Red Hat OpenStack Services on OpenShift 18.0

配置持久性存储

在 OpenShift 中为 Red Hat OpenStack Services 配置存储服务

Last Updated: 2024-11-05

Red Hat OpenStack Services on OpenShift 18.0 配置持久性存储

在 OpenShift 中为 Red Hat OpenStack Services 配置存储服务

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

在 OpenShift 部署中，为 Red Hat OpenStack Services 中的 block、image、object 和 file 存储配置服务。

目录

对红帽文档提供反馈 4

第 1 章 配置持久性存储 **5**
存储解决方案 5

第 2 章 集成 RED HAT CEPH STORAGE **6**
2.1. 创建 RED HAT CEPH STORAGE 池 6
2.2. 创建 RED HAT CEPH STORAGE SECRET 7
2.3. 获取 RED HAT CEPH STORAGE 文件系统标识符 8
2.4. 配置 CONTROL PLANE 以使用 RED HAT CEPH STORAGE 集群 8
2.5. 配置数据平面以使用 RED HAT CEPH STORAGE 集群 13
2.6. 配置外部 CEPH 对象网关后端 16
2.7. 为外部 RED HAT CEPH STORAGE 集群使用 TLS 配置 RGW 20
2.8. 为带有依赖项的卷或镜像启用延迟删除 26
2.9. RED HAT CEPH STORAGE RBD 集成故障排除 27
2.10. 自定义和管理 RED HAT CEPH STORAGE 33

第 3 章 配置块存储服务(CINDER) **35**
3.1. 术语 35
3.2. RED HAT OPENSTACK SERVICES ON OPENSHIFT (RHOSO)中的块存储服务(CINDER)的改进. 36
3.3. 配置传输协议 37
3.4. 配置初始默认值 42
3.5. 配置 API 服务 43
3.6. 配置调度程序服务 46
3.7. 配置卷服务 50
3.8. 配置后端可用区 54
3.9. 配置通用 NFS 后端 56
3.10. 配置 NFS 转换目录 59
3.11. 配置自动数据库清理 60
3.12. 保留作业 61
3.13. 解决主机名冲突 62
3.14. 使用其他容器镜像 63

第 4 章 配置块存储备份服务 **65**
4.1. 备份的存储后端 65
4.2. 为备份设置副本数量 66
4.3. 备份性能注意事项 67
4.4. 为备份设置选项 67
4.5. 启用数据压缩 69
4.6. 为块存储备份配置 CEPH RBD 后端 71
4.7. 为备份配置 OBJECT STORAGE 服务(SWIFT)后端 72
4.8. 为备份配置 NFS 后端 73
4.9. 为备份配置 S3 后端 75
4.10. 块存储卷备份元数据 76

第 5 章 配置镜像服务(GLANCE) **77**
5.1. 为镜像服务配置块存储后端 77
5.2. 配置对象存储后端 79
5.3. 配置 NFS 后端 80

第 6 章 配置 OBJECT STORAGE 服务 (SWIFT) **83**
6.1. 使用 PERSISTENTVOLUME 在 OPENSHIFT 节点上部署 OBJECT STORAGE 服务 83
6.2. 对象存储环 84

6.3. RING 分区电源 85
6.4. 增加环分区电源 85

第 7 章 配置共享文件系统服务(MANILA) 88
7.1. 启用共享文件系统服务 89
7.2. 配置原生 CEPHFS 后端 91
7.3. 配置 CEPHFS-NFS 后端 93
7.4. 配置替代后端 94
7.5. 配置多个后端 98
7.6. 验证共享文件系统服务的部署 101
7.7. 验证多个后端的部署 102
7.8. 为后端创建可用区 103
7.9. 更改允许的 NAS 协议 104
7.10. 查看后端存储容量 105
7.11. 配置自动数据库清理 106

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。与我们分享您的成功秘诀。

在 JIRA 中提供文档反馈

使用 Create Issue 表单在 OpenShift (RHOSO)或更早版本的 Red Hat OpenStack Platform (RHOSP)上提供有关 Red Hat OpenStack Services 文档的反馈。当您为 RHOSO 或 RHOSP 文档创建问题时，这个问题将在 RHOSO Jira 项目中记录，您可以在其中跟踪您的反馈的进度。

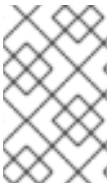
要完成 Create Issue 表单，请确保您已登录到 JIRA。如果您没有红帽 JIRA 帐户，您可以在 https://issues.redhat.com 创建一个帐户。

1. 点击以下链接打开 **Create Issue** 页面： Create Issue
2. 完成 **Summary** 和 **Description** 字段。在 **Description** 字段中，包含文档 URL、章节号以及问题的详细描述。不要修改表单中的任何其他字段。
3. 点 **Create**。

第 1 章 配置持久性存储

当您在 OpenShift 上部署 Red Hat OpenStack Services (RHOSO)时，您可以将部署配置为使用 Red Hat Ceph Storage 作为存储的后端，您可以为块、镜像、对象和文件存储配置 RHOSO 存储服务。

您可以将外部 Red Hat Ceph Storage 集群与 Compute 服务(nova)和一个或多个 RHOSO 存储服务的组合集成，您可以创建超融合基础架构(HCI)环境。RHOSO 支持 Red Hat Ceph Storage 7.1 或更高版本。有关创建超融合基础架构(HCI)环境的详情，请参考 *部署超融合基础架构环境*。



注意

Red Hat OpenShift Data Foundation (ODF)可用于外部模式，以便与 Red Hat Ceph Storage 集成。不支持在内部模式中使用 ODF。有关以外部模式部署 ODF 的更多信息，请参阅 以外部模式部署 OpenShift Data Foundation。

RHOSO 识别两种类型的存储 - 临时和持久性：

- 临时存储与特定 Compute 实例关联。当该实例终止时，是关联的临时存储。这种类型的存储可用于运行时要求，如存储实例的操作系统。
- 持久性存储被设计为独立于任何正在运行的实例来存活（持久性）。此存储用于需要被不同实例或特定实例生命周期之外的任何数据。

RHOSO 存储服务与以下持久性存储类型对应：

- 块存储服务(cinder)：卷
- 镜像服务(glance)：镜像
- Object Storage 服务(swift)： Objects
- 共享文件系统服务(manila)：共享

所有持久性存储服务将数据存储在存储后端中。Red Hat Ceph Storage 可以用作所有四个服务的后端，在使用 Red Hat Ceph Storage 时，OpenStack 服务的功能和功能会被优化。

存储解决方案

RHOSO 支持以下存储解决方案：

- 使用 Ceph RBD 后端、iSCSI、FC 或 NVMe-TCP 存储协议或通用 NFS 后端配置块存储服务。
- 使用 Ceph RBD、块存储、对象存储或 NFS 后端配置镜像服务。
- 将 Object Storage 服务配置为在 OpenShift 节点或外部数据平面节点上使用 PersistentVolumes (PV)。
- 使用原生 CephFS、Ceph-NFS 或替代后端（如 NetApp 或 Pure Storage）配置共享文件系统服务。

有关为您的 RHOSO 部署规划存储解决方案和相关要求的详情，如联网和安全性，请参阅 规划 *部署中的规划存储和共享* 文件系统。

为了促进最佳实践的使用，红帽具有适用于 OpenStack 后端的认证流程。为了提高支持性和互操作性，请确保您的存储后端已通过 RHOSO 认证。您可以在 红帽生态系统目录 中检查认证状态。Ceph RBD 在所有 RHOSO 版本中作为后端认证。

第 2 章 集成 RED HAT CEPH STORAGE

您可以在 OpenShift (RHOSO)上配置 Red Hat OpenStack Services，以便与外部 Red Hat Ceph Storage 集群集成。此配置将以下服务连接到 Red Hat Ceph Storage 集群：

- Block Storage 服务 (cinder)
- Image 服务 (glance)
- Object Storage 服务 (swift)
- 计算服务(nova)
- 共享文件系统服务(manila)

要将 Red Hat Ceph Storage 配置为 RHOSO 存储的后端，请完成以下任务：

1. 验证 Red Hat Ceph Storage 是否已部署，并且所有必需的服务都在运行。
2. 在 Red Hat Ceph Storage 集群中创建 Red Hat Ceph Storage 池。
3. 在 Red Hat Ceph Storage 集群中创建 Red Hat Ceph Storage secret，以提供 Red Hat Ceph Storage 集群的 RHOSO 服务访问权限。
4. 获取 Ceph 文件系统标识符。
5. 配置 OpenStackControlPlane CR，以使用 Red Hat Ceph Storage 集群作为后端。
6. 配置 OpenStackDataPlane CR，以使用 Red Hat Ceph Storage 集群作为后端。

先决条件

- 访问 Red Hat Ceph Storage 集群。
- RHOSO control plane 安装在一个可正常工作的 RHOSO 集群中。

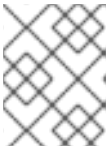
2.1. 创建 RED HAT CEPH STORAGE 池

在 Red Hat Ceph Storage 集群服务器上为每个使用集群的 RHOSO 服务创建池。

流程

1. 为 Compute 服务(vms)、块存储服务(volumes)和镜像服务(images)创建池：

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



注意

在创建池时，设置适当的放置组(PG)号，如 Red Hat Ceph *Storage 策略指南中的*放置组 中所述。

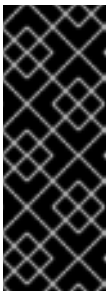
2. 可选：如果在 control plane 中启用了共享文件系统服务(manila)，则创建 **cephfs** 卷。这会自动启用 CephFS 元数据服务(MDS)，并在 Ceph 集群中创建必要的数据和元数据池：

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

3. 可选：在 Red Hat Ceph Storage 集群上部署 NFS 服务，以便将 CephFS 与 NFS 搭配使用：

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

- 将 **<vip>** 替换为分配给 NFS 服务的 IP 地址。NFS 服务应当隔离在一个可与所有 Red Hat OpenStack 用户共享的网络中。有关自定义 NFS 服务的更多信息，请参阅 NFS 集群和导出管理。

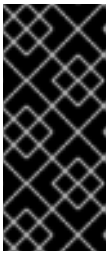


重要

当您为共享文件系统服务部署 NFS 服务时，请不要选择要公开 NFS 的自定义端口。仅支持 2049 的默认 NFS 端口。您必须启用 Red Hat Ceph Storage **ingress** 服务，并将 **ingress-mode** 设置为 **haproxy-protocol**。否则，您不能将基于 IP 的访问规则与共享文件系统服务一起使用。对于生产环境中的安全性，红帽不推荐在共享上提供对 **0.0.0.0/0** 的访问，以将其挂载到客户端机器上。

4. 为 RHOSO 创建 cephx 密钥以访问池：

```
$ 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'
```



重要

如果 control plane 中启用了共享文件系统服务，请将 **osd** caps 替换为以下内容：

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. 导出 cephx 密钥：

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

6. 导出配置文件：

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

2.2. 创建 RED HAT CEPH STORAGE SECRET

创建一个 secret，以便服务可以访问 Red Hat Ceph Storage 集群。

流程

1. 将创建 Red Hat Ceph Storage 池 流程中创建的 **cephx** 密钥和配置文件传输到可在 **openstack** 命名空间中创建资源的主机。
2. Base64 对这些文件进行编码，并将其存储在 **KEY** 和 **CONF** 环境变量中：

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

3. 创建 YAML 文件以创建 **Secret** 资源。
4. 使用环境变量，将 **Secret** 配置添加到 YAML 文件中：

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

5. 保存 YAML 文件。
6. 创建 **Secret** 资源：

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

- 将 **<secret_configuration_file>** 替换为您创建的 YAML 文件的名称。

注意

本节中的示例使用 **openstack** 作为 Red Hat Ceph Storage 用户的名称。 **Secret** 资源中的文件名必须与此用户名匹配。

例如，如果用于用户名 **openstack2** 的文件名为 **/etc/ceph/ceph.client.openstack2.keyring**，则机密数据行应为 **ceph.client.openstack2.keyring: $KEY**。

2.3. 获取 RED HAT CEPH STORAGE 文件系统标识符

Red Hat Ceph Storage File System Identifier (FSID)是集群的唯一标识符。FSID 用于配置和管理集群与 RHOSO 的互操作性。

流程

- 从 Red Hat Ceph Storage secret 中提取 FSID：
  $ FSID=$(oc get secret ceph-conf-files -o json | jq -r '.data."ceph.conf"' | base64 -d | grep fsid | sed -e 's/fsid = //')

2.4. 配置 CONTROL PLANE 以使用 RED HAT CEPH STORAGE 集群

您必须将 **OpenStackControlPlane** CR 配置为使用 Red Hat Ceph Storage 集群。配置包括以下任务：

1. 确认 Red Hat Ceph Storage 集群和相关服务具有正确的网络配置。
2. 配置 control plane 以使用 Red Hat Ceph Storage secret。
3. 将镜像服务(glance)配置为使用 Red Hat Ceph Storage 集群。
4. 将块存储服务(cinder)配置为使用 Red Hat Ceph Storage 集群。
5. 可选：配置共享文件系统服务(manila)，以将原生 CephFS 或 CephFS-NFS 与 Red Hat Ceph Storage 集群一起使用。

注意

本例不包括使用 Red Hat Ceph Storage 配置块存储备份服务(**cinder-backup**)。

流程

1. 检查 **NodeNetworkConfigurationPolicy** (**nncp**)自定义资源中定义的存储接口，以确认它具有与 Red Hat Ceph Storage 集群 **public_network** 相同的网络配置。这需要通过 Storage 网络启用对 Red Hat Ceph **Storage** 集群的访问。**Storage** 网络应具有与 Red Hat Ceph Storage 集群的 **public_network** 相同的网络配置。
RHOSO 不需要访问 Red Hat Ceph Storage 集群的 **cluster_network**。

注意

如果不会影响工作负载性能，则存储网络可以使用路由(L3)连接从外部 Red Hat Ceph **Storage** 集群 **public_network** 不同，只要将适当的路由添加到 **Storage** 网络，以访问外部 Red Hat Ceph Storage 集群 **public_network**。

2. 检查 **OpenStackControlPlane** CR 中默认镜像服务的 **networkAttachments**，以确认默认镜像服务已配置为访问 **存储网络** ：

```
glance:
    enabled: true
    template:
      databaseInstance: openstack
      storage:
        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. 确认块存储服务已配置为通过 MetalLB 访问存储网络。

4. **可选：确认共享文件系统服务被配置为通过 ManilaShare 访问 存储网络。**

5. **确认 Compute 服务(nova)已配置为访问 存储网络。**

6. **确认 Red Hat Ceph Storage 配置文件 /etc/ceph/ceph.conf 包含 Red Hat Ceph Storage 集群监控器的 IP 地址。这些 IP 地址必须在 Storage 网络 IP 地址范围中。**

7. **打开 openstack_control_plane.yaml 文件，以编辑 OpenStackControlPlane CR。**

8. **添加 extraMounts 参数，以定义需要访问 Red Hat Ceph Storage secret 的服务。**

以下是为此目的使用 extraMounts 参数的示例。如果您使用共享文件系统服务(manila)，只在传播列表中包括 ManilaShare ：

```
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
```

- **将 <ceph-conf-files > 替换为 创建 Red Hat Ceph Storage secret 时创建的 Secret CR 的名称。**

9. **将 customServiceConfig 参数添加到 glance 模板，将镜像服务配置为使用 Red Hat Ceph Storage 集群：**

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  glance:
    template:
      customServiceConfig: |
        [DEFAULT]
        enabled_backends = default_backend:rbd
        [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
      databaseInstance: openstack
      databaseAccount: glance
      secret: osp-secret
      storage:
        storageRequest: 10G
  extraMounts:
    - name: v1
      region: r1
      extraVol:
        - propagation:
          - GlanceAPI
          extraVolType: Ceph
          volumes:
          - name: ceph
            secret:
              secretName: ceph-conf-files
          mounts:
          - name: ceph
            mountPath: "/etc/ceph"
            readOnly: true
```

当您将 Red Hat Ceph Storage 用作镜像服务的后端时，默认启用 image-conversion。如需更多信息，请参阅 规划 *部署中的规划* 存储和共享文件系统。

10. **将 customServiceConfig 参数添加到 cinder 模板，将块存储服务配置为使用 Red Hat Ceph Storage 集群。有关使用块存储备份的详情，请参考配置块存储备份服务。**

```
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)

使用实际的 FSID 替换。FSID 本身不需要被视为 secret。如需更多信息，请参阅 获取 Red Hat Ceph Storage FSID。

11. 可选：将 customServiceConfig 参数添加到 manila 模板，将共享文件系统服务配置为使用原生 CephFS 或 CephFS-NFS 和 Red Hat Ceph Storage 集群。如需更多信息，请参阅配置共享文件系统服务(manila)。

以下示例公开原生 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
```

以下示例通过 NFS 公开 CephFS：

```
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. **将更新应用到 OpenStackControlPlane CR：**

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

2.5. 配置数据平面以使用 RED HAT CEPH STORAGE 集群

配置数据平面以使用 Red Hat Ceph Storage 集群。

流程

1. **创建一个 ConfigMap，其中包含 nova_compute 容器内计算服务(nova)配置文件 /etc/nova/nova.conf.d/ 的额外内容。此额外内容指示计算服务使用 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) **此文件名必须遵循 swig-< name>-nova.conf 的命名约定。文件由 Compute 服务按字母顺序评估。以 01 开头的文件名将在以 02 开头的文件名之前由 Compute 服务评估。当同一配置选项在多个文件中发生时，最后一个配置选项读取成功。**

(2) **$FSID 值应包含实际的 FSID，如 获取 Ceph FSID 部分所述。FSID 本身不需要被视为 secret。**

2. **创建 default nova 服务的自定义版本，以使用新的 ConfigMap，本例中为 ceph-nova。**

```
apiVersion: dataplane.openstack.org/v1beta1
kind: OpenStackDataPlaneService
metadata:
  name: nova-custom-ceph (1)
spec:
  label: dataplane-deployment-nova-custom-ceph
  dataSources:
   - configMapRef:
       name: ceph-nova
   - secretRef:
       name: nova-cell1-compute-config
   - secretRef:
       name: nova-migration-ssh-key
  playbook: osp.edpm.nova
```

(1) **自定义服务命名为 nova-custom-ceph。它不能命名为 nova，因为 nova 是不可更改的默认服务。任何名称与默认服务名称相同的自定义服务都会在协调过程中被覆盖。**

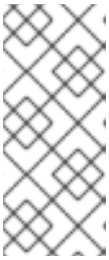
3. **应用 ConfigMap 和自定义服务更改：**

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

4. **更新 OpenStackDataPlaneNodeSet 服务列表，将 nova 服务替换为新的自定义服务（本例中为 nova-custom-ceph），添加 ceph-client 服务，并使用 extraMounts 参数来定义对 Ceph Storage secret 的访问。**

```
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
```



注意

在 libvirt 和 nova-custom-ceph 服务之前，必须添加 ceph-client 服务。ceph-client 服务通过分发 Red Hat Ceph Storage 客户端文件将 EDPM 节点配置为 Red Hat Ceph Storage 服务器的客户端。

5. **保存对 services 列表的更改。**

6. 创建 **OpenStackDataPlaneDeployment CR：**

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

- 将 **<dataplanedeployment_cr_file >** 替换为您的文件的名称。

结果

当 **nova-custom-ceph** 服务 **Ansible** 作业运行时，作业会从 **ConfigMap** 复制到计算服务主机。它还将使用 **virsh secretProducer** 命令，以便 **libvirt** 服务检索由 **FSID** 的 **cephx secret**。

- 作业完成后在 **EDPM** 节点上运行以下命令，以确认作业结果：

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

2.6. 配置外部 CEPH 对象网关后端

您可以通过完成以下高级别任务，将外部 **Ceph** 对象网关(RGW)配置为充当对象存储服务(swift)后端：

1. 配置 **RGW**，以验证 **Identity** 服务(keystone)中的用户及其角色，以使用外部 **RGW** 服务进行身份验证。

2. 部署和配置 **RGW** 服务，以处理对象存储请求。

您可以使用 **openstack** 客户端工具配置对象存储服务。

2.6.1. 配置 RGW 身份验证

您必须配置 **RGW**，以验证 **Identity** 服务(keystone)中的用户及其角色，以便与外部 **RGW** 服务进行身份验证。

先决条件

- 您已部署了可正常工作的 **OpenStack control plane**。

流程

1. **在 control plane 上创建 Object Storage 服务：**

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

2. **创建名为 swift 的用户：**

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

- **将 <swift_password > 替换为要分配给 swift 用户的密码。**

3. **为 swift 用户创建角色：**

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

4. **将 swift 用户添加到系统角色中：**

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

5. **将 RGW 端点 IP 地址导出到变量并创建 control plane 端点：**

```
$ 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;
```

- **将 <rgw_endpoint_ip_address_storage > 替换为存储网络上 RGW 端点的 IP 地址。这是内部服务如何访问 RGW。**

- **将 <rgw_endpoint_ip_address_external > 替换为外部网络上 RGW 端点的 IP 地址。这是云用户如何将对象写入 RGW 的方式。**

注意

两个端点 IP 地址都是代表虚拟 IP 地址（由 haproxy 和 keepalived 所有）的端点，用于访问在 配置和部署 RGW 服务的步骤中将部署在 Red Hat Ceph Storage 集群中的 RGW 后端。

6. 将 swiftoperator 角色添加到 control plane admin 组：

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

2.6.2. 配置和部署 RGW 服务

配置和部署 RGW 服务以处理对象存储请求。

流程

1. 登录到 Red Hat Ceph Storage Controller 节点。

2. 创建名为 /tmp/rgw_spec.yaml 的文件并添加 RGW 部署参数：

```
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>
```

- **将 <host_1& gt ; , < host_ 2> , …, <host_n > 替换为部署 RGW 实例的 Ceph 节点的名称。**
- **将 <storage_network > 替换为用于解析绑定 radosgw 进程的接口的网络范围。**
- **将 <storage_network_vip > 替换为用作 haproxy 前端的虚拟 IP (VIP)。这与配置 RGW 身份验证过程中 Object Storage 服务端点($RGW_ENDPOINT)的配置 相同。**
- **可选：使用外部网络上的额外 VIP 替换 <external_network_vip>，以用作 haproxy 前端。此地址用于从外部网络连接到 RGW。**

3. **保存该文件。**

4. **输入 cephadm shell 并挂载 rgw_spec.yaml 文件。**

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

5. **在集群中添加 RGW 相关配置：**

```
$ 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
```

- 将 **<keystone_endpoint >** 替换为 **Identity** 服务端点。**EDPM** 节点能够解析内部端点，但不能解析公共端点。不要从 **URL** 省略 **URIScheme**，它必须是 **http://** 或 **https://**。

- 将 **< swift_password >** 替换为上一步中分配给 **swift** 用户的密码。

6. 使用 **Orchestrator** 部署 **RGW** 配置：

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

2.7. 为外部 RED HAT CEPH STORAGE 集群使用 TLS 配置 RGW

使用 **TLS** 配置 **RGW**，以便 **control plane** 服务可以解析外部 **Red Hat Ceph Storage** 集群主机名。

此流程将 **Ceph RGW** 配置为模拟 **Object Storage** 服务(**swift**)。它创建一个 **DNS** 区域和证书，以便 **https://rgw-external.ceph.local:8080** 等 **URL** 被注册为 **Identity** 服务(**keystone**)端点。这可让 **Red Hat OpenStack Services on OpenShift (RHOSO)**客户端解析主机并信任证书。

由于 **RHOSO pod** 需要安全地访问由 **Red Hat OpenShift Container Platform (RHOCP)**托管的 **HTTPS** 端点，所以此过程用于为该端点创建 **DNS** 域和证书。

在此过程中，会创建一个 **DNSData** 域，在示例中 **ceph.local**，以便 **pod** 可以为 **RHOCP** 上托管的服务将主机名映射到 **IP** 地址。然后，使用 **CoreDNS** 服务为域配置 **DNS** 转发。最后，使用 **RHOSO** 公共根证书颁发机构创建证书。

您必须将 **RHOCP** 中创建的证书和密钥文件复制到托管 **RGW** 的节点，以便它们可以成为 **Ceph** 编排器 **RGW** 规范的一部分。

流程

1. 为外部 **Ceph** 集群创建 **DNSData** 自定义资源(**CR**)。

注意

创建 DNSData CR 创建一个新的 dnsmasq pod，它可以在关联的 DNSData CR 中读取和解析 DNS 信息。

以下是 DNSData CR 的示例：

```
apiVersion: network.openstack.org/v1beta1
kind: DNSData
metadata:
  labels:
    component: ceph-storage
    service: ceph
  name: ceph-storage
  namespace: openstack
spec:
  dnsDataLabelSelectorValue: dnsdata
  hosts:
    - hostnames:
      - ceph-rgw-internal-vip.ceph.local
      ip: 172.18.0.2
    - hostnames:
      - ceph-rgw-external-vip.ceph.local
      ip: 10.10.10.2
```

注意

在本例中，假设 IP 地址 172.18.0.2 的主机托管 Ceph RGW 端点，以访问私有存储网络。此主机通过 CR，以便创建 DNS A 和 PTR 记录。这可让使用主机名 ceph-rgw-internal-vip.ceph.local 访问主机。

还假设主机位于 IP 地址 10.10.10.2 的主机上，用于外部网络上访问 Ceph RGW 端点。此主机通过 CR，以便创建 DNS A 和 PTR 记录。这可让使用主机名 ceph-rgw-external-vip.ceph.local 访问主机。

本例中的主机列表不是所需主机的确定列表。它提供给演示目的。替换适合您的环境的主机。

2. **将 CR 应用到您的环境：**

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

- 将 <ceph_dns_yaml > 替换为 DNSData CR 文件的名称。

3. 使用向 ceph.local 域的请求的 dnsmasq 更新 CoreDNS CR。有关 DNS 转发的更多信息，请参阅 RHOCP *网络指南中的* 使用 DNS 转发。

4. 列出 openstack domain DNS 集群 IP：

```
$ oc get svc dnsmasq-dns
```

以下是这个命令的输出示例：

```
$ oc get svc dnsmasq-dns
dnsmasq-dns     LoadBalancer   10.217.5.130   192.168.122.80    53:30185/UDP     160m
```

5. 记录命令输出中的转发信息。

6. 列出 CoreDNS CR：

```
$ oc -n openshift-dns describe dns.operator/default
```

7. 编辑 CoreDNS CR，并使用转发信息更新它。

以下是带有转发信息更新的 CoreDNS CR 示例：

```
apiVersion: operator.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: "2024-03-25T02:49:24Z"
  finalizers:
  - dns.operator.openshift.io/dns-controller
  generation: 3
  name: default
  resourceVersion: "164142"
  uid: 860b0e61-a48a-470e-8684-3b23118e6083
spec:
  cache:
    negativeTTL: 0s
    positiveTTL: 0s
  logLevel: Normal
  nodePlacement: {}
```

```
  operatorLogLevel: Normal
  servers:
  - forwardPlugin:
      policy: Random
      upstreams:
      - 10.217.5.130:53
    name: ceph
    zones:
    - ceph.local
  upstreamResolvers:
    policy: Sequential
    upstreams:
    - port: 53
      type: SystemResolvConf
```

以下是添加到 CR 中的内容：

```
....
   servers:
  - forwardPlugin:
      policy: Random
      upstreams:
      - 10.217.5.130:53 (1)
    name: ceph
    zones:
    - ceph.local
....
```

(1)

从 oc get svc dnsmasq-dns 命令记录的转发信息。

8.

使用 DNSData CR 中的主机名创建一个 证书 CR。

以下是 证书 CR 的示例 ：

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: cert-ceph-rgw
  namespace: openstack
spec:
  duration: 43800h0m0s
  issuerRef: {'group': 'cert-manager.io', 'kind': 'Issuer', 'name': 'rootca-public'}
  secretName: cert-ceph-rgw
  dnsNames:
    - ceph-rgw-internal-vip.ceph.local
    - ceph-rgw-external-vip.ceph.local
```

注意

证书 **issuerRef** 设置为 **RHOSO** 的 **root** 证书颁发机构**(CA)**。部署 **control plane** 时会自动创建此 **CA**。**CA** 的默认名称为 **rootca-public**。**RHOSO pod** 信任这个新证书，因为使用了 **root CA**。

9. 将 **CR** 应用到您的环境：

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

- 将 **<ceph_cert_yaml >** 替换为 证书 **CR** 文件的名称。

10. 从应用 证书 **CR** 时创建的 **secret** 中提取证书和密钥数据：

```
$ oc get secret <ceph_cert_secret_name> -o yaml
```

- 将 **<ceph_cert_secret_name >** 替换为 **Certificate CR** 的 **secretName** 字段中使用的名称。

注意

此命令使用类似如下的 **data** 部分输出 **YAML**：

```
[stack@osp-storage-04 ~]$ oc get secret cert-ceph-rgw -o yaml
apiVersion: v1
data:
  ca.crt: <CA>
  tls.crt: <b64cert>
  tls.key: <b64key>
kind: Secret
```

<b64cert> 和 **<b64key>** 值是您在下一步中必须使用的 **base64** 编码证书和密钥字符串。

11. 提取和 **base64** 解码上一步中获取的证书和密钥信息，并将它们的串联保存到 **Ceph** 对象网关服务规格中。

规格文件的 rgw 部分类似如下：

```
service_type: rgw
service_id: rgw
service_name: rgw.rgw
placement:
  hosts:
  - host1
  - host2
networks:
  - 172.18.0.0/24
spec:
  rgw_frontend_port: 8082
  rgw_realm: default
  rgw_zone: default
  ssl: true
  rgw_frontend_ssl_certificate: |
  -----BEGIN CERTIFICATE-----
  MIIDkzCCAfugAwIBAgIRAKNgGd++xV9cBOrwDAeEdQUwDQYJKoZIhvcNAQELBQAw
  <redacted>
  -----BEGIN RSA PRIVATE KEY-----
  MIIEpQIBAAKCAQEAyTL1XRJDcSuaBLpqasAuLsGU2LQdMxuEdw3tE5voKUNnWgjB
  <redacted>
  -----END RSA PRIVATE KEY-----
```

规格文件的 ingress 部分类似如下：

```
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_interface_networks:
  - 172.18.0.0/24
  virtual_ip: 172.18.0.2/24
  ssl_cert: |
  -----BEGIN CERTIFICATE-----
  MIIDkzCCAfugAwIBAgIRAKNgGd++xV9cBOrwDAeEdQUwDQYJKoZIhvcNAQELBQAw
  <redacted>
  -----BEGIN RSA PRIVATE KEY-----
  MIIEpQIBAAKCAQEAyTL1XRJDcSuaBLpqasAuLsGU2LQdMxuEdw3tE5voKUNnWgjB
  <redacted>
  -----END RSA PRIVATE KEY-----
```

在上例中，rgw_frontend_ssl_certificate 和 ssl_cert 包含上一步中的 < b64cert> 和 < b64key> 中的 base64 解码值，两者之间没有空格。

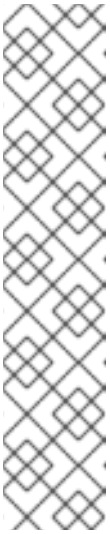
12. 使用服务规格部署 Ceph 对象网关的步骤， 以使用 SSL 部署 Ceph RGW。

13. 连接到 openstackclient pod。

14. 验证转发信息是否已成功更新。

```
$ curl --trace - <host_name>
```

- 将 <host_name > 替换为之前添加到 DNSData CR 的外部主机的名称。



注意

以下是此命令的输出示例，其中 openstackclient pod 成功解析主机名，且不会遇到 SSL 验证错误。

```
sh-5.1$ curl https://rgw-external-vip.ceph.local:8080
<?xml version="1.0" encoding="UTF-8"?><ListAllMyBucketsResult
xmlns="http://s3.amazonaws.com/doc/2006-03-01/"><Owner>
<ID>anonymous</ID><DisplayName></DisplayName></Owner>
<Buckets></Buckets></ListAllMyBucketsResult>
.1$
sh-5.1$
```

2.8. 为带有依赖项的卷或镜像启用延迟删除

当您将 Ceph RBD 用作块存储服务(cinder)或镜像服务(glance)的后端时，您可以在 Ceph RBD 克隆 v2 API 中启用延迟删除。

通过延迟删除，您可以从块存储服务或镜像服务中删除卷，即使 Ceph RBD 卷或快照依赖于它们，例如，块存储服务或计算服务(nova)在不同存储池中创建的 COW 克隆。卷从块存储服务中删除，或者镜像已从镜像服务中删除，但它仍然存储在 Ceph RBD 中用于依赖项的垃圾箱中。只有在没有依赖项时，卷或镜像才会从 Ceph RBD 中删除。

限制：

- 当您在现有环境中启用 Clone v2 延迟删除时，该功能仅适用于新卷或镜像。

流程

1. 验证 Ceph Storage 集群中的客户端正在运行的 Ceph 版本：

```
$ cephadm shell -- ceph osd get-require-min-compat-client
```

输出示例：

```
luminous
```

2. 要将集群设置为使用 Clone v2 API 和延迟删除功能，请将 min-compat-client 设置为 mimic。只有运行 Ceph 版本 13.2.x (Mimic)的集群中的客户端才能访问依赖项的镜像：

```
$ cephadm shell -- ceph osd set-require-min-compat-client mimic
```

3. 使用 m 后缀调度以分钟为单位的 垃圾清除 间隔：

```
$ rbd trash purge schedule add --pool <pool> <30m>
```

- 将 & lt;pool > 替换为关联的存储池的名称，如块存储服务中的卷。
- 将 <30m > 替换为您要为 垃圾清除 指定的间隔（以分钟为单位）。

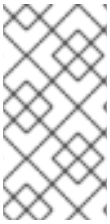
4. 验证已为池设置了垃圾清除调度：

```
$ rbd trash purge schedule list --pool <pool>
```

2.9. RED HAT CEPH STORAGE RBD 集成故障排除

Compute (nova)、块存储(cinder)和镜像(glance)服务可以与红帽 Ceph Storage RBD 集成，以将其用作存储后端。如果此集成无法按预期工作，您可以执行增量故障排除过程来逐渐消除可能的原因。

以下示例演示了如何对镜像服务集成进行故障排除。您可以调整相同的步骤来对 Compute 和块存储服务集成进行故障排除。



注意

如果您在完成此步骤前发现问题的原因，则不需要执行任何后续步骤。您可以退出这个过程并解决问题。

流程

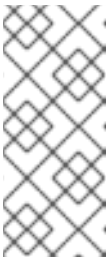
1. 通过评估 **Ready** 条件是否为 **True** 来确定 **control plane** 中的任何部分没有正确部署：

```
$ oc get -n openstack OpenStackControlPlane \
  -o jsonpath="{range .items[0].status.conditions[?(@.status!='True')]}{.type} is {.status} due to {.message}{'\n'}{end}"
```

a. 如果您识别未正确部署的服务，请检查该服务的状态。

以下示例检查 **Compute** 服务的状态：

```
$ oc get -n openstack Nova/nova \
  -o jsonpath="{range .status.conditions[?(@.status!='True')]}{.type} is {.status} due to {.message}{'\n'}{end}"
```

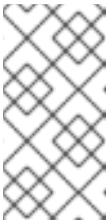


注意

您可以使用 **oc get pods -n openstack** 命令以及特定服务的日志（**oc logs -n openstack < service_pod_name>** 命令检查所有部署的服务的状态。将 **<service_pod_name >** 替换为您要检查的服务 **pod** 的名称。

b. 如果您识别没有正确部署的 **Operator**，请检查 **Operator** 的状态：

```
$ oc get pods -n openstack-operators -lopenstack.org/operator-name
```



注意

使用 **oc logs -n openstack-operators -lopenstack.org/operator-name=<operator_name>** 命令来检查 **Operator** 日志。

2. 检查 **data plane** 部署的 **Status** ：

```
$ oc get -n openstack OpenStackDataPlaneDeployment
```

a. 如果 data plane 部署的 Status 是 False，请检查关联的 Ansible 作业的日志：

```
$ oc logs -n openstack job/<ansible_job_name>
```

将 <ansible_job_name > 替换为关联的作业的名称。作业名称列在 oc get -n openstack OpenStackDataPlaneDeployment 命令的 Message 字段中。

3. 检查 data plane 节点设置部署的 Status ：

```
$ oc get -n openstack OpenStackDataPlaneNodeSet
```

a. 如果 data plane 节点设置部署的 Status 为 False，请检查关联的 Ansible 作业的日志：

```
$ oc logs -n openstack job/<ansible_job_name>
```

- 将 <ansible_job_name > 替换为关联的作业的名称。它列在 oc get -n openstack OpenStackDataPlaneNodeSet 命令的 Message 字段中。

4. 如果任何 pod 处于 CrashLookBackOff 状态，您可以使用 oc debug 命令重复它们以排查目的：

```
oc debug <pod_name>
```

将 <pod_name > 替换为要重复的 pod 的名称。

提示

您还可以在以下对象调试活动中使用 oc debug 命令：

- **要在第一个容器以外的容器中运行 /bin/sh，命令默认行为，使用 oc debug -container <pod_name> <container_name>。这对 pod 很有用，如第一个容器跟踪文件的 API，第二个容器是您要调试的容器。如果使用这个命令表单，您必须首先使用 oc get pods | grep <search_string& gt; 命令来查找容器名称。**

- **要在调试过程中将流量路由到 pod，请使用命令表单 oc debug <pod_name> --keep-labels=true。**

- **要创建任何创建 pod 的资源，如 Deployments、StatefulSets 和 Nodes ，请使用命令形式 oc debug <resource_type>/<resource_name>。创建 StatefulSet 示例为 oc debug StatefulSet/cinder-scheduler。**

5. **连接到 pod，并确认 /etc/ceph 目录中存在 ceph.client.openstack.keyring 和 ceph.conf 文件。**

注意

如果 pod 处于 CrashLookBackOff 状态，请使用 oc debug 命令，如上一步中所述，复制 pod 并将流量路由到它。

```
$ oc rsh <pod_name>
```

- **将 <pod_name > 替换为适用的 pod 的名称。**

提示

如果缺少 Ceph 配置文件，请检查 OpenStackControlPlane CR 中的 extraMounts 参数。

6. **通过从 pod 连接 Ceph Monitor 的 IP 和端口，确认 pod 具有与 Red Hat Ceph Storage 集群的网络连接。IP 和端口信息位于 /etc/ceph.conf 中。**

以下是此过程的示例：

```
$ oc get pods | grep glance | grep external-api-0
glance-06f7a-default-external-api-0                   3/3     Running     0             2d3h
$ oc debug --container glance-api glance-06f7a-default-external-api-0
Starting pod/glance-06f7a-default-external-api-0-debug-p24v9, command was:
/usr/bin/dumb-init --single-child -- /bin/bash -c /usr/local/bin/kolla_set_configs &&
/usr/local/bin/kolla_start
Pod IP: 192.168.25.50
If you don't see a command prompt, try pressing enter.
sh-5.1# cat /etc/ceph/ceph.conf
# Ansible managed

[global]

fsid = 63bdd226-fbe6-5f31-956e-7028e99f1ee1
mon host = [v2:192.168.122.100:3300/0,v1:192.168.122.100:6789/0],
[v2:192.168.122.102:3300/0,v1:192.168.122.102:6789/0],
[v2:192.168.122.101:3300/0,v1:192.168.122.101:6789/0]


[client.libvirt]
admin socket = /var/run/ceph/$cluster-$type.$id.$pid.$cctid.asok
log file = /var/log/ceph/qemu-guest-$pid.log

sh-5.1# python3
Python 3.9.19 (main, Jul 18 2024, 00:00:00)
[GCC 11.4.1 20231218 (Red Hat 11.4.1-3)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import socket
>>> s = socket.socket()
>>> ip="192.168.122.100"
>>> port=3300
>>> s.connect((ip,port))
>>>
```

提示

如果无法连接到 Ceph 监控器，请对集群和 pod 之间的网络连接进行故障排除。上例使用 Python 套接字从 ceph.conf 文件连接到 Red Hat Ceph Storage 集群的 IP 和端口。

执行 s.connect （(ip,port) 函数时有两个潜在的结果：

- **如果命令成功执行，且没有类似以下示例的错误，pod 和集群间的网络连接可以正常工作。如果连接正常工作，则命令执行将根本不提供返回值。**
- **如果命令需要很长时间才能执行并返回类似以下示例的错误，则 pod 和集群间的网络连接无法正常工作。应进一步调查该连接以对连接进行故障排除。**

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TimeoutError: [Errno 110] Connection timed out
```

7. **如以下示例所示，检查 cephx 密钥：**

```
bash-5.1$ cat /etc/ceph/ceph.client.openstack.keyring
[client.openstack]
   key = "<redacted>"c
   caps mgr = allow *
   caps mon = profile rbd
   caps osd = profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=backups, profile
rbd pool=images
bash-5.1$
```

8. **从 caps osd 参数列出池的上下文，如下例所示：**

```
$ /usr/bin/rbd --conf /etc/ceph/ceph.conf \
--keyring /etc/ceph/ceph.client.openstack.keyring \
--cluster ceph --id openstack | wc -l
```

提示

如果这个命令返回数字 0 或更高版本，cephx 密钥提供足够的权限来连接，并从中读取信息，即 Red Hat Ceph Storage 集群。

如果此命令未完成，但已确认至集群的网络连接，请与 Ceph 管理员合作来获取正确的 cephx 密钥环。

另外，存储网络上可能存在 MTU 不匹配。如果网络使用巨型帧(MTU 值为 9000)，则必须使用接口的服务器之间的所有交换机端口都必须更新，以支持巨型帧。如果交换机没有进行此更改，则在 Ceph 应用层可能会出现问题。使用网络验证所有主机都可以通过所需的 MTU 进行通信，例如 ping -M do -s 8972 <ip_address>。

9. **将测试数据发送到 Ceph 集群上的 images 池。**

以下是执行此任务的示例：

```
# DATA=$(date | md5sum | cut -c-12)
# POOL=images
# RBD="/usr/bin/rbd --conf /etc/ceph/ceph.conf --keyring
/etc/ceph/ceph.client.openstack.keyring --cluster ceph --id openstack"
# $RBD create --size 1024 $POOL/$DATA
```

提示

可以从集群中读取数据，但没有权限向它写入数据，即使 cephx 密钥环中授予了写入权限。如果已授予写入权限，但您无法向集群写入数据，这可能表示集群过载且无法写入新数据。

在示例中，rbd 命令没有成功完成，并已取消。因此，它会确认集群本身没有用于写入新数据的资源。这个问题已在集群本身中解决。客户端配置没有不正确的。

2.10. 自定义和管理 RED HAT CEPH STORAGE

Red Hat OpenStack Services on OpenShift (RHOSO) 18.0 支持 Red Hat Ceph Storage 7。有关自定义和管理 Red Hat Ceph Storage 7 的信息，请参阅 Red Hat Ceph Storage 文档。以下指南包含这些任务的关键信息和程序：

- 管理指南
- 配置指南
- 操作指南
- 数据安全性和强化指南
- 仪表板指南
- 故障排除指南

第 3 章 配置块存储服务(CINDER)

块存储服务(cinder)通过卷提供对远程块存储设备的访问，以提供持久存储。块存储服务具有三个强制服务： api、scheduler 和 volume ；另一个可选服务，备份。

所有块存储服务都使用 OpenStackControlPlane 自定义资源(CR)的 cinder 部分进行配置：

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

全局配置选项直接应用于 cinder 和 template 部分。特定于服务的配置选项会出现在其关联的部分下。以下示例演示了应用块存储服务配置的所有部分，以及每个部分中应用了哪些配置类型：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    <global-options>
    template:
      <global-options>
      cinderAPI:
        <cinder-api-options>
      cinderScheduler:
        <cinder-scheduler-options>
      cinderVolumes:
        <name1>: <cinder-volume-options>
        <name2>: <cinder-volume-options>
      cinderBackup:
        <cinder-backup-options>
```

3.1. 术语

以下术语对于了解块存储服务(cinder)非常重要：

- 存储后端：存储卷数据的物理存储系统。
- Cinder 驱动程序：块存储服务的一部分，支持与存储后端通信。It 使用 volume_driver 和

backup_driver 选项配置。

- Cinder 后端：使用其配置对 cinder 驱动程序进行分组的逻辑表示。此分组用于管理和解决特定存储后端中存在的卷。此逻辑结构的名称使用 volume_backend_name 选项进行配置。
- 存储池：给定存储后端中的卷的逻辑分组。
- Cinder 池：存储池的块存储服务中的表示。
- 卷主机：块存储服务地址卷的方式。有两个不同的表示： short (<hostname>@<backend-name>)和 full (<hostname>@<backend-name>""<pool-name>)。
- quota ：每个项目定义的限值，以限制使用块存储特定资源。

3.2. RED HAT OPENSTACK SERVICES ON OPENSHIFT (RHOSO)中的块存储服务(CINDER)的改进.

以下功能改进已集成到块存储服务中：

- 为多个卷后端轻松部署。
- 后端部署不会影响正在运行的卷后端。
- 后端添加和删除不会影响运行的后端。
- 后端配置更改不会影响其他正在运行的后端。
- 每个后端都可以使用自己的特定供应商容器镜像。构建包含两个驱动程序依赖关系的自定义镜像不再需要。
- Pacemaker 已被 Red Hat OpenShift Container Platform (RHOCP)功能替代。

- 改进了对服务代码进行故障排除的方法。

3.3. 配置传输协议

部署使用不同的传输协议来连接卷。Block Storage 服务(cinder)支持以下传输协议：

- iSCSI
- 光纤频道（FC）
- NVMe over TCP (NVMe-TCP)
- NFS
- Red Hat Ceph Storage RBD

使用卷（如块存储服务(cinder)卷和备份服务）的 control plane 服务可能需要支持 Red Hat OpenShift Container Platform (RHOCP)集群来使用 iscsid 和 multipathd 模块，具体取决于使用的存储阵列。 这些模块必须在执行这些卷的服务的所有节点上可用。要使用这些传输协议，请创建一个 MachineConfig CR 来定义这些模块的执行位置。如需有关 MachineConfig 的更多信息，请参阅了解 Machine Config operator。

重要

使用 MachineConfig 更改节点的配置会导致节点重新引导。在应用 'MachineConfig' 以确保 RHOCP 工作负载的完整性之前，请咨询您的 RHOCP 管理员。

本节中的步骤旨在指导这些协议的一般配置。存储后端供应商将提供有关如何连接到其特定解决方案的配置信息。

除了特定于协议的配置外，红帽建议配置多路径，而不考虑使用的传输协议。完成传输协议配置后，请参阅为流程 配置多路径。

注意

这些服务会在 **EDPM** 节点上自动启动。

3.3.1. 配置 iSCSI 协议

从 **RHOCP** 节点连接到 **iSCSI** 卷需要 **iSCSI** 启动器服务。**iscsid service** 模块必须有一个实例用于正常 **RHOCP** 使用、**OpenShift CSI** 插件用法和 **RHOSO** 服务。将 **MachineConfig** 应用到适用的节点，以配置节点以使用 **iSCSI** 协议。

注意

如果 **iscsid** 服务模块已在运行，则不需要这个过程。

流程

1. 创建 **MachineConfig CR**，为 **iscsid** 模块配置节点。

   以下示例在所有 **RHOCP worker** 节点上启动带有默认配置的 **iscsid** 服务：

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
    service: cinder
  name: 99-worker-cinder-enable-iscsid
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
      - enabled: true
        name: iscsid.service
```

2. 保存该文件。

3. 应用 **MachineConfig CR** 文件。

```
$ oc apply -f <machine_config_file> -n openstack
```

- 将 <machine_config_file > 替换为 MachineConfig CR 文件的名称。

3.3.2. 配置光纤通道协议

不需要额外的节点配置来使用 Fibre Channel 协议连接到卷。虽然所有使用 Fibre Channel 的节点都有一个主机总线适配器(HBA)卡。除非 RHOCP 部署中的所有 worker 节点都有一个 HBA 卡，否则您必须在 control plane 配置中使用 nodeSelector 来选择要卷和备份服务的节点，以及使用块存储服务为其存储后端的镜像服务实例。

3.3.3. 配置 NVMe over TCP (NVMe-TCP)协议

从 RHOCP 节点连接到 NVMe-TCP 卷需要 nvme 内核模块。

流程

1. 创建 MachineConfig CR，为 nvme 内核模块配置节点。

   以下示例使用所有 RHOCP worker 节点中的默认配置启动 nvme 内核模块：

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
    service: cinder
  name: 99-worker-cinder-load-nvme-fabrics
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - path: /etc/modules-load.d/nvme_fabrics.conf
          overwrite: false
          mode: 420
          user:
            name: root
          group:
            name: root
          contents:
            source: data:,nvme-fabrics%0Anvme-tcp
```

2. 保存该文件。

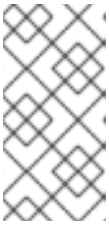
3. 应用 **MachineConfig CR** 文件。

```
$ oc apply -f <machine_config_file> -n openstack
```

- 将 **<machine_config_file >** 替换为 **MachineConfig CR** 文件的名称。

4. 节点重启后，验证 **nvme-fabrics** 模块是否已加载并支持主机上的 **ANA** ：

```
cat /sys/module/nvme_core/parameters/multipath
```



注意

虽然 **ANA** 不使用 **Linux** 多路径设备映射器，但 **multipathd** 必须正在运行，以便 **Compute** 节点能够在将卷连接到实例时使用多路径。

3.3.4. 配置多路径

在 **RHOCP** 节点上配置多路径需要 **MachineConfig CR**，用于在节点上创建一个 **multipath.conf** 文件并启动该服务。

注意

此流程中提供的示例只创建最小的 **multipath.conf** 文件。生产部署可能要求根据您的环境添加特定的硬件供应商。请咨询适当的系统管理员以获取部署所需的任何值。

流程

1. 创建 **MachineConfig CR** 来配置节点多路径。

以下示例创建了 **multipath.conf** 文件，并在所有 **RHOCP** 节点上启动 **multipathd** 模块：

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
    service: cinder
  name: 99-worker-cinder-enable-multipathd
```

```
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - path: /etc/multipath.conf
          overwrite: false
          mode: 384
          user:
            name: root
          group:
            name: root
          contents:
            source:
data:,defaults%20%7B%0A%20%20user_friendly_names%20no%0A%20%20recheck_wwid%
20yes%0A%20%20skip_kpartx%20yes%0A%20%20find_multipaths%20yes%0A%7D%0A%0
Ablacklist%20%7B%0A%7D
    systemd:
      units:
      - enabled: true
        name: multipathd.service
```

注意

以下是本例创建的 multipath.conf 的内容：

```
defaults {
  user_friendly_names no
  recheck_wwid yes
  skip_kpartx yes
  find_multipaths yes
}

blacklist {
}
```

2. **保存该文件。**

3. **应用 MachineConfig CR 文件。**

```
$ oc apply -f <machine_config_file> -n openstack
```

- **将 <machine_config_file > 替换为 MachineConfig CR 文件的名称。**

注意

在 RHOSO 部署中，默认启用 use_multipath_for_image_xfer 配置选项。

3.4. 配置初始默认值

块存储服务(cinder)有一组初始默认值，应在服务首次启用时进行配置。它们必须在主 customServiceConfig 部分中定义。部署后，会使用 openstack 客户端修改这些初始默认值。

流程

1. 打开 OpenStackControlPlane CR 文件，openstack_control_plane.yaml。

2. 编辑 CR 文件并添加块存储服务全局配置。

   以下示例演示了块存储服务初始配置：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    enabled: true
    template:
      customServiceConfig: |
        [DEFAULT]
        quota_volumes = 20
        quota_snapshots = 15
```

   有关所有初始默认参数的完整列表，请参阅 Initial default parameters。

3. 更新 control plane：

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

4. 等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

3.4.1. 初始默认参数

在服务首次启用时，应配置这些初始默认参数。

参数	描述
default_volume_type	为所有用户提供默认卷类型。任何非默认值的默认值类型不会被自动创建。默认值为 **__DEFAULT__**。
no_snapshot_gb_quota	提供一个值来确定快照计数是否针对 GB 配额。默认值为 **false**。
per_volume_size_limit	以 GB 为单位提供每个卷的最大大小。默认值为 **1000**。
quota_volumes	提供每个项目允许的卷数量。默认值为 **10**。
quota_snapshots	提供每个项目允许的快照数量。默认值为 **10**。
quota_consistencygroups	提供每个项目允许的快照数量。默认值为 **10**。
quota_groups	提供每个项目允许的编号组。默认值为 **10**。
quota_gigabytes	以 GB 为单位为每个项目提供卷和快照允许的存储总量。默认值为 **1000**。
quota_backups	提供每个项目允许的数字备份。默认值为 **10**。
quota_backup_gigabytes	提供允许每个项目的备份的总存储量（以 GB 为单位）。默认值为 **1000**。

3.5. 配置 API 服务

块存储服务(cinder)为用户和其他 RHOSO 服务提供与服务的所有外部交互的 API 接口。

流程

1. 打开 **OpenStackControlPlane CR** 文件，**openstack_control_plane.yaml**。

2. 编辑 **CR** 文件并添加内部 **Red Hat OpenShift Container Platform (RHOCP)**负载均衡器的配置。

   以下示例演示了负载均衡器配置：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      cinderAPI:
        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
```

3. 编辑 **CR** 文件，并添加 **API** 服务副本数量的配置。红帽建议在具有三个副本的 **Active-Active** 配置中运行 **cinderAPI** 服务。

   以下示例演示了将 **cinderAPI** 服务配置为使用三个副本：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      cinderAPI:
        replicas: 3
```

4. 编辑 **CR** 文件并配置 **cinderAPI** 选项。这些选项在 **cinderAPI** 部分下的 **customServiceConfig** 部分中配置。

以下示例演示了配置 cinderAPI 服务选项并在所有服务上启用调试：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      customServiceConfig: |
        [DEFAULT]
        debug = true
      cinderAPI:
        customServiceConfig: |
          [DEFAULT]
          osapi_volume_workers = 3
```

有关常用 cinderAPI 服务选项参数的列表，请参阅 API 服务选项参数。

5. 保存该文件。

6. 更新 control plane：

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

7. 等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

3.5.1. API 服务选项参数

提供了 API 服务选项参数，用于配置块存储服务的 cinderAPI 部分。

参数	描述
api_rate_limit	提供用于确定是否启用 API 速率限制的值。默认值为 **false**。
debug	提供一个值，来确定日志级别是否设置为 **DEBUG**，而不是默认的 **INFO**。默认值为 **false**。可以在不重启的情况下动态设置日志记录级别。
osapi_max_limit	为集合资源在单个响应中返回的最大项目数提供一个值。默认值为 **1000**。
osapi_volume_workers	为分配给 API 组件的 worker 数量提供一个值。默认为可用的 CPU 数量。

3.6. 配置调度程序服务

块存储服务(cinder)有一个调度程序服务(cinderScheduler)，它负责做出决策，例如选择哪个后端接收新卷，无论是否有足够的可用空间来执行操作，确定现有卷应移到某些特定操作的位置。

红帽建议只使用单个 cinderScheduler 实例来调度一致性并简化故障排除。虽然 cinderScheduler 可以使用多个实例运行，但建议使用服务默认 副本：1。

流程

1. **打开 OpenStackControlPlane CR 文件，openstack_control_plane.yaml。**

2. **编辑 CR 文件，并添加服务下线检测超时的配置。**

   以下示例演示了此配置：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      customServiceConfig: |
        [DEFAULT]
        report_interval = 20 1
        service_down_time = 120 2
```

1. **块存储服务组件之间的秒数，以心跳形式通过数据库报告操作状态。默认值为 10。**

2

从组件的最后一个心跳以来，其被视为无法正常工作的最大秒数。默认值为 60。

注意

红帽建议在 CR 的 cinder 级别配置这些值，而不是 cinderScheduler，以便这些值一致地应用到所有组件。

3. 编辑 CR 文件并添加统计报告间隔的配置。

以下示例演示了在 cinder 级别配置这些值以将其全局应用到所有服务：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      customServiceConfig: |
        [DEFAULT]
        backend_stats_polling_interval = 120 1
        backup_driver_stats_polling_interval = 120 2
```

1

从卷对后端用量统计之间的请求间隔的秒数。默认值为 60

2

从卷对备份服务的用量统计之间的请求间隔秒数。默认值为 60。

以下示例演示了在 cinderVolume 和 cinderBackup 级别上配置这些值，以便在服务级别自定义设置。

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

```
cinderBackup:
  customServiceConfig: |
    [DEFAULT]
    backup_driver_stats_polling_interval = 120 (1)
    < rest of the config >
cinderVolumes:
  nfs:
    customServiceConfig: |
      [DEFAULT]
      backend_stats_polling_interval = 120 (2)
```

(1) 从卷对后端用量统计之间的请求间隔的秒数。默认值为 60

(2) 从卷对备份服务的用量统计之间的请求间隔秒数。默认值为 60。

注意

生成使用量统计可以是某些后端的资源密集型。设置这些值太低可能会影响后端性能。您可能需要调整这些设置的配置，以更好地适合个别后端。

4. 执行自定义 **cinderScheduler** 服务所需的任何其他配置。

   有关自定义 **cinderScheduler** 服务的更多配置选项，请参阅 调度程序服务参数。

5. 保存该文件。

6. 更新 **control plane**：

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

7. 等待 **RHOCP** 创建与 **OpenStackControlPlane CR** 相关的资源。运行以下命令来检查状态：

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

3.6.1. 调度程序服务参数

为配置块存储服务的 cinderScheduler 部分提供了调度程序服务参数

参数	描述
debug	为日志记录级别提供设置。当此参数为 true 时，日志级别设置为 DEBUG，而不是 INFO。默认值为 false。
scheduler_max_attempts	提供调度卷的最大尝试次数的设置。默认值为 3
scheduler_default_filters	提供一个设置，用于过滤在请求中未指定时用于过滤主机的类名称。这是一个以逗号分隔的列表。默认为 AvailabilityZoneFilter,CapacityFilter,CapabilitiesFilter。
scheduler_default_weighers	为用于权衡主机的 weigher 类名称提供设置。这是一个以逗号分隔的列表。默认为 CapacityWeigher。
scheduler_weight_handler	为处理程序提供设置，以便在权衡后选择主机或池。值 cinder.scheduler.weights.OrderedHostWeightHandler 从传递过滤的主机列表中选择第一个主机，以及值 cinder.scheduler.weights.stochastic.stochasticHostWeightHandler，每个池都会有几率与每个池权重成比例。默认为 cinder.scheduler.weights.OrderedHostWeightHandler。

以下是参数表中的过滤器类名称的说明：

- AvailabilityZoneFilter
  - 过滤掉所有不符合所请求卷的可用区要求的后端。

- **CapacityFilter**
  - 仅选择具有足够空间的后端来容纳卷。
- **CapabilitiesFilter**
  - 仅选择可以支持卷中任何指定设置的后端。
- **InstanceLocality**
  - 将集群配置为使用到同一节点的 volumes local。

3.7. 配置卷服务

块存储服务(cinder)有一个卷服务(cinderVolumes 部分)，它负责管理与卷、快照和组相关的操作。这些操作包括创建、删除和克隆卷，以及创建快照。

此服务需要访问 OpenStackControlPlane CR 的 networkAttachments 中的存储后端（存储）和存储管理(storageMgmt)网络。某些操作（如创建空卷或快照）不需要卷服务和存储后端之间的任何数据移动。然而，其他操作（如将数据从一个存储后端迁移到另一个存储后端）都要求数据通过卷服务进行访问。

卷服务配置在 cinderVolumes 部分中执行，其参数在 customServiceConfig , customServiceConfig Secrets,networkAttachments,replicas, 和 nodeSelector 部分中设置。

卷服务不能有多个副本。

流程

1. 打开 OpenStackControlPlane CR 文件，openstack_control_plane.yaml。
2. 编辑 CR 文件并添加后端配置。

以下示例演示了 Red Hat Ceph Storage 后端的服务配置：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      customServiceConfig: |
        [DEFAULT]
        debug = true
      cinderVolumes:
        ceph: (1)
          networkAttachments: (2)
          - storage
          customServiceConfig: |
            [ceph]
            volume_backend_name = ceph (3)
            volume_driver = cinder.volume.drivers.rbd.RBDDriver (4)
```

(1) 单个后端的配置区域。每个唯一的后端都需要单独的配置区域。默认不会部署后端。块存储服务卷服务将不会运行，除非部署期间至少配置了一个后端。有关配置后端的更多信息，请参阅 块存储服务(cinder)后端和 多个块存储服务(cinder)后端。

(2) 后端网络连接的配置区域。

(3) 分配给此后端的名称。

(4) 用于连接到此后端的驱动程序。

有关常用卷服务参数的列表，请参阅 卷服务参数。

3. 保存该文件。

4. **更新 control plane：**

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

5. **等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：**

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

3.7.1. 卷服务参数

提供了卷服务参数，用于配置块存储服务的 cinderVolumes 部分

参数	描述
backend_availability_zone	为后端可用区提供设置。这是在 **[DEFAULT]** 部分中设置的。默认值为 **storage_availability_zone**。
volume_backend_name	为给定驱动程序实施的后端名称提供设置。没有默认值。
volume_driver	提供用于创建卷的驱动程序的设置。它以特定类的 Python 命名空间的形式提供。没有默认值。
enabled_backends	为要使用的后端名称列表提供设置。这些后端名称应该由唯一 [CONFIG] 组及其选项来支持。这是以逗号分隔的值列表。默认值为带有 **volume_backend_name** 选项的部分名称。
image_conversion_dir	为在镜像转换过程中用于临时存储的目录提供设置。默认值为 **/var/lib/cinder/conversion**。
backend_stats_polling_interval	提供设置，用于卷从存储后端发出用量统计之间的秒数。默认值为 **60**。

3.7.2. 块存储服务(cinder)后端

每个块存储服务后端都应该在 cinderVolumes 部分中有一个单独的配置部分。这样可确保每个后端在专用 pod 中运行。这个方法有以下优点：

- 增加隔离。
- 添加和删除后端的速度会很快，不会影响其他正在运行的后端。
- 配置更改不会影响其他正在运行的后端。
- 自动将卷 pod 分散到不同的节点上。

每个块存储服务后端使用存储传输协议来访问卷中的数据。每个存储传输协议都有单独的要求，如 配置传输协议 中所述。存储协议信息还应在单独的供应商安装指南中提供。

注意

红帽建议使用独立 pod 配置每个后端。在基于 director 的 RHOSP 版本中，所有后端都在单个 cinder-volume 容器中运行。这不再是推荐的做法。

默认不会部署后端。块存储服务卷服务将不会运行，除非部署期间至少配置了一个后端。

所有存储供应商都提供一个安装指南，其中包含用于供应商驱动程序的最佳实践、部署配置和配置选项。这些安装指南提供了为部署正确配置卷服务所需的特定配置信息。红帽生态系统目录 中提供了安装指南。

有关集成和认证供应商驱动程序的更多信息，请参阅 *集成合作伙伴内容*。

如需有关 Red Hat Ceph Storage 后端配置的信息，请参阅 集成 Red Hat Ceph Storage 和 部署超融合基础架构环境。

有关配置通用（非供应商特定）NFS 后端的详情，请参考 配置通用 NFS 后端。

注意

红帽不推荐在生产环境中使用通用 NFS 驱动程序。

3.7.3. 多个块存储服务(cinder)后端

通过在 cinderVolumes 配置部分中添加多个独立条目来部署多个块存储服务后端。每个后端都在独立的 pod 中运行。

以下配置示例部署两个独立后端；一个用于 iSCSI，另一个用于 NFS：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      cinderVolumes:
        nfs:
          networkAttachments:
          - storage
          customServiceConfigSecrets:
          - cinder-volume-nfs-secrets
          customServiceConfig: |
         [nfs]
         volume_backend_name=nfs
        iSCSI:
          networkAttachments:
          - storage
          - storageMgmt
          customServiceConfig: |
         [iscsi]
         volume_backend_name=iscsi
```

3.8. 配置后端可用区

为卷服务后端配置后端可用区(AZ)和备份服务，以对用户进行分组云基础架构服务。AZ 映射到故障域和计算资源，以实现高可用性、容错和资源调度。

例如，您可以创建一个具有特定硬件的 AZ 节点，用户可以在创建需要该硬件的实例时选择这些硬件。

注意

部署后，AZ 使用 RESKEY:availability_zones 卷类型额外规格创建。

只要卷类型没有限制 AZ，用户可以直接在 AZ 中创建卷。

流程

1. **打开 OpenStackControlPlane CR 文件，openstack_control_plane.yaml。**

2. **编辑 CR 文件并添加 AZ 配置。**

   以下示例演示了一个 AZ 配置：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
name: openstack
spec:
  cinder:
    template:
      cinderVolumes:
        nfs:
          networkAttachments:
          - storage
          - storageMgmt
          customServiceConfigSecrets:
          - cinder-volume-nfs-secrets
          customServiceConfig: |
              [nfs]
              volume_backend_name=nfs
              backend_availability_zone=zone1 (1)
        iSCSI:
          networkAttachments:
          - storage
          - storageMgmt
          customServiceConfig: |
              [iscsi]
              volume_backend_name=iscsi
              backend_availability_zone=zone2
```

(1) **与后端关联的可用区。**

3. 保存该文件。

4. 更新 control plane：

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

5. 等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

3.9. 配置通用 NFS 后端

块存储服务(cinder)可以配置通用 NFS 后端，以便为卷和备份提供替代的存储解决方案。

块存储服务支持通用 NFS 解决方案，请考虑以下事项：

- 红帽建议使用经过认证的存储后端和驱动程序。红帽不推荐在生产环境中使用来自通用 NFS 后端的 NFS 存储。与认证的存储后端和驱动程序相比，通用 NFS 后端的功能有限。例如，通用 NFS 后端不支持卷加密和卷多附加等功能。有关支持的驱动程序的详情，请查看 红帽生态系统目录。

- 对于 Block Storage (cinder)和 Compute (nova)服务，您必须使用 NFS 版本 4.0 或更高版本。RHOSO 不支持早期版本的 NFS。

- RHOSO 不支持 NetApp NAS 安全功能。它干扰正常的卷操作。这个功能必须使用以下参数在特定后端配置中的 customServiceConfig 中禁用：

```
nas_secure_file_operation=false
nas_secure_file_permissions=false
```

- 不要配置 nfs_mount_options 选项。默认值为 RHOSO 环境的最佳 NFS 选项。如果您在配置多个服务以共享同一 NFS 服务器时遇到问题，请联系红帽支持团队。

流程

1. 创建 Secret CR 来存储卷连接信息。

   以下是 Secret CR 的示例：

```
apiVersion: v1
kind: Secret
metadata:
  name: cinder-volume-nfs-secrets (1)
type: Opaque
stringData:
  cinder-volume-nfs-secrets: |
 [nfs]
 nas_host=192.168.130.1
 nas_share_path=/var/nfs/cinder
```

   (1) 在 cinderVolumes 后端配置中包括它时使用的名称。

2. 保存该文件。

3. 更新 control plane：

```
$ oc apply -f <secret_file_name> -n openstack
```

   - 将 <secret_file_name> 替换为包含 Secret CR 的文件的名称。

4. 打开 OpenStackControlPlane CR 文件，openstack_control_plane.yaml。

5. 编辑 CR 文件并添加通用 NFS 后端的配置。

以下示例演示了此配置：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      cinderVolumes:
        nfs:
          networkAttachments: 1
          - 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:
          - cinder-volume-nfs-secrets 2
```

1

storageMgmt 网络没有列出，因为通用 NFS 没有管理界面。

2

来自 Secret CR 的名称。

注意

如果您要配置多个通用 NFS 后端，请确保每个配置都处于单独的配置部分中，以便每个 pod 都被投票到每个后端。

6. **保存该文件。**

7. **更新 control plane：**

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

8. 等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

3.10. 配置 NFS 转换目录

当块存储服务(cinder)执行镜像格式转换时，空间有限，对大型镜像服务(glance)镜像的转换可能会导致完全使用节点根磁盘空间。您可以使用外部 NFS 共享进行转换，以防止完全填写节点上的空间。

流程

1. 打开 OpenStackControlPlane CR 文件，openstack_control_plane.yaml。

2. 编辑 CR 文件并添加转换目录的配置。

以下示例演示了转换目录配置：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  extraMounts:
    extraVol:
      - propagation:
        - CinderVolume
        volumes:
        - name: cinder-conversion
          nfs:
            path: <nfs_share_path> 1
            server: <nfs_server> 2
        mounts:
        - name: cinder-conversion
          mountPath: /var/lib/cinder/conversion
          readOnly: true
```

1

2 提供转换目录的服务器的 IP 地址。

注意

提供的示例演示了如何创建供所有卷服务 pod 使用的通用转换目录。

也可以为每个卷服务 pod 定义转换目录。为此，请按上面所示的 extraMount 定义 每个转换目录，但在 OpenStackControlPlane CR 文件的 cinder 部分中。然后，您要将 传播 值设置为特定卷部分的名称，而不是 CinderVolume。

3. 保存该文件。

4. 更新 control plane：

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

5. 等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

3.11. 配置自动数据库清理

块存储服务(cinder)执行数据库条目的软提取。这意味着数据库条目标记为删除，但实际上不会从数据库中删除。这允许审核已删除的资源。

这些标记为删除的数据库行将无限增长，并在未清除时消耗资源。RHOSO 会在设置天数后自动清除标记为删除的数据库条目。默认情况下，在 30 天后标记为要删除的记录会被清除。您可以为清除作业配置不同的记录年龄和调度。

流程

1. 打开 openstack_control_plane.yaml 文件，以编辑 OpenStackControlPlane CR。

2. 将 dbPurge 参数添加到 cinder 模板，以根据您要配置的服务配置数据库清理。

   以下是使用 dbPurge 参数配置块存储服务的示例：

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

   1. 在清除记录前，记录标记为删除的天数。默认值为 30。最小值为 1。

   2. 以 crontab 格式运行作业的时间计划。默认值为 1 0 * *。此默认值等同于每日 00:01。

3. 更新 control plane：

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

3.12. 保留作业

块存储服务(cinder)需要自动运行的维护操作。有些操作是一次性的，有些是定期的。这些操作使用 OpenShift 作业运行。

如果作业及其 pod 在完成后自动删除，则无法检查这些操作的日志。但是，您可以使用 OpenStackControlPlane CR 中的 preserveJob 字段停止自动删除作业并保留它们。

Example:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      preserveJobs: true
```

3.13. 解决主机名冲突

块存储服务(cinder)中的大多数存储后端都需要连接到它们的主机具有唯一的主机名。这些主机名用于识别权限和地址，如 iSCSI 启动器名称、HBA 和 WWPN。

由于您在 OpenShift 中部署，块存储服务卷和备份报告的主机名不是 OpenShift 主机名，而是改为 pod 名称。

这些 pod 名称使用预先确定的模板组成：* 对于 volumes: cinder-volume-<backend_key>-0 * 用于备份： cinder-backup-<replica-number>

如果您在多个部署中使用相同的存储后端，则可能无法遵守唯一的主机名要求，从而导致操作问题。要解决这个问题，您可以使用 uniquePodNames 字段请求安装程序具有唯一的 pod 名称，因此唯一的主机名。

当您将 uniquePodNames 字段设置为 true 时，会将一个简短哈希添加到 pod 名称中，用于处理主机名冲突。

Example:

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

```
spec:
  cinder:
    uniquePodNames: true
```

3.14. 使用其他容器镜像

Red Hat OpenStack Services on OpenShift (RHOSO)服务使用特定发行版本和版本的容器镜像进行部署。有时，部署需要为该发行版本和版本生成的容器镜像以外的容器镜像。常见原因包括：

- 部署热修复.
- 使用经过认证的、供应商提供的容器镜像。

安装程序使用的容器镜像通过 OpenStackVersion CR 控制。在部署服务期间，openstack operator 会自动创建一个 OpenStackVersion CR。或者，也可以在 OpenStackControlPlane CR 的应用程序之前手动创建，但安装了 openstack operator 后。这允许容器镜像为任何服务和组件单独指定。

这种设计的粒度取决于服务。例如，在块存储服务(cinder)中，所有 cinderAPI、cinderScheduler 和 cinderBackup pod 必须具有相同的镜像。但是，对于卷服务，会为每个 cinderVolumes 定义容器镜像。

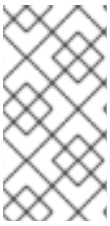
以下示例演示了具有两个后端的 OpenStackControlPlane 配置：一个名为 ceph，另一个名为 custom-fc。custom-fc 后端需要一个经过认证的、供应商提供的容器镜像。此外，我们必须将其他服务镜像配置为使用热修补代码中的非标准镜像。

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  cinder:
    template:
      cinderVolumes:
        ceph:
          networkAttachments:
          - storage
< . . . >
        custom-fc:
          networkAttachments:
          - storage
```

以下示例演示了我们的 OpenStackVersion CR 可能类似于正确设置容器镜像。

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackVersion
metadata:
  name: openstack
spec:
  customContainerImages:
    cinderAPIImages: <custom-api-image>
    cinderBackupImages: <custom-backup-image>
    cinderSchedulerImages: <custom-scheduler-image>
    cinderVolumeImages:
      custom-fc: <vendor-volume-volume-image>
```

- **将 <custom-api-image > 替换为要使用的 API 服务镜像的名称。**
- **将 <custom-backup-image > 替换为要使用的 Backup 服务镜像的名称。**
- **将 <custom-scheduler-image > 替换为要使用的调度程序服务镜像的名称。**
- **将 <vendor-volume-volume-image > 替换为要使用的认证、厂商提供的镜像的名称。**



注意

OpenStackVersion CR 中的 name 属性必须与 OpenStackControlPlane CR 中的相同属性匹配。

第 4 章 配置块存储备份服务

Block Storage 服务(cinder)提供了一个可选的备份服务，您可以在 OpenShift (RHOSO)环境中的 Red Hat OpenStack Services 中部署。

用户可以使用块存储备份服务为其块存储卷创建和恢复完整或增量备份。

卷备份是保存到备份存储库的块存储卷内容的持久副本。

您可以在 OpenStackControlPlane CR 中的 glance 模板的 cinderBackup 部分下配置备份服务。

先决条件

- 在工作站上安装了 oc 命令行工具。
- 以具有 cluster-admin 权限的用户身份登录到可访问 RHOSO 控制平面的工作站。
- 您已在 OpenStack Control Plane 中为块存储服务启用了备份服务。

4.1. 备份的存储后端

您可以将以下存储后端用于块存储备份：

- 使用 Red Hat Ceph Storage 时，Red Hat Ceph Storage RBD 是默认后端。有关更多信息，请参阅配置 control plane 以使用 Red Hat Ceph Storage 集群。
- Object Storage 服务 (swift)
- NFS
- S3

有关备份的其他后端选项的详情，请参考 OSP18 Cinder 备用存储。

您可以使用备份服务备份块存储服务(cinder)支持的任何后端的卷，无论您选择用于备份的后端是什么。您只能为备份配置一个后端，而您可以为卷配置多个后端。

备份的后端没有 RHOCP 节点的传输协议要求。但是，备份 pod 需要连接到卷，卷的后端具有传输协议要求。

4.2. 为备份设置副本数量

您可以通过将副本设置为大于 1 的值，在主动-主动模式下运行块存储备份组件的多个实例。默认值为 0。

流程

1. 打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并将以下参数添加到 cinder 模板，以便为 cinderBackup 参数设置副本数：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  …
  cinder:
    template:
      cinderBackup: |
        replicas: <number_of_replicas>
...
```

- 将 <number_of_replicas > 替换为大于 1 的值。

2. 更新 control plane：

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

3. 等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：

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

- 当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

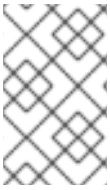
4.3. 备份性能注意事项

块存储备份服务的一些功能，如增量备份、从快照创建备份，数据压缩可能会降低备份操作的性能。

仅捕获对卷的定期更改，增量备份操作可以最小化资源使用量。但是，增量备份操作的性能比完整备份操作低。当您创建增量备份时，必须首先读取卷中的所有数据，并与完整备份和后续增量备份中的数据进行比较。

有些卷后端支持通过将快照直接附加到备份主机来从快照创建备份，这比将快照克隆到卷中要快。如果您用于卷的后端不支持此功能，您可以从快照创建卷，并使用该卷作为备份。但是，从快照创建卷的额外步骤可能会影响备份操作的性能。

您可以配置块存储备份服务，以启用或禁用备份存储后端的数据压缩。如果您启用数据压缩，备份操作需要额外的 CPU 电源，但它们会降低网络带宽和存储空间。



注意

您不能在 Red Hat Ceph Storage 后端中使用数据压缩。

4.4. 为备份设置选项

cinderBackup 参数从 OpenStackControlPlane CR 中的 cinder 模板的顶级 customServiceConfig 部分中继承配置。但是，cinderBackup 参数也有自己的 customServiceConfig 部分。

下表描述了适用于所有后端驱动程序的配置选项。

表 4.1. 备份驱动程序的配置选项

选项	描述	值类型	默认值
debug	当设置为 **true** 时，日志级别设置为 **DEBUG**，而不是默认的 **INFO** 级别。您还可以使用动态日志级别 API 功能动态为调度程序设置调试日志级别，而无需重启。	布尔值	**false**
backup_service_inithost_offload	在备份服务启动过程中卸载待处理的备份删除。如果设置为 **false**，备份服务将保持关闭，直到所有待处理的备份都被删除。	布尔值	**true**
storage_availability_zone	备份服务的可用区。	字符串	**nova**
backup_workers	在备份 pod 中启动的进程数。通过并发备份提高性能。	整数	**1**
backup_max_operations	在每个 pod 上执行的最大并发内存数量，以及可能在 CPU 上执行的重度操作（备份和恢复）。数量限制 pod 中的所有 worker，而不是跨 pod 的所有 worker。0 代表没有限制。	整数	**15**
backup_native_threads_pool_size	用于备份数据相关操作的原生线程池大小。大多数备份驱动程序主要依赖于这个选项，您可以增加不依赖于它的特定驱动程序的值。	整数	**60**

流程

1. **打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并将以下参数添加到 cinder 模板，以设置配置选项。在本例中，您可以启用调试日志，对进程数量加倍，并将每个 pod 的最大操作数增加到 20。**

   Example:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack
spec:
  …
  cinder:
    template:
      customServiceConfig: |
        [DEFAULT]
        debug = true
      cinderBackup:
```

```
          customServiceConfig: |
           [DEFAULT]
           backup_workers = 2
           backup_max_operations = 20

...
```

2. 更新 control plane：

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

3. 等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

4.5. 启用数据压缩

默认情况下，备份使用 zlib 压缩算法压缩。

数据压缩需要额外的 CPU 电源，但使用较少的网络带宽和存储空间。

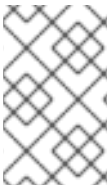
您可以使用 OpenStackControlPlane CR 中的 backup_compression_algorithm 参数更改备份的数据压缩算法或禁用数据压缩。

以下选项可用于数据压缩。

表 4.2. 数据压缩选项

选项	描述
无,off, 或 no	不要使用压缩。

zlib 或 **gzip**	使用 Deflate 压缩算法。
bz2z 或 **bzip2**	使用 Burrows-Wheeler 转换压缩。
zstd	使用 Zstandard 压缩算法。



注意

您不能为 Red Hat Ceph Storage 后端驱动程序指定数据压缩算法。

流程

1. **打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并将以下参数添加到 cinder 模板，以启用数据压缩。在本例中，您可以使用 Object Storage 服务(swift)后端启用数据压缩：**

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  cinder:
    template:
      cinderBackup
        customServiceConfig: |
          [DEFAULT]
          backup_driver = cinder.backup.drivers.nfs.SwiftBackupDriver
          backup_compression_algorithm = zstd
        networkAttachments:
        - storage
```

2. **更新 control plane：**

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

3. **等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：**

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

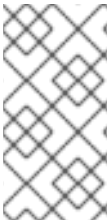
当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

4.6. 为块存储备份配置 CEPH RBD 后端

您可以使用 Red Hat Ceph Storage RADOS 块设备(RBD)作为存储后端配置块存储服务(cinder)备份服务。



注意

如果您使用 Ceph RBD 作为备份与 Ceph RBD 卷的后端，则增量备份的性能效率更高。

有关 Ceph RBD 的更多信息，请参阅配置 control plane 以使用 Red Hat Ceph Storage 集群。

先决条件

- 确保存储后端、Red Hat OpenShift 集群和 Compute 节点之间的网络连接。

流程

1. 打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并将以下参数添加到 cinder 模板，将 Ceph RBD 配置为备份的后端：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  cinder:
    template:
      cinderBackup
        customServiceConfig: |
          [DEFAULT]
          backup_driver = cinder.backup.drivers.nfs.CephBackupDriver
          backup_ceph_pool = backups
          backup_ceph_user = openstack
        networkAttachments:
        - storage
        replicas: 1
```

2. 更新 control plane：

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

3. 等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

4.7. 为备份配置 OBJECT STORAGE 服务(SWIFT)后端

您可以使用 Object Storage 服务(swift)作为存储后端，配置块存储服务(cinder)备份服务。

先决条件

- 确保存储后端、Red Hat OpenShift 集群和 Compute 节点之间的网络连接。
- 验证 Object Storage 服务是否在 OpenShift (RHOSO)部署的 Red Hat OpenStack Services 中活跃。

对象存储服务后端的默认容器是 volumebackups。您可以使用 backup_swift_container 配置选项更改默认容器。

流程

1. 打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并将以下参数添加到 cinder 模板，将 Object Storage 服务配置为备份的后端：

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

```
 cinder:
  template:
    cinderBackup
      customServiceConfig: |
        [DEFAULT]
        backup_driver = cinder.backup.drivers.nfs.SwiftBackupDriver
      networkAttachments:
      - storage
      replicas: 1
```

2. **更新 control plane：**

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

3. **等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：**

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

4.8. 为备份配置 NFS 后端

您可以使用 NFS 作为存储后端配置块存储服务(cinder)备份服务。

先决条件

- **确保存储后端、Red Hat OpenShift 集群和 Compute 节点之间的网络连接。**

流程

1. **创建 secret CR 文件，如 cinder-backup-nfs-secrets.yaml，并为 NFS 共享添加以下配置：**

```
apiVersion: v1
kind: Secret
```

```
metadata:
  labels:
    service: cinder
    component: cinder-backup
  name: cinder-backup-nfs-secrets
type: Opaque
stringData:
  nfs-secrets.conf: |
    [DEFAULT]
    backup_share = <192.168.1.2:/Backups>
    backup_mount_options = <optional>
```

- **将 <192.168.1.2:/Backups > 替换为 NFS 共享的 IP 地址。**
- **将 <optional > 替换为您的 NFS 共享的挂载选项。**

2. **打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并将以下参数添加到 cinder 模板，以便为 NFS 共享添加 secret，并将 NFS 配置为备份的后端：**

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  cinder:
    template:
      cinderBackup
        customServiceConfig: |
          [DEFAULT]
          backup_driver = cinder.backup.drivers.nfs.NFSBackupDriver
        customServiceConfigSecrets:
        - cinder-backup-nfs-secrets
        networkAttachments:
        - storage
        replicas: 1
```

3. **更新 control plane：**

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

4. **等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：**

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 **-w** 选项附加到 **get** 命令的末尾，以跟踪部署进度。

4.9. 为备份配置 S3 后端

您可以使用 **S3** 作为存储后端配置块存储服务**(cinder)**备份服务。

先决条件

- 确保存储后端和 **Red Hat OpenShift** 集群之间的网络连接。

流程

1. 打开 **OpenStackControlPlane CR** 文件 **openstack_control_plane.yaml**，并将以下参数添加到 **cinder** 模板，将 **S3** 配置为备份的后端：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  cinder:
    template:
      cinderBackup
        customServiceConfig: |
          [DEFAULT]
          backup_driver = cinder.backup.drivers.s3.S3BackupDriver
          backup_s3_endpoint_url = <user supplied>
          backup_s3_store_access_key = <user supplied>
          backup_s3_store_secret_key = <user supplied>
          backup_s3_store_bucket = volumebackups
          backup_s3_ca_cert_file = /etc/pki/tls/certs/ca-bundle.crt
        networkAttachments:
        - storage
```

2. 更新 **control plane**：

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

3. 等待 **RHOCP** 创建与 **OpenStackControlPlane CR** 相关的资源。运行以下命令来检查状态：

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

4.10. 块存储卷备份元数据

当您创建块存储卷的备份时，此备份的元数据存储在块存储服务数据库中。当从备份中恢复卷时，块存储备份服务会使用这个元数据。

重要

为确保备份在块存储服务数据库的灾难性丢失时存下，您可以手动导出和存储此备份的元数据。在出现灾难性数据库丢失后，您需要创建新的块存储数据库，然后手动将此备份元数据重新导入到其中。

第 5 章 配置镜像服务(GLANCE)

Image 服务(glance)为磁盘和服务器镜像提供发现、注册和交付服务。它提供复制或存储服务器镜像快照的功能。您可以将存储的镜像用作模板，比安装服务器操作系统和单独配置服务，快速、一致地编写新的服务器。

您可以为镜像服务配置以下后端（存储）：

- 使用 Red Hat Ceph Storage 时，RADOS 块设备(RBD)是默认后端。有关更多信息，请参阅配置 control plane 以使用 Red Hat Ceph Storage 集群。
- Block Storage (cinder)。
- Object Storage (swift)。
- NFS.

先决条件

- 在工作站上安装了 oc 命令行工具。
- 以具有 cluster-admin 权限的用户身份登录到可访问 RHOSO 控制平面的工作站。

5.1. 为镜像服务配置块存储后端

您可以使用 Block Storage 服务(cinder)作为存储后端来配置 Image 服务(glance)。

先决条件

- 确保存储后端、Red Hat OpenShift 集群和 Compute 节点之间的网络连接。
- 确保满足放置、网络和传输协议要求。例如，如果您的块存储服务后端是光纤通道(FC)，运行镜像服务 API 的节点必须具有主机总线适配器(HBA)。对于 FC、iSCSI 和 NVMe over Fabrics

(NVMe-oF)，将节点配置为支持协议并使用多路径。如需更多信息，请参阅配置传输协议。

流程

1. 打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并在 glance 模板中添加以下参数，将块存储服务配置为后端：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  ...
  glance:
    template:
      glanceAPIs:
        default:
          replicas: 3 # recommendation is 3 when deploying the service
      ...
      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. 更新 control plane：

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

3. 等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

5.2. 配置对象存储后端

您可以将 Object Storage 服务(swift)配置为存储后端，配置 Image 服务(glance)。

先决条件

- 确保存储后端、Red Hat OpenShift 集群和 Compute 节点之间的网络连接。

流程

1. 打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并在 glance 模板中添加以下参数，将对象存储服务配置为后端：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
  ...
  glance:
    template:
      glanceAPIs:
        default:
          replicas: 3 # recommendation is 3 when deploying the service
      ...
      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. 更新 control plane：

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

3. 等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

5.3. 配置 NFS 后端

当您在镜像服务(glance)上挂载 NFS 共享时，镜像服务不管理该操作。镜像服务将数据写入文件系统，但不知道后端是一个 NFS 共享。

如果您使用 NFS 作为镜像服务后端，红帽建议以下最佳实践来降低风险：

- 使用可靠的生产级 NFS 后端。
- 确保网络传播到 OpenShift control plane，部署镜像服务，并且镜像服务具有指向网络的 NetworkAttachmentDefinition (NAD)。此配置可确保镜像服务 pod 可以访问 NFS 服务器。
- 设置底层文件系统权限。写入权限必须存在于用作存储的共享文件系统中。
- 确保运行 glance-api 进程的用户和组在本地文件系统上没有挂载点的写入权限。这意味着进程可以检测到可能的挂载失败，并在写尝试过程中将存储置于只读模式。

限制：

- 在 OpenShift 上的 Red Hat OpenStack Services (RHOSO)中，您无法在 pod 规格中设置客户端 NFS 挂载选项。您可以使用以下方法之一设置 NFS 挂载选项：
  - 设置服务器端挂载选项。
  - 使用 /etc/nfsmount.conf。

- 使用带有挂载选项的 PersistentVolume 挂载 NFS 卷。

流程

1. 打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并在 spec 部分添加 extraMounts 参数，以添加 NFS 共享的导出路径和 IP 地址。该路径映射到 /var/lib/glance/images，其中镜像服务 API 存储并检索镜像：

```
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
...
```

- 将 <nfs_export_path > 替换为 NFS 共享的导出路径。
- 将 <nfs_ip_address > 替换为 NFS 共享的 IP 地址。此 IP 地址必须是覆盖网络的一部分，该网络可由镜像服务访问。

2. 在 glance 模板中添加以下参数，将 NFS 配置为后端：

```
...
spec:
  extraMounts:
    ...
    glance:
    template:
      glanceAPIs:
        default:
```

```
      replicas: 3 # recommendation is 3 when deploying the service
    ...
    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. **更新 control plane：**

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

4. **等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：**

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

第 6 章 配置 OBJECT STORAGE 服务 (SWIFT)

您可以将 Object Storage 服务(swift)配置为在 OpenShift 节点或外部数据平面节点上使用 PersistentVolumes (PV)。

当您在 OpenShift 节点上使用 PV 时，此配置仅限于每个节点的一个 PV。对象存储服务需要多个 PV。要最大化可用性和数据持久性，您可以在不同的节点上创建这些 PV，每个节点只有一个 PV。

您可以使用外部数据平面节点在大型存储部署中具有更大的灵活性，您可以在其中使用每个节点使用多个磁盘来部署更大的对象存储集群。

先决条件

- 在工作站上安装了 oc 命令行工具。
- 以具有 cluster-admin 权限的用户身份登录到可访问 RHOSO 控制平面的工作站。

6.1. 使用 PERSISTENTVOLUME 在 OPENSHIFT 节点上部署 OBJECT STORAGE 服务

在默认对象存储服务(swift)部署中至少使用两个 swiftProxy 副本和三个 swiftStorage 副本。您可以增加这些值，以在更多节点和磁盘之间分发存储。

ringReplicas 值定义集群中的对象副本数。例如，如果您设置了 ringReplicas: 3 和 swiftStorage/replicas: 5，每个对象存储在 3 个不同的 PersistentVolume (PV)上，并且总计有 5 个 PV。

流程

1. 打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并将以下参数添加到 swift 模板：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack-control-plane
  namespace: openstack
spec:
  ...
  swift:
```

```
    enabled: true
    template:
      swiftProxy:
        replicas: 2
      swiftRing:
        ringReplicas: 3
      swiftStorage:
        replicas: 3
        storageClass: <swift-storage>
        storageRequest: 100Gi
...
```

- **增加 swiftProxy/replicas: 值，以在更多节点之间分发代理实例。**
- **替换 ringReplicas: 值，以定义集群中您想要的对象副本数。**
- **增加 swiftStorage/replicas: 值，以定义集群中的 PV 数量。**
- **将 < swift-storage > 替换为您希望 Object Storage 服务使用的存储类的名称。**

2. **更新 control plane：**

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

3. **等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：**

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

6.2. 对象存储环

Object Storage 服务(swift)使用名为 ring 的数据结构，在集群中分发分区空间。此分区空间是对象存

储服务中数据持久性引擎的核心。通过 ring，对象存储服务可以快速轻松地在集群间同步每个分区。

Ring 包含有关对象存储分区的信息，以及如何在 OpenShift (RHOSO)部署上的 Red Hat OpenStack Services 中的不同节点和磁盘之间分发。当任何对象存储组件与数据交互时，会在环本地执行快速查找，以确定每个对象的可能分区。

对象存储服务有三个环来存储以下类型的数据：

- 帐户信息
- 容器，便于在帐户下组织对象
- 对象副本

6.3. RING 分区电源

Ring 电源决定了资源（如帐户、容器或对象）被映射的分区。分区包含在路径中，资源存储在后端文件系统中。因此，更改分区电源需要将资源重新定位到后端文件系统中的新路径中。

在大量填充的集群中，重新定位过程会非常耗时。为避免停机，在集群仍然运行时重新定位资源。您必须在不临时丢失对数据的访问或影响进程性能（如复制和审计）的情况下这样做。如需增加环分区功能的帮助，请联系红帽支持。

当您将单独的节点用于对象存储服务(swift)时，请使用更高的分区电源值。

对象存储服务使用修改 *的哈希环* 将数据分布到磁盘和节点上。默认有三个环：一个用于帐户，一个用于容器，另一个用于对象。每个环使用一个称为 *分区电源* 的固定参数。此参数设置可创建的最大分区数。

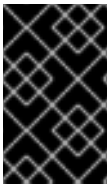
6.4. 增加环分区电源

您只能为新容器及其对象更改分区 power 参数，因此您必须在初始部署前设置这个值。

默认分区节能值为 10。如果使用三个副本，请参考下表来选择适当的分区功能：

表 6.1. 根据可用磁盘的数量的适当分区电源值

分区 Power	最大磁盘数
10	~ 35
11	~ 75
12	~ 150
13	~ 250
14	~ 500



重要

设置过高的分区电源值（例如，只有 14 代表 40 个磁盘）对复制时间造成负面影响。

流程

1. 打开 OpenStackControlPlane CR 文件，openstack_control_plane.yaml，并在 swift 模板的 swiftRing 参数下更改 partPower 的值：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack-control-plane
  namespace: openstack
spec:
  ...
  swift:
    enabled: true
    template:
      swiftProxy:
        replicas: 2
      swiftRing:
        partPower: 12
        ringReplicas: 3
...
```

- 将 <12 > 替换为您要为分区电源设置的值。

提示

您还可以为新容器配置额外的对象服务器 ring。如果要在最初使用低分区电源的对象存储服务部署中添加更多磁盘，这非常有用。

第 7 章 配置共享文件系统服务(MANILA)

当您部署共享文件系统服务(manila)时，您可以选择一个或多个受支持的后端，如原生 CephFS、CephFS-NFS、NetApp 等。

有关支持的后端设备和驱动程序的完整列表，请参阅 Red Hat OpenStack Platform 中红帽知识库文章、组件、插件和驱动程序支持的 Manila 部分。

先决条件

- 在工作站上安装了 oc 命令行工具。
- 以具有 cluster-admin 权限的用户身份登录到可访问 RHOSO 控制平面的工作站。
- 您已计划共享文件系统服务的网络。如需更多信息，请参阅 规划 *部署中的规划* 共享文件系统服务 的网络。
- 您已启用了共享文件系统服务。如需更多信息，请参阅启用共享文件系统服务。
- 对于原生 CephFS 或 CephFS-NFS：
  - Red Hat Ceph Storage 集群中必须存在 CephFS 文件系统。有关更多信息，请参阅 集成 Red Hat Ceph Storage。
  - 必须存在一个具有 CephX 功能（大写）的 Ceph 用户，才能在 CephFS 文件系统上执行操作。有关更多信息，请参阅 集成 Red Hat Ceph Storage。
- 对于 CephFS-NFS：
  - Ceph Storage 集群中必须存在 ceph nfs 服务。有关更多信息，请参阅 集成 Red Hat Ceph Storage。
  - 您已为 NFS 导出创建了一个隔离的 StorageNFS 网络，以及在网络服务(neutron)中对

应的 StorageNFS 共享提供商网络。StorageNFS 共享提供商网络映射到数据中心的隔离 StorageNFS 网络。

- NFS 服务在网络中隔离，您可以与 OpenShift (RHOSO)用户共享的所有 Red Hat OpenStack 服务共享。有关自定义 NFS 服务的更多信息，请参阅 Red Hat Ceph Storage *文件系统指南中的* NFS 集群和导出管理。

重要

当您为共享文件系统服务部署 NFS 服务时，请不要选择要公开 NFS 的自定义端口。仅支持 2049 的默认 NFS 端口。您必须启用 Red Hat Ceph Storage ingress 服务，并将 ingress-mode 设置为 haproxy-protocol。否则，您不能将基于 IP 的访问规则与共享文件系统服务一起使用。对于生产环境中的安全性，红帽不推荐在共享上提供对 0.0.0.0/0 的访问，以将其挂载到客户端机器上。

7.1. 启用共享文件系统服务

您可以启用共享文件系统服务(manila)，在 OpenShift (RHOSO)部署中在 Red Hat OpenStack Services 中置备远程、可共享的文件系统。这些文件系统称为共享，它们允许云中的项目共享 POSIX 兼容存储。共享可以通过读/写访问模式同时挂载到多个计算实例、裸机计算、容器或容器的 pod。

启用共享文件系统服务时，您可以使用以下后端配置服务：

- Red Hat Ceph Storage CephFS
- Red Hat Ceph Storage CephFS-NFS
- 通过第三方供应商存储系统进行 NFS 或 CIFS

先决条件

- 您已计划共享文件系统服务的网络。如需更多信息，请参阅 规划 *部署中的规划* 共享文件系统服务 的网络。

流程

1.

打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并将以下参数添加到 spec 部分以启用共享文件系统服务：

```
spec:
  ...
  manila:
    enabled: true
    apiOverride:
      route: {}
    template:
      databaseInstance: openstack
      secret: osp-secret
      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
      manilaAPI:
        replicas: 3
      manilaScheduler:
        replicas: 3
      manilaShares:
        share1:
          networkAttachments:
          - storage
          replicas: 0 # backend needs to be configured
```

注意

您必须为共享文件系统服务配置后端。如果您没有为共享文件系统服务配置后端，则部署该服务但不激活(副本： 0)。

2. 更新 control plane：

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

3. 等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

7.2. 配置原生 CEPHFS 后端

您可以使用原生 CephFS 作为存储后端配置共享文件系统服务(manila)。

限制：

您可以将原生 CephFS 后端公开给可信用户，但采取以下安全措施：

- 将存储网络配置为提供商网络。
- 应用基于角色的访问控制(RBAC)策略来保护存储供应商网络。
- 创建私有共享类型。

先决条件

- 隔离存储网络。
- 确保存储后端、Red Hat OpenShift 集群和 Compute 节点之间的网络连接。
- 您已创建了 Red Hat Ceph Storage secret。如需更多信息，请参阅 集成 Red Hat Ceph Storage

流程

1. 打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并在 spec 部分添加 extraMounts 参数，以显示 Ceph 配置文件：
    -

```
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. **将以下参数添加到 manila 模板，以配置原生 CephFS 后端：**

```
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. 更新 control plane：

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

4. 等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

7.3. 配置 CEPHFS-NFS 后端

您可以使用 CephFS-NFS 作为存储后端配置共享文件系统服务(manila)。

先决条件

- 隔离存储网络在 OpenShift 上的共享管理器 pod 上配置，以便共享文件系统服务可以与 Red Hat Ceph Storage 集群通信。
- 对于 NFS 流量，红帽建议使用隔离的 NFS 网络。此网络不需要供 OpenShift 上共享文件系统服务的共享管理器容器集使用，但必须为最终用户拥有的 Compute 实例可用。
- 确保存储后端、Red Hat OpenShift 集群和 Compute 节点之间的网络连接。

流程

1. 打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并将以下参数添加到 manila 模板：

```
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. **更新 control plane：**

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

3. **等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：**

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

7.4. 配置替代后端

要使用替代后端（如 NetApp 或 Pure Storage）配置共享文件系统服务(manila)，请完成以下高级别任务：

1. 创建服务器连接 secret。

2. 配置 OpenStackControlPlane CR，以使用替代存储系统作为共享文件系统服务的后端。

先决条件

- 您已准备了替代存储系统以供 OpenShift (RHOSO)上的 Red Hat OpenStack Services 使用。

- Red Hat OpenShift 集群、Compute 节点和替代存储系统之间的网络连接。

7.4.1. 创建服务器连接 secret

为替代后端创建服务器连接 secret，以防止将服务器连接信息直接放在 OpenStackControlPlane CR 中。

流程

1. 创建一个包含您替代后端的服务器连接信息的配置文件。在本例中，您要为 NetApp 后端创建 secret。

   以下是配置文件内容的示例：

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

   - 将 <netapp_ip > 替换为服务器的 IP 地址。

   - 将 <netapp_user > 替换为登录用户名。

- 将 <netapp_password& gt; 替换为登录密码。
- 将 <netappvserver& gt; 替换为 vserver 名称。如果配置 driver_handles_share_servers=True 模式，则不需要这个选项。

2. 保存配置文件。
3. 根据配置文件创建 secret：

```
$ oc create secret generic <secret_name> --from-file=<configuration_file_name>
```

- 将 <secret_name > 替换为您要分配给 secret 的名称。
- 将 <configuration_file_name > 替换为您创建的配置文件的名称。

4. 删除配置文件。

7.4.2. 配置替代后端

您可以使用替代存储后端（如 NetApp 后端）配置共享文件系统服务(manila)。

先决条件

- 确保存储后端、Red Hat OpenShift 集群和 Compute 节点之间的网络连接。

流程

1. 打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并将以下参数添加到 manila 模板：

```
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. **更新 control plane：**

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

3. **等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：**

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

7.4.3. 自定义配置文件

当您为共享文件系统服务(manila)配置替代后端时，您可能需要使用其他配置文件。您可以使用

OpenStackControlPlane CR 文件中的 extraMounts 参数将这些配置文件作为相关共享管理器 pod 中的 OpenShift configMap 或 secret 对象提供。

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
...
```

7.4.4. 自定义存储驱动程序镜像

当您为共享文件系统服务(manila)配置替代后端时，您可能需要使用 红帽生态系统目录 上的厂商使用自定义 manilaShares 容器镜像。

您可以使用 customContainerImages 参数将容器镜像的路径添加到 OpenStackVersion CR 文件中。

如需更多信息，请参阅 *集成 合作伙伴内容 中的 部署合作伙伴 容器镜像*。

7.5. 配置多个后端

您可以为共享文件系统服务(manila)部署多个后端，如 CephFS-NFS 后端、原生 CephFS 后端和第三方后端。仅为每个 pod 添加一个后端。

先决条件

- **当您从需要外部软件组件的存储厂商中使用后端驱动程序时，您必须在部署过程中覆盖共享文件系统服务的标准容器镜像。您可以在 红帽生态系统目录 中找到自定义容器镜像，例如 Dell EMC unity 存储系统的 Dell EMC unity 容器镜像。**

- **确保存储后端、 Red Hat OpenShift 集群和 Compute 节点之间的网络连接。**

流程

1. **打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并在 manila 模板中添加以下参数来配置后端。在本例中，有 CephFS-NFS 后端、原生 CephFS 后端和 Pure 存储后端：**

```
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. **为每个要使用的后端添加配置：**

   - **为 CephFS-NFS 后端添加配置：**

```
...
    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
    ...
```

- 为**原生 CephFS 后端添加配置：**

```
    ...
      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
      ...
```

- 为 **Pure 存储后端添加配置：**

```
    ...
      customContainerImages:
        manilaShareImages:
          pure: registry.connect.redhat.com/purestorage/openstack-manila-share-
pure:rhoso18
      manilaShares:
        cephfsnfs:
        ...
        cephfs:
        ...
        pure:
          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. **更新 control plane：**

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

4. **等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：**

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

7.6. 验证共享文件系统服务的部署

在部署或对问题进行故障排除时，验证共享文件系统服务(manila)的服务是否正在运行，以及它们是否已启动。

验证 **manila Pod 是否正在运行。pod 数量取决于您为共享文件系统服务的不同组件配置的副本数。**

当您验证 pod 是否正在运行时，您可以使用共享文件系统服务 API 检查服务的状态。

流程

1. **列出 manila pod 以验证它们是否正在运行：**

```
$ oc -n openstack get pod -l service=manila
```

输出示例：

```
NAME                          READY   STATUS      RESTARTS        AGE
manila-api-0                  2/2     Running     0               43h
manila-api-1                  2/2     Running     0               43h
manila-api-2                  2/2     Running     0               43h
manila-db-purge-28696321-tkl9g   0/1     Completed   0             41h
manila-db-purge-28697761-zxxzc   0/1     Completed   0             17h
manila-scheduler-0            2/2     Running     0               43h
manila-scheduler-1            2/2     Running     0               43h
manila-scheduler-2            2/2     Running     0               43h
manila-share-share1-0           2/2     Running     0             43h
```

2. **从您的工作站访问 OpenStackClient pod 的远程 shell：**

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

3. **运行 openstack share service list 命令：**

```
$ openstack share service list
```

**示例 output:----------
| Binary | Host | Zone | Status | State | Updated At |----------
| 1 | manila-scheduler | hostgroup | enabled | up | 2024-07-25T17:40:27.323342 | 4 | manila-share | hostgroup@cephfsnfs | nova | enabled | up | 2024-07-25T17:40:49.115386|----------**

4. **验证每个服务的 Status 条目是否为 up。如果没有，请检查相关的日志文件。**

5. **退出 openstackclient pod:**

```
$ exit
```

7.7. 验证多个后端的部署

使用 openstack share service list 命令来验证共享文件系统服务(manila)的存储后端是否已成功部署。如果您在多个后端中使用健康检查，则 ping 测试也会返回一个响应，即使其中一个后端没有响应，

因此这不是验证部署的一种可靠方法。

流程

1. 从您的工作站访问 OpenStackClient pod 的远程 shell：

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

2. 确认共享文件系统服务后端列表：

```
$ openstack share service list
```

每个成功部署的后端的状态都显示为 enabled，状态显示为 up。

3. 退出 openstackclient pod:

```
$ exit
```

7.8. 为后端创建可用区

您可以为共享文件系统服务后端创建可用区(AZ)，以明确为用户对云基础架构和服务进行分组。将 AZ 映射到故障域和计算资源，以实现高可用性、容错和资源调度。例如，您可以创建一个具有特定硬件的 Compute 节点的 AZ，用户可以在创建需要该硬件的实例时指定它们。

部署后，使用 availability_zones 共享类型额外规格，将共享类型限制为一个或多个 AZ。只要共享类型没有限制它们，用户可以在 AZ 中直接创建共享。

流程

以下示例部署两个后端，其中 CephFS 是 zone 1，NetApp 是 zone 2。

1. 打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并将以下参数添加到 manila 模板：

```
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. **更新 control plane：**

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

3. **等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：**

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

7.9. 更改允许的 NAS 协议

您可以使用共享文件系统服务(manila)在 NFS、CephFS 或 CIFS 网络附加存储(NAS)协议中导出共享。默认情况下，共享文件系统服务启用 NFS 和 CIFS，这可能不受部署中的后端的支持。

您可以更改 enabled_share_protocols 参数，并只列出要在云中允许的协议。例如，如果您的部署中的后端支持 NFS 和 CIFS，您可以更改默认值并仅启用一个协议。您分配的 NAS 协议必须由共享文件系统服务部署的后端支持。

不是所有存储后端驱动程序都支持 CIFS 协议。有关哪些认证存储系统支持 CIFS 的信息，请参阅 红帽生态系统目录。

流程

1. **打开 OpenStackControlPlane CR 文件 openstack_control_plane.yaml，并将以下参数添加到 manila 模板。在这个示例中，您可以启用 NFS 协议：**

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

2. **更新 control plane：**

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

3. **等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。运行以下命令来检查状态：**

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

当状态为 "Setup complete" 时，会创建 OpenStackControlPlane 资源。

提示

将 -w 选项附加到 get 命令的末尾，以跟踪部署进度。

7.10. 查看后端存储容量

共享文件系统服务(manila)的调度程序组件根据容量、置备配置、放置提示以及后端存储系统驱动程序检测到和公开的功能，进行智能放置决策。您可以使用共享类型和额外规格来修改放置决策。

流程

1. 从您的工作站访问 OpenStackClient pod 的远程 shell：

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

2. 运行以下命令来查看可用的后端存储容量：

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

3. 退出 openstackclient pod:

```
$ exit
```

7.11. 配置自动数据库清理

共享文件系统服务(manila)服务自动清除标记为删除的数据库条目。默认情况下，记录标记为删除 30 天。您可以为清除作业配置不同的记录年龄和调度。

流程

1. 打开 openstack_control_plane.yaml 文件，以编辑 OpenStackControlPlane CR。

2. 将 dbPurge 参数添加到 manila 模板，以配置数据库清理。

   以下是使用 dbPurge 参数配置共享文件系统服务的示例：

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

1. 在清除记录前，记录标记为删除的天数。默认值为 30 天。最小值为 1 天。

2

以 crontab 格式运行作业的时间计划。默认值为 1 0 * *。此默认值等同于每日 00:01。

3. **更新 control plane：**

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