Red Hat OpenStack Platform 17.0-Beta

Distributed compute node and storage deployment

Deploying Red Hat OpenStack Platform distributed compute node technologies
Deploying Red Hat OpenStack Platform distributed compute node technologies

OpenStack Team
rhos-docs@redhat.com
Abstract

You can deploy Red Hat OpenStack Platform (RHOSP) with a distributed compute node (DCN) architecture for edge site operability with heat stack separation. Each site can have its own Ceph storage back end for Image service (glance) multi store.
Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE .............................................................. 4

CHAPTER 1. UNDERSTANDING DCN ................................................................ 5
   1.1. REQUIRED SOFTWARE FOR DISTRIBUTED COMPUTE NODE ARCHITECTURE 5
   1.2. MULTISTACK DESIGN ........................................................................ 6
   1.3. DCN STORAGE .................................................................................. 6
   1.4. DCN EDGE ...................................................................................... 6

CHAPTER 2. PLANNING A DISTRIBUTED COMPUTE NODE (DCN) DEPLOYMENT .. 7
   2.1. CONSIDERATIONS FOR STORAGE ON DCN ARCHITECTURE ........... 7
   2.2. CONSIDERATIONS FOR NETWORKING ON DCN ARCHITECTURE ...... 7
   2.3. STORAGE TOPOLOGIES AND ROLES AT THE EDGE ....................... 9
      2.3.1. Roles for edge deployments ...................................................... 12

CHAPTER 3. PREPARING OVERCLOUD TEMPLATES FOR DCN DEPLOYMENT ..... 14
   3.1. PREREQUISITES FOR USING SEPARATE HEAT STACKS .................... 14
   3.2. LIMITATIONS OF THE EXAMPLE SEPARATE HEAT STACKS DEPLOYMENT 14
   3.3. DESIGNING YOUR SEPARATE HEAT STACKS DEPLOYMENT ............... 14
   3.4. MANAGING SEPARATE HEAT STACKS ............................................ 15
   3.5. RETRIEVING THE CONTAINER IMAGES .......................................... 15
   3.6. CREATING FAST DATAPATH ROLES FOR THE EDGE ....................... 16

CHAPTER 4. INSTALLING THE CENTRAL LOCATION ...................................... 18
   4.1. DEPLOYING THE CENTRAL CONTROLLERS WITHOUT EDGE STORAGE 18
   4.2. DEPLOYING THE CENTRAL SITE WITH STORAGE ............................. 20
   4.3. INTEGRATING EXTERNAL CEPH ..................................................... 23

CHAPTER 5. DEPLOY THE EDGE WITHOUT STORAGE .................................... 27
   5.1. DEPLOYING EDGE NODES WITHOUT STORAGE ............................... 27
   5.2. EXCLUDING SPECIFIC IMAGE TYPES AT THE EDGE ....................... 29

CHAPTER 6. DEPLOYING STORAGE AT THE EDGE ....................................... 31
   6.1. DEPLOYING EDGE SITES WITH HYPERCONVERGED STORAGE .......... 31
   6.2. USING A PRE-INSTALLED RED HAT CEPH STORAGE CLUSTER AT THE EDGE 34
   6.3. UPDATING THE CENTRAL LOCATION .............................................. 36
      6.3.1. Clearing residual data after interrupted Image service processes .... 38
   6.4. DEPLOYING RED HAT CEPH STORAGE DASHBOARD ON DCN .......... 39

CHAPTER 7. DEPLOYING WITH KEY MANAGER .......................................... 41
   7.1. DEPLOYING EDGE SITES WITH KEY MANAGER ............................. 41

CHAPTER 8. PRECACHING GLANCE IMAGES INTO NOVA ............................ 42
   8.1. RUNNING THE TRIPLEO_NOVA_IMAGE_CACHE.YML ANSIBLE PLAYBOOK 42
   8.2. PERFORMANCE CONSIDERATIONS .............................................. 43
   8.3. OPTIMIZING THE IMAGE DISTRIBUTION TO DCN SITES ................. 44
   8.4. CONFIGURING THE NOVA-CACHE CLEANUP .................................. 44

CHAPTER 9. TLS-E FOR DCN ..................................................................... 45
   9.1. DEPLOYING DISTRIBUTED COMPUTE NODE ARCHITECTURE WITH TLS-E 45

CHAPTER 10. CREATING A CEPH KEY FOR EXTERNAL ACCESS .................. 47
   10.1. CREATING A CEPH KEY FOR EXTERNAL ACCESS ......................... 47
   10.2. USING EXTERNAL CEPH KEYS .................................................... 48
APPENDIX A. DEPLOYMENT MIGRATION OPTIONS ............................................. 50
A.1. VALIDATING EDGE STORAGE ......................................................... 50
   A.1.1. Importing from a local file .................................................... 50
   A.1.2. Importing an image from a web server .................................. 51
   A.1.3. Copying an image to a new site ............................................ 51
   A.1.4. Confirming that an instance at an edge site can boot with image based volumes ......................................................... 52
   A.1.5. Confirming image snapshots can be created and copied between sites ................................................................. 53
A.2. MIGRATING TO A SPINE AND LEAF DEPLOYMENT ......................... 53
A.3. MIGRATING TO A MULTISTACK DEPLOYMENT ................................... 54
A.4. BACKING UP AND RESTORING ACROSS EDGE SITES ....................... 54
MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
CHAPTER 1. UNDERSTANDING DCN

Distributed compute node (DCN) architecture is for edge use cases allowing remote compute and storage nodes to be deployed remotely while sharing a common centralised control plane. DCN architecture allows you to position workloads strategically closer to your operational needs for higher performance.

The central location can consist of any role, however at a minimum, requires three controllers. Compute nodes can exist at the edge, as well as at the central location.

DCN architecture is a hub and spoke routed network deployment. DCN is comparable to a spine and leaf deployment for routed provisioning and control plane networking with Red Hat OpenStack Platform director.

- The hub is the central site with core routers and a datacenter gateway (DC-GW).
- The spoke is the remote edge, or leaf.

Edge locations do not have controllers, making them architecturally different from traditional deployments of Red Hat OpenStack Platform:

- Control plane services run remotely, at the central location.
- Pacemaker is not installed.
- The Block Storage service (cinder) runs in active/active mode.
- Etcd is deployed as a distributed lock manager (DLM).

![Diagram of DCN architecture]

1.1. REQUIRED SOFTWARE FOR DISTRIBUTED COMPUTE NODE ARCHITECTURE

The following table shows the software and minimum versions required to deploy Red Hat OpenStack Platform in a distributed compute node (DCN) architecture:
### 1.2. MULTISTACK DESIGN

When you deploy Red Hat OpenStack Platform (RHOSP) with a DCN design, you use Red Hat director’s capabilities for multiple stack deployment and management to deploy each site as a distinct stack.

Managing a DCN architecture as a single stack is unsupported, unless the deployment is an upgrade from Red Hat OpenStack Platform 13. There are no supported methods to split an existing stack, however you can add stacks to a pre-existing deployment. For more information, see Section A.3, “Migrating to a multistack deployment”.

The central location is a traditional stack deployment of RHOSP, however you are not required to deploy Compute nodes or Red Hat Ceph storage with the central stack.

With DCN, you deploy each location as a distinct availability zone (AZ).

### 1.3. DCN STORAGE

You can deploy each edge site, either without storage, or with Ceph on hyperconverged nodes. The storage you deploy is dedicated to the site you deploy it on.

DCN architecture uses Glance multistore. For edge sites deployed without storage, additional tooling is available so that you can cache and store images in the Compute service (nova) cache. Caching glance images in nova provides the faster boot times for instances by avoiding the process of downloading images across a WAN link. For more information, see Chapter 8, Precaching glance images into nova.

### 1.4. DCN EDGE

With Distributed Compute Node architecture, the central location is deployed with the control nodes that manage the edge locations. When you then deploy an edge location, you deploy only compute nodes, making edge sites architecturally different from traditional deployments of Red Hat OpenStack Platform. At edge locations:

- Control plane services run remotely at the central location.
- Pacemaker does not run at DCN sites.
- The Block Storage service (cinder) runs in active/active mode.
- Etcd is deployed as a distributed lock manager (DLM).
CHAPTER 2. PLANNING A DISTRIBUTED COMPUTE NODE (DCN) DEPLOYMENT

When you plan your DCN architecture, check that the technologies that you need are available and supported.

2.1. CONSIDERATIONS FOR STORAGE ON DCN ARCHITECTURE

The following features are not currently supported for DCN architectures:

- Copying a volume snapshot between edge sites. You can work around this by creating an image from the volume and using glance to copy the image. After the image is copied, you can create a volume from it.
- Ceph Rados Gateway (RGW) at the edge.
- CephFS at the edge.
- Instance high availability (HA) at the edge sites.
- Live migration between edge sites or from the central location to edge sites. You can still live migrate instances within a site boundary.
- RBD mirroring between sites.

Additionally, you must consider the following:

- You must upload images to the central location before copying them to edge sites; a copy of each image must exist in the Image service (glance) at the central location.
- Before you create an instance at an edge site, you must have a local copy of the image at that edge site.
- You must use the RBD storage driver for the Image, Compute and Block Storage services.
- For each site, assign a unique availability zone, and use the same value for the NovaComputeAvailabilityZone and CinderStorageAvailabilityZone parameters.
- You can migrate an offline volume from an edge site to the central location, or vice versa. You cannot migrate volumes directly between edge sites.

2.2. CONSIDERATIONS FOR NETWORKING ON DCN ARCHITECTURE

The following features are not currently supported for DCN architectures:

- Octavia
- DHCP on DPDK nodes
- Contrack for TC Flower Hardware Offload

Contrack for TC Flower Hardware Offload is available on DCN as a Technology Preview, and therefore using these solutions together is not fully supported by Red Hat. This feature should only be used with DCN for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see Scope of Coverage Details.
The following ML2/OVS technologies are fully supported:

- OVS-DPDK without DHCP on the DPDK nodes
- SR-IOV
- TC Flower hardware offload, without conntrack
- Neutron availability zones (AZs) with networker nodes at the edge, with one AZ per site
- Routed provider networks

The following ML2/OVN networking technologies are fully supported:

- OVS-DPDK without DHCP on the DPDK nodes
- SR-IOV (without DHCP
- TC flower hardware offload, without conntrack
- Router provider networks
- OVN GW (networker node) with Neutron AZs supported

Additionally, you must consider the following:

- Network latency: Balance the latency as measured in round-trip time (RTT), with the expected number of concurrent API operations to maintain acceptable performance. Maximum TCP/IP throughput is inversely proportional to RTT. You can mitigate some issues with high-latency connections with high bandwidth by tuning kernel TCP parameters. Contact Red Hat support if a cross-site communication exceeds 100 ms.

- Network drop outs: If the edge site temporarily loses connection to the central site, then no OpenStack control plane API or CLI operations can be executed at the impacted edge site for the duration of the outage. For example, Compute nodes at the edge site are consequently unable to create a snapshot of an instance, issue an auth token, or delete an image. General OpenStack control plane API and CLI operations remain functional during this outage, and can continue to serve any other edge sites that have a working connection. Image type: You must use raw images when deploying a DCN architecture with Ceph storage.

- Image sizing:
  - Overcloud node images - overcloud node images are downloaded from the central undercloud node. These images are potentially large files that will be transferred across all necessary networks from the central site to the edge site during provisioning.
  - Instance images: If there is no block storage at the edge, then the Image service images traverse the WAN during first use. The images are copied or cached locally to the target edge nodes for all subsequent use. There is no size limit for glance images. Transfer times vary with available bandwidth and network latency.
    - If there is block storage at the edge, then the image is copied over the WAN asynchronously for faster boot times at the edge.
  - Provider networks: This is the recommended networking approach for DCN deployments. If you use provider networks at remote sites, then you must consider that the Networking service (neutron) does not place any limits or checks on where you can attach available networks. For
example, if you use a provider network only in edge site A, you must ensure that you do not try to attach to the provider network in edge site B. This is because there are no validation checks on the provider network when binding it to a Compute node.

- Site-specific networks: A limitation in DCN networking arises if you use networks that are specific to a certain site: When you deploy centralized neutron controllers with Compute nodes, there are no triggers in neutron to identify a certain Compute node as a remote node. Consequently, the Compute nodes receive a list of other Compute nodes and automatically form tunnels between each other; the tunnels are formed from edge to edge through the central site. If you use VXLAN or Geneve, every Compute node at every site forms a tunnel with every other Compute node and Controller node, whether or not they are local or remote. This is not an issue if you are using the same neutron networks everywhere. When you use VLANs, neutron expects that all Compute nodes have the same bridge mappings, and that all VLANs are available at every site.

- Additional sites: If you need to expand from a central site to additional remote sites, you can use the openstack CLI on Red Hat OpenStack Platform director to add new network segments and subnets.

- If edge servers are not pre-provisioned, you must configure DHCP relay for introspection and provisioning on routed segments.

- Routing must be configured either on the cloud or within the networking infrastructure that connects each edge site to the hub. You should implement a networking design that allocates an L3 subnet for each Red Hat OpenStack Platform cluster network (external, internal API, and so on), unique to each site.

### 2.3. STORAGE TOPOLOGIES AND ROLES AT THE EDGE

When you deploy Red Hat OpenStack platform with a distributed compute node architecture, you must decide if you need storage at the edge. Based on storage and performance needs, you can deploy each site with one of three configurations. Not all edge sites must have an identical configuration.

**DCN without storage**

To deploy this architecture, use the **Compute** role.

Without block storage at the edge:
The Object Storage (swift) service at the control plane is used as an Image (glance) service backend.

- Multi-backend image service is not available.
  - Images are cached locally at edge sites in Nova. For more information see Chapter 8, *Precaching glance images into nova*.

- The instances are stored locally on the Compute nodes.

- Volume services such as Block Storage (cinder) are not available at edge sites.

**IMPORTANT**

If you do not deploy the central location with Red Hat Ceph storage, you will not have the option of deploying an edge site with storage at a later time.

For more information about deploying without block storage at the edge, see Section 5.1, "Deploying edge nodes without storage".

**DCN with storage**

To deploy DCN with storage you must also deploy Red Hat Ceph Storage at the central location. You must use the `dcn-storage.yaml` and `cephadm.yaml` environment files. For edge sites that include non-hyperconverged Red Hat Ceph Storage nodes, use the `DistributedCompute`, `DistributedComputeScaleOut`, `CephAll`, and `CephStorage` roles.

With block storage at the edge:

- Red Hat Ceph Block Devices (RBD) is used as an Image (glance) service backend.

- Multi-backend Image service (glance) is available so that images may be copied between the central and DCN sites.

- The Block Storage (cinder) service is available at all sites and is accessed by using the Red Hat Ceph Block Devices (RBD) driver.

- The Block Storage (cinder) service runs on the Compute nodes, and Red Hat Ceph Storage runs separately on dedicated storage nodes.
Nova ephemeral storage is backed by Ceph (RBD). For more information, see Section 4.2, “Deploying the central site with storage”.

### DCN with hyperconverged storage

To deploy this configuration you must also deploy Red Hat Ceph Storage at the central location. You need to configure the `dcn-storage.yaml` and `cephadm.yaml` environment files. Use the `DistributedComputeHCI` and `DistributedComputeHCIScaleOut` roles. You can also use the `DistributedComputeScaleOut` role to add Compute nodes that do not participate in providing Red Hat Ceph Storage services.

With hyperconverged storage at the edge:

- Red Hat Ceph Block Devices (RBD) is used as an Image (glance) service backend.
- Multi-backend Image service (glance) is available so that images may be copied between the central and DCN sites.
- The Block Storage (cinder) service is available at all sites and is accessed by using the Red Hat Ceph Block Devices (RBD) driver.
- Both the Block Storage service and Red Hat Ceph Storage run on the Compute nodes. For more information, see Section 6.1, “Deploying edge sites with hyperconverged storage”.

When you deploy Red Hat OpenStack Platform in a distributed compute architecture, you have the option of deploying multiple storage topologies, with a unique configuration at each site. You must deploy the central location with Red Hat Ceph storage to deploy any of the edge sites with storage.
2.3.1. Roles for edge deployments

The following roles are available for edge deployments. Select the appropriate roles for your environment based on your chosen configuration.

Compute

The **Compute** role is used for edge deployments without storage.

DistributedCompute

The **DistributedCompute** role is used at the edge for storage deployments without hyperconverged nodes. The **DistributedCompute** role includes the **GlanceApiEdge** service, which ensures that Image services are consumed at the local edge site rather than at the central hub location. You can deploy up to three nodes using the **DistributedCompute** role. For any additional nodes use the **DistributedComputeScaleOut** role.

DistributedComputeScaleOut

The **DistributedComputeScaleOut** role includes the **HAproxyEdge** service, which enables instances created on the DistributedComputeScaleOut role to proxy requests for Image services to nodes that provide that service at the edge site. After you deploy three nodes with a role of **DistributedCompute**, you can use the DistributedComputeScaleOut role to scale compute resources. There is no minimum number of hosts required to deploy with the **DistributedComputeScaleOut** role. This role is used at the edge for storage deployments without hyperconverged nodes.

DistributedComputeHCI

The **DistributedComputeHCI** role enables a hyperconverged deployment at the edge by including Ceph Management and OSD services. You must use exactly three nodes when using the **DistributedComputeHCI** role. This role is used for storage deployments with fully converged nodes.

DistributedComputeHCIScaleOut

The **DistributedComputeHCIScaleOut** role includes the **Ceph OSD** service, which allows storage capacity to be scaled with compute when more nodes are added to the edge. This role also includes the **HAproxyEdge** service to redirect image download requests to the **GlanceAPIEdge** nodes at the edge site. This role enables a hyper converged deployment at the edge. You must use exactly three nodes when using the **DistributedComputeHCI** role. This role is used at the edge for storage deployments with hyperconverged nodes.

CephAll
The **CephAll** role includes the Ceph OSD, Ceph mon, and Ceph Mgr services. This role is used at the edge for storage deployments without hyperconverged nodes. You can deploy up to three nodes using the CephAll role. For any additional storage capacity use the CephStorage role.

**CephStorage**

The **CephStorage** role includes the Ceph OSD service. This role is used at the edge for storage deployments without hyperconverged nodes. If three CephAll nodes do not provide enough storage capacity, then add as many CephStorage nodes as needed.
CHAPTER 3. PREPARING OVERCLOUD TEMPLATES FOR DCN DEPLOYMENT

3.1. PREREQUISITES FOR USING SEPARATE HEAT STACKS

Your environment must meet the following prerequisites before you create a deployment using separate heat stacks:

- An installed instance of Red Hat OpenStack Platform director 17.0-Beta.
- For Ceph Storage users: access to Red Hat Ceph Storage 5.
- For the central location: three nodes that are capable of serving as central Controller nodes. All three Controller nodes must be in the same heat stack. You cannot split Controller nodes, or any of the control plane services, across separate heat stacks.
- Ceph storage is a requirement at the central location if you plan to deploy Ceph storage at the edge.
- For each additional DCN site: three HCI compute nodes.
- All nodes must be pre-provisioned or able to PXE boot from the central deployment network. You can use a DHCP relay to enable this connectivity for DCNs.
- All nodes have been introspected by ironic.
- Red Hat recommends leaving the <role>HostnameFormat parameter as the default value: %stackname%-%role%-%index%. If you do not include the %stackname% prefix, your overcloud uses the same hostnames for distributed compute nodes in different stacks. Ensure that your distributed compute nodes use the %stackname% prefix to distinguish nodes from different edge sites. For example, if you deploy two edge sites named dcn0 and dcn1, the stack name prefix helps you to distinguish between dcn0-distributedcompute-0 and dcn1-distributedcompute-0 when you run the openstack server list command on the undercloud.
- Source the centralrc authentication file to schedule workloads at edge sites as well as at the central location. You do not require authentication files that are automatically generated for edge sites.

3.2. LIMITATIONS OF THE EXAMPLE SEPARATE HEAT STACKS DEPLOYMENT

This document provides an example deployment that uses separate heat stacks on Red Hat OpenStack Platform. This example environment has the following limitations:

- Spine/Leaf networking - The example in this guide does not demonstrate routing requirements, which are required in distributed compute node (DCN) deployments.
- Ironic DHCP Relay - This guide does not include how to configure Ironic with a DHCP relay.

3.3. DESIGNING YOUR SEPARATE HEAT STACKS DEPLOYMENT

To segment your deployment within separate heat stacks, you must first deploy a single overcloud with the control plane. You can then create separate stacks for the distributed compute node (DCN) sites. The following example shows separate stacks for different node types:
- Controller nodes: A separate heat stack named `central`, for example, deploys the controllers. When you create new heat stacks for the DCN sites, you must create them with data from the `central` stack. The Controller nodes must be available for any instance management tasks.

- DCN sites: You can have separate, uniquely named heat stacks, such as `dcn0`, `dcn1`, and so on. Use a DHCP relay to extend the provisioning network to the remote site.

**NOTE**

You must create a separate availability zone (AZ) for each stack.

### 3.4. MANAGING SEPARATE HEAT STACKS

The procedures in this guide show how to organize the environment files for three heat stacks: `central`, `dcn0`, and `dcn1`. Red Hat recommends that you store the templates for each heat stack in a separate directory to keep the information about each deployment isolated.

**Procedure**

1. Define the `central` heat stack:
   
   ```bash
   $ mkdir central
   $ touch central/overrides.yaml
   ```

2. Extract data from the `central` heat stack into a common directory for all DCN sites:
   
   ```bash
   $ mkdir dcn-common
   $ touch dcn-common/overrides.yaml
   $ touch dcn-common/central-export.yaml
   ```

   The `central-export.yaml` file is created later by the `openstack overcloud export` command. It is in the `dcn-common` directory because all DCN deployments in this guide must use this file.

3. Define the `dcn0` site:
   
   ```bash
   $ mkdir dcn0
   $ touch dcn0/overrides.yaml
   ```

To deploy more DCN sites, create additional `dcn` directories by number.

**NOTE**

The touch is used to provide an example of file organization. Each file must contain the appropriate content for successful deployments.

### 3.5. RETRIEVING THE CONTAINER IMAGES

Use the following procedure, and its example file contents, to retrieve the container images you need for deployments with separate heat stacks. You must ensure the container images for optional or edge-specific services are included by running the `openstack container image prepare` command with edge site’s environment files.

For more information, see [Preparing container images](#).
### Procedure

1. Add your Registry Service Account credentials to `containers.yaml`:

```yaml
parameter_defaults:
  ContainerImagePrepare:
    - push_destination: true
  set:
    ceph_namespace: registry.redhat.io/rhceph
    ceph_image: rhceph-5-rhel9
    ceph_tag: latest
    name_prefix: openstack-
    namespace: registry.redhat.io/rhosp17-rhel9
    tag: latest
  ContainerImageRegistryCredentials:
    # https://access.redhat.com/RegistryAuthentication
    registry.redhat.io:
      registry-service-account-username: registry-service-account-password
```

2. Generate the environment file as `images-env.yaml`:

```bash
sudo openstack tripleo container image prepare \
  -e containers.yaml \
  --output-env-file images-env.yaml
```

The resulting `images-env.yaml` file is included as part of the overcloud deployment procedure for the stack for which it is generated.

### 3.6. CREATING FAST DATAPATH ROLES FOR THE EDGE

To use fast datapath services at the edge, you must create a custom role that defines both fast datapath and edge services. When you create the roles file for deployment, you can include the newly created role that defines services needed for both distributed compute node architecture and fast datapath services such as DPDK or SR-IOV.

For example, create a custom role for distributedCompute with DPDK:

#### Prerequisites

A successful undercloud installation. For more information, see [Installing the undercloud](#).

#### Procedure

1. Log in to the undercloud host as the `stack` user.

2. Copy the default `roles` directory:

```bash
cp -r /usr/share/openstack-tripleo-heat-templates/roles ~/.
```

3. Create a new file named `DistributedComputeDpdk.yaml` from the `DistributedCompute.yaml` file:

```bash
cp roles/DistributedCompute.yaml roles/DistributedComputeDpdk.yaml
```
4. Add DPDK services to the new `DistributedComputeDpdk.yaml` file. You can identify the parameters that you need to add by identifying the parameters in the `ComputeOvsDpdk.yaml` file that are not present in the `DistributedComputeDpdk.yaml` file.

```bash
diff -u roles/DistributedComputeDpdk.yaml roles/ComputeOvsDpdk.yaml
```

In the output, the parameters that are preceded by `+` are present in the `ComputeOvsDpdk.yaml` file but are not present in the `DistributedComputeDpdk.yaml` file. Include these parameters in the new `DistributedComputeDpdk.yaml` file.

5. Use the `DistributedComputeDpdk.yaml` to create a `DistributedComputeDpdk` roles file:

```bash
openstack overcloud roles generate --roles-path ~/roles/ -o ~/roles/roles-custom.yaml
```

You can use this same method to create fast datapath roles for SR-IOV, or a combination of SR-IOV and DPDK for the edge to meet your requirements.

If you are planning to deploy edge sites without block storage, see the following:

- Chapter 4, *Installing the central location*
- Section 5.1, “Deploying edge nodes without storage”

If you plan to deploy edge sites with Red Hat Ceph Storage, see the following:

- Chapter 4, *Installing the central location*
- Section 6.1, “Deploying edge sites with hyperconverged storage”
CHAPTER 4. INSTALLING THE CENTRAL LOCATION

When you deploy the central location for distributed compute node (DCN) architecture, you can deploy the cluster:

- With or without Compute nodes
- With or without Red Hat Ceph Storage

If you deploy Red Hat OpenStack Platform without Red Hat Ceph Storage at the central location, you cannot deploy any of your edge sites with Red Hat Ceph storage. Additionally, you do not have the option of adding Red Hat Ceph Storage to the central location later by redeploying.

4.1. DEPLOYING THE CENTRAL CONTROLLERS WITHOUT EDGE STORAGE

You can deploy a distributed compute node cluster without Block storage at edge sites if you use the Object Storage service (swift) as a back end for the Image service (glance) at the central location. A site deployed without block storage cannot be updated later to have block storage due to the differing role and networking profiles for each architecture.

Important: The following procedure uses lvm as the backend for Cinder which is not supported for production. You must deploy a certified block storage solution as a backend for Cinder.

Deploy the central controller cluster in a similar way to a typical overcloud deployment. This cluster does not require any Compute nodes, so you can set the Compute count to 0 to override the default of 1. The central controller has particular storage and Oslo configuration requirements. Use the following procedure to address these requirements.

**Prerequisites**

- You must create network_data.yaml and vip_data.yaml files specific to your environment. You can find sample files in /usr/share/openstack-tripleo-heat-templates/network-data-samples.
- You must create an overcloud-baremetal-deploy.yaml file specific to your environment. You can find sample files in /usr/share/openstack-tripleo-heat-templates/baremetal-samples.

**Procedure**

The following procedure outlines the steps for the initial deployment of the central location.

**NOTE**

The following steps detail the deployment commands and environment files associated with an example DCN deployment without glance multistore. These steps do not include unrelated, but necessary, aspects of configuration, such as networking.

1. Log in to the undercloud as the stack user.
2. Source the stackrc file:
   ```bash
   [stack@director ~]$ source /home/stack/stackrc
   ```
3. Generate an environment file:
4. In the home directory, create directories for each stack that you plan to deploy.

```bash
mkdir /home/stack/central
mkdir /home/stack/dcn0
mkdir /home/stack/dcn1
```

5. Provision networks for the overcloud. This command takes a definition file for overcloud networks as input. You must use the output file in your command to deploy the overcloud:

```bash
(undercloud)$ openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

6. Provision virtual IPs for the overcloud. This command takes a definition file for virtual IPs as input. You must use the output file in your command to deploy the overcloud:

```bash
(undercloud)$ openstack overcloud network vip provision \
--stack central \
--output /home/stack/central/overcloud-vip-deployed.yaml \
/home/stack/central/vip_data.yaml
```

7. Provision bare metal instances. This command takes a definition file for bare metal nodes as input. You must use the output file in your command to deploy the overcloud:

```bash
(undercloud)$ openstack overcloud node provision \
--stack central \
--network-config \
--o /home/stack/central/deployed_metal.yaml \
~/overcloud-baremetal-deploy.yaml
```

8. Create a file called `central/overrides.yaml` with settings similar to the following:

```yaml
parameter_defaults:
  NtpServer:
    - 0.pool.ntp.org
    - 1.pool.ntp.org
  GlanceBackend: swift

- ControllerCount: 3 specifies that three nodes will be deployed. These will use swift for glance, lvm for cinder, and host the control-plane services for edge compute nodes.
- ComputeCount: 0 is an optional parameter to prevent Compute nodes from being deployed with the central Controller nodes.
- GlanceBackend: swift uses Object Storage (swift) as the Image Service (glance) back end.
```

The resulting configuration interacts with the distributed compute nodes (DCNs) in the following ways:
The Image service on the DCN creates a cached copy of the image it receives from the central Object Storage back end. The Image service uses HTTP to copy the image from Object Storage to the local disk cache.

NOTE
The central Controller node must be able to connect to the distributed compute node (DCN) site. The central Controller node can use a routed layer 3 connection.

9. Configure the naming conventions for your site in the site-name.yaml environment file. The Nova availability zone, Cinder storage availability zone must match:

```bash
cat > /home/stack/central/site-name.yaml << EOF
parameter_defaults:
  NovaComputeAvailabilityZone: central
  ControllerExtraConfig:
    nova::availability_zone::default_schedule_zone: central
  NovaCrossAZAttach: false
EOF
```

10. Deploy the central Controller node. For example, you can use a deploy.sh file with the following contents:

```bash
openstack overcloud deploy --templates
  --deployed-server
  --stack central
  --templates /usr/share/openstack-tripleo-heat-templates/
  -n /home/stack/network_data.yaml
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml
  -e /usr/share/openstack-tripleo-heat-templates/environments/podman.yaml
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml
  -e /home/stack/dcn0/overcloud-networks-deployed.yaml
  -e /home/stack/dcn0/overcloud-vip-deployed.yaml
  -e /home/stack/dcn0/deployed_metal.yaml
```

NOTE
You must include heat templates for the configuration of networking in your openstack overcloud deploy command. Designing for edge architecture requires spine and leaf networking. See Spine Leaf Networking for more details.

4.2. DEPLOYING THE CENTRAL SITE WITH STORAGE

To deploy the Image service with multiple stores and Ceph Storage as the back end, complete the following steps:

Prerequisites

- You must create network_data.yaml and vip_data.yaml files specific to your environment. You can find sample files in /usr/share/openstack-tripleo-heat-templates/network-data-samples.
You must create an `overcloud-baremetal-deploy.yaml` file specific to your environment. You can find sample files in /usr/share/openstack-tripleo-heat-templates/baremetal-samples.

- You have hardware for a Ceph cluster at the central location and in each availability zone, or in each geographic location where storage services are required.
- You have hardware for three Image Service (glance) servers at a central location and in each availability zone, or in each geographic location where storage services are required. At edge locations, the Image service is deployed to the DistributedComputeHCI nodes.

**Procedure**

Deploy the Red Hat OpenStack Platform central location so that the Image service (glance) can be used with multiple stores.

1. Log in to the undercloud as the stack user.
2. Source the stackrc file:
   ```bash
   [stack@director ~]$ source /home/stack/stackrc
   ```
3. Generate an environment file `/home/stack/central/central-images-env.yaml`:
   ```bash
   sudo openstack tripleo container image prepare \
   -e containers.yaml \
   --output-env-file /home/stack/central/central-images-env.yaml
   ```
4. Generate roles for the central location using roles appropriate for your environment:
   ```bash
   openstack overcloud roles generate Compute Controller CephStorage \
   -o /home/stack/central/central_roles.yaml
   ```
5. In the home directory, create directories for each stack that you plan to deploy:
   ```bash
   mkdir /home/stack/central
   mkdir /home/stack/dcn0
   mkdir /home/stack/dcn1
   ```
6. Provision networks for the overcloud. This command takes a definition file for overcloud networks as input. You must use the output file in your command to deploy the overcloud:
   ```bash
   openstack overcloud network provision \
   --output /home/stack/central/overcloud-networks-deployed.yaml \
   /home/stack/network_data.yaml
   ```
7. Provision virtual IPs for the overcloud. This command takes a definition file for virtual IPs as input. You must use the output file in your command to deploy the overcloud:
   ```bash
   openstack overcloud network vip provision \
   --stack central \
   --output /home/stack/central/overcloud-vip-deployed.yaml \
   /home/stack/vip_data.yaml
   ```
8. Provision bare metal instances. This command takes a definition file for bare metal nodes as input. You must use the output file in your command to deploy the overcloud:

```
openstack overcloud node provision \
--stack central \
--network-config \
-o /home/stack/central/deployed_metal.yaml \
/home/stack/overcloud-baremetal-deploy.yaml
```

9. If you are deploying the central location with hyperconverged storage, you must create an initial-ceph.conf configuration file using the following parameters. For more information see [link]:

```
$ cat <<EOF > /home/stack/central/initial-ceph.conf
[osd]
osd_memory_target_autotune = true
osd_numa_auto_affinity = true
[mgr]
mgr/cephadm/autotune_memory_target_ratio = 0.2
EOF
```

10. Use the deployed_metal.yaml file as input to the openstack overcloud ceph deploy command. The openstack overcloud ceph deploy command outputs a yaml file that describes the deployed Ceph cluster:

```
openstack overcloud ceph deploy \
/home/stack/central/deployed_metal.yaml \
--config /home/stack/central/initial-ceph.conf \
--output /home/stack/central/deployed_ceph.yaml \
--container-image-prepare ~/containers.yaml \
--network-data /home/stack/network-data.yaml \
--cluster central \
--roles-data /home/stack/central/central_roles.yaml
```

1 Include initial-ceph.com only when deploying hyperconverged infrastructure.

11. Verify a functional Ceph deployment before continuing. Use ssh to connect to a server running the ceph-mon service. In an HCI deployment, this is a controller node. Run the following command:

```
cephadm shell --config /etc/ceph/central.conf \
--keyring /etc/ceph/central.client.admin.keyring
```

**NOTE**

You must use the --config and --keyring parameters.

12. Configure the naming conventions for your site in the site-name.yaml environment file. The Nova availability zone and the Cinder storage availability zone must match:

```
cat > /home/stack/central/site-name.yaml << EOF
parameter_defaults:
```
NovaComputeAvailabilityZone: central
ControllerExtraConfig:
    nova::availability_zone::default_schedule_zone: central
NovaCrossAZAttach: false
CinderStorageAvailabilityZone: central
GlanceBackendID: central
EOF

13. Configure a glance.yaml template with contents similar to the following:

```yaml
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'central rbd glance store'
  GlanceBackendID: central
  CephClusterName: central
```

14. Deploy the stack for the central location:

```bash
openstack overcloud deploy --templates \[b][c]
  --deployed-server \]
  --stack central \]
  --templates /usr/share/openstack-tripleo-heat-templates/ \]
  -r /home/stack/central/central_roles.yaml \]
  -n ~/network-data.yaml \]
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \]
  -e /usr/share/openstack-tripleo-heat-templates/environments/podman.yaml \]
  -e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \]
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \]
  -e /home/stack/central/overcloud-networks-deployed.yaml \]
  -e /home/stack/central/overcloud-vip-deployed.yaml \]
  -e /home/stack/central/deployed_metal.yaml \]
  -e ~/control-plane/glance.yaml
```

15. After you have deployed the overcloud for the central location, data that is needed as input for additional stack deployments for edge sites is exported and placed in the `/home/stack/overcloud-deploy` directory. Ensure that this control-plane-export.yaml file is present:

```bash
stat ~/overcloud-deploy/control-plane/central-export.yaml
```

16. Export Ceph specific data:

```bash
openstack overcloud export ceph \]
  --stack central \]
  --output-file ~/central_ceph_external.yaml
```

### 4.3. INTEGRATING EXTERNAL CEPH

You can deploy the central location of a distributed compute node (DCN) architecture and integrate a pre-deployed Red Hat Ceph Storage solution.
Prerequisites

- You must create `network_data.yaml` and `vip_data.yaml` files specific to your environment. You can find sample files in `/usr/share/openstack-tripleo-heat-templates/network-data-samples`. For more information, see [link].

- You must create an `overcloud-baremetal-deploy.yaml` file specific to your environment. You can find sample files in `/usr/share/openstack-tripleo-heat-templates/baremetal-samples`. For more information, see [link].

- Hardware for a Ceph cluster at the central location and in each availability zone, or in each geographic location where storage services are required.

The following is an example deployment of two or more stacks:

- One stack at the central location called `central`.
- One stack at an edge site called `dcn0`.
- Additional stacks deployed similarly to `dcn0`, such as `dcn1`, `dcn2`, and so on.

![Diagram of Ceph cluster deployment]

Procedure

You can install the central location so that it is integrated with a pre-existing Red Hat Ceph Storage solution by following the process documented in Integrating an Overcloud with an Existing Red Hat Ceph Cluster. There are no special requirements for integrating Red Hat Ceph Storage with the central site of a DCN deployment, however you must still complete DCN specific steps before deploying the overcloud:

1. Log in to the undercloud as the stack user.
2. Source the stackrc file:
   
   ```bash
   [stack@director ~]$ source ~/stackrc
   ```
   
   ```bash
   sudo openstack tripleo container image prepare \
   -e containers.yaml \
   --output-env-file ~/.central/central-images-env.yaml
   ```
4. In the home directory, create directories for each stack that you plan to deploy. Use this to separate templates designed for their respective sites.
mkdir /home/stack/central
mkdir /home/stack/dcn0
mkdir /home/stack/dcn1

5. Provision networks for the overcloud. This command takes a definition file for overcloud networks as input. You must use the output file in your command to deploy the overcloud:

```bash
openstack overcloud network provision \ 
  --output /home/stack/central/overcloud-networks-deployed.yaml \ 
  ~/network_data.yaml
```

6. Provision virtual IPs for the overcloud. This command takes a definition file for virtual IPs as input. You must use the output file in your command to deploy the overcloud:

```bash
openstack overcloud network vip provision \ 
  --stack central \ 
  --output ~/overcloud-vip-deployed.yaml \ 
  ~/vip_data.yaml
```

7. Provision bare metal instances. This command takes a definition file for bare metal nodes as input. You must use the output file in your command to deploy the overcloud:

```bash
openstack overcloud node provision \ 
  --stack central \ 
  --network-config \ 
  -o /home/stack/central/deployed_metal.yaml \ 
  ~/overcloud-baremetal-deploy.yaml
```

8. Configure the naming conventions for your site in the `site-name.yaml` environment file. The Compute (nova) availability zone and the Block Storage (cinder) availability zone must match:

```bash
cat > /home/stack/central/site-name.yaml << EOF
parameter_defaults:
  NovaComputeAvailabilityZone: central
  ControllerExtraConfig:
    nova::availability_zone::default_schedule_zone: central
  NovaCrossAZAttach: false
  CinderStorageAvailabilityZone: central
  GlanceBackendID: central
EOF
```

9. Configure a `glance.yaml` template with contents similar to the following:

```bash
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'central rbd glance store'
  GlanceBackendID: central
  CephClusterName: central
```

10. Deploy the central location:

```bash
openstack overcloud deploy \
```
11. After you have deployed the overcloud for the central location, data that is needed as input for additional stack deployments for edge sites is exported and placed in the `/home/stack/overcloud-deploy` directory. Ensure that this `control-plane-export.yaml` file is present:

```
stat ~/overcloud-deploy/control-plane/control-plane-export.yaml
```

12. Export Ceph specific data:

```
openstack overcloud export ceph
--stack central
--output-file ~/central_ceph_external.yaml
```
CHAPTER 5. DEPLOY THE EDGE WITHOUT STORAGE

You can deploy a distributed compute node (DCN) cluster without block storage at edge sites if you use the Object Storage service (swift) as a back end for the Image service (glance) at the central location. If you deploy a site without block storage, you cannot update it later to have block storage.

**IMPORTANT**

The following procedure uses lvm as the back end for the Block Storage service (cinder), which is not supported for production. You must deploy a certified block storage solution as a back end for the Block Storage service.

5.1. DEPLOYING EDGE NODES WITHOUT STORAGE

When you deploy Compute nodes at an edge site, you use the central location as the control plane. You can add a new DCN stack to your deployment and reuse the configuration files from the central location to create new environment files.

**Prerequisites**

- You must create the `network_data.yaml` file specific to your environment. You can find sample files in `/usr/share/openstack-tripleo-heat-templates/network-data-samples`. For more information, see [link].

- You must create an `overcloud-baremetal-deploy.yaml` file specific to your environment. You can find sample files in `/usr/share/openstack-tripleo-heat-templates/baremetal-samples`. For more information, see [link].

**Procedure**

1. Log in to the undercloud as the stack user.

2. Source the stackrc file:

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

3. Generate an environment file `~/dcn0/dcn0-images-env.yaml`:

   ```bash
   sudo[es] openstack tripleo container image prepare \
   -e containers.yaml \ 
   --output-env-file ~/dcn0/dcn0-images-env.yaml
   ```

4. Generate a roles file for the edge location. Generate roles for the edge location using roles appropriate for your environment:

   ```bash
   (undercloud)$ openstack overcloud roles \
   generate Compute DistributedComputeScaleOut \ 
   -o /home/stack/dcn0/dcn0__roles.yaml
   ```

5. If you are using ML2/OVS for networking overlay, you must edit the Compute role include the NeutronDhcpAgent and NeutronMetadataAgent services:

   a. Create a role file for the Compute role:

      ```bash
      ```
openstack overcloud roles \
generate Compute \
-o /home/stack/dcn0/dcn0_roles.yaml

.. Edit the /home/stack/dcn0/dcn0_roles.yaml file to include the **NeutronDhcpAgent** and **NeutronMetadataAgent** services:

+...

   - OS::TripleO::Services::MySQLClient
   - OS::TripleO::Services::NeutronBgpVpnBagpipe
   + - OS::TripleO::Services::NeutronDhcpAgent
   + - OS::TripleO::Services::NeutronMetadataAgent
   - OS::TripleO::Services::NeutronLinuxbridgeAgent
   - OS::TripleO::Services::NeutronVppAgent
   - OS::TripleO::Services::NovaAZConfig
   - OS::TripleO::Services::NovaCompute

+ For more information, see Preparing for a routed provider network.

1. Provision networks for the overcloud. This command takes a definition file for overcloud networks as input. You must use the output file in your command to deploy the overcloud:

   (undercloud)$ openstack overcloud network provision \
   --output /home/stack/dcn0/overcloud-networks-deployed.yaml \
   /home/stack/dcn0/network_data.yaml

2. Provision bare metal instances. This command takes a definition file for bare metal nodes as input. You must use the output file in your command to deploy the overcloud:

   (undercloud)$ openstack overcloud node provision \
   --stack dcn0 \
   --network-config \
   -o /home/stack/dcn0/deployed_metal.yaml \
   ~/overcloud-baremetal-deploy.yaml

3. Configure the naming conventions for your site in the site-name.yaml environment file.

   parameter_defaults:
   NovaComputeAvailabilityZone: dcn0
   ControllerExtraConfig:
   nova::availability_zone::default_schedule_zone: dcn0
   NovaCrossAZAttach: false

4. Deploy the stack for the dcn0 edge site:

   openstack overcloud deploy --templates \
   --deployed-server \ 
   --stack dcn0 \ 
   --templates /usr/share/openstack-tripleo-heat-templates/ \
   -r /home/stack/dcn0/dcn0_roles.yaml \ 
   -n /home/stack/network_data.yaml \
5. After you have deployed the overcloud for the dcn0 location, data that is needed as input for the central location redeploy is exported and placed in the /home/stack/overcloud-deploy/<stack> directory. Ensure that this file is present:

```
stat /home/stack/overcloud-deploy/dcn0/dcn0-export.yaml
```

### 5.2. EXCLUDING SPECIFIC IMAGE TYPES AT THE EDGE

By default, Compute nodes advertise all image formats that they support. If your Compute nodes do not use Ceph storage, you can exclude RAW images from the image format advertisement. The RAW image format consumes more network bandwidth and local storage than QCOW2 images and is inefficient when used at edge sites without Ceph storage. Use the `NovaImageTypeExcludeList` parameter to exclude specific image formats:

- **IMPORTANT**
  Do not use this parameter at edge sites with Ceph, because Ceph requires RAW images.

- **NOTE**
  Compute nodes that do not advertise RAW images cannot host instances created from RAW images. This can affect snapshot-redeploy and shelving.

**Prerequisites**

- Red Hat OpenStack Platform director is installed
- The central location is installed
- Compute nodes are available for a DCN deployment

**Procedure**

1. Log in to the undercloud host as the `stack` user.
2. Source the `stackrc` credentials file:
   ```
   $ source ~/stackrc
   ```
3. Include the `NovaImageTypeExcludeList` parameter in one of your custom environment files:
   ```yaml
   parameter_defaults:
     NovaImageTypeExcludeList:
       - raw
   ```
4. Include the environment file that contains the `NovaImageTypeExcludeList` parameter in the overcloud deployment command, along with any other environment files relevant to your deployment:

```bash
openstack overcloud deploy --templates \
-n network_data.yaml \
-r roles_data.yaml \
-e <environment_files> \
-e <new_environment_file>
```
CHAPTER 6. DEPLOYING STORAGE AT THE EDGE

You can leverage Red Hat OpenStack Platform director to extend distributed compute node deployments to include distributed image management and persistent storage at the edge with the benefits of using Red Hat OpenStack Platform and Ceph Storage.

6.1. DEPLOYING EDGE SITES WITH HYPERCONVERGED STORAGE

After you deploy the central site, build out the edge sites and ensure that each edge location connects primarily to its own storage back end, as well as to the storage back end at the central location. A spine and leaf networking configuration should be included with this configuration, with the addition of the storage and storage_mgmnt networks that ceph needs. For more information, see Spine Leaf Networking. You must have connectivity between the storage network at the central location and the storage network at each edge site so that you can move Image service (glance) images between sites.

Ensure that the central location can communicate with the mons and OSDs at each of the edge sites. However, you should terminate the storage management network at site location boundaries because the storage management network is used for OSD rebalancing.

Prerequisites

- You must create the network_data.yaml file specific to your environment. You can find sample files in /usr/share/openstack-tripleo-heat-templates/network-data-samples. For more information, see [link].

- You must create an overcloud-baremetal-deploy.yaml file specific to your environment. You can find sample files in /usr/share/openstack-tripleo-heat-templates/baremetal-samples. For more information, see [link].

- You have hardware for three Image Service (glance) servers at a central location and in each availability zone, or in each geographic location where storage services are required. At edge locations, the Image service is deployed to the DistributedComputeHCI nodes.

Procedure

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

   [stack@director ~]$ source ~/stackrc

3. Generate an environment file ~/dcn0/dcn0-images-env.yaml:

   sudo openstack tripleo container image prepare
   -e containers.yaml
   --output-env-file /home/stack/dcn0/dcn0-images-env.yaml

4. Generate the appropriate roles for the dcn0 edge location:

   openstack overcloud roles generate DistributedComputeHCI
   DistributedComputeHCI-ScaleOut
   -o ~/dcn0/dcn0_roles.yaml

5. Provision networks for the overcloud. This command takes a definition file for overcloud networks as input. You must use the output file in your command to deploy the overcloud:

   (undercloud)$ openstack overcloud network provision
   --output /home/stack/dcn0/overcloud-networks-deployed.yaml
   /home/stack/network_data.yaml

6. Provision bare metal instances. This command takes a definition file for bare metal nodes as input. You must use the output file in your command to deploy the overcloud:

   (undercloud)$ openstack overcloud node provision
   --stack dcn0
   --network-config
   -o /home/stack/dcn0/deployed_metal.yaml
   /home/stack/overcloud-baremetal-deploy.yaml

7. If you are deploying the edge site with hyperconverged storage, you must create an `initial-ceph.conf` configuration file with the following parameters. For more information, see [link]:

   [osd]
   osd_memory_target_autotune = true
   osd_numa_auto_affinity = true
   [mgr]
   mgr/cephadm/autotune_memory_target_ratio = 0.2

8. Use the `deployed_metal.yaml` file as input to the `openstack overcloud ceph deploy` command. The `openstack overcloud ceph deploy command` outputs a yaml file that describes the deployed Ceph cluster:

   openstack overcloud ceph deploy
   /home/stack/dcn0/deployed_metal.yaml
   --stack dcn0
   --config ~/dcn0/initial-ceph.conf
   --output ~/dcn0/deployed_ceph.yaml
   --container-image-prepare ~/containers.yaml
Include initial-ceph.conf only when deploying hyperconverged infrastructure.

9. Configure the naming conventions for your site in the site-name.yaml environment file. The Nova availability zone and the Cinder storage availability zone must match:

```yaml
parameter_defaults:
  NovaComputeAvailabilityZone: dcn0
  ControllerExtraConfig:
    nova::availability_zone::default_schedule_zone: dcn0
  NovaCrossAZAttach: false
  CinderStorageAvailabilityZone: dcn0
  CinderVolumeCluster: dcn0
  GlanceBackendID: dcn0
```

10. Configure a glance.yaml template with contents similar to the following:

```yaml
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'dcn0 rbd glance store'
  GlanceBackendID: dcn0
  GlanceMultistoreConfig:
    central:
      GlanceBackend: rbd
      GlanceStoreDescription: 'central rbd glance store'
      CephClusterName: central
```

11. Deploy the stack for the dcn0 location:

```
openstack overcloud deploy --templates \
  --deployed-server \
  --stack dcn0 \
  --templates /usr/share/openstack-tripleo-heat-templates/ \
  -r ~/dcn0/dcn0_roles.yaml \
  -n ~/dcn0/network-data.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/podman.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm-rbd-only.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
  -e /home/stack/overcloud-deploy/central/central-export.yaml \
  -e /home/stack/dcn0/overcloud-networks-deployed.yaml \
  -e /home/stack/dcn0/overcloud-vip-deployed.yaml \
  -e /home/stack/dcn0/deployed_metal.yaml \
  -e ~/control-plane/glance.yaml
```
6.2. USING A PRE-INSTALLED RED HAT CEPH STORAGE CLUSTER AT THE EDGE

You can configure Red Hat OpenStack Platform to use a pre-existing Ceph cluster. This is called an external Ceph deployment.

Prerequisites

- You must have a preinstalled Ceph cluster that is local to your DCN site so that latency requirements are not exceeded.

Procedure

1. Create the following pools in your Ceph cluster. If you are deploying at the central location, include the backups and metrics pools:

   ```
   [root@ceph ~]# ceph osd pool create volumes <_PGnum_>
   [root@ceph ~]# ceph osd pool create images <_PGnum_>
   [root@ceph ~]# ceph osd pool create vms <_PGnum_>
   [root@ceph ~]# ceph osd pool create backups <_PGnum_>
   [root@ceph ~]# ceph osd pool create metrics <_PGnum_>
   ```

   Replace `<_PGnum_> with the number of placement groups. You can use the Ceph Placement Groups (PGs) per Pool Calculator to determine a suitable value.

2. Create the OpenStack client user in Ceph to provide the Red Hat OpenStack Platform environment access to the appropriate pools:

   ```
   ceph auth add client.openstack mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx pool=volumes, allow rwx pool=vms, allow rwx pool=images'
   ```

   Save the provided Ceph client key that is returned. Use this key as the value for the CephClientKey parameter when you configure the undercloud.

   **NOTE**

   If you run this command at the central location and plan to use Cinder backup or telemetry services, add allow rwx pool=backups, allow pool=metrics to the command.

3. Save the file system ID of your Ceph Storage cluster. The value of the fsid parameter in the [global] section of your Ceph configuration file is the file system ID:

   ```
   [global]
   fsid = 4b5c8c0a-f660-454b-a1b4-9747aa737d19
   ... 
   ```

   Use this value as the value for the CephClusterFSID parameter when you configure the undercloud.

4. On the undercloud, create an environment file to configure your nodes to connect to the unmanaged Ceph cluster. Use a recognizable naming convention, such as ceph-external-<SITE>.yaml where SITE is the location for your deployment, such as ceph-external-central.yaml, ceph-external-dcn1.yaml, and so on.
parameter_defaults:
  # The cluster FSID
  CephClusterFSID: '4b5c8c0a-ff60-454b-a1b4-9747aa737d19'
  # The CephX user auth key
  CephClientKey: 'AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJEQAsRnRQ=='
  # The list of IPs or hostnames of the Ceph monitors
  CephExternalMonHost: '172.16.1.7, 172.16.1.8, 172.16.1.9'
  # The desired name of the generated key and conf files
  CephClusterName: dcn1

a. Use the previously saved values for the CephClusterFSID and CephClientKey parameters.

b. Use a comma delimited list of ip addresses from the Ceph monitors as the value for the CephExternalMonHost parameter.

c. You must select a unique value for the CephClusterName parameter amongst edge sites. Reusing a name will result in the configuration file being overwritten.

5. If you deployed Red Hat Ceph storage using Red Hat OpenStack Platform director at the central location, then you can export the ceph configuration to an environment file central_ceph_external.yaml. This environment file connects DCN sites to the central hub Ceph cluster, so the information is specific to the Ceph cluster deployed in the previous steps:

```bash
sudo -E openstack overcloud export ceph
  --stack central
  --config-download-dir /var/lib/mistral
  --output-file ~/dcn-common/central_ceph_external.yaml
```

If the central location has Red Hat Ceph Storage deployed externally, then you cannot use the openstack overcloud export ceph command to generate the central_ceph_external.yaml file. You must create the central_ceph_external.yaml file manually instead:

```
parameter_defaults:
  CephExternalMultiConfig:
    - cluster: "central"
      fsid: "3161a3b4-e5ff-42a0-9f53-860403b29a33"
      external_cluster_mon_ips: "172.16.11.84, 172.16.11.87, 172.16.11.92"
      keys:
        - name: "client.openstack"
          caps:
            mgr: "allow *"
            mon: "profile rbd"
            osd: "profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=images"
          key: "AQD29WteAAAAABAAAphgOjFD7nyjdYe8Lz0mQ5Q=="
          mode: "0600"
        dashboard_enabled: false
        ceph_conf_overrides:
          client:
            keyring: /etc/ceph/central.client.openstack.keyring
```

6. Create an environment file with similar details about each site with an unmanaged Red Hat Ceph Storage cluster for the central location. The openstack overcloud export ceph command does not work for sites with unmanaged Red Hat Ceph Storage clusters. When you update the central location, this file will allow the central location the storage clusters at your edge sites as secondary locations.
7. Use the ceph-ansible-external.yaml, ceph-external-<SITE>.yaml, and the central_ceph_external.yaml environment files when deploying the overcloud:

```bash
openstack overcloud deploy \ 
  --stack dcn1 \ 
  --templates /usr/share/openstack-tripleo-heat-templates/ \ 
  -r ~/dcn1/roles_data.yaml \ 
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml \ 
  -e /usr/share/openstack-tripleo-heat-templates/environments/dcn-hci.yaml \ 
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \ 
  -e ~/dcn1/ceph-external-dcn1.yaml \ 
  .... \ 
  -e ~/dcn-common/central-export.yaml \ 
  -e ~/dcn-common/central_ceph_external.yaml \ 
  -e ~/dcn1/dcn_ceph_keys.yaml \ 
  -e ~/dcn1/role-counts.yaml \ 
  -e ~/dcn1/ceph.yaml \ 
  -e ~/dcn1/site-name.yaml \ 
  -e ~/dcn1/tuning.yaml \ 
  -e ~/dcn1/glance.yaml
```

8. Redeploy the central location after all edge locations have been deployed.

## 6.3. UPDATING THE CENTRAL LOCATION

**WARNING**

The content for this feature is available in this release as a *Documentation Preview*, and therefore is not fully verified by Red Hat. Use it only for testing, and do not use in a production environment.

After you configure and deploy all of the edge sites using the sample procedure, update the configuration at the central location so that the central Image service can push images to the edge sites.
WARNING

This procedure restarts the Image service (glance) and interrupts any long running Image service process. For example, if an image is being copied from the central Image service server to a DCN Image service server, that image copy is interrupted and you must restart it. For more information, see Clearing residual data after interrupted Image service processes.

Procedure

1. Create a `~/central/glance_update.yaml` file similar to the following. This example includes a configuration for two edge sites, dcn0 and dcn1:

   ```yaml
   parameter_defaults:
     GlanceEnabledImportMethods: web-download,copy-image
     GlanceBackend: rbd
     GlanceStoreDescription: 'central rbd glance store'
     CephClusterName: central
     GlanceBackendID: central
     GlanceMultistoreConfig:
       dcn0:
         GlanceBackend: rbd
         GlanceStoreDescription: 'dcn0 rbd glance store'
         CephClientUserName: 'openstack'
         CephClusterName: dcn0
         GlanceBackendID: dcn0
       dcn1:
         GlanceBackend: rbd
         GlanceStoreDescription: 'dcn1 rbd glance store'
         CephClientUserName: 'openstack'
         CephClusterName: dcn1
         GlanceBackendID: dcn1
   ```

2. Create the `dcn_ceph.yaml` file. In the following example, this file configures the glance service at the central site as a client of the Ceph clusters of the edge sites, dcn0 and dcn1.

   ```bash
   openstack overcloud export ceph \
   --stack dcn0,dcn1 \
   --output-file ~/central/dcn_ceph.yaml
   ```

3. Redeploy the central site using the original templates and include the newly created `dcn_ceph.yaml` and `glance_update.yaml` files.

   ```bash
   openstack overcloud deploy --templates \
   --deployed-server \
   --stack central \
   --templates /usr/share/openstack-tripleo-heat-templates/ \
   -r ~/control-plane/central_roles.yaml \
   -n ~/network-data.yaml \
   -e /usr/share/openstack-tripleo-heat-templates.environments/network-environment.yaml \
   -e /usr/share/openstack-tripleo-heat-templates.environments/podman.yaml \
   ```
openstack overcloud deploy \
   --stack central \
   --templates /usr/share/openstack-tripleo-heat-templates/ \
   -r ~/central/central_roles.yaml \
   ...
   -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
   -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
   -e ~/central/central-images-env.yaml \
   -e ~/central/role-counts.yaml \
   -e ~/central/site-name.yaml \
   -e ~/central/ceph.yaml \
   -e ~/central/ceph_keys.yaml \
   -e ~/central/glance.yaml \
   -e ~/central/dcn_ceph_external.yaml 

1. On a controller at the central location, restart the cinder-volume service. If you deployed the central location with the cinder-backup service, then restart the cinder-backup service too:

   ssh heat-admin@controller-0 sudo pcs resource restart openstack-cinder-volume
   ssh heat-admin@controller-0 sudo pcs resource restart openstack-cinder-backup

6.3.1. Clearing residual data after interrupted Image service processes

When you restart the central location, any long-running Image service (glance) processes are interrupted. Before you can restart these processes, you must first clean up residual data on the Controller node that you rebooted, and in the Ceph and Image service databases.

Procedure

1. Check and clear residual data in the Controller node that was rebooted. Compare the files in the glance-api.conf file for staging store with the corresponding images in the Image service database, for example <image_ID>.raw.

   • If these corresponding images show importing status, you must recreate the image.
   
   • If the images show active status, you must delete the data from staging and restart the copy import.

2. Check and clear residual data in Ceph stores. The images that you cleaned from the staging area must have matching records in their stores property in the Ceph stores that contain the image. The image name in Ceph is the image id in the Image service database.
3. Clear the Image service database. Clear any images that are in importing status from the import jobs there were interrupted:

   $ glance image-delete <image_id>

6.4. DEPLOYING RED HAT CEPH STORAGE DASHBOARD ON DCN

Procedure

To deploy the Red Hat Ceph Storage Dashboard to the central location, see Adding the Red Hat Ceph Storage Dashboard to an overcloud deployment. These steps should be completed prior to deploying the central location.

To deploy Red Hat Ceph Storage Dashboard to edge locations, complete the same steps that you completed for central, however you must complete the following following:

- Ensure that the ManageNetworks parameter has a value of false in your templates for deploying the edge site. When you set ManageNetworks to false, Edge sites will use the existing networks that were already created in the central stack:

  ```yaml
  parameter_defaults:
    ManageNetworks: false
  ```

- You must deploy your own solution for load balancing in order to create a high availability virtual IP. Edge sites do not deploy haproxy, nor pacemaker. When you deploy Red Hat Ceph Storage Dashboard to edge locations, the deployment is exposed on the storage network. The dashboard is installed on each of the three DistributedComputeHCI nodes with distinct IP addresses without a load balancing solution.

You can create an additional network to host virtual IP where the Ceph dashboard can be exposed. You must not be reusing network resources for multiple stacks. For more information on reusing network resources, see Reusing network resources in multiple stacks.

To create this additional network resource, use the provided network_data_dashboard.yaml heat template. The name of the created network is StorageDashboard.

Procedure

1. Log in to Red Hat OpenStack Platform Director as stack.

2. Generate the DistributedComputeHCIDashboard role and any other roles appropriate for your environment:

   ```bash
   openstack overcloud roles generate DistributedComputeHCIDashboard -o ~/dnc0/roles.yaml
   ```

3. Include the roles.yaml and the network_data_dashboard.yaml in the overcloud deploy command:

   ```bash
   $ openstack overcloud deploy --templates \
   -r ~/<dcn>/<dcn_site_roles>.yaml \
   -n /usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml \
   -e <overcloud_environment_files> \
   ... \
   -e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm-rbd-
   ```
only.yaml
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/ceph-dashboard.yaml

NOTE
The deployment provides the three ip addresses where the dashboard is enabled on the storage network.

Verification
To confirm the dashboard is operational at the central location and that the data it displays from the Ceph cluster is correct, see Accessing Ceph Dashboard.

You can confirm that the dashboard is operating at an edge location through similar steps, however, there are exceptions as there is no load balancer at edge locations.

1. Retrieve dashboard admin login credentials specific to the selected stack from /var/lib/mistral/<stackname>/ceph-ansible/group_vars/all.yml

2. Within the inventory specific to the selected stack, /var/lib/mistral/<stackname>/ceph-ansible/inventory.yml, locate the DistributedComputeHCI role hosts list and save all three of the storage_ip values. In the example below the first two dashboard IPs are 172.16.11.84 and 172.16.11.87:

   DistributedComputeHCI:
   hosts:
     dcn1-distributed-compute-hci-0:
       ansible_host: 192.168.24.16
     ...
     storage_hostname: dcn1-distributed-compute-hci-0.storage.localdomain
     storage_ip: 172.16.11.84
     ...
     dcn1-distributed-compute-hci-1:
       ansible_host: 192.168.24.22
     ...
     storage_hostname: dcn1-distributed-compute-hci-1.storage.localdomain
     storage_ip: 172.16.11.87

3. You can check that the Ceph Dashboard is active at one of these IP addresses if they are accessible to you. These IP addresses are on the storage network and are not routed. If these IP addresses are not available, you must configure a load balancer for the three IP addresses that you get from the inventory to obtain a virtual IP address for verification.
CHAPTER 7. DEPLOYING WITH KEY MANAGER

If you have deployed edge sites previous to the release of Red Hat OpenStack Platform 16.1.2, you will need to regenerate roles.yaml to implement this feature: To implement the feature, regenerate the roles.yaml file used for the DCN site’s deployment.

```
$ openstack overcloud roles generate DistributedComputeHCI DistributedComputeHCIScaleOut -o ~/dcn0/roles_data.yaml
```

7.1. DEPLOYING EDGE SITES WITH KEY MANAGER

If you want to include access to the Key Manager (barbican) service at edge sites, you must configure barbican at the central location. For information on installing and configuring barbican, see Deploying Barbican.

- You can configure access to barbican from DCN sites by including the `/usr/share/openstack-tripleo-heat-templates/environments/services/barbican-edge.yaml`

```
openstack overcloud deploy \n  --stack dcn0 \n  --templates /usr/share/openstack-tripleo-heat-templates/ \n  -r ~/dcn0/roles_data.yaml \n  .... \n  -e /usr/share/openstack-tripleo-heat-templates/environments/services/barbican-edge.yaml
```
CHAPTER 8. PRECACHING GLANCE IMAGES INTO NOVA

When you configure OpenStack Compute to use local ephemeral storage, glance images are cached to quicken the deployment of instances. If an image that is necessary for an instance is not already cached, it is downloaded to the local disk of the Compute node when you create the instance.

The process of downloading a glance image takes a variable amount of time, depending on the image size and network characteristics such as bandwidth and latency.

If you attempt to start an instance, and the image is not available on the on the Ceph cluster that is local, launching an instance will fail with the following message:

```
Build of instance 3c04e982-c1d1-4364-b6bd-f876e99325b aborted: Image 20c5ff9d-5f54-4b74-830f-88e78b9999ed is unacceptable: No image locations are accessible
```

You see the following in the Compute service log:

```
'Image %s is not on my ceph and [workarounds]/ never_download_image_if_on_rbd=True; refusing to fetch and upload.',
```

The instance fails to start due to a parameter in the `nova.conf` configuration file called `never_download_image_if_on_rbd`, which is set to `true` by default for DCN deployments. You can control this value using the heat parameter `NovaDisableImageDownloadToRbd` which you can find in the `dcn-hci.yaml` file.

If you set the value of `NovaDisableImageDownloadToRbd` to `false` prior to deploying the overcloud, the following occurs:

- The Compute service (nova) will automatically stream images available at the central location if they are not available locally.
- You will not be using a COW copy from glance images.
- The Compute (nova) storage will potentially contain multiple copies of the same image, depending on the number of instances using it.
- You may saturate both the WAN link to the central location as well as the nova storage pool.

Red Hat recommends leaving this value set to true, and ensuring required images are available locally prior to launching an instance. For more information on making images available to the edge, see Section A.1.3, “Copying an image to a new site”.

For images that are local, you can speed up the creation of VMs by using the `tripleo_nova_image_cache.yml` ansible playbook to pre-cache commonly used images or images that are likely to be deployed in the near future.

8.1. RUNNING THE TRIPLEO_NOVA_IMAGE_CACHE.YML ANSIBLE PLAYBOOK

Prerequisites

- Authentication credentials to the correct API in the shell environment.

Before the command provided in each step, you must ensure that the correct authentication file is sourced.
Procedure

1. Create an ansible inventory file for the stack. You can specify multiple stacks in a comma delimited list to cache images at more than one site:

   ```
   $ source stackrc
   $ tripleo-ansible-inventory --plan central,dcn0,dcn1 --static-yaml-inventory inventory.yaml
   ```

2. Create a list of image IDs that you want to pre-cache:
   a. Retrieve a comprehensive list of available images:

   ```
   $ source centralrc
   $ openstack image list
   +--------------------------------------+---------+--------+
   | ID                                   | Name    | Status |
   +--------------------------------------+---------+--------+
   | 07bc2424-753b-4f65-9da5-5a99d8383fe6 | image_0 | active |
   | d5187afa-c821-4f22-aa4b-4e76382bef86 | image_1 | active |
   +--------------------------------------+---------+--------+
   ```

   b. Create an ansible playbook argument file called `nova_cache_args.yml`, and add the IDs of the images that you want to pre-cache:

   ```
   ---
   tripleo_nova_image_cache_images:
   - id: 07bc2424-753b-4f65-9da5-5a99d8383fe6
   - id: d5187afa-c821-4f22-aa4b-4e76382bef86
   ```

3. Run the `tripleo_nova_image_cache.yml` ansible playbook:

   ```
   $ source centralrc
   $ ansible-playbook -i inventory.yaml --extra-vars "@nova_cache_args.yml"
   /usr/share/ansible/tripleo-playbooks/tripleo_nova_image_cache.yml
   ```

8.2. PERFORMANCE CONSIDERATIONS

You can specify the number of images that you want to download concurrently with the ansible `forks` parameter, which defaults to a value of 5. You can reduce the time to distribute this image by increasing the value of the `forks` parameter, however you must balance this with the increase in network and glance-api load.

Use the `--forks` parameter to adjust concurrency as shown:

```
ansible-playbook -i inventory.yaml --forks 10 --extra-vars "@nova_cache_args.yml"
/usr/share/ansible/tripleo-playbooks/tripleo_nova_image_cache.yml
```
8.3. OPTIMIZING THE IMAGE DISTRIBUTION TO DCN SITES

You can reduce WAN traffic by using a proxy for glance image distribution. When you configure a proxy:

- Glance images are downloaded to a single Compute node that acts as the proxy.
- The proxy redistributes the glance image to other Compute nodes in the inventory.

You can place the following parameters in the `nova_cache_args.yml` ansible argument file to configure a proxy node.

Set the `tripleo_nova_image_cache_use_proxy` parameter to `true` to enable the image cache proxy.

The image proxy uses secure copy `scp` to distribute images to other nodes in the inventory. SCP is inefficient over networks with high latency, such as a WAN between DCN sites. Red Hat recommends that you limit the playbook target to a single DCN location, which correlates to a single stack.

Use the `tripleo_nova_image_cache_proxy_hostname` parameter to select the image cache proxy. The default proxy is the first compute node in the ansible inventory file. Use the `tripleo_nova_image_cache_plan` parameter to limit the playbook inventory to a single site:

```yaml
tripleo_nova_image_cache_use_proxy: true
tripleo_nova_image_cache_proxy_hostname: dcn0-novacompute-1
tripleo_nova_image_cache_plan: dcn0
```

8.4. CONFIGURING THE NOVA-CACHE CLEANUP

A background process runs periodically to remove images from the nova cache when both of the following conditions are true:

- The image is not in use by an instance.
- The age of the image is greater than the value for the nova parameter `remove_unused_original_minimum_age_seconds`.

The default value for the `remove_unused_original_minimum_age_seconds` parameter is **86400**, the value is expressed in seconds and is equal to 24 hours. You can control this value with the `NovaImageCacheTTL` tripleo-heat-templates parameter during the initial deployment, or during a stack update of your cloud:

```yaml
parameter_defaults:
  NovaImageCacheTTL: 604800 # Default to 7 days for all compute roles
  Compute2Parameters:
    NovaImageCacheTTL: 1209600 # Override to 14 days for the Compute2 compute role
```

When you instruct the playbook to pre-cache an image that already exists on a Compute node, ansible does not report a change, but the age of the image is reset to 0. Run the ansible play more frequently than the value of the `NovaImageCacheTTL` parameter to maintain a cache of images.
You can enable TLS (transport layer security) on clouds designed for distributed compute node infrastructure. You have the option of either enabling TLS for public access only, or enabling TLS on every network with TLS-e, which allows for encryption on all internal and external dataflows.

You cannot enable public access on edge stacks as edge sites do not have public endpoints. For more information on TLS for public access, see [Enabling SSL/TLS on Overcloud Public Endpoints](#).

### 9.1. Deploying Distributed Compute Node Architecture with TLS-E

**Prerequisites**

When you configure TLS-e on Red Hat OpenStack Platform (RHOSP) distributed compute node architecture with Red Hat Identity Manager (IdM), take the following actions based on the version of Red Hat Enterprise Linux deployed for Red Hat Identity Manager.

**Red Hat Enterprise Linux 8.4**

1. On the Red Hat Identity Management node, allowed trusted subnets to an ACL in the **ipa-ext.conf** file:

   ```
   acl "trusted_network" {
   localnets;
   localhost;
   192.168.24.0/24;
   192.168.25.0/24;
   };
   ```

   1. In the **/etc/named/ipa-options-ext.conf** file, allow recursion, and query cache:

   ```
   allow-recursion { trusted_network; };
   allow-query-cache { trusted_network; };
   ```

2. Restart the `named-pkcs11` service:

   ```
   systemctl restart named-pkcs11
   ```

**Red Hat Enterprise Linux 8.2**

[1]: #
If you have Red Hat Identity Manager (IdM) on Red Hat Enterprise Linux (RHEL) 8.2, you must upgrade Red Hat Enterprise Linux and then follow the directions for RHEL 8.4

**Red Hat Enterprise Linux 7.x**

If you have Red Hat Identity Manager (IdM) on Red Hat Enterprise Linux (RHEL) 7.x, you must add an access control instruction (ACI) for your domain name manually. For example, if the domain name is `redhat.local`, run the following commands on Red Hat Identity Manager to configure the ACI:

```bash
ADMIN_PASSWORD=redhat_01
DOMAIN_LEVEL_1=local
DOMAIN_LEVEL_2=redhat

cat << EOF | ldapmodify -x -D "cn=Directory Manager" -w $ADMIN_PASSWORD
dn: cn=dns,dc=${DOMAIN_LEVEL_2},dc=${DOMAIN_LEVEL_1}
changetype: modify
add: aci
aci: (targetattr = "aaaarecord || arecord || cnamerecord || idnsname || objectclass || pptrrecord")
(targetfilter = "(&(objectclass=idnsrecord)(|(aaaarecord=)(arecord=)(cnamerecord=)(ptrrecord=)
(idnsZoneActive=TRUE)))")
(version 3.0; acl "Allow hosts to read DNS A/AAA/CNAME/PTR records";
allow (read,search,compare) userdn = "ldap:///fqdn=*,cn=computers,cn=accounts,dc=${DOMAIN_LEVEL_2},dc=${DOMAIN_LEVEL_1}";
EOF
```

**Procedure**

For distributed compute node (DCN) architectures, it is required to use the ansible-based `tripleo-ipa` method of implementing TLS-e as opposed to the previous `novajoin` method. For more information on deploying TLS-e with `tripleo-ipa` see [Implementing TLS-e with Ansible](#).

To deploy TLS-e with `tripleo-ipa` for DCN architectures, you will need to also complete the following steps:

1. If you are deploying storage at the edge, include the following parameters in your modified `tripleo` heat templates for edge stacks:

   ```
   TEMPLATES=/usr/share/openstack-tripleo-heat-templates
   resource_registry:
   OS::TripleO::Services::IpaClient:
   ${TEMPLATES}/deployment/ipa/ipaservices-baremetal-ansible.yaml
   ```

   Due to differences in design between the central and edge locations, do not include the following files in edge stacks:

   **tls-everywhere-endpoints-dns.yaml**
   
   This file is ignored at edge sites, the endpoints that it sets are overridden by the endpoints exported from the central stack.

   **haproxy-public-tls-certmonger.yaml**
   
   This file causes a failed deployment as there are no public endpoints at the edge.
CHAPTER 10. CREATING A CEPH KEY FOR EXTERNAL ACCESS

WARNING

The content for this feature is available in this release as a Documentation Preview, and therefore is not fully verified by Red Hat. Use it only for testing, and do not use in a production environment.

External access to Ceph storage is access to Ceph from any site that is not local. Ceph storage at the central location is external for edge (DCN) sites, just as Ceph storage at the edge is external for the central location.

When you deploy the central or DCN sites with Ceph storage, you have the option of using the default openstack keyring for both local and external access. Alternatively, you can create a separate key for access by non-local sites.

If you decide to use additional Ceph keys for access to your external sites, each key must have the same name. The key name is external in the examples that follow.

If you use a separate key for access by non-local sites, you have the additional security benefit of being able to revoke and re-issue the external key in response to a security event without interrupting local access. However, using a separate key for external access will result in the loss of access to some features, such as cross availability zone backups and offline volume migration. You must balance the needs of your security posture against the desired feature set.

By default, the keys for the central and all DCN sites will be shared.

10.1. CREATING A CEPH KEY FOR EXTERNAL ACCESS

Complete the following steps to create an external key for non-local access.

Process

1. Create a Ceph key for external access. This key is sensitive. You can generate the key using the following:

   ```
   python3 -c 'import os,struct,time,base64; key = os.urandom(16) ;
   header = struct.pack("<hii", 1, int(time.time()), 0, len(key)) ;
   print(base64.b64encode(header + key).decode())'
   ```

2. In the directory of the stack you are deploying, create a `ceph_keys.yaml` environment file with contents like the following, using the output from the previous command for the key:

   ```yaml
   parameter_defaults:
   CephExtraKeys:
     - name: "client.external"
       caps:
         mgr: *allow *"
   ```
mon: "profile rbd"
  osd: "profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=images"
  key: "AQD29WteAAAAABAAphgOjFD7nyjdYe8Lz0mQ5Q=="
  mode: "0600"

3. Include the ceph_keys.yaml environment file in the deployment of the site. For example, to deploy the central site with with the ceph_keys.yaml environment file, run a command like the following:

```bash
overcloud deploy
  --stack central
  --templates /usr/share/openstack-tripleo-heat-templates/
  ...
  -e ~/central/ceph_keys.yaml
```

### 10.2. USING EXTERNAL CEPH KEYS

You can only use keys that have already been deployed. For information on deploying a site with an external key, see Section 10.1, "Creating a Ceph key for external access". This should be done for both central and edge sites.

- When you deploy an edge site that will use an external key provided by central, complete the following:

  1. Create dcn_ceph_external.yaml environment file for the edge site. You must include the cephx-key-client-name option to specify the deployed key to include.

```bash
sudo -E openstack overcloud export ceph
  --stack central
  --config-download-dir /var/lib/mistral
  --cephx-key-client-name external
  --output-file ~/dcn-common/dcn_ceph_external.yaml
```

  2. Include the dcn_ceph_external.yaml file so that the edge site can access the Ceph cluster at the central site. Include the ceph_keys.yaml file to deploy an external key for the Ceph cluster at the edge site.

- When you update the central location after deploying your edge sites, ensure the central location to use the dcn external keys:

  1. Ensure that the CephClientUserName parameter matches the key being exported. If you are using the name external as shown in these examples, create glance_update.yaml to be similar to the following:

```yaml
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'central rbd glance store'
  CephClusterName: central
  GlanceBackendID: central
  GlanceMultistoreConfig:
    dcn0:
      GlanceBackend: rbd
      GlanceStoreDescription: 'dcn0 rbd glance store'
      CephClientUserName: 'external'
```
2. Use the `openstack overcloud export ceph` command to include the `external` keys for DCN edge access from the central location. To do this you must provide a comma-delimited list of stacks for the `--stack` argument, and include the `cephx-key-client-name` option:

```
sudo -E openstack overcloud export ceph
    --stack dcn0,dcn1,dcn2
    --config-download-dir /var/lib/mistral
    --cephx-key-client-name external
    --output-file ~/central/dcn_ceph_external.yaml
```

3. Redeploy the central site using the original templates and include the newly created `dcn_ceph_external.yaml` and `glance_update.yaml` files.

```
openstack overcloud deploy
    --stack central
    --templates /usr/share/openstack-tripleo-heat-templates/
    -r ~/central/central_roles.yaml
    ...
    -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml
    -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml
    -e ~/central/central-images-env.yaml
    -e ~/central/role-counts.yaml
    -e ~/central/site-name.yaml
    -e ~/central/ceph.yaml
    -e ~/central/ceph_keys.yaml
    -e ~/central/glance.yaml
    -e ~/central/dcn_ceph_external.yaml
```
APPENDIX A. DEPLOYMENT MIGRATION OPTIONS

This section includes topics related validation of DCN storage, as well as migrating or changing architectures.

A.1. VALIDATING EDGE STORAGE

Ensure that the deployment of central and edge sites are working by testing glance multi-store and instance creation.

You can import images into glance that are available on the local filesystem or available on a web server.

NOTE

Always store an image copy in the central site, even if there are no instances using the image at the central location.

Prerequisites

1. Check the stores that are available through the Image service by using the `glance stores-info` command. In the following example, three stores are available: central, dcn1, and dcn2. These correspond to glance stores at the central location and edge sites, respectively:

```
$ glance stores-info

+----------+----------------------------------------------------------------------------------+
| Property | Value                                                                            |
+----------+----------------------------------------------------------------------------------+
| stores   | ["default": "true", "id": "central", "description": "central rbd glance store"], |
|          |{"id": "dcn0", "description": "dcn0 rbd glance store"},                          |
|          |{"id": "dcn1", "description": "dcn1 rbd glance store"}]                          |
+----------+----------------------------------------------------------------------------------+
```

A.1.1. Importing from a local file

You must upload the image to the central location’s store first, then copy the image to remote sites.

1. Ensure that your image file is in RAW format. If the image is not in raw format, you must convert the image before importing it into the Image service:

```
file cirros-0.5.1-x86_64-disk.img

cirros-0.5.1-x86_64-disk.img: QEMU QCOW2 Image (v3), 117440512 bytes

qemu-img convert -f qcow2 -O raw cirros-0.5.1-x86_64-disk.img cirros-0.5.1-x86_64-disk.raw

Import the image into the default back end at the central site:

```

```

```

glance image-create \
--disk-format raw --container-format bare \
--name cirros --file cirros-0.5.1-x86_64-disk.raw \
--store central

```
```
A.1.2. Importing an image from a web server

If the image is hosted on a web server, you can use the GlanceImageImportPlugins parameter to upload the image to multiple stores.

This procedure assumes that the default image conversion plugin is enabled in glance. This feature automatically converts QCOW2 file formats into RAW images, which are optimal for Ceph RBD. You can confirm that a glance image is in RAW format by running the glance image-show ID | grep disk_format.

Procedure

1. Use the image-create-via-import parameter of the glance command to import an image from a web server. Use the --stores parameter.

```
# glance image-create-via-import 
--disk-format qcow2 
--container-format bare 
--name cirros 
--uri http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img 
--import-method web-download 
--stores central,dcn1
```

In this example, the qcow2 cirros image is downloaded from the official Cirros site, converted to RAW by glance, and imported into the central site and edge site 1 as specified by the --stores parameter.

Alternatively you can replace --stores with --all-stores True to upload the image to all of the stores.

A.1.3. Copying an image to a new site

You can copy existing images from the central location to edge sites, which gives you access to previously created images at newly established locations.

1. Use the UUID of the glance image for the copy operation:

```
ID=$(openstack image show cirros -c id -f value)
glance image-import $ID --stores dcn0,dcn1 --import-method copy-image
```

NOTE

In this example, the --stores option specifies that the cirros image will be copied from the central site to edge sites dcn1 and dcn2. Alternatively, you can use the --all-stores True option, which uploads the image to all the stores that don't currently have the image.

2. Confirm a copy of the image is in each store. Note that the stores key, which is the last item in the properties map, is set to central,dcn0,dcn1:

```
$ openstack image show $ID | grep properties
| properties       | direct_url=
| rbd://d25504ce-432d-b6fa-79854d786f2b/images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076/snap, locations=[{"url:
| rbd://d25504ce-459f-432d-b6fa-79854d786f2b/images/8083c7e7-32d8-4f7a-b1da-
```

APPENDIX A. DEPLOYMENT MIGRATION OPTIONS
NOTE

Always store an image copy in the central site even if there is no VM using it on that site.

A.1.4. Confirming that an instance at an edge site can boot with image based volumes

You can use an image at the edge site to create a persistent root volume.

Procedure

1. Identify the ID of the image to create as a volume, and pass that ID to the `openstack volume create` command:

   IMG_ID=$(openstack image show cirros -c id -f value)
   openstack volume create --size 8 --availability-zone dcn0 pet-volume-dcn0 --image $IMG_ID

2. Identify the volume ID of the newly created volume and pass it to the `openstack server create` command:

   VOL_ID=$(openstack volume show -f value -c id pet-volume-dcn0)
   openstack server create --flavor tiny --key-name dcn0-key --network dcn0-network --security-group basic --availability-zone dcn0 --volume $VOL_ID pet-server-dcn0

3. You can verify that the volume is based on the image by running the rbd command within a ceph-mon container at the dcn0 edge site to list the volumes pool.

   $ sudo podman exec ceph-mon-$HOSTNAME rbd --cluster dcn0 -p volumes ls -l
   NAME SIZE PARENT FMT PROT LOCK
   volume-28c6fc32-047b-4306-ad2d-de2be02716b7 8 GiB images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076@snap excl
   $

4. Confirm that you can create a cinder snapshot of the root volume of the instance. Ensure that the server is stopped to quiesce data to create a clean snapshot. Use the `--force` option, because the volume status remains `in-use` when the instance is off.

   openstack server stop pet-server-dcn0
   openstack volume snapshot create pet-volume-dcn0-snap --volume $VOL_ID --force
   openstack server start pet-server-dcn0

5. List the contents of the volumes pool on the dcn0 Ceph cluster to show the newly created snapshot.
A.1.5. Confirming image snapshots can be created and copied between sites

1. Verify that you can create a new image at the dcn0 site. Ensure that the server is stopped to quiesce data to create a clean snapshot:

```
NOVA_ID=$(openstack server show pet-server-dcn0 -f value -c id)
openstack server stop $NOVA_ID
openstack server image create --name cirros-snapshot $NOVA_ID
openstack server start $NOVA_ID
```

2. Copy the image from the dcn0 edge site back to the hub location, which is the default back end for glance:

```
IMAGE_ID=$(openstack image show cirros-snapshot -f value -c id)
glance image-import $IMAGE_ID --stores central --import-method copy-image
```

For more information on glance multistore operations, see [Image service with multiple stores](#).

A.2. MIGRATING TO A SPINE AND LEAF DEPLOYMENT

It is possible to migrate an existing cloud with a pre-existing network configuration to one with a spine leaf architecture. For this, the following conditions are needed:

- All bare metal ports must have their `physical-network` property value set to `ctlplane`.
- The parameter `enable_routed_networks` is added and set to `true` in undercloud.conf, followed by a re-run of the undercloud installation command, `openstack undercloud install`.

Once the undercloud is re-deployed, the overcloud is considered a spine leaf, with a single leaf `leaf0`. You can add additional provisioning leaves to the deployment through the following steps.

1. Add the desired subnets to undercloud.conf as shown in [Configuring routed spine-leaf in the undercloud](#).

2. Re-run the undercloud installation command, `openstack undercloud install`.

3. Add the desired additional networks and roles to the overcloud templates, `network_data.yaml` and `roles_data.yaml` respectively.

**NOTE**

If you are using the `{network.name}`InterfaceRoutes parameter in the network configuration file, then you’ll need to ensure that the `NetworkDeploymentActions` parameter includes the value `UPDATE`. 
4. Finally, re-run the overcloud installation script that includes all relevant heat templates for your cloud deployment.

A.3. MIGRATING TO A MULTISTACK DEPLOYMENT

You can migrate from a single stack deployment to a multistack deployment by treating the existing deployment as the central site, and adding additional edge sites.

The ability to migrate from single to multistack in this release is a Technology Preview, and therefore is not fully supported by Red Hat. It should only be used for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see Scope of Coverage Details.

You cannot split the existing stack. You can scale down the existing stack to remove compute nodes if needed. These compute nodes can then be added to edge sites.

NOTE

This action creates workload interruptions if all compute nodes are removed.

A.4. BACKING UP AND RESTORING ACROSS EDGE SITES

You can back up and restore Block Storage service (cinder) volumes across distributed compute node (DCN) architectures in edge site and availability zones. The cinder-backup service runs in the central availability zone (AZ), and backups are stored in the central AZ. The Block Storage service does not store backups at DCN sites.

Prerequisites

- The central site is deployed with the cinder-backup.yaml environment file located in /usr/share/openstack-tripleo-heat-templates/environments. For more information, see Block Storage backup service deployment.
- The Block Storage service (cinder) CLI is available.
- All sites must use a common openstack cephx client name. For more information, see Creating a Ceph key for external access.

Procedure

1. Create a backup of a volume in the first DCN site:

   $ cinder --os-volume-api-version 3.51 backup-create --name <volume_backup> --availability-zone <az_central> <edge_volume>

   - Replace <volume_backup> with a name for the volume backup.
   - Replace <az_central> with the name of the central availability zone that hosts the cinder-backup service.
   - Replace <edge_volume> with the name of the volume that you want to back up.
If you experience issues with Ceph keyrings, you might need to restart the `cinder-backup` container so that the keyrings copy from the host to the container successfully.

2. Restore the backup to a new volume in the second DCN site:

```bash
$cinder --os-volume-api-version 3.51 create --availability-zone <az_2> --name <new_volume> --backup-id <volume_backup> <volume_size>
```

- Replace `<az_2>` with the name of the availability zone where you want to restore the backup.
- Replace `<new_volume>` with a name for the new volume.
- Replace `<volume_backup>` with the name of the volume backup that you created in the previous step.
- Replace `<volume_size>` with a value in GB equal to or greater than the size of the original volume.