Red Hat OpenStack Services on OpenShift 18.0-beta

Configuring storage

Configuring storage services for Red Hat OpenStack Services on OpenShift

Last Updated: 2024-06-06

Red Hat OpenStack Services on OpenShift 18.0-beta Configuring storage

Configuring storage services for Red Hat OpenStack Services on OpenShift

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This guide contains procedures for configuring persistent and ephemeral storage services for your Red Hat OpenStack Services on OpenShift environment.

Table of Contents

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.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Tell us how we can make it better.

Providing documentation feedback in Jira

Use the Create Issue form to provide feedback on the documentation. The Jira issue will be created in the Red Hat OpenStack Services on OpenShift Jira project, where you can track the progress of your feedback.

1. Ensure that you are logged in to Jira. If you do not have a Jira account, create an account to submit feedback.
2. Click the following link to open a the **Create Issue** page: Create Issue
3. Complete the **Summary** and **Description** fields. In the **Description** field, include the documentation URL, chapter or section number, and a detailed description of the issue. Do not modify any other fields in the form.
4. Click **Create**.

CHAPTER 1. CONFIGURING STORAGE IN RED HAT OPENSTACK SERVICES ON OPENSHIFT (RHOSO)

When you deploy Red Hat OpenStack Services on OpenShift (RHOSO), you can configure your deployment to use Red Hat Ceph Storage as the back end for storage. You can integrate an external Red Hat Ceph Storage cluster with the Compute service (nova) and a combination of one or more RHOSO storage services, and you can create a Hyperconverged Infrastructure (HCI) environment.

RHOSO provides the following storage services:

- Block Storage service (cinder)
- Image service (glance)
- Object Storage service (swift)
- Shared File Systems service (manila)

For information about integrating Red Hat Ceph Storage with your RHOSO deployment, see Integrating Red Hat Ceph Storage. For information about creating a HCI environment, see Configuring a Hyperconverged Infrastructure environment.

RHOSO supports the following storage solutions:

- Configure the Block Storage service with a Ceph RBD back end, iSCSI, FC, or NVMe-TCP storage protocols, or a generic NFS back end. For information about third-party back ends for the Block Storage service, see OSP18 Cinder Alternative Storage .
- Configure the Image service with a Ceph RBD, Block Storage, Object Storage, or NFS back end.
- Configure the Object Storage service to use PersistentVolumes (PVs) on OpenShift nodes or disks on external data plane nodes.
- Configure the Shared File Systems service with a native CephFS, Ceph-NFS, or alternative back end, such as NetApp or Pure Storage.

RHOSO recognizes two types of storage - ephemeral and persistent:

- Ephemeral storage is associated with a specific Compute instance. When that instance is terminated, so is the associated ephemeral storage. This type of storage is useful for runtime requirements, such as storing the operating system of an instance.
- Persistent storage is designed to survive (persist) independent of any running instance. This storage is used for any data that needs to be reused, either by different instances or beyond the life of a specific instance.

For information about planning the storage solution and related requirements for your RHOSO deployment, for example, networking and security, see the **Planning storage** chapter in *Planning your deployment*.

CHAPTER 2. INTEGRATING RED HAT CEPH STORAGE

You can configure Red Hat OpenStack Services on OpenShift (RHOSO) to integrate with an external Red Hat Ceph Storage cluster. This configuration connects the following services to a Red Hat Ceph Storage cluster:

- Block Storage service (cinder)
- Image service (glance)
- Object Storage service (swift)
- Compute service (nova)
- Shared File Systems service (manila)

If you want to deploy a Red Hat Ceph Storage Hyper Converged Infrastructure (HCI), see Configuring a Hyperconverged Infrastructure environment.

To configure Red Hat Ceph Storage as the back end for RHOSO storage, complete the following tasks:

1. Create the Red Hat Ceph Storage pools on the Red Hat Ceph Storage cluster.
2. Create a Red Hat Ceph Storage secret on the Red Hat Ceph Storage cluster to provide RHOSO services access to the Red Hat Ceph Storage cluster.
3. Obtain the Ceph File System Identifier.
4. Configure the OpenStackControlPlane CR to use the Red Hat Ceph Storage cluster as the back end.
5. Configure the OpenStackDataPlane CR to use the Red Hat Ceph Storage cluster as the back end.

Prerequisites

- Access to a Red Hat Ceph Storage cluster. If you intend to host Red Hat Ceph Storage on data plane nodes (HCI), then complete Configuring a Hyperconverged Infrastructure environment first.
- The RHOSO control plane is installed on an operational Red Hat OpenShift Platform cluster.

2.1. CREATING RED HAT CEPH STORAGE POOLS

Create pools on the Red Hat Ceph Storage cluster server for each RHOSO service that uses the cluster.

Procedure

1. Create pools for the Compute service (vms), the Block Storage service (volumes), and the Image service (images):

```
$ for P in vms volumes images; do
  cephadm shell -- ceph osd pool create $P;
  cephadm shell -- ceph osd pool application enable $P rbd;
done
```

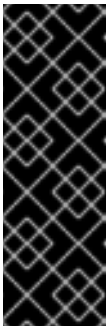
2. Optional: Create the **cephfs** volume if the Shared File Systems service (manila) is enabled in the control plane. This automatically enables the CephFS Metadata service (MDS) and creates the necessary data and metadata pools on the Ceph cluster:

```
$ cephadm shell -- ceph fs volume create cephfs
```

3. Optional: Deploy an NFS service on the Red Hat Ceph Storage cluster to use CephFS with NFS:

```
$ cephadm shell -- ceph nfs cluster create cephfs \
--ingress --virtual-ip=<vip> \
--ingress-mode=haproxy-protocol
```

- Replace **<vip>** with the IP address assigned to the NFS service. The NFS service should be isolated on a network that can be shared with all Red Hat OpenStack users. See NFS cluster and export management, for more information about customizing the NFS service.



IMPORTANT

When you deploy an NFS service for the Shared File Systems service, do not select a custom port to expose NFS. Only the default NFS port of 2049 is supported. You must enable the Red Hat Ceph Storage **ingress** service and set the **ingress-mode** to **haproxy-protocol**. Otherwise, you cannot use IP-based access rules with the Shared File Systems service. For security in production environments, Red Hat does not recommend providing access to **0.0.0.0/0** on shares to mount them on client machines.

4. Create a cephx key for RHOSO to use to access pools:

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

IMPORTANT

If the Shared File Systems service is enabled in the control plane, replace **osd** caps with the following:

osd 'allow class-read object_prefix rbd_children, allow rwx pool=vms, allow rwx pool=volumes, allow rwx pool=images, allow rwx pool=cephfs.cephfs.data'

5. Export the cephx key:

```
$ cephadm shell -- ceph auth get client.openstack > /etc/ceph/ceph.client.openstack.keyring
```

6. Export the configuration file:

```
$ cephadm shell -- ceph config generate-minimal-conf > /etc/ceph/ceph.conf
```

2.2. CREATING A RED HAT CEPH STORAGE SECRET

Create a secret so that services can access the Red Hat Ceph Storage cluster.

Procedure

1. Transfer the **cephx** key and configuration file created in the Creating Red Hat Ceph Storage pools procedure to a host that can create resources in the **openstack** namespace.
2. Base64 encode these files and store them in **KEY** and **CONF** environment variables:

```
$ KEY=$(cat /etc/ceph/ceph.client.openstack.keyring | base64 -w 0)
$ CONF=$(cat /etc/ceph/ceph.conf | base64 -w 0)
```

3. Create a YAML file to create the **Secret** resource.
4. Using the environment variables, add the **Secret** configuration to the YAML file:

```
apiVersion: v1
data:
  ceph.client.openstack.keyring: $KEY
  ceph.conf: $CONF
kind: Secret
metadata:
  name: ceph-conf-files
  namespace: openstack
type: Opaque
```

5. Save the YAML file.
6. Create the **Secret** resource:

```
$ oc create -f <secret_configuration_file>
```

- Replace **<secret_configuration_file>** with the name of the YAML file you created.

NOTE

The examples in this section use **openstack** as the name of the Red Hat Ceph Storage user. The file name in the **Secret** resource must match this user name.

For example, if the file name used is **/etc/ceph/ceph.client.openstack2.keyring**, then the secret data line should be **ceph.client.openstack2.keyring: $KEY**.

2.3. OBTAINING THE RED HAT CEPH STORAGE FILE SYSTEM IDENTIFIER

The Red Hat Ceph Storage File System Identifier (FSID) is a unique identifier for the cluster. The FSID is used in configuration and verification of cluster interoperability with RHOSO.

Procedure

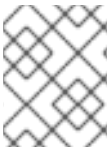
- Extract the FSID from the Red Hat Ceph Storage secret:

```
$ FSID=$(oc get secret ceph-conf-files -o json | jq -r '.data."ceph.conf"' | base64 -d | grep fsid
| sed -e 's/fsid = //')
```

2.4. CONFIGURING THE CONTROL PLANE TO USE THE RED HAT CEPH STORAGE CLUSTER

You must configure the **OpenStackControlPlane** CR to use the Red Hat Ceph Storage cluster. Configuration includes the following tasks:

1. Confirming the Red Hat Ceph Storage cluster and the associated services have the correct network configuration.
2. Configuring the control plane to use the Red Hat Ceph Storage secret.
3. Configuring the Image service (glance) to use the Red Hat Ceph Storage cluster.
4. Configuring the Block Storage service (cinder) to use the Red Hat Ceph Storage cluster.
5. Optional: Configuring the Shared File Systems service (manila) to use native CephFS or CephFS-NFS with the Red Hat Ceph Storage cluster.

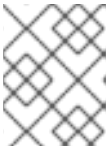


NOTE

This example does not include configuring Block Storage backup service (**cinder-backup**) with Red Hat Ceph Storage.

Procedure

1. Check the storage interface defined in your **NodeNetworkConfigurationPolicy** (**nncp**) custom resource to confirm that it has the same network configuration as the **public_network** of the Red Hat Ceph Storage cluster. This is required to enable access to the Red Hat Ceph Storage cluster through the **Storage** network. The **Storage** network should have the same network configuration as the **public_network** of the Red Hat Ceph Storage cluster.



NOTE

It is not necessary for RHOSO to access the **cluster_network** of the Red Hat Ceph Storage cluster.

2. Check the **networkAttachments** for the default Image service instance in the **OpenStackControlPlane** CR to confirm that the default Image service is configured to access the **Storage** network:

```
glance:
    enabled: true
    template:
      databaseInstance: openstack
      storageClass: ""
      storageRequest: 10G
      glanceAPIs:
        default
          replicas: 3
          override:
            service:
```

```
      internal:
        metadata:
          annotations:
            metallb.universe.tf/address-pool: internalapi
            metallb.universe.tf/allow-shared-ip: internalapi
            metallb.universe.tf/loadBalancerIPs: 172.17.0.80
        spec:
          type: LoadBalancer
    networkAttachments:
    - storage
```

3. Confirm the Block Storage service is configured to access the **Storage** network through MetalLB.

4. Optional: Confirm the Shared File Systems service is configured to access the **Storage** network through ManilaShare.

5. Confirm the Compute service (nova) is configured to access the **Storage** network.

6. Confirm the Red Hat Ceph Storage configuration file, **/etc/ceph/ceph.conf**, contains the IP addresses of the Red Hat Ceph Storage cluster monitors. These IP addresses must be within the **Storage** network IP address range.

7. Open your **openstack_control_plane.yaml** file to edit the **OpenStackControlPlane** CR.

8. Add the **extraMounts** parameter to define the services that require access to the Red Hat Ceph Storage secret.
   The following is an example of using the **extraMounts** parameter for this purpose. Only include **ManilaShare** in the propagation list if you are using the Shared File Systems service (manila):

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  extraMounts:
    - name: v1
      region: r1
      extraVol:
        - propagation:
          - CinderVolume
          - GlanceAPI
          - ManilaShare
          extraVolType: Ceph
          volumes:
          - name: ceph
            projected:
              sources:
              - secret:
                  name: <ceph-conf-files>
          mounts:
          - name: ceph
            mountPath: "/etc/ceph"
            readOnly: true
```

   - Replace **<ceph-conf-files>** with the name of your Secret CR created in Creating a Red Hat Ceph Storage secret.

9. Add the **customServiceConfig** parameter to the **glance** template to configure the Image service to use the Red Hat Ceph Storage cluster:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
    ...
 glance:
   enabled: true
   template:
     databaseInstance: openstack
     databaseUser: glance
     customServiceConfig: |
       [DEFAULT]
       enabled_backends = default_backend:rbd
       enabled_import_methods=[web-download,glance-direct]
       [glance_store]
       default_backend = default_backend
       [default_backend]
       rbd_store_ceph_conf = /etc/ceph/ceph.conf
       store_description = "RBD backend"
       rbd_store_pool = images
       rbd_store_user = openstack
   glanceAPIs:
     default:
       preserveJobs: false
       replicas: 1
   secret: osp-secret
   storageClass: ""
   storageRequest: 10G
 extraMounts:
   - name: v1
     region: r1
     extraVol:
       - propagation:
         - Glance
         extraVolType: Ceph
         volumes:
           - name: ceph
             projected:
               sources:
               - secret:
                   name: ceph-conf-files
         mounts:
           - name: ceph
             mountPath: "/etc/ceph"
             readOnly: true
```

10. Add the **customServiceConfig** parameter to the **cinder** template to configure the Block Storage service to use the Red Hat Ceph Storage cluster:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  extraMounts:
    ...
```

```
cinder:
  template:
    cinderVolumes:
      ceph:
        customServiceConfig: |
          [DEFAULT]
          enabled_backends=ceph
          [ceph]
          volume_backend_name=ceph
          volume_driver=cinder.volume.drivers.rbd.RBDDriver
          rbd_ceph_conf=/etc/ceph/ceph.conf
          rbd_user=openstack
          rbd_pool=volumes
          rbd_flatten_volume_from_snapshot=False
          rbd_secret_uuid=$FSID (1)
```

(1) Replace with the actual FSID. The FSID itself does not need to be considered secret. For more information, see Obtaining the Red Hat Ceph Storage FSID .

11. Optional: Add the **customServiceConfig** parameter to the **manila** template to configure the Shared File Systems service to use native CephFS or CephFS-NFS with the Red Hat Ceph Storage cluster. For more information, see Configuring the Shared File Systems service (manila):
The following example exposes native CephFS:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  extraMounts:
    ...
    manila:
      template:
  manilaAPI:
          customServiceConfig: |
            [DEFAULT]
            enabled_share_protocols=cephfs
        manilaShares:
          share1:
            customServiceConfig: |
              [DEFAULT]
              enabled_share_backends=cephfs
              [cephfs]
              driver_handles_share_servers=False
              share_backend_name=cephfs
              share_driver=manila.share.drivers.cephfs.driver.CephFSDriver
              cephfs_conf_path=/etc/ceph/ceph.conf
              cephfs_auth_id=openstack
              cephfs_cluster_name=ceph
              cephfs_volume_mode=0755
              cephfs_protocol_helper_type=CEPHFS
```

The following example exposes CephFS with NFS:

```
apiVersion: core.openstack.org/v1beta1
```

```
kind: OpenStackControlPlane
spec:
  extraMounts:
    ...
    manila:
      template:
        manilaAPI:
          customServiceConfig: |
            [DEFAULT]
            enabled_share_protocols=nfs
        manilaShares:
          share1:
            customServiceConfig: |
              [DEFAULT]
              enabled_share_backends=cephfsnfs
              [cephfsnfs]
              driver_handles_share_servers=False
              share_backend_name=cephfsnfs
              share_driver=manila.share.drivers.cephfs.driver.CephFSDriver
              cephfs_conf_path=/etc/ceph/ceph.conf
              cephfs_auth_id=openstack
              cephfs_cluster_name=ceph
              cephfs_volume_mode=0755
              cephfs_protocol_helper_type=NFS
              cephfs_nfs_cluster_id=cephfs
```

12. Apply the updates to the **OpenStackControlPlane** CR:

```
$ oc apply -f openstack_control_plane.yaml
```

2.5. CONFIGURING THE DATA PLANE TO USE THE RED HAT CEPH STORAGE CLUSTER

Configure the data plane to use the Red Hat Ceph Storage cluster.

Procedure

1. Create a **ConfigMap** with additional content for the Compute service (nova) configuration file **/etc/nova/nova.conf.d/** inside the **nova_compute** container. This additional content directs the Compute service to use Red Hat Ceph Storage RBD.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ceph-nova
data:
  03-ceph-nova.conf: | 1
    [libvirt]
    images_type=rbd
    images_rbd_pool=vms
    images_rbd_ceph_conf=/etc/ceph/ceph.conf
    images_rbd_glance_store_name=default_backend
    images_rbd_glance_copy_poll_interval=15
```

```
images_rbd_glance_copy_timeout=600
rbd_user=openstack
rbd_secret_uuid=$FSID (2)
```

(1) This file name must follow the naming convention of **##-<name>-nova.conf**. Files are evaluated by the Compute service alphabetically. A filename that starts with **01** will be evaluated by the Compute service before a filename that starts with **02**.

(2) The **$FSID** value should contain the actual **FSID** as described in the Obtaining the Ceph FSID section. The **FSID** itself does not need to be considered secret.

2. Create a custom version of the default **nova** operator to use the new **ConfigMap**, which in this case is called **ceph-nova**.

```
apiVersion: dataplane.openstack.org/v1beta1
kind: OpenStackDataPlaneService
metadata:
  name: nova-custom-ceph (1)
spec:
  label: dataplane-deployment-nova-custom-ceph
  configMaps:
    - ceph-nova
  secrets:
    - nova-cell1-compute-config
  playbook: osp.edpm.nova
```

(1) The custom service is named **nova-custom-ceph**. It cannot be named **nova** because **nova** is an unchangeable default service. Any custom service that has the same name as a default service name will be overwritten during reconciliation.

3. Apply the **ConfigMap** and custom service changes:

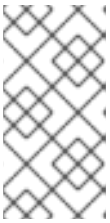
```
$ oc create -f ceph-nova.yaml
```

4. Update the **OpenStackDataPlaneNodeSet** services list to replace the **nova** service with the new custom service (in this case called **nova-custom-ceph**), add the **ceph-client** service, and use the **extraMounts** parameter to define access to the Ceph Storage secret.

```
apiVersion: dataplane.openstack.org/v1beta1
kind: OpenStackDataPlaneNodeSet
spec:
  ...
  roles:
    edpm-compute:
      ...
      services:
        - configure-network
        - validate-network
        - install-os
        - configure-os
        - run-os
        - ceph-client
        - ovn
```

```
      - libvirt
      - nova-custom-ceph
      - telemetry

  nodeTemplate:
    extraMounts:
    - extraVolType: Ceph
      volumes:
      - name: ceph
        secret:
          secretName: ceph-conf-files
      mounts:
      - name: ceph
        mountPath: "/etc/ceph"
        readOnly: true
```



NOTE

The **ceph-client** service must be added before the **libvirt** and **nova-custom-ceph** services. The **ceph-client** service configures EDPM nodes as clients of a Red Hat Ceph Storage server by distributing the Red Hat Ceph Storage client files.

5. Save the changes to the services list.
6. Create an **OpenStackDataPlaneDeployment** CR:

```
$ oc create -f <dataplanedeployment_cr_file>
```

- Replace **<dataplanedeployment_cr_file>** with the name of your file.

NOTE

An example of an **OpenStackDataPlaneDeployment** CR file is available here: link:https://github.com/openstack-k8s-operators/dataplane-operator/blob/main/config/samples/dataplane_v1beta1_openstackdataplanede

Result

When the **nova-custom-ceph** service Ansible job runs, the job copies overrides from the **ConfigMaps** to the Compute service hosts. It will also use **virsh secret-*** commands so the **libvirt** service retrieves the **cephx** secret by **FSID**.

- Run the following command on an EDPM node after the job completes to confirm the job results:

```
$ podman exec libvirt_virtsecretd virsh secret-get-value $FSID
```

2.6. CONFIGURING THE OBJECT STORAGE SERVICE (SWIFT) WITH AN EXTERNAL CEPH OBJECT GATEWAY BACK END

You can configure an external Ceph Object Gateway (RGW) to act as an Object Storage service (swift) back end, by completing the following high-level tasks:

1. Configure the RGW to verify users and their roles in the Identity service (keystone) to authenticate with the external RGW service.

2. Deploy and configure a RGW service to handle object storage requests.

You use the **openstack** client tool to configure the Object Storage service.

2.6.1. Configuring RGW authentication

You must configure RGW to verify users and their roles in the Identity service (keystone) to authenticate with the external RGW service.

Prerequisites

- You have deployed an operational OpenStack control plane.

Procedure

1. Create the Object Storage service on the control plane:

```
$ openstack service create --name swift --description "OpenStack Object Storage" object-store
```

2. Create a user called **swift**:

```
$ openstack user create --project service --password <swift_password> swift
```

   - Replace **<swift_password>** with the password to assign to the **swift** user.

3. Create roles for the **swift** user:

```
$ openstack role create swiftoperator
$ openstack role create ResellerAdmin
```

4. Add the **swift** user to system roles:

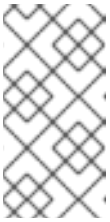
```
$ openstack role add --user swift --project service member
$ openstack role add --user swift --project service admin
```

5. Export the RGW endpoint IP addresses to variables and create control plane endpoints:

```
$ export RGW_ENDPOINT_STORAGE=<rgw_endpoint_ip_address_storage>
$ export RGW_ENDPOINT_EXTERNAL=<rgw_endpoint_ip_address_external>
$ openstack endpoint create --region regionOne object-store public http://$RGW_ENDPOINT_EXTERNAL:8080/swift/v1/AUTH_%\(tenant_id\)s;
$ openstack endpoint create --region regionOne object-store internal http://$RGW_ENDPOINT_STORAGE:8080/swift/v1/AUTH_%\(tenant_id\)s;
```

   - Replace **<rgw_endpoint_ip_address_storage>** with the IP address of the RGW endpoint on the storage network. This is how internal services will access RGW.

   - Replace **<rgw_endpoint_ip_address_external>** with the IP address of the RGW endpoint on the external network. This is how cloud users will write objects to RGW.



NOTE

Both endpoint IP addresses are the endpoints that represent the Virtual IP addresses, owned by **haproxy** and **keepalived**, used to reach the RGW backends that will be deployed in the Red Hat Ceph Storage cluster in the procedure Configuring and Deploying the RGW service .

6. Add the **swiftoperator** role to the control plane **admin** group:

```
$ openstack role add --project admin --user admin swiftoperator
```

2.6.2. Configuring and deploying the RGW service

Configure and deploy a RGW service to handle object storage requests.

Procedure

1. Log in to a Red Hat Ceph Storage Controller node.
2. Create a file called **/tmp/rgw_spec.yaml** and add the RGW deployment parameters:

```
service_type: rgw
service_id: rgw
service_name: rgw.rgw
placement:
  hosts:
    - <host_1>
    - <host_2>
    ...
    - <host_n>
networks:
- <storage_network>
spec:
  rgw_frontend_port: 8082
  rgw_realm: default
  rgw_zone: default
---
service_type: ingress
service_id: rgw.default
service_name: ingress.rgw.default
placement:
  count: 1
spec:
  backend_service: rgw.rgw
  frontend_port: 8080
  monitor_port: 8999
  virtual_ips_list:
  - <storage_network_vip>
  - <external_network_vip>
  virtual_interface_networks:
  - <storage_network>
```

- Replace **<host_1>**, **<host_2>**, ..., **<host_n>** with the name of the Ceph nodes where the RGW instances are deployed.

- Replace **<storage_network>** with the network range used to resolve the interfaces where **radosgw** processes are bound.
- Replace **<storage_network_vip>** with the virtual IP (VIP) used as the **haproxy** front end. This is the same address configured as the Object Storage service endpoint (**$RGW_ENDPOINT**) in the Configuring RGW authentication procedure.
- Optional: Replace **<external_network_vip>** with an additional VIP on an external network to use as the **haproxy** front end. This address is used to connect to RGW from an external network.

3. Save the file.
4. Enter the cephadm shell and mount the **rgw_spec.yaml** file.

```
$ cephadm shell -m /tmp/rgw_spec.yaml
```

5. Add RGW related configuration to the cluster:

```
$ ceph config set global rgw_keystone_url "https://<keystone_endpoint>"
$ ceph config set global rgw_keystone_verify_ssl false
$ ceph config set global rgw_keystone_api_version 3
$ ceph config set global rgw_keystone_accepted_roles "member, Member, admin"
$ ceph config set global rgw_keystone_accepted_admin_roles "ResellerAdmin, swiftoperator"
$ ceph config set global rgw_keystone_admin_domain default
$ ceph config set global rgw_keystone_admin_project service
$ ceph config set global rgw_keystone_admin_user swift
$ ceph config set global rgw_keystone_admin_password "$SWIFT_PASSWORD"
$ ceph config set global rgw_keystone_implicit_tenants true
$ ceph config set global rgw_s3_auth_use_keystone true
$ ceph config set global rgw_swift_versioning_enabled true
$ ceph config set global rgw_swift_enforce_content_length true
$ ceph config set global rgw_swift_account_in_url true
$ ceph config set global rgw_trust_forwarded_https true
$ ceph config set global rgw_max_attr_name_len 128
$ ceph config set global rgw_max_attrs_num_in_req 90
$ ceph config set global rgw_max_attr_size 1024
```

- Replace **<keystone_endpoint>** with the Identity service internal endpoint. The EDPM nodes are able to resolve the internal endpoint but not the public one. Do not omit the URIScheme from the URL, it must be either **http://** or **https://**.
- Replace **<swift_password>** with the password assigned to the swift user in the previous step.

6. Deploy the RGW configuration using the Orchestrator:

```
$ ceph orch apply -i /mnt/rgw_spec.yaml
```

CHAPTER 3. CONFIGURING A HYPERCONVERGED INFRASTRUCTURE ENVIRONMENT

This section describes how to deploy a Hyperconverged Infrastructure (HCI) environment. A HCI environment contains data plane nodes that host both Ceph Storage and the Compute service.

Create an HCI environment, by completing the following high-level tasks:

1. Configuring the data plane node networking.
2. Installing Red Hat Ceph Storage on the data plane nodes.
3. Configuring Red Hat OpenStack Services on OpenShift (RHOSO) to use the Red Hat Ceph Storage cluster.

3.1. DATA PLANE NODE SERVICES LIST

Create an **OpenStackDataPlaneNodeSet** CR to configure data plane nodes. The **dataplane-operator** reconciles the **OpenStackDataPlaneNodeSet** CR when an **OpenStackDataPlaneDeployment** CR is created.

These CRs have a service list similar to the following example:

```
apiVersion: dataplane.openstack.org/v1beta1
kind: OpenStackDataPlaneNodeSet
spec:
  ...
  services:
    - configure-network
    - validate-network
    - install-os
    - configure-os
    - run-os
    - ovn
    - libvirt
    - nova
```

Only the services in the services list are configured.

Red Hat Ceph Storage must be deployed on the data plane node after the **Storage** network and NTP are configured but before the Compute service is configured. This means you must edit the services list and make other changes to the CR. Throughout this section, you edit the services list to complete the configuration of the HCI environment.

3.2. CONFIGURING THE DATA PLANE NODE NETWORKS

You must configure the data plane node networks to accommodate the Red Hat Ceph Storage networking requirements.

Prerequisites

- Control plane deployment is complete but has not yet been modified to use Ceph Storage.
- The data plane nodes have been provisioned with an operating system.

- The data plane nodes are accessible through an SSH key that Ansible can use.
- The data plane nodes have disks available to be used as Ceph OSDs.
- There are a minimum of three available data plane nodes. Ceph Storage clusters must have a minimum of three nodes to ensure redundancy.

Procedure

1. Create an **OpenStackDataPlaneNodeSet** CRD file to represent the data plane nodes.

NOTE

Do not create the CR in Red Hat OpenShift yet.

2. Add the **ceph-hci-pre** service to the list before the **configure-os** service and remove all other service listings after **run-os**.
The following is an example of the edited list:

```
apiVersion: dataplane.openstack.org/v1beta1
kind: OpenStackDataPlaneNodeSet
spec:
  ...
  services:
    - download-cache
    - bootstrap
    - configure-network
    - validate-network
    - install-os
    - ceph-hci-pre
    - configure-os
    - ssh-known-hosts
    - run-os
    - reboot-os
```

NOTE

Note the services that you remove from the list. You add them back to the list later.

3. (Optional) The **ceph-hci-pre** service prepares EDPM nodes to host Red Hat Ceph Storage services after network configuration using the **edpm_ceph_hci_pre edpm-ansible** role. By default, the **edpm_ceph_hci_pre_enabled_services** parameter of this role only contains RBD, RGW, and NFS services. If other services, such as the Dashboard, are deployed with HCI nodes; they must be added to the **edpm_ceph_hci_pre_enabled_services** parameter list. For more information about this role, see edpm_ceph_hci_pre role.

4. Configure the Red Hat Ceph Storage **cluster_network** for storage management traffic between OSDs. Modify the CR to set **edpm-ansible** variables so that the **edpm_network_config** role configures a storage management network which Ceph uses as the **cluster_network**.
The following example has 3 nodes. It assumes the storage management network range is **172.20.0.0/24** and that it is on **VLAN23**. The bolded lines are additions for the **cluster_network**:

```
apiVersion: dataplane.openstack.org/v1beta1
kind: OpenStackDataPlaneNodeSet
metadata:
  name: openstack-edpm
  namespace: openstack
spec:
  env:
  - name: ANSIBLE_FORCE_COLOR
    value: "True"
  networkAttachments:
  - ctlplane
  nodeTemplate:
    ansible:
      ansiblePort: 22
      ansibleUser: cloud-admin
      ansibleVars:
        edpm_ceph_hci_pre_enabled_services:
        - ceph_mon
        - ceph_mgr
        - ceph_osd
        - ceph_rgw
        - ceph_nfs
        - ceph_rgw_frontend
        - ceph_nfs_frontend
        edpm_fips_mode: check
        edpm_iscsid_image: {{ registry_url }}/openstack-iscsid:{{ image_tag }}
        edpm_logrotate_crond_image: {{ registry_url }}/openstack-cron:{{ image_tag }}
        edpm_network_config_hide_sensitive_logs: false
        edpm_network_config_os_net_config_mappings:
          edpm-compute-0:
            nic1: 52:54:00:1e:af:6b
            nic2: 52:54:00:d9:cb:f4
          edpm-compute-1:
            nic1: 52:54:00:f2:bc:af
            nic2: 52:54:00:f1:c7:dd
          edpm-compute-2:
            nic1: 52:54:00:dd:33:14
            nic2: 52:54:00:50:fb:c3
        edpm_network_config_template: |
          ---
          {% set mtu_list = [ctlplane_mtu] %}
          {% for network in nodeset_networks %}
          {{ mtu_list.append(lookup(vars, networks_lower[network] ~ _mtu)) }}
          {%- endfor %}
          {% set min_viable_mtu = mtu_list | max %}
          network_config:
          - type: ovs_bridge
            name: {{ neutron_physical_bridge_name }}
            mtu: {{ min_viable_mtu }}
            use_dhcp: false
            dns_servers: {{ ctlplane_dns_nameservers }}
            domain: {{ dns_search_domains }}
            addresses:
            - ip_netmask: {{ ctlplane_ip }}/{{ ctlplane_cidr }}
            routes: {{ ctlplane_host_routes }}
            members:
```

```
          - type: interface
            name: nic2
            mtu: {{ min_viable_mtu }}
            # force the MAC address of the bridge to this interface
            primary: true
        {% for network in nodeset_networks %}
          - type: vlan
            mtu: {{ lookup(vars, networks_lower[network] ~ _mtu) }}
            vlan_id: {{ lookup(vars, networks_lower[network] ~ _vlan_id) }}
            addresses:
            - ip_netmask:
                {{ lookup(vars, networks_lower[network] ~ _ip) }}/{{ lookup(vars,
networks_lower[network] ~ _cidr) }}
            routes: {{ lookup(vars, networks_lower[network] ~ _host_routes) }}
        {% endfor %}
      edpm_neutron_metadata_agent_image: {{ registry_url }}/openstack-neutron-metadata-
agent-ovn:{{ image_tag }}
      edpm_nodes_validation_validate_controllers_icmp: false
      edpm_nodes_validation_validate_gateway_icmp: false
      edpm_selinux_mode: enforcing
      edpm_sshd_allowed_ranges:
      - 192.168.122.0/24
      - 192.168.111.0/24
      edpm_sshd_configure_firewall: true
      enable_debug: false
      gather_facts: false
      image_tag: current-podified
      neutron_physical_bridge_name: br-ex
      neutron_public_interface_name: eth0
      service_net_map:
        nova_api_network: internalapi
        nova_libvirt_network: internalapi
      storage_mgmt_cidr: "24"
      storage_mgmt_host_routes: []
      storage_mgmt_mtu: 9000
      storage_mgmt_vlan_id: 23
      storage_mtu: 9000
      timesync_ntp_servers:
      - hostname: pool.ntp.org
  ansibleSSHPrivateKeySecret: dataplane-ansible-ssh-private-key-secret
  managementNetwork: ctlplane
  networks:
  - defaultRoute: true
    name: ctlplane
    subnetName: subnet1
  - name: internalapi
    subnetName: subnet1
  - name: storage
    subnetName: subnet1
  - name: tenant
    subnetName: subnet1
  nodes:
    edpm-compute-0:
      ansible:
        host: 192.168.122.100
      hostName: compute-0
```

```
    networks:
    - defaultRoute: true
      fixedIP: 192.168.122.100
      name: ctlplane
      subnetName: subnet1
    - name: internalapi
      subnetName: subnet1
    - name: storage
      subnetName: subnet1
    - name: storagemgmt
      subnetName: subnet1
    - name: tenant
      subnetName: subnet1
  edpm-compute-1:
    ansible:
      host: 192.168.122.101
    hostName: compute-1
    networks:
    - defaultRoute: true
      fixedIP: 192.168.122.101
      name: ctlplane
      subnetName: subnet1
    - name: internalapi
      subnetName: subnet1
    - name: storage
      subnetName: subnet1
    - name: storagemgmt
      subnetName: subnet1
    - name: tenant
      subnetName: subnet1
  edpm-compute-2:
    ansible:
      host: 192.168.122.102
    hostName: compute-2
    networks:
    - defaultRoute: true
      fixedIP: 192.168.122.102
      name: ctlplane
      subnetName: subnet1
    - name: internalapi
      subnetName: subnet1
    - name: storage
      subnetName: subnet1
    - name: storagemgmt
      subnetName: subnet1
    - name: tenant
      subnetName: subnet1
preProvisioned: true
services:
- bootstrap
- configure-network
- validate-network
- install-os
- ceph-hci-pre
- configure-os
```

```
- ssh-known-hosts
- run-os
- reboot-os
```

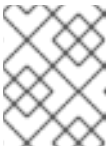
NOTE

It is not necessary to add the storage management network to the networkAttachments key.

5. Apply the CR:

```
$ oc apply -f <dataplane_cr_file>
```

- Replace **<dataplane_cr_file>** with the name of your file.



NOTE

Ansible does not configure or validate the networks until the **OpenStackDataPlaneDeployment** CRD is created.

6. Create an **OpenStackDataPlaneDeployment** CRD, as described in Deploying the data plane, which has the **OpenStackDataPlaneNodeSet** CRD file defined above to have Ansible configure the services on the data plane nodes.

7. To confirm the network is configured, complete the following steps:

   a. SSH into a data plane node.

   b. Use the **ip a** command to display configured networks.

   c. Confirm the storage networks are in the list of configured networks.

3.2.1. Red Hat Ceph Storage MTU settings

The example in this procedure changes the MTU of the storage and storage_mgmt networks from 1500 to 9000. An MTU of 9000 is known as a jumbo frame. Even though it is not mandatory to increase the MTU, jumbo frames are used for improved storage performance. If jumbo frames are used, all network switch ports in the data path must be configured to support jumbo frames. MTU changes must also be made for services using the Storage network running on OpenShift.

To change the MTU for the OpenShift services connecting to the data plane nodes, update the Node Network Configuration Policy (NNCP) for the base interface and the VLAN interface. It is not necessary to update the Network Attachment Definition (NAD) if the main NAD interface already has the desired MTU. If the MTU of the underlying interface is set to 9000, and it is not specified for the VLAN interface on top of it, then it will default to the value from the underlying interface.

If the MTU values are not consistent, issues can occur on the application layer that can cause the Red Hat Ceph Storage cluster to not reach quorum or not support authentication using the CephX protocol. If the MTU is changed and you observe these types of problems, verify all hosts that use the network using jumbo frames can communicate at the chosen MTU value with the ping command, for example:

```
$ ping -M do -s 8972 172.20.0.100
```

3.3. CONFIGURING AND DEPLOYING RED HAT CEPH STORAGE ON DATA PLANE NODES

Use the **cephadm** utility to configure and deploy Red Hat Ceph Storage for an HCI environment.

3.3.1. The cephadm utility

Use the **cephadm** utility to configure and deploy Red Hat Ceph Storage on the data plane nodes. The **cephadm** package must be deployed on at least one data plane node before proceeding; **edpm-ansible** does not deploy Red Hat Ceph Storage.

For additional information and procedures for deploying Red Hat Ceph Storage, see Red Hat Ceph Storage installation in the *Red Hat Ceph Storage Installation Guide* .

3.3.2. Configuring and deploying Red Hat Ceph Storage

Configure and deploy Red Hat Ceph Storage by editing the configuration file and using the **cephadm** utility.

Procedure

1. Edit the Red Hat Ceph Storage configuration file.
2. Add the **Storage** and **Storage Management** network ranges. Red Hat Ceph Storage uses the **Storage** network as the Red Hat Ceph Storage **public_network** and the **Storage Management** network as the **cluster_network**.
   The following example is for a configuration file entry where the **Storage** network range is **172.18.0.0/24** and the **Storage Management** network range is **172.20.0.0/24**:

   ```
   [global]
   public_network = 172.18.0.0/24
   cluster_network = 172.20.0.0/24
   ```

3. Add collocation boundaries between the Compute service and Ceph OSD services. Boundaries should be set between collocated Compute service and Ceph OSD services to reduce CPU and memory contention.
   The following is an example for a Ceph configuration file entry with these boundaries set:

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

   In this example, the **osd_memory_target_autotune** parameter is set to **true** so that the OSD daemons adjust memory consumption based on the **osd_memory_target** option. The **autotune_memory_target_ratio** defaults to 0.7. This means 70 percent of the total RAM in the system is the starting point from which any memory consumed by non-autotuned Ceph daemons is subtracted. The remaining memory is divided between the OSDs; assuming all OSDs have **osd_memory_target_autotune** set to true. For HCI deployments, you can set **mgr/cephadm/autotune_memory_target_ratio** to 0.2 so that more memory is available for the Compute service.

For additional information about service collocation, see Collocating services in a HCI environment for NUMA nodes.

NOTE

If these values need to be adjusted after the deployment, use the **ceph config set osd <key> <value>** command.

4. Deploy Ceph Storage with the edited configuration file on a data plane node:
$ cephadm bootstrap --config <config_file> --mon-ip <data_plane_node_ip>

   - Replace **<config_file>** with the name of your Ceph configuration file.
   - Replace **<data_plane_node_ip>** with the **Storage** network IP address of the data plane node on which Red Hat Ceph Storage will be installed.

5. After the Red Hat Ceph Storage cluster is bootstrapped on the first EDPM node, see Red Hat Ceph Storage installation in the *Red Hat Ceph Storage Installation Guide* to add the other EDPM nodes to the Ceph cluster.

3.3.2.1. Collocating services in a HCI environment for NUMA nodes

A two-NUMA node system can host a latency sensitive Compute service workload on one NUMA node and a Ceph OSD workload on the other NUMA node. To configure Ceph OSDs to use a specific NUMA node not being used by the the Compute service, use either of the following Ceph OSD configurations:

- **osd_numa_node** sets affinity to a NUMA node (-1 for none).
- **osd_numa_auto_affinity** automatically sets affinity to the NUMA node where storage and network match.

If there are network interfaces on both NUMA nodes and the disk controllers are on NUMA node 0, do the following:

1. Use a network interface on NUMA node 0 for the storage network
2. Host the Ceph OSD workload on NUMA node 0.
3. Host the Compute service workload on NUMA node 1 and have it use the network interfaces on NUMA node 1.

Set **osd_numa_auto_affinity** to true, as in the initial Ceph configuration file. Alternatively, set the **osd_numa_node** directly to 0 and clear the **osd_numa_auto_affinity** parameter so that it defaults to **false**.

When a hyperconverged cluster backfills as a result of an OSD going offline, the backfill process can be slowed down. In exchange for a slower recovery, the backfill activity has less of an impact on the collocated Compute service (nova) workload. Red Hat Ceph Storage has the following defaults to control the rate of backfill activity.

- **osd_recovery_op_priority = 3**
- **osd_max_backfills = 1**
- **osd_recovery_max_active_hdd = 3**

- **osd_recovery_max_active_ssd = 10**

3.3.3. Confirming Red Hat Ceph Storage deployment

Confirm Red Hat Ceph Storage is deployed before proceeding.

Procedure

1. Connect to a data plane node by using SSH.
2. View the status of the Red Hat Ceph Storage cluster:

```
$ cephadm shell -- ceph -s
```

3.3.4. Confirming Red Hat Ceph Storage tuning

Ensure that Red Hat Ceph Storage is properly tuned before proceeding.

Procedure

1. Connect to a data plane node by using SSH.
2. Verify overall Red Hat Ceph Storage tuning with the following commands:

```
$ ceph config dump | grep numa
$ ceph config dump | grep autotune
$ ceph config dump | get mgr
```

3. Verify the tuning of an OSD with the following commands:

```
$ ceph config get <osd_number> osd_memory_target
$ ceph config get <osd_number> osd_memory_target_autotune
$ ceph config get <osd_number> osd_numa_auto_affinity
```

   - Replace **<osd_number>** with the number of an OSD. For example, to refer to OSD 11, use **osd.11**.

4. Verify the default backfill values of an OSD with the following commands:

```
$ ceph config get <osd_number> osd_recovery_op_priority
$ ceph config get <osd_number> osd_max_backfills
$ ceph config get <osd_number> osd_recovery_max_active_hdd
$ ceph config get <osd_number> osd_recovery_max_active_ssd
```

   - Replace **<osd_number>** with the number of an OSD. For example, to refer to OSD 11, use **osd.11**.

3.4. CONFIGURING THE DATA PLANE TO USE THE COLLOCATED RED HAT CEPH STORAGE SERVER

Although the Red Hat Ceph Storage cluster is physically collocated with the Compute services on the data plane nodes, it is treated as logically separated. Red Hat Ceph Storage must be configured as the storage solution before the data plane nodes can use it.

Prerequisites

- Complete the procedures in the section Integrating Red Hat Ceph Storage .

Procedure

1. Edit the **OpenStackDataPlaneNodeSet** CR.
2. To define the **cephx** key and configuration file for the Compute service (nova), use the **extraMounts** parameter.
   The following is an example of using the **extraMounts** parameter for this purpose:

```
apiVersion: dataplane.openstack.org/v1beta1
kind: OpenStackDataPlane
spec:
  roles:
    edpm-compute:
      nodeTemplate:
        extraMounts:
        - extraVolType: Ceph
          volumes:
          - name: ceph
            secret:
              secretName: ceph-conf-files
          mounts:
          - name: ceph
            mountPath: "/etc/ceph"
            readOnly: true
```

3. Locate the **services** list in the CR.
4. Edit the **services** list to restore all of the services removed in Configuring the data plane node networks. Restoring the full **services** list allows the remaining jobs to be run that complete the configuration of the HCI environment.
   The following is an example of a full **services** list with the additional services in bold:

```
apiVersion: dataplane.openstack.org/v1beta1
kind: OpenStackDataPlaneNodeSet
spec:
  ...
  services:
    - bootstrap
    - configure-network
    - validate-network
    - install-os
    - ceph-hci-pre
    - configure-os
    - ssh-known-hosts
    - run-os
    - reboot-os
    - install-certs
    - ceph-client
    - ovn
```

```
    - neutron-metadata
    - libvirt
    - nova-custom-ceph
```

NOTE

In addition to restoring the default service list, the **ceph-client `service is added after the run-os service. The `ceph-client** service configures EDPM nodes as clients of a Red Hat Ceph Storage server. This service distributes the files necessary for the clients to connect to the Red Hat Ceph Storage server.

5. Create a **ConfigMap** to set the **reserved_host_memory_mb** parameter to a value appropriate for your configuration.
   The following is an example of a ConfigMap used for this purpose:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: reserved-memory-nova
data:
  04-reserved-memory-nova.conf: |
    [DEFAULT]
    reserved_host_memory_mb=75000
```

NOTE

The value for the **reserved_host_memory_mb** parameter may be set so that the Compute service scheduler does not give memory to a virtual machine that a Ceph OSD on the same server needs. The example reserves 5 GB per OSD for 10 OSDs per host in addition to the default reserved memory for the hypervisor. In an IOPS-optimized cluster, you can improve performance by reserving more memory for each OSD. The 5 GB number is provided as a starting point which can be further tuned if necessary.

6. Add **reserved-memory-nova** to the **configMaps** list by editing the **OpenStackDataPlaneService/nova-custom-ceph** file:

```
kind: OpenStackDataPlaneService
<...>
spec:
  configMaps:
  - ceph-nova
  - reserved-memory-nova
```

7. Apply the CR changes.

```
$ oc apply -f <dataplane_cr_file>
```

- Replace **<dataplane_cr_file>** with the name of your file.

NOTE

Ansible does not configure or validate the networks until the **OpenStackDataPlaneDeployment** CRD is created.

8. Create an **OpenStackDataPlaneDeployment** CRD, as described in Deploying the data plane, which has the **OpenStackDataPlaneNodeSet** CRD file defined above to have Ansible configure the services on the data plane nodes.

CHAPTER 4. CONFIGURING THE BLOCK STORAGE SERVICE (CINDER)

You can configure the following back ends for the Block Storage service (cinder):

- Ceph RBD.
- iSCSI, FC, or NVMe-TCP storage protocols. For information about configuring a third-party back end for the Block Storage service, see OSP18 Cinder Alternative Storage .
- NFS.

4.1. CONFIGURING AN NFS BACK END

You can configure the Block Storage service (cinder) with a generic NFS back end to provide an alternative storage solution by completing the following high level tasks:

1. To ensure network connectivity between the NFS server, the Red Hat OpenShift cluster, and the Compute nodes, complete the following tasks:
   a. Confirm all Block Storage services are operational.
   b. Create a test volume from an Image service (glance) image.
   c. Boot a VM from the test volume or attach a VM to the test volume.
2. Create a secret containing NFS server connection information.
3. Configure the **OpenStackControlPlane** custom resource (CR) to use the NFS storage as the back end for the Block Storage service.

NOTE

When using Red Hat OpenStack in a production environment, use a certified third-party NFS driver. The generic NFS driver is not recommended for a production environment.

4.1.1. Creating the NFS server connection secret

Create a server connection secret to prevent placing server connection information directly in the **OpenStackControlPlane** CRD.

Procedure

1. Create a configuration file that contains NFS server connection information.
   The following is an example of the contents of a configuration file:

```
[nfs]
nas_host=192.168.130.1
nas_share_path=/var/nfs/cinder
```

2. Save the configuration file.
3. Create the secret based on the configuration file:
   $ oc create secret generic <secret_name> --from-file=<configuration_file_name>

- Replace **<secret_name>** with the name you wish to assign to the secret.
- Replace **<configuration_file_name>** with the name of the configuration file you created.

4. Delete the configuration file.

4.1.2. Configuring the control plane to use the generic NFS driver

Configure the Block Storage service (cinder) in the **OpenStackControlPlane** CR to use NFS storage.

NOTE

Use a certified third-party NFS driver when using Red Hat OpenStack Services on OpenShift (RHOSO) in a production environment. The generic NFS driver is not recommended for a production environment.

Procedure

1. Edit the **OpenStackControlPlane** CR.
2. Add the **customServiceConfig** parameter to the **cinder** template to configure the Block Storage service.
   The following is an example of using the **customServiceConfig** parameter to configure the Block Storage service:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      cinderVolumes:
        nfs:
          replicas: 1
          networkAttachments:
          - storage
          customServiceConfig: |
            [nfs]
            volume_backend_name=nfs
            volume_driver=cinder.volume.drivers.nfs.NfsDriver
            nfs_snapshot_support=true
            nas_secure_file_operations=false
            nas_secure_file_permissions=false
          customServiceConfigSecrets:
          - <nfs_secret_name> 1
```

1 The name of your secret created in Creating the NFS server connection secret .

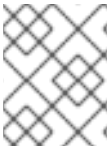
3. Apply the CR changes:

```
$ oc apply -f <control_plane_file>
```

- Replace **<control_plane_file>** with the name of your **OpenStackControlPlane** CR file.

4.2. CONFIGURING AUTOMATIC DATABASE CLEANUP

The Block Storage (cinder) and Shared File Systems (manila) services automatically purge database entries marked for deletion for a set number of days. By default, records are marked for deletion for 30 days. You can configure a different record age and schedule for purge jobs.



NOTE

The Image service (glance) also purges database entries automatically but this functionality is not currently user configurable.

Procedure

1. Open your **openstack_control_plane.yaml** file to edit the **OpenStackControlPlane** CR.
2. Add the **dbPurge** parameter to the **cinder** or **manila** template to configure database cleanup depending on the service you want to configure.
   The following is an example of using the **dbPurge** parameter to configure the Block Storage service:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      dbPurge:
        age: 20 1
        schedule: 1 0 * * 0 2
```

1. The number of days a record has been marked for deletion before it is purged. The default value is 30 days. The minimum value is 1 day.
2. The schedule of when to run the job in a crontab format. The default value is **1 0 * * *** . This default value is equivalent to **00:01** daily.

3. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml
```

CHAPTER 5. CONFIGURING THE IMAGE SERVICE (GLANCE)

The Image service (glance) provides discovery, registration, and delivery services for disk and server images. It provides the ability to copy or snapshot a server image, and immediately store it. You can use stored images as templates to commission new servers quickly and more consistently than installing a server operating system and individually configuring services.

You can configure the following back ends (stores) for the Image service:

- RADOS Block Device (RBD) is the default back end when you use Red Hat Ceph Storage.
- Block Storage (cinder).
- Object Storage (swift).
- NFS.

5.1. CONFIGURING A CEPH RBD BACK END

You can configure the Image service (glance) with Red Hat Ceph Storage RADOS Block Device (RBD) as the storage back end.

Prerequisites

- Ensure network connectivity between the storage back end, the Red Hat OpenShift cluster, and the Compute nodes.

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the following parameters to the **glance** template to configure Ceph RBD as the back end:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
    ...
  glance:
    template:
      databaseInstance: openstack
      databaseUser: glance
      customServiceConfig: |
        [DEFAULT]
        enabled_backends = default_backend:rbd
        enabled_import_methods=[web-download,glance-direct]
        [glance_store]
        default_backend = default_backend
        [default_backend]
        rbd_store_ceph_conf = /etc/ceph/ceph.conf
        store_description = "RBD backend"
        rbd_store_pool = images
        rbd_store_user = openstack
        rbd_thin_provisioning = True
...
```

2. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

3. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

The **OpenStackControlPlane** resources are created when the status is "Setup complete".

TIP

Append the **-w** option to the end of the **get** command to track deployment progress.

Image conversion

When you use Red Hat Ceph Storage as a back end for the Image service, **image-conversion** is enabled by default with a dedicated PersistentVolumeClaim (PVC), **glance-conversion**. The PVC is mounted to the **/var/lib/glance/os_glance_staging_store** path. You can run the **oc describe pod** command to see the **glance-conversion** PVC:

Example output:

```
...
    Mounts:
      /etc/ceph from ceph (ro)
      /etc/my.cnf from config-data (ro,path="my.cnf")
      /usr/local/bin/container-scripts from scripts (ro)
      /var/lib/config-data/default from config-data (ro)
      /var/lib/glance from glance (rw)
      /var/lib/glance/os_glance_staging_store from glance-conversion (rw)
      /var/lib/kolla/config_files/config.json from config-data (ro,path="glance-api-config.json")
      /var/log/glance from logs (rw)
...
```

The PVC is only created for an external instance to store the staging data of an uploaded image.

5.2. CONFIGURING A BLOCK STORAGE BACK END

You can configure the Image service (glance) with the Block Storage service (cinder) as the storage back end.

Prerequisites

- Ensure network connectivity between the storage back end, the Red Hat OpenShift cluster, and the Compute nodes.

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the following parameters to the **glance** template to configure the Block Storage service as the back end:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
```

```
spec:
    ...
  glance:
    template:
      customServiceConfig: |
        [DEFAULT]
        enabled_backends = default_backend:cinder
        [glance_store]
        default_backend = default_backend
        [default_backend]
        rootwrap_config = /etc/glance/rootwrap.conf
        description = Default cinder backend
        cinder_store_user_name = {{ .ServiceUser }}
        cinder_store_password = {{ .ServicePassword }}
        cinder_store_project_name = servicecinder_catalog_info volumev3::publicURL
...
```

2. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

3. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

The **OpenStackControlPlane** resources are created when the status is "Setup complete".

TIP

Append the **-w** option to the end of the **get** command to track deployment progress.

5.3. CONFIGURING AN OBJECT STORAGE BACK END

You can configure the Image service (glance) with the Object Storage service (swift) as the storage back end.

Prerequisites

- Ensure network connectivity between the storage back end, the Red Hat OpenShift cluster, and the Compute nodes.

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the following parameters to the **glance** template to configure the Object Storage service as the back end:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
    …
```

```
    glance:
      template:
        customServiceConfig: |
          [DEFAULT]
          enabled_backends = default_backend:swift
          [glance_store]
          default_backend = default_backend
          [default_backend]
          swift_store_create_container_on_put = True
          swift_store_auth_version = 3
          swift_store_auth_address = {{ .KeystoneInternalURL }}
          swift_store_key = {{ .ServicePassword }}
          swift_store_user = service:glance
          swift_store_endpoint_type = internalURL
...
```

2. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

3. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

The **OpenStackControlPlane** resources are created when the status is "Setup complete".

TIP

Append the **-w** option to the end of the **get** command to track deployment progress.

5.4. CONFIGURING AN NFS BACK END

When you mount an NFS share on the Image service (glance), the Image service does not manage the operation. The Image service writes data to the file system but is unaware that the back end is an NFS share.

If you use NFS as an Image service back end, Red Hat recommends the following best practices to mitigate risk:

- Use a reliable production-grade NFS back end.
- Make sure the network is propagated to the OpenShift control plane, where the Image service is deployed, and the Image service has a NetworkAttachmentDefinition (NAD) that points to the network. This configuration ensures that the Image service pods can reach the NFS server.
- Set underlying file system permissions. Write permissions must be present in the shared file system that you use as a store.
- Ensure that the user and the group that the **glance-api** process runs on do not have write permissions on the mount point at the local file system. This means that the process can detect possible mount failure and put the store into read-only mode during a write attempt.

Limitations

- In Red Hat OpenStack Services on OpenShift (RHOSO), you cannot set client-side NFS mount options in a pod spec. You can set NFS mount options in one of the following ways:
  - Set server-side mount options.
  - Use **/etc/nfsmount.conf**.
  - Mount NFS volumes by using PersistentVolumes, which have mount options.

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the **extraMounts** parameter in the **spec** section to add the export path and IP address of the NFS share. The path is mapped to **/var/lib/glance/images**, where the Image service API stores and retrieves images:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
...
spec:
  extraMounts:
  - extraVol:
    - extraVolType: Nfs
      mounts:
      - mountPath: /var/lib/glance/images
        name: nfs
      propagation:
      - Glance
      volumes:
      - name: nfs
        nfs:
          path: {{ <nfs_export_path> }}
          server: {{ <nfs_ip_address> }}
    name: r1
    region: r1
```

   - Replace **<nfs_export_path>** with the export path of your NFS share.
   - Replace **<nfs_ip_address>** with the IP address of your NFS share. This IP address must be part of the overlay network that is reachable by the Image service.

2. Add the following parameters to the **glance** template to configure NFS as the back end:

```
...
spec:
  extraMounts:
    ...
  glance:
    template:
      customServiceConfig: |
        [DEFAULT]
        enabled_backends = default_backend:file
        [glance_store]
        default_backend = default_backend
```

```
      [default_backend]
      filesystem_store_datadir = /var/lib/glance/images
    databaseInstance: openstack
    glanceAPIs:
...
```

3. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

4. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

The **OpenStackControlPlane** resources are created when the status is "Setup complete".

TIP

Append the **-w** option to the end of the **get** command to track deployment progress.

CHAPTER 6. CONFIGURING THE OBJECT STORAGE SERVICE (SWIFT)

You can configure the Object Storage service (swift) to use PersistentVolumes (PVs) on OpenShift nodes or disks on external data plane nodes.

When you use PVs on OpenShift nodes, this configuration is limited to a single PV per node. The Object Storage service requires multiple PVs. To maximize availability and data durability, you create these PVs on different nodes, and only use one PV per node.

You can use external data plane nodes for more flexibility in larger storage deployments, where you can use multiple disks per node to deploy a larger Object Storage cluster.

Prerequisites

- You have the **oc** and **podman** command line tools installed on your workstation.
- You are logged on to a workstation that has access to the RHOSO control plane as a user with **cluster-admin** privileges.

6.1. DEPLOYING THE OBJECT STORAGE SERVICE ON OPENSHIFT NODES BY USING PERSISTENTVOLUMES

You use at least two **swiftProxy** replicas and three **swiftStorage** replicas in a default Object Storage service (swift) deployment. You can increase these values to distribute storage across more nodes and disks.

The **ringReplicas** value defines the number of object copies in the cluster. For example, if you set **ringReplicas: 3** and **swiftStorage/replicas: 5**, every object is stored on 3 different PersistentVolumes (PVs), and there are 5 PVs in total.

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the following parameters to the **swift** template:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack-galera-network-isolation
  namespace: openstack
spec:
  ...
  swift:
    enabled: true
    template:
      swiftProxy:
        replicas: 2
      swiftRing:
        ringReplicas: 3
      swiftStorage:
        replicas: 3
```

```
        storageClass: <swift-storage>
        storageRequest: 100Gi
...
```

- Increase the **swiftProxy/replicas:** value to distribute proxy instances across more nodes.
- Replace the **ringReplicas:** value to define the number of object copies you want in your cluster.
- Increase the **swiftStorage/replicas:** value to define the number of PVs in your cluster.
- Replace **<swift-storage>** with the name of the storage class you want the Object Storage service to use.

2. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

3. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

The **OpenStackControlPlane** resources are created when the status is "Setup complete".

TIP

Append the **-w** option to the end of the **get** command to track deployment progress.

6.2. OBJECT STORAGE RINGS

The Object Storage service (swift) uses a data structure called the ring to distribute partition space across the cluster. This partition space is core to the data durability engine in the Object Storage service. With rings, the Object Storage service can quickly and easily synchronize each partition across the cluster.

Rings contain information about Object Storage partitions and how partitions are distributed among the different nodes and disks in your Red Hat OpenStack Services on OpenShift (RHOSO) deployment. When any Object Storage component interacts with data, a quick lookup is performed locally in the ring to determine the possible partitions for each object.

The Object Storage service has three rings to store the following types of data:

- Account information
- Containers, to facilitate organizing objects under an account
- Object replicas

6.3. RING PARTITION POWER

The ring power determines the partition to which a resource, such as an account, container, or object, is mapped. The partition is included in the path under which the resource is stored in a back-end file system. Therefore, changing the partition power requires relocating resources to new paths in the back-

end file systems.

In a heavily populated cluster, a relocation process is time consuming. To avoid downtime, relocate resources while the cluster is still operating. You must do this without temporarily losing access to data or compromising the performance of processes, such as replication and auditing. For assistance with increasing ring partition power, contact Red Hat Support.

When you use separate nodes for the Object Storage service (swift), use a higher partition power value.

The Object Storage service distributes data across disks and nodes using modified *hash rings*. There are three rings by default: one for accounts, one for containers, and one for objects. Each ring uses a fixed parameter called *partition power*. This parameter sets the maximum number of partitions that can be created.

6.4. INCREASING RING PARTITION POWER

You can only change the partition power parameter for new containers and their objects, so you must set this value before initial deployment.

The default partition power value is **10**. Refer to the following table to select an appropriate partition power if you use three replicas:

Table 6.1. Appropriate partition power values per number of available disks

Partition Power	Maximum number of disks
10	~ 35
11	~ 75
12	~ 150
13	~ 250
14	~ 500

IMPORTANT

Setting an excessively high partition power value (for example, **14** for only 40 disks) negatively impacts replication times.

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and change the value for **partPower** under the **swiftRing** parameter in the **swift** template:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack-galera-network-isolation
  namespace: openstack
spec:
```

```
...
swift:
  enabled: true
  template:
    swiftProxy:
      replicas: 2
    swiftRing:
      partPower: 12
      ringReplicas: 3
...
```

- Replace **<12>** with the value you want to set for partition power.

TIP

You can also configure an additional object server ring for new containers. This is useful if you want to add more disks to an Object Storage service deployment that initially uses a low partition power.

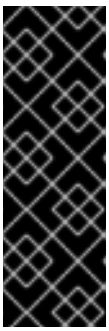
CHAPTER 7. CONFIGURING THE SHARED FILE SYSTEMS SERVICE (MANILA)

When you deploy the Shared File Systems service (manila), you can choose one or more supported back ends, such as native CephFS, CephFS-NFS, NetApp, and others.

For a complete list of supported back-end appliances and drivers, see the Manila section of the Red Hat Knowledge Article, Component, Plug-In, and Driver Support in Red Hat OpenStack Platform .

Prerequisites

- You have the **oc** command line tool installed on your workstation.
- You are logged on to a workstation that has access to the RHOSO control plane as a user with **cluster-admin** privileges.
- For native CephFS or CephFS-NFS, a CephFS file system must exist on the Red Hat Ceph Storage cluster. For more information, see Integrating Red Hat Ceph Storage .
- For native CephFS or CephFS-NFS, a Ceph user must exist that has CephX capabilities (caps) to perform operations on the CephFS file system. For more information, see Integrating Red Hat Ceph Storage.
- For CephFS-NFS only, a **ceph nfs** service must exist in the Ceph Storage cluster. For more information, see Integrating Red Hat Ceph Storage .
- For back ends where **driver_handles_share_servers=false**, you configure the networking in advance rather than dynamically in the back end for the Shared File Systems service.
- For a CephFS-NFS back end, you create an isolated StorageNFS network for NFS exports and a corresponding StorageNFS shared provider network in the Networking service (neutron). The StorageNFS shared provider network maps to the isolated StorageNFS network of the data center.
- Ensure that the NFS service is isolated on a network that you can share with all Red Hat OpenStack Services on OpenShift (RHOSO) users. For more information about customizing the NFS service, see NFS cluster and export management in the Red Hat Ceph Storage *File System Guide*.



IMPORTANT

When you deploy an NFS service for the Shared File Systems service, do not select a custom port to expose NFS. Only the default NFS port of 2049 is supported. You must enable the Red Hat Ceph Storage **ingress** service and set the **ingress-mode** to **haproxy-protocol**. Otherwise, you cannot use IP-based access rules with the Shared File Systems service. For security in production environments, Red Hat does not recommend providing access to **0.0.0.0/0** on shares to mount them on client machines.

7.1. CONFIGURING A NATIVE CEPHFS BACK END

You can configure the Shared File Systems service (manila) with native CephFS as the storage back end.

Limitations

You can expose a native CephFS back end to trusted users, but take the following security measures:

- Configure the storage network as a provider network.
- Apply role-based access control (RBAC) policies to secure the storage provider network.
- Create a private share type.

Prerequisites

- An isolated storage network.
- Ensure network connectivity between the storage back end, the Red Hat OpenShift cluster, and the Compute nodes.
- You have created a Red Hat Ceph Storage secret. For more information, see Integrating Red Hat Ceph Storage

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the **extraMounts** parameter in the **spec** section to present the Ceph configuration files:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  extraMounts:
    - name: v1
      region: r1
      extraVol:
        - propagation:
          - ManilaShare
          extraVolType: Ceph
          volumes:
          - name: ceph
            projected:
              sources:
              - secret:
                  name: <ceph-conf-files>
          mounts:
          - name: ceph
            mountPath: "/etc/ceph"
            readOnly: true
```

2. Add the following parameters to the **manila** template to configure the native CephFS back end:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
    ...
  manila:
    enabled: true
    template:
      manilaAPI:
```

```
      replicas: 3
      customServiceConfig: |
        [DEFAULT]
        debug = true
        enabled_share_protocols=cephfs
    manilaScheduler:
      replicas: 3
    manilaShares:
      cephfsnative:
        replicas: 1
        networkAttachments:
        - storage
        customServiceConfig: |
          [DEFAULT]
          enabled_share_backends=cephfs
          [cephfs]
          driver_handles_share_servers=False
          share_backend_name=cephfs
          share_driver=manila.share.drivers.cephfs.driver.CephFSDriver
          cephfs_conf_path=/etc/ceph/ceph.conf
          cephfs_auth_id=openstack
          cephfs_cluster_name=ceph
          cephfs_volume_mode=0755
          cephfs_protocol_helper_type=CEPHFS
...
```

3. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

4. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

The **OpenStackControlPlane** resources are created when the status is "Setup complete".

TIP

Append the **-w** option to the end of the **get** command to track deployment progress.

7.2. CONFIGURING A CEPHFS-NFS BACK END

You can configure the Shared File Systems service (manila) with CephFS-NFS as the storage back end.

Prerequisites

- The isolated storage network is configured on the share manager pod on OpenShift so that the Shared File Systems service can communicate with the Red Hat Ceph Storage cluster.
- For NFS traffic, Red Hat recommends using an isolated NFS network. This network does not need to be available to the share manager pod for the Shared File Systems service on OpenShift, but it must be available to Compute instances owned by end users.

- Ensure network connectivity between the storage back end, the Red Hat OpenShift cluster, and the Compute nodes.

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the following parameters to the **manila** template:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
    ...
  manila:
    enabled: true
    template:
      manilaAPI:
        replicas: 3
        customServiceConfig: |
          [DEFAULT]
          debug = true
          enabled_share_protocols=nfs
      manilaScheduler:
        replicas: 3
      manilaShares:
        share1:
          customServiceConfig: |
              [DEFAULT]
              enabled_share_backends=cephfsnfs
              [cephfsnfs]
              driver_handles_share_servers=False
              share_backend_name=cephfs
              share_driver=manila.share.drivers.cephfs.driver.CephFSDriver
              cephfs_auth_id=openstack
              cephfs_cluster_name=ceph
              cephfs_nfs_cluster_id=cephfs
              cephfs_protocol_helper_type=NFS
...
```

2. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

3. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

The **OpenStackControlPlane** resources are created when the status is "Setup complete".

TIP

Append the **-w** option to the end of the **get** command to track deployment progress.

7.3. CONFIGURING ALTERNATIVE BACK ENDS

To configure the Shared File Systems service (manila) with an alternative back end, for example, NetApp or Pure Storage, complete the following high level tasks:

1. Create the server connection secret.
2. Configure the **OpenStackControlPlane** CR to use the alternative storage system as the back end for the Shared File Systems service.

Prerequisites

- You have prepared the alternative storage system for consumption by Red OpenStack Services on OpenShift (RHOSO).
- Network connectivity between the Red Hat OpenShift cluster, the Compute nodes, and the alternative storage system.

7.3.1. Creating the server connection secret

Create a server connection secret for an alternative back end to prevent placing server connection information directly in the **OpenStackControlPlane** CR.

Procedure

1. Create a configuration file that contains the server connection information for your alternative back end. In this example, you are creating the **secret** for a NetApp back end.
   The following is an example of the contents of a configuration file:

   ```
   [netapp]
   netapp_server_hostname = <netapp_ip>
   netapp_login = <netapp_user>
   netapp_password = <netapp_password>
   netapp_vserver = <netappvserver>
   ```

   - Replace **<netapp_ip>** with the IP address of the server.
   - Replace **<netapp_user>** with the login user name.
   - Replace **<netapp_password>** with the login password.
   - Replace **<netappvserver>** with the vserver name. You do not need this option if configuring the **driver_handles_share_servers=True** mode.
2. Save the configuration file.
3. Create the secret based on the configuration file:
   $ oc create secret generic <secret_name> --from-file=<configuration_file_name>
   - Replace **<secret_name>** with the name you want to assign to the secret.
   - Replace **<configuration_file_name>** with the name of the configuration file you created.
4. Delete the configuration file.

7.3.2. Configuring an alternative back end

You can configure the Shared File Systems service (manila) with an alternative storage back end, for example, a NetApp back end.

Prerequisites

- Ensure network connectivity between the storage back end, the Red Hat OpenShift cluster, and the Compute nodes.

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the following parameters to the **manila** template:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
    ...
    manila:
    enabled: true
    template:
      manilaAPI:
        replicas: 3
        customServiceConfig: |
          [DEFAULT]
          debug = true
          enabled_share_protocols=cifs
      manilaScheduler:
        replicas: 3
      manilaShares:
        share1:
          networkAttachments:
          - storage
          customServiceConfigSecrets:
          - manila_netapp_secret
          customServiceConfig: |
            [DEFAULT]
            debug = true
            enabled_share_backends=netapp
            [netapp]
       driver_handles_share_servers=False
            share_backend_name=netapp
            share_driver=manila.share.drivers.netapp.common.NetAppDriver
            netapp_storage_family=ontap_cluster
...
```

2. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

3. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

The **OpenStackControlPlane** resources are created when the status is "Setup complete".

TIP

Append the **-w** option to the end of the **get** command to track deployment progress.

7.3.3. Custom configuration files

When you configure an alternative back end for the Shared File Systems service (manila), you might need to use additional configuration files. You can use the **extraMounts** parameter in your **OpenStackControlPlane** CR file to present these configuration files as OpenShift **configMap** or **secret** objects in the relevant share manager pod.

Example:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
...
  extraMounts:
    - name: v1
      region: r1
      extraVol:
        - propagation:
          - sharepod1
          extraVolType: Undefined
          volumes:
          - name: backendconfig
            projected:
              sources:
              - secret:
                  name: manila-sharepod1-secrets
          mounts:
          - name: backendconfig
            mountPath: /etc/manila/drivers
            readOnly: true
...
```

7.3.4. Custom storage driver images

When you configure an alternative back end for the Shared File Systems service (manila), you might need to use a custom **manilaShares** container image from the vendor on the Red Hat Ecosystem Catalog. You can add the path to the container image to your **OpenStackControlPlane** CR file to use it in the relevant share manager pod.

Example:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  manila:
    template:
      manilaShares:
        pure-storage:
          networkAttachments:
          - storage
```

```
        containerImage: image-registry.openshift-image-registry.svc:5000/openshift/manila-share-
pure:oso18
...
```

7.4. CONFIGURING MULTIPLE BACK ENDS

You can deploy multiple back ends for the Shared File Systems service (manila), for example, a CephFS-NFS back end, a native CephFS back end, and a third-party back end. Add one back end only per pod.

Prerequisites

- When you use a back-end driver from a storage vendor that requires external software components, you must override the standard container image for the Shared File Systems service during deployment. You can find custom container images, for example, the Dell EMC Unity container image for a Dell EMC Unity storage system, at Red Hat Ecosystem Catalog .
- Ensure network connectivity between the storage back end, the Red Hat OpenShift cluster, and the Compute nodes.

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the following parameters to the **manila** template to configure the back ends. In this example, there is a CephFS-NFS back end, a native CephFS back end, and a Pure Storage back end:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
    ...
      manila:
          enabled: true
          template:
            manilaAPI:
              replicas: 3
              customServiceConfig: |
                [DEFAULT]
                debug = true
                enabled_share_protocols=nfs,cephfs,cifs
            manilaScheduler:
              replicas: 3
...
```

2. Add the configuration for each back end you want to use:
   - Add the configuration for the CephFS-NFS back end:

```
        ...
            customServiceConfig: |
            ...
            manilaShares:
              cephfsnfs:
                networkAttachments:
                - storage
                customServiceConfig: |
```

```
            [DEFAULT]
            enabled_share_backends=cephfsnfs
            [cephfsnfs]
            driver_handles_share_servers=False
            share_backend_name=cephfs
            share_driver=manila.share.drivers.cephfs.driver.CephFSDriver
            cephfs_auth_id=openstack
            cephfs_cluster_name=ceph
            cephfs_nfs_cluster_id=cephfs
            cephfs_protocol_helper_type=NFS
          replicas: 1
    ...
```

- Add the configuration for the native CephFS back end:

```
    ...
      customServiceConfig: |
      ...
      manilaShares:
       cephfsnfs:
       ...
       cephfs:
         networkAttachments:
         - storage
         customServiceConfig: |
            [DEFAULT]
            enabled_share_backends=cephfs
            [cephfs]
            driver_handles_share_servers=False
            share_backend_name=cephfs
            share_driver=manila.share.drivers.cephfs.driver.CephFSDriver
            cephfs_conf_path=/etc/ceph/ceph.conf
            cephfs_auth_id=openstack
            cephfs_protocol_helper_type=CEPHFS
          replicas: 1
     ...
```

- Add the configuration for the Pure Storage back end:

```
    ...
      customServiceConfig: |
      ...
      manilaShares:
       cephfsnfs:
       ...
       cephfs:
       ...
       pure:
         containerImage: image-registry.openshift-image-
registry.svc:5000/openshift/manila-share-pure:oso18
         networkAttachments:
         - storage
         customServiceConfigSecret: |
         - manila-pure-secret
         customServiceConfig: |
```

```
            [DEFAULT]
            debug = true
            enabled_share_backends=pure
            [pure]
            driver_handles_share_servers=False
            share_backend_name=pure

share_driver=manila.share.drivers.purestorage.flashblade.FlashBladeShareDriver
  ...
```

3. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

4. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

The **OpenStackControlPlane** resources are created when the status is "Setup complete".

TIP

Append the **-w** option to the end of the **get** command to track deployment progress.

7.5. CONFIRMING DEPLOYMENT OF MULTIPLE BACK ENDS

Use the **openstack share service list** command to verify that the storage back ends for the Shared File Systems service (manila) deployed successfully. If you use a health check on multiple back ends, a ping test returns a response even if one of the back ends is unresponsive, so this is not a reliable way to verify your deployment.

Procedure

1. Access the remote shell for the **OpenStackClient** pod from your workstation:

```
$ oc rsh -n openstack openstackclient
```

2. Confirm the list of Shared File Systems service back ends:

```
$ openstack share service list
```

The status of each successfully deployed back end shows as **enabled** and the state shows as **up**.

3. Exit the **openstackclient** pod:

```
$ exit
```

7.6. CREATING AVAILABILITY ZONES FOR BACK ENDS

You can create availability zones (AZs) for Shared File Systems service back ends to group cloud

infrastructure and services logically for users. Map the AZs to failure domains and compute resources for high availability, fault tolerance, and resource scheduling. For example, you can create an AZ of Compute nodes that have specific hardware that users can specify when they create an instance that requires that hardware.

Post deployment, use the **availability_zones** share type extra specification to limit share types to one or more AZs. Users can create a share directly in an AZ as long as the share type does not restrict them.

Procedure

The following example deploys two back ends where **CephFS** is zone 1 and **NetApp** is zone 2.

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the following parameters to the **manila** template:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  ...
 manila:
  enabled: true
  template:
   manilaShares:
    cephfs:
     customServiceConfig: |
      [cephfs]
      backend_availability_zone = zone_1
     ...
    netapp:
     customServiceConfig: |
      [netapp]
      backend_availability_zone = zone_2
     ...
```

2. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

3. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

The **OpenStackControlPlane** resources are created when the status is "Setup complete".

TIP

Append the **-w** option to the end of the **get** command to track deployment progress.

7.7. CHANGING THE ALLOWED NAS PROTOCOLS

You can use the Shared File Systems service (manila) to export shares in the NFS, CephFS, or CIFS network attached storage (NAS) protocols. By default, the Shared File Systems service enables NFS and CIFS, and this may not be supported by the back ends in your deployment.

You can change the **enabled_share_protocols** parameter and list only the protocols that you want to allow in your cloud. For example, if back ends in your deployment support both NFS and CIFS, you can change the default value and enable only one protocol. The NAS protocols that you assign must be supported by the back ends in your Shared File Systems service deployment.

Not all storage back-end drivers support the CIFS protocol. For information about which certified storage systems support CIFS, see the Red Hat Ecosystem Catalog .

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the following parameters to the **manila** template. In this example, you enable the NFS protocol:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  ...
    manila:
      enabled: true
      template:
        manilaAPI:
          customServiceConfig: |
            [DEFAULT]
            enabled_share_protocols = NFS
        ...
```

2. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

3. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

The **OpenStackControlPlane** resources are created when the status is "Setup complete".

TIP

Append the **-w** option to the end of the **get** command to track deployment progress.

7.8. VIEWING BACK-END STORAGE CAPACITY

The scheduler component of the Shared File Systems service (manila) makes intelligent placement decisions based on several factors such as capacity, provisioning configuration, placement hints, and the capabilities that the back-end storage system driver detects and exposes. You can use share types and extra specifications to modify placement decisions.

Procedure

1. Access the remote shell for the **OpenStackClient** pod from your workstation:

```
$ oc rsh -n openstack openstackclient
```

2. Run the following command to view the available back-end storage capacity:

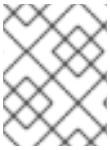
```
$ openstack share pool list --detail
```

3. Exit the **openstackclient** pod:

```
$ exit
```

7.9. CONFIGURING AUTOMATIC DATABASE CLEANUP

The Block Storage (cinder) and Shared File Systems (manila) services automatically purge database entries marked for deletion for a set number of days. By default, records are marked for deletion for 30 days. You can configure a different record age and schedule for purge jobs.



NOTE

The Image service (glance) also purges database entries automatically but this functionality is not currently user configurable.

Procedure

1. Open your **openstack_control_plane.yaml** file to edit the **OpenStackControlPlane** CR.

2. Add the **dbPurge** parameter to the **cinder** or **manila** template to configure database cleanup depending on the service you want to configure.
   The following is an example of using the **dbPurge** parameter to configure the Block Storage service:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      dbPurge:
        age: 20 1
        schedule: 1 0 * * 0 2
```

1. The number of days a record has been marked for deletion before it is purged. The default value is 30 days. The minimum value is 1 day.

2. The schedule of when to run the job in a crontab format. The default value is **1 0 * * ***. This default value is equivalent to **00:01** daily.

3. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml
```