determine which OpenStack Networking (neutron) container to use. Use a null value to set to the standard **neutron-server** container. Set to **ovn** to use OVN-based containers. Set to **odl** to use OpenDaylight-based containers.

Key	Description
tag	The tag that the director uses to identify the images to pull from the source registry. You usually keep this key set to **latest**.



NOTE

The **set** section might contains several parameters that begin with **openshift_**. These parameters are for various scenarios involving OpenShift-on-OpenStack.

3.5. CHECKING THE DIRECTOR CONFIGURATION

Check the **/usr/share/python-tripleoclient/undercloud.conf.sample** for new or deprecated parameters that might be applicable to your environment. Modify these parameters in your current **/home/stack/undercloud.conf** file. In particular, note the following parameters:

- **container_images_file**, which you should set to the absolute location of your **containers-prepare-parameter.yaml** file.
- **enabled_drivers**, which you should remove. The older drivers have now been replaced by **hardware_types**.
- **generate_service_certificate**, which now defaults to **true**. Switch to **false** if your undercloud did not originally use SSL and you have no intention to enable SSL . Note that enabling SSL on the undercloud requires providing extra environment files during upgrade to establish trust between the undercloud and overcloud nodes

3.6. DIRECTOR CONFIGURATION PARAMETERS

The following list contains information about parameters for configuring the **undercloud.conf** file. Keep all parameters within their relevant sections to avoid errors.

Defaults

The following parameters are defined in the **[DEFAULT]** section of the **undercloud.conf** file:

additional_architectures

A list of additional (kernel) architectures that an overcloud supports. Currently the overcloud supports **ppc64le** architecture.

NOTE

When enabling support for ppc64le, you must also set **ipxe_enabled** to **False**

certificate_generation_ca

The **certmonger** nickname of the CA that signs the requested certificate. Use this option only if you have set the **generate_service_certificate** parameter. If you select the **local** CA, certmonger extracts the local CA certificate to **/etc/pki/ca-trust/source/anchors/cm-local-ca.pem** and adds the certificate to the trust chain.

clean_nodes

Defines whether to wipe the hard drive between deployments and after introspection.

cleanup

Cleanup temporary files. Set this to **False** to leave the temporary files used during deployment in place after the command is run. This is useful for debugging the generated files or if errors occur.

container_images_file

Heat environment file with container image information. This can either be:

- Parameters for all required container images
- Or the **ContainerImagePrepare** parameter to drive the required image preparation. Usually the file containing this parameter is named **containers-prepare-parameter.yaml**.

custom_env_files

Additional environment file to add to the undercloud installation.

deployment_user

The user installing the undercloud. Leave this parameter unset to use the current default user (**stack**).

discovery_default_driver

Sets the default driver for automatically enrolled nodes. Requires **enable_node_discovery** enabled and you must include the driver in the **enabled_hardware_types** list.

docker_insecure_registries

A list of insecure registries for **docker** to use. Use this parameter if you want to pull images from another source, such as a private container registry. In most cases, docker has the certificates to pull container images from either the Red Hat Container Catalog or from your Satellite server if the undercloud is registered to Satellite.

docker_registry_mirror

An optional **registry-mirror** configured in **/etc/docker/daemon.json**

enable_ironic; enable_ironic_inspector; enable_mistral; enable_tempest; enable_validations; enable_zaqar

Defines the core services to enable for director. Leave these parameters set to **true**.

enable_ui

Defines whether to install the director web UI. Use this parameter to perform overcloud planning and deployments through a graphical web interface. Note that the UI is only available with SSL/TLS enabled using either the **undercloud_service_certificate** or **generate_service_certificate**.

enable_node_discovery

Automatically enroll any unknown node that PXE-boots the introspection ramdisk. New nodes use the **fake_pxe** driver as a default but you can set **discovery_default_driver** to override. You can also use introspection rules to specify driver information for newly enrolled nodes.

enable_novajoin

Defines whether to install the **novajoin** metadata service in the Undercloud.

enable_routed_networks

Defines whether to enable support for routed control plane networks.

enable_swift_encryption

Defines whether to enable Swift encryption at-rest.

enable_telemetry

Defines whether to install OpenStack Telemetry services (gnocchi, aodh, panko) in the undercloud. Set **enable_telemetry** parameter to **true** if you want to install and configure telemetry services automatically. The default value is **false**, which disables telemetry on the undercloud. This parameter is required if using other products that consume metrics data, such as Red Hat CloudForms.

enabled_hardware_types

A list of hardware types to enable for the undercloud.

generate_service_certificate

Defines whether to generate an SSL/TLS certificate during the undercloud installation, which is used for the **undercloud_service_certificate** parameter. The undercloud installation saves the resulting certificate **/etc/pki/tls/certs/undercloud-[undercloud_public_vip].pem**. The CA defined in the **certificate_generation_ca** parameter signs this certificate.

heat_container_image

URL for the heat container image to use. Leave unset.

heat_native

Use native heat templates. Leave as **true**.

hieradata_override

Path to **hieradata** override file that configures Puppet hieradata on the director, providing custom configuration to services beyond the **undercloud.conf** parameters. If set, the undercloud installation copies this file to the **/etc/puppet/hieradata** directory and sets it as the first file in the hierarchy. See Configuring hieradata on the undercloud for details on using this feature.

inspection_extras

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

inspection_interface

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

inspection_runbench

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

ipa_otp

Defines the one time password to register the Undercloud node to an IPA server. This is required when **enable_novajoin** is enabled.

ipxe_enabled

Defines whether to use iPXE or standard PXE. The default is **true**, which enables iPXE. Set to **false** to set to standard PXE.

local_interface

The chosen interface for the director's Provisioning NIC. This is also the device the director uses for DHCP and PXE boot services. Change this value to your chosen device. To see which device is connected, use the **ip addr** command. For example, this is the result of an **ip addr** command:

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
1000
    link/ether 52:54:00:75:24:09 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.178/24 brd 192.168.122.255 scope global dynamic eth0
       valid_lft 3462sec preferred_lft 3462sec
    inet6 fe80::5054:ff:fe75:2409/64 scope link
```

```
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noop state DOWN
    link/ether 42:0b:c2:a5:c1:26 brd ff:ff:ff:ff:ff:ff
```

In this example, the External NIC uses **eth0** and the Provisioning NIC uses **eth1**, which is currently not configured. In this case, set the **local_interface** to **eth1**. The configuration script attaches this interface to a custom bridge defined with the **inspection_interface** parameter.

local_ip

The IP address defined for the director's Provisioning NIC. This is also the IP address that the director uses for DHCP and PXE boot services. Leave this value as the default **192.168.24.1/24** unless you use a different subnet for the Provisioning network, for example, if it conflicts with an existing IP address or subnet in your environment.

local_mtu

MTU to use for the **local_interface**. Do not exceed 1500 for the undercloud.

local_subnet

The local subnet to use for PXE boot and DHCP interfaces. The **local_ip** address should reside in this subnet. The default is **ctlplane-subnet**.

net_config_override

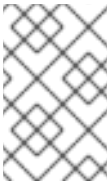
Path to network configuration override template. If you set this parameter, the undercloud uses a JSON format template to configure the networking with **os-net-config**. The undercloud ignores the network parameters set in **undercloud.conf**. See **/usr/share/python-tripleoclient/undercloud.conf.sample** for an example.

output_dir

Directory to output state, processed heat templates, and Ansible deployment files.

overcloud_domain_name

The DNS domain name to use when deploying the overcloud.



NOTE

When configuring the overcloud, the **CloudDomain** parameter must be set to a matching value. Set this parameter in an environment file when you configure your overcloud.

roles_file

The roles file to override for undercloud installation. It is highly recommended to leave unset so that the director installation uses the default roles file.

scheduler_max_attempts

Maximum number of times the scheduler attempts to deploy an instance. This value must be greater or equal to the number of bare metal nodes that you expect to deploy at once to work around potential race condition when scheduling.

service_principal

The Kerberos principal for the service using the certificate. Use this parameter only if your CA requires a Kerberos principal, such as in FreeIPA.

subnets

List of routed network subnets for provisioning and introspection. See Subnets for more information. The default value includes only the **ctlplane-subnet** subnet.

templates

Heat templates file to override.

undercloud_admin_host

The IP address defined for the director Admin API when using SSL/TLS. This is an IP address for administration endpoint access over SSL/TLS. The director configuration attaches the director's IP address to its software bridge as a routed IP address, which uses the **/32** netmask.

undercloud_debug

Sets the log level of undercloud services to **DEBUG**. Set this value to **true** to enable.

undercloud_enable_selinux

Enable or disable SELinux during the deployment. It is highly recommended to leave this value set to **true** unless you are debugging an issue.

undercloud_hostname

Defines the fully qualified host name for the undercloud. If set, the undercloud installation configures all system host name settings. If left unset, the undercloud uses the current host name, but the user must configure all system host name settings appropriately.

undercloud_log_file

The path to a log file to store the undercloud install/upgrade logs. By default, the log file is **install-undercloud.log** within the home directory. For example, **/home/stack/install-undercloud.log**.

undercloud_nameservers

A list of DNS nameservers to use for the undercloud hostname resolution.

undercloud_ntp_servers

A list of network time protocol servers to help synchronize the undercloud date and time.

undercloud_public_host

The IP address defined for the director Public API when using SSL/TLS. This is an IP address for accessing the director endpoints externally over SSL/TLS. The director configuration attaches this IP address to the director software bridge as a routed IP address, which uses the **/32** netmask.

undercloud_service_certificate

The location and filename of the certificate for OpenStack SSL/TLS communication. Ideally, you obtain this certificate from a trusted certificate authority. Otherwise, generate your own self-signed certificate.

undercloud_update_packages

Defines whether to update packages during the undercloud installation.

Subnets

Each provisioning subnet is a named section in the **undercloud.conf** file. For example, to create a subnet called **ctlplane-subnet**, use the following sample in your **undercloud.conf** file:

```
[ctlplane-subnet]
cidr = 192.168.24.0/24
dhcp_start = 192.168.24.5
dhcp_end = 192.168.24.24
inspection_iprange = 192.168.24.100,192.168.24.120
gateway = 192.168.24.1
masquerade = true
```

You can specify as many provisioning networks as necessary to suit your environment.

gateway

The gateway for the overcloud instances. This is the undercloud host, which forwards traffic to the External network. Leave this as the default **192.168.24.1** unless you use a different IP address for the director or want to use an external gateway directly.

NOTE

The director configuration also enables IP forwarding automatically using the relevant **sysctl** kernel parameter.

cidr

The network that the director uses to manage overcloud instances. This is the Provisioning network, which the undercloud **neutron** service manages. Leave this as the default **192.168.24.0/24** unless you use a different subnet for the Provisioning network.

masquerade

Defines whether to masquerade the network defined in the **cidr** for external access. This provides the Provisioning network with a degree of network address translation (NAT) so that the Provisioning network has external access through the director.

dhcp_start; dhcp_end

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

3.7. UPGRADING THE DIRECTOR

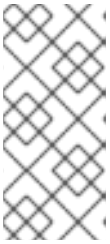
Complete the following steps to upgrade the director.

Procedure

1. Run the following command to upgrade the director on the undercloud:

```
$ openstack undercloud upgrade
```

This command launches the director configuration script. The director upgrades its packages and configures its services to suit the settings in the **undercloud.conf**. This script takes several minutes to complete.



NOTE

The director configuration script prompts for confirmation before proceeding. Bypass this confirmation using the **-y** option:

```
$ openstack undercloud upgrade -y
```

2. The script also starts all OpenStack Platform service containers on the undercloud automatically. Check the enabled containers using the following command:

```
[stack@director ~]$ sudo docker ps
```

3. The script adds the **stack** user to the **docker** group to ensure that the **stack** user has access to container management commands. Refresh the **stack** user permissions with the following command:

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

The command prompts you to log in again. Enter the stack user password.

4. To initialize the **stack** user to use the command line tools, run the following command:

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

The prompt now indicates OpenStack commands authenticate and execute against the undercloud;

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

The director upgrade is complete.

3.8. UPGRADING THE OVERCLOUD IMAGES

You must replace your current overcloud images with new versions. The new images ensure that the director can introspect and provision your nodes using the latest version of 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**):

```
$ rm -rf ~/images/*
```

2. Extract the archives:

```
$ cd ~/images
$ for i in /usr/share/rhosp-director-images/overcloud-full-latest-14.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-14.0.tar; do tar -xvf $i; done
$ cd ~
```

3. Import the latest images into the director:

```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

4. Configure your nodes to use the new images:

```
$ openstack overcloud node configure $(openstack baremetal node list -c UUID -f value)
```

5. Verify the existence of the new images:

```
$ openstack image list
$ ls -l /var/lib/ironic/httpboot/
```

IMPORTANT

When deploying overcloud nodes, ensure that the Overcloud image version corresponds to the respective Heat template version. For example, use only the OpenStack Platform 14 images with the OpenStack Platform 14 Heat templates.

3.9. UNDERCLOUD POST-UPGRADE NOTES

- If you use a local set of core templates in your **stack** user home directory, ensure that you update the templates using the recommended workflow in "Using Customized Core Heat Templates". You must update the local copy before upgrading the overcloud.

3.10. NEXT STEPS

The undercloud upgrade is complete. You can now prepare the overcloud for the upgrade.

CHAPTER 4. PREPARING FOR THE OVERCLOUD UPGRADE

This process prepares the overcloud for the upgrade process.

Prerequisites

- You have upgraded the undercloud to the latest version.

4.1. RED HAT SUBSCRIPTION MANAGER (RHSM) COMPOSABLE SERVICE

The **rhsm** composable service provides a method to register overcloud nodes through Ansible. Each role in the default **roles_data** file contains a **OS::TripleO::Services::Rhsm** resource, which is disabled by default. To enable the service, register the resource to the **rhsm** composable service file:

```
resource_registry:
  OS::TripleO::Services::Rhsm: /usr/share/openstack-tripleo-heat-
templates/extraconfig/services/rhsm.yaml
```

The **rhsm** composable service accepts a **RhsmVars** parameter, which allows you to define multiple sub-parameters relevant to your registration. For example:

```
parameter_defaults:
  RhsmVars:
    rhsm_repos:
      - rhel-7-server-rpms
      - rhel-7-server-extras-rpms
      - rhel-7-server-rh-common-rpms
      - rhel-ha-for-rhel-7-server-rpms
      - rhel-7-server-openstack-14-rpms
      - rhel-7-server-rhceph-3-osd-rpms
      - rhel-7-server-rhceph-3-mon-rpms
      - rhel-7-server-rhceph-3-tools-rpms
    rhsm_username: "myusername"
    rhsm_password: "p@55w0rd!"
    rhsm_org_id: "1234567"
```

You can also use the **RhsmVars** parameter in combination with role-specific parameters (e.g. **ControllerParameters**) to provide flexibility when enabling specific repositories for different nodes types.

4.2. SWITCHING TO THE RHSM COMPOSABLE SERVICE

The previous **rhel-registration** method runs a bash script to handle the overcloud registration. The scripts and environment files for this method are located in the core Heat template collection at **/usr/share/openstack-tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/**.

Complete the following steps to switch from the **rhel-registration** method to the **rhsm** composable service.

Procedure

1. Exclude the **rhel-registration** environment files from future deployments operations. In most cases, exclude the following files:
   - **rhel-registration/environment-rhel-registration.yaml**
   - **rhel-registration/rhel-registration-resource-registry.yaml**
2. If you use a custom **roles_data** file, ensure that each role in your **roles_data** file contains the **OS::TripleO::Services::Rhsm** composable service. For example:

```
- name: Controller
  description: |
    Controller role that has all the controller services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  ...
  ServicesDefault:
    ...
    - OS::TripleO::Services::Rhsm
    ...
```

3. Add the environment file for **rhsm** composable service parameters to future deployment operations.

This method replaces the **rhel-registration** parameters with the **rhsm** service parameters and changes the Heat resource that enables the service from:

```
resource_registry:
  OS::TripleO::NodeExtraConfig: rhel-registration.yaml
```

To:

```
resource_registry:
  OS::TripleO::Services::Rhsm: /usr/share/openstack-tripleo-heat-
templates/extraconfig/services/rhsm.yaml
```

You can also include the **/usr/share/openstack-tripleo-heat-templates/environments/rhsm.yaml** environment file with your deployment to enable the service.

4.3. RHEL-REGISTRATION TO RHSM MAPPINGS

rhel-registration	rhsm / RhsmVars
rhel_reg_method	**rhsm_method**
rhel_reg_org	**rhsm_org_id**
rhel_reg_pool_id	**rhsm_pool_ids**
rhel_reg_activation_key	**rhsm_activation_key**
rhel_reg_auto_attach	**rhsm_autosubscribe**

rhel-registration	rhsm / RhsmVars
rhel_reg_sat_url	**rhsm_satellite_url**
rhel_reg_repos	**rhsm_repos**
rhel_reg_user	**rhsm_username**
rhel_reg_password	**rhsm_password**
rhel_reg_http_proxy_host	**rhsm_rhsm_proxy_hostname**
rhel_reg_http_proxy_port	**rhsm_rhsm_proxy_port**
rhel_reg_http_proxy_username	**rhsm_rhsm_proxy_user**
rhel_reg_http_proxy_password	**rhsm_rhsm_proxy_password**

4.4. UPDATING COMPOSABLE SERVICES

This section contains information about new and deprecated composable services.

- If you use the default **roles_data** file, these services are included automatically.
- If you use a custom **roles_data** file, add the new services and remove the deprecated services for each relevant role.

Controller Nodes

The following services have been deprecated for Controller nodes. Remove them from your Controller role.

Service	Reason
OS::TripleO::Services::CeilometerApi **OS::TripleO::Services::CeilometerCollector** **OS::TripleO::Services::CeilometerExpirer**	OpenStack Platform no longer includes Ceilometer services.
OS::TripleO::Services::RabbitMQ	This service has been substituted for two new services: **OS::TripleO::Services::OsloMessagingRpc** **OS::TripleO::Services::OsloMessagingNotify**

The following services are new for Controller nodes. Add them to your Controller role.

Service	Reason
OS::TripleO::Services::CinderBackendNVMeOF	Only required if enabling the Block Storage (cinder) NVMeOF backend,
OS::TripleO::Services::ContainerImagePrepare	Run the commands to automatically pull and prepare container images relevant to the services in your overcloud.
OS::TripleO::Services::DesignateApi **OS::TripleO::Services::DesignateCentral** **OS::TripleO::Services::DesignateProducer** **OS::TripleO::Services::DesignateWorker** **OS::TripleO::Services::DesignateMDNS** **OS::TripleO::Services::DesignateSink**	Services for DNS-as-a-Service (designate).
OS::TripleO::Services::IronicInspector	Service for Bare Metal Introspection for the overcloud.
OS::TripleO::Services::IronicNeutronAgent	The networking agent for OpenStack Bare Metal (ironic).
OS::TripleO::Services::OsloMessagingRpc **OS::TripleO::Services::OsloMessagingNotify**	Replacement services for the **OS::TripleO::Services::RabbitMQ** service.
OS::TripleO::Services::Xinetd	Service to remove **xinetd**, which is no longer used in a containerized overcloud.

Compute Nodes

The following services are new for Compute nodes. Add them to your Compute role.

Service	Reason
OS::TripleO::Services::NovaLibvirtGuests	Service to enable **libvirt-guests** on Compute nodes.

All Nodes

The following services are new for all nodes. Add them to all roles.

Service	Reason

Service	Reason
OS::TripleO::Services::MetricsQdr	Service to enable Qpid Dispatch Router service for metrics and monitoring.
OS::TripleO::Services::Rhsm	Service to enable Ansible-based Red Hat Subscription Management.

4.5. DEPRECATED PARAMETERS

The following parameters are deprecated and have been replaced with role-specific 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**

Update these parameters in your custom environment files.

If your OpenStack Platform environment still requires these deprecated parameters, the default **roles_data** file allows their use. However, if you are using a custom **roles_data** file and your overcloud still requires these deprecated parameters, you can allow access to them by editing the **roles_data** file and adding the following to each role:

Controller Role

```
- name: Controller
  uses_deprecated_params: True
  deprecated_param_extraconfig: 'controllerExtraConfig'
  deprecated_param_flavor: 'OvercloudControlFlavor'
  deprecated_param_image: 'controllerImage'
  ...
```

Compute Role

```
- name: Compute
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  ...
```

Object Storage Role

```
- name: ObjectStorage
  uses_deprecated_params: True
  deprecated_param_metadata: 'SwiftStorageServerMetadata'
  deprecated_param_ips: 'SwiftStorageIPs'
  deprecated_param_image: 'SwiftStorageImage'
  deprecated_param_flavor: 'OvercloudSwiftStorageFlavor'
  ...
```

4.6. DEPRECATED CLI OPTIONS

Some command line options are outdated or deprecated in favor of using Heat template parameters, which you include in the **parameter_defaults** section of an environment file. The following table maps deprecated options to their Heat template equivalents.

Table 4.1. Mapping deprecated CLI options to Heat template parameters

Option	Description	Heat Template Parameter
--control-scale	The number of Controller nodes to scale out	**ControllerCount**
--compute-scale	The number of Compute nodes to scale out	**ComputeCount**
--ceph-storage-scale	The number of Ceph Storage nodes to scale out	**CephStorageCount**
--block-storage-scale	The number of Cinder nodes to scale out	**BlockStorageCount**

Option	Description	Heat Template Parameter
--swift-storage-scale	The number of Swift nodes to scale out	**ObjectStorageCount**
--control-flavor	The flavor to use for Controller nodes	**OvercloudControllerFlavor**
--compute-flavor	The flavor to use for Compute nodes	**OvercloudComputeFlavor**
--ceph-storage-flavor	The flavor to use for Ceph Storage nodes	**OvercloudCephStorageFlavor**
--block-storage-flavor	The flavor to use for Cinder nodes	**OvercloudBlockStorageFlavor**
--swift-storage-flavor	The flavor to use for Swift storage nodes	**OvercloudSwiftStorageFlavor**
--neutron-flat-networks	Defines the flat networks to configure in neutron plugins. Defaults to "datacentre" to permit external network creation	**NeutronFlatNetworks**
--neutron-physical-bridge	An Open vSwitch bridge to create on each hypervisor. This defaults to "br-ex". Typically, this should not need to be changed	**HypervisorNeutronPhysicalBridge**
--neutron-bridge-mappings	The logical to physical bridge mappings to use. Defaults to mapping the external bridge on hosts (br-ex) to a physical name (datacentre). Use this for the default floating network	**NeutronBridgeMappings**
--neutron-public-interface	Defines the interface to bridge onto br-ex for network nodes	**NeutronPublicInterface**
--neutron-network-type	The tenant network type for Neutron	**NeutronNetworkType**
--neutron-tunnel-types	The tunnel types for the Neutron tenant network. To specify multiple values, use a comma separated string	**NeutronTunnelTypes**

Option	Description	Heat Template Parameter
--neutron-tunnel-id-ranges	Ranges of GRE tunnel IDs that you want to make available for tenant network allocation	**NeutronTunnelIdRanges**
--neutron-vni-ranges	Ranges of VXLAN VNI IDs that you want to make available for tenant network allocation	**NeutronVniRanges**
--neutron-network-vlan-ranges	The Neutron ML2 and Open vSwitch VLAN mapping range to support. Defaults to permitting any VLAN on the 'datacentre' physical network	**NeutronNetworkVLANRanges**
--neutron-mechanism-drivers	The mechanism drivers for the neutron tenant network. Defaults to "openvswitch". To specify multiple values, use a comma-separated string	**NeutronMechanismDrivers**
--neutron-disable-tunneling	Disables tunneling in case you aim to use a VLAN segmented network or flat network with Neutron	No parameter mapping.
--validation-errors-fatal	The overcloud creation process performs a set of pre-deployment checks. This option exits if any fatal errors occur from the pre-deployment checks. It is advisable to use this option as any errors can cause your deployment to fail.	No parameter mapping
--ntp-server	Sets the NTP server to use to synchronize time	**NtpServer**

These parameters have been removed from Red Hat OpenStack Platform. It is recommended that you convert your CLI options to Heat parameters and add them to an environment file.

4.7. COMPOSABLE NETWORKS

This version of Red Hat OpenStack Platform introduces a new feature for composable networks. If you use a custom **roles_data** file, edit the file to add the composable networks to each role. For example, for Controller nodes:

```
- name: Controller
  networks:
    - External
    - InternalApi
```

```
    - Storage
    - StorageMgmt
    - Tenant
```

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

The following table contains a 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**

Role	Networks Required
OpenDaylight	**External**, **InternalApi**, **Tenant**
Redis	**InternalApi**
Sahara	**InternalApi**
Swift API	**Storage**
Swift Storage	**StorageMgmt**
Telemetry	**InternalApi**

IMPORTANT

In previous versions, the ***NetName** parameters (e.g. **InternalApiNetName**) changed the names of the default networks. This is no longer supported. Use a custom composable network file. For more information, see "Using Composable Networks" in the *Advanced Overcloud Customization* guide.

4.8. UPDATING NETWORK INTERFACE TEMPLATES

A new feature in OpenStack Platform 14 allows you to specify routes for each network in the overcloud's **network_data** file. To accommodate this new feature, the network interface templates now require parameters for the route list in each network. These parameters use the following format:

[network-name]InterfaceRoutes

Even if your overcloud does not use a routing list, you must still include these parameters for each network interface template.

- If you use one of the default NIC template sets, these parameters are included automatically.
- If you use a custom set of static NIC template, add these new parameters to the **parameters** of each role's template.

Red Hat OpenStack Platform includes a script to automatically add the missing parameters to your template files.

Procedure

1. Change to the director's core template collection:

```
$ cd /usr/share/openstack-tripleo-heat-templates
```

2. Run the **merge-new-params-nic-config-script.py** python script in the tools directory. For example, to update a custom Controller node NIC template, run the script with the following options:

```
$ python ./tools/merge-new-params-nic-config-script.py --role-name Controller -t
/home/stack/ccsosp-templates/custom-nics/controller.yaml
```

Note the following options used with this script:

- **--role-name** defines the name of the role to use as a basis for the template update.
- **-t, --template** defines the filename of the NIC template to update.
- **-n, --network-data** defines the relative path to the **network_data** file. Use this option for custom **network_data** files. If omitted, the script uses the default file.
- **-r, --roles-data**, defines the relative path to the **roles_data.yaml** file. Use this option for custom **roles_data** files. If omitted, the script uses the default file.

3. The script saves a copy of the original template and adds a timestamp extension to the copy's filename. To compare the differences between the original and updated template, run the following command:

```
$ diff /home/stack/ccsosp-templates/custom-nics/controller.yaml.[TIMESTAMP]
/home/stack/ccsosp-templates/custom-nics/controller.yaml
```

Replace **[TIMESTAMP]** with the timestamp on the original filename.

The output displays the new route parameters for that role:

```
StorageMgmtInterfaceRoutes:
  default: []
  description: >
    Routes for the storage_mgmt network traffic. JSON route e.g. [{'destination':'10.0.0.0/16',
'nexthop':'10.0.0.1'}] Unless
    the default is changed, the parameter is automatically resolved from the subnet
host_routes attribute.
  type: json
TenantInterfaceRoutes:
  default: []
  description: >
    Routes for the tenant network traffic. JSON route e.g. [{'destination':'10.0.0.0/16',
'nexthop':'10.0.0.1'}] Unless
    the default is changed, the parameter is automatically resolved from the subnet
host_routes attribute.
  type: json
ExternalInterfaceRoutes:
  default: []
  description: >
    Routes for the external network traffic. JSON route e.g. [{'destination':'10.0.0.0/16',
'nexthop':'10.0.0.1'}] Unless
    the default is changed, the parameter is automatically resolved from the subnet
host_routes attribute.
  type: json
InternalApiInterfaceRoutes:
  default: []
  description: >
    Routes for the internal_api network traffic. JSON route e.g. [{'destination':'10.0.0.0/16',
'nexthop':'10.0.0.1'}] Unless
    the default is changed, the parameter is automatically resolved from the subnet
```

```
    host_routes attribute.
      type: json
    ManagementInterfaceRoutes:
      default: []
      description: >
        Routes for the management network traffic. JSON route e.g. [{'destination':'10.0.0.0/16',
    'nexthop':'10.0.0.1'}] Unless
        the default is changed, the parameter is automatically resolved from the subnet
    host_routes attribute.
      type: json
    StorageInterfaceRoutes:
      default: []
      description: >
        Routes for the storage network traffic. JSON route e.g. [{'destination':'10.0.0.0/16',
    'nexthop':'10.0.0.1'}] Unless
        the default is changed, the parameter is automatically resolved from the subnet
    host_routes attribute.
      type: json
```

For more information, see "Isolating Networks".

4.9. PREPARING BLOCK STORAGE SERVICE TO RECEIVE CUSTOM CONFIGURATION FILES

When upgrading to the containerized environment, use the **CinderVolumeOptVolumes** parameter to add docker volume mounts. This enables custom configuration files on the host to be made available to the cinder-volume service when it's running in a container.

For example:

```
parameter_defaults:
  CinderVolumeOptVolumes:
    /etc/cinder/nfs_shares1:/etc/cinder/nfs_shares1
    /etc/cinder/nfs_shares2:/etc/cinder/nfs_shares2
```

4.10. NEXT STEPS

The overcloud preparation stage is complete. You can now perform an upgrade of the overcloud to 14 using the steps in Chapter 5, *Upgrading the Overcloud*.

CHAPTER 5. UPGRADING THE OVERCLOUD

This process upgrades the overcloud.

Prerequisites

- You have upgraded the undercloud to the latest version.
- You have prepared your custom environment files to accommodate the changes in the upgrade.

5.1. RELEVANT FILES FOR UPGRADE

The following is a list of new and modified files for the overcloud upgrade.

Roles

- If you use custom roles, include the updated **roles_data** file with new and deprecated services.

Network

- If you use isolated networks, include the **network_data** file.
- If you use custom NIC template, include the new versions.

Environment File

- Include the **containers-prepare-parameter.yaml** file created during the undercloud upgrade.
- Replace the **rhel-registration** environment files with the environment file to configure the Ansible-based Red Hat Subscription Management service.
- Include any additional environment files relevant to your overcloud configuration.

5.2. RUNNING THE OVERCLOUD UPGRADE PREPARATION

The upgrade requires running **openstack overcloud upgrade prepare** command, which performs the following tasks:

- Updates the overcloud plan to OpenStack Platform 14
- Prepares the nodes for the upgrade

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Run the upgrade preparation command:

```
$ openstack overcloud upgrade prepare \
    --templates \
```

```
-e /home/stack/containers-prepare-parameter.yaml \
-e <ENVIRONMENT FILE>
```

Include the following options relevant to your environment:

- Custom configuration environment files (**-e**)
- The environment file (**containers-prepare-parameter.yaml**) with your new container image locations (**-e**). In most cases, this is the same environment file that the undercloud uses.
- If applicable, your custom roles (**roles_data**) file using **--roles-file**.
- If applicable, your composable network (**network_data**) file using **--networks-file**.
- If you use a custom stack name, pass the name with the **--stack** option.

3. Wait until the upgrade preparation completes.

5.3. RUNNING THE CONTAINER IMAGE PREPARATION

The overcloud requires the OpenStack Platform 14 container images before performing the upgrade. This involves executing the **container_image_prepare** external upgrade process. To execute this process, run the **openstack overcloud external-upgrade run** command against tasks tagged with the **container_image_prepare** tag. These tasks perform the following operations:

- Automatically prepare all container image configuration relevant to your environment.
- Pull the relevant container images to your undercloud, unless you have previously disabled this option.

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Run the **openstack overcloud external-upgrade run** command against tasks tagged with the **container_image_prepare** tag:

```
$ openstack overcloud external-upgrade run --tags container_image_prepare
```

- If you use a custom stack name, pass the name with the **--stack** option.

5.4. UPGRADING CONTROLLER AND CUSTOM ROLE NODES

Use the following process to upgrade all the Controller nodes, split Controller services, and other custom nodes to OpenStack Platform 14. The process involves running the **openstack overcloud upgrade run** command and including the **--nodes** option to restrict operations to only the selected nodes:

```
$ openstack overcloud upgrade run --nodes [ROLE]
```

Substitute **[ROLE]** for the name of a role or a comma-separated list of roles.

If your overcloud uses monolithic Controller nodes, run this command against the **Controller** role.

If your overcloud uses split Controller services, use the following guide to upgrade the node role in the following order:

- All roles that use Pacemaker. For example: **ControllerOpenStack**, **Database**, **Messaging**, and **Telemetry**.
- **Networker** nodes
- Any other custom roles

Do not upgrade the following nodes yet:

- **Compute** nodes
- **CephStorage** nodes

You will upgrade these nodes at a later stage.

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. If you use monolithic Controller nodes, run the upgrade command against the **Controller** role:

```
$ openstack overcloud upgrade run --nodes Controller
```

   - If you use a custom stack name, pass the name with the **--stack** option.

3. If you use Controller services split across multiple roles:

   a. Run the upgrade command for roles with Pacemaker services:

```
$ openstack overcloud upgrade run --nodes ControllerOpenStack
$ openstack overcloud upgrade run --nodes Database
$ openstack overcloud upgrade run --nodes Messaging
$ openstack overcloud upgrade run --nodes Telemetry
```

   - If you use a custom stack name, pass the name with the **--stack** option.

   a. Run the upgrade command for the **Networker** role:

```
$ openstack overcloud upgrade run --nodes Networker
```

   - If you use a custom stack name, pass the name with the **--stack** option.

   a. Run the upgrade command for any remaining custom roles, except for **Compute** or **CephStorage** roles:

```
$ openstack overcloud upgrade run --nodes ObjectStorage
```

   - If you use a custom stack name, pass the name with the **--stack** option.

5.5. UPGRADING ALL COMPUTE NODES

This process upgrades all remaining Compute nodes to OpenStack Platform 14. The process involves running the **openstack overcloud upgrade run** command and including the **--nodes Compute** option to restrict operations to the Compute nodes only.

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Run the upgrade command:

```
$ openstack overcloud upgrade run --nodes Compute
```

   - If you use a custom stack name, pass the name with the **--stack** option.

3. Wait until the Compute node upgrade completes.

5.6. UPGRADING ALL CEPH STORAGE NODES

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

- Running the **openstack overcloud upgrade run** command and including the **--nodes CephStorage** option to restrict operations to the Ceph Storage nodes.
- Running the **openstack overcloud ceph-upgrade run** command to perform an upgrade to a containerized Red Hat Ceph Storage 3 cluster.

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Run the upgrade command:

```
$ openstack overcloud upgrade run --nodes CephStorage
```

   - If you use a custom stack name, pass the name with the **--stack** option.

3. Wait until the node upgrade completes.
4. Run the Ceph Storage external upgrade process. For example:

```
$ openstack overcloud external-upgrade run --tags ceph
```

   - If you use a custom stack name, pass the name with the **--stack** option.
   - To pass any additional overrides, see in Section 5.6.1, "Custom parameters for upgrades".

5. Wait until the Ceph Storage node upgrade completes.

5.6.1. Custom parameters for upgrades

When migrating Ceph to containers, each Ceph monitor and OSD is stopped sequentially. The migration does not continue until the same service that was stopped is successfully restarted. Ansible waits 15 seconds (the delay) and checks 5 times for the service to start (the retries). If the service does not restart, the migration stops so the operator can intervene.

Depending on the size of the Ceph cluster, you may need to increase the retry or delay values. The exact names of these parameters and their defaults are as follows:

```
health_mon_check_retries: 5
health_mon_check_delay: 15
health_osd_check_retries: 5
health_osd_check_delay: 15
```

To change the default values and make the cluster check 30 times and wait 40 seconds between each check, pass the following parameters in a yaml file with **-e** using the **openstack overcloud deploy** command:

```
parameter_defaults:
  CephAnsibleExtraConfig:
    health_osd_check_delay: 40
    health_osd_check_retries: 30
```

5.7. PERFORMING ONLINE DATABASE UPGRADES

Some overcloud components require an online upgrade (or migration) of their databases tables. This involves executing the **online_upgrade** external upgrade process. To execute this process, run the **openstack overcloud external-upgrade run** command against tasks tagged with the **online_upgrade** tag. This performs online database upgrades to the following components:

- OpenStack Block Storage (cinder)
- OpenStack Compute (nova)
- OpenStack Bare Metal (ironic) if enabled in the overcloud

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Run the **openstack overcloud external-upgrade run** command against tasks tagged with the **online_upgrade** tag:

```
$ openstack overcloud external-upgrade run --tags online_upgrade
```

   - If you use a custom stack name, pass the name with the **--stack** option.

5.8. FINALIZING THE UPGRADE

The 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 14 and allows you to perform standard **openstack overcloud deploy** functions in the future.

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Run the upgrade finalization command:

```
$ openstack overcloud upgrade converge \
    --templates \
    -e <ENVIRONMENT FILE>
```

   Include the following options relevant to your environment:

   - Custom configuration environment files (**-e**).
   - If you use a custom stack name, pass the name with the **--stack** option.
   - If applicable, your custom roles (**roles_data**) file using **--roles-file**.
   - If applicable, your composable network (**network_data**) file using **--networks-file**.

3. Wait until the upgrade finalization completes.