Red Hat OpenShift Data Foundation 4.15

管理混合和多云资源

有关如何使用多云对象网关(NooBaa)在混合云或多云环境中管理存储资源的说明。

Last Updated: 2024-05-14

Red Hat OpenShift Data Foundation 4.15 管理混合和多云资源

有关如何使用多云对象网关(NooBaa)在混合云或多云环境中管理存储资源的说明。

法律通告

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

本文档解释了如何在混合云或多云环境中管理存储资源。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。请告诉我们如何让它更好。

要提供反馈，请创建一个 Bugzilla ticket：

1. 进入 Bugzilla 网站。
2. 在 **Component** 部分中，选择 **文档**。
3. 在 **Description** 中输入您要提供的信息。包括文档相关部分的链接。
4. 点 **Submit Bug**。

第 1 章 关于 MULTICLOUD 对象网关

Multicloud 对象网关 (MCG) 是 OpenShift 的轻量级对象存储服务，允许用户启动小规模，然后根据需要在多个集群中、多个集群中和云原生存储中进行扩展。

第 2 章 使用应用程序访问多云对象网关

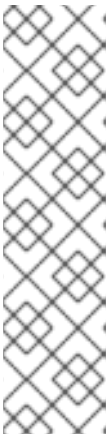
您可以使用任何以 AWS S3 或使用 AWS S3 软件开发套件 (SDK) 的代码为目标的应用程序访问对象服务。应用程序需要指定多云对象网关(MCG)端点、访问密钥和 secret 访问密钥。您可以使用您的终端或 MCG CLI 来检索此信息。

如需有关访问 RADOS 对象网关(RGW)S3 端点的信息，请参阅访问 RADOS 对象网关 S3 端点。

先决条件

- 正在运行的 OpenShift Data Foundation 平台。
- 下载 MCG 命令行界面以简化管理。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```



注意

指定使用订阅管理器启用存储库的适当架构。

- 对于 IBM Power，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- 对于 IBM Z，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- 或者，您也可以从 下载红帽 OpenShift Data Foundation 页面 上的 OpenShift Data Foundation RPM 安装 MCG 软件包。

注意

根据您的架构选择正确的产品变体。

您可以通过两种方式访问相关的端点、访问密钥和 secret 访问密钥：

- 从终端访问 Multicloud 对象网关
- 使用 MCG 命令行界面访问 Multicloud 对象网关

例如：

使用虚拟主机风格访问 MCG 存储桶

如果客户端应用程序尝试访问 https://*<bucket-name>*.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com

<bucket-name>

是 MCG 存储桶的名称
例如：https://mcg-test-bucket.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com

DNS 条目需要 **mcg-test-bucket.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com** 来指向 S3 服务。

重要

确保您有一个 DNS 条目，以便使用虚拟主机风格将客户端应用程序指向 MCG 存储桶。

2.1. 从终端访问 MULTICLOUD 对象网关

流程

运行 **describe** 命令，以查看有关多云对象网关(MCG)端点的信息，包括其访问密钥（**AWS_ACCESS_KEY_ID** 值）和 secret 访问密钥（**AWS_SECRET_ACCESS_KEY** 值）。

```
# oc describe noobaa -n openshift-storage
```

输出结果类似如下：

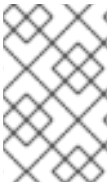
```
Name:         noobaa
Namespace:    openshift-storage
Labels:       <none>
Annotations:  <none>
API Version:  noobaa.io/v1alpha1
Kind:         NooBaa
Metadata:
  Creation Timestamp:  2019-07-29T16:22:06Z
  Generation:          1
  Resource Version:    6718822
  Self Link:           /apis/noobaa.io/v1alpha1/namespaces/openshift-storage/noobaas/noobaa
  UID:                 019cfb4a-b21d-11e9-9a02-06c8de012f9e
Spec:
Status:
  Accounts:
    Admin:
      Secret Ref:
        Name:          noobaa-admin
        Namespace:     openshift-storage
  Actual Image:        noobaa/noobaa-core:4.0
  Observed Generation:  1
  Phase:               Ready
  Readme:

  Welcome to NooBaa!
  -----------------

  Welcome to NooBaa!
    -----------------
    NooBaa Core Version:
    NooBaa Operator Version:

    Lets get started:

    1. Connect to Management console:
```

```
    Read your mgmt console login information (email & password) from secret: "noobaa-admin".

      kubectl get secret noobaa-admin -n openshift-storage -o json | jq '.data|map_values(@base64d)'

    Open the management console service - take External IP/DNS or Node Port or use port
forwarding:

      kubectl port-forward -n openshift-storage service/noobaa-mgmt 11443:443 &
      open https://localhost:11443

  2. Test S3 client:

    kubectl port-forward -n openshift-storage service/s3 10443:443 &
(1)
    NOOBAA_ACCESS_KEY=$(kubectl get secret noobaa-admin -n openshift-storage -o json | jq -r
'.data.AWS_ACCESS_KEY_ID|@base64d')
(2)
    NOOBAA_SECRET_KEY=$(kubectl get secret noobaa-admin -n openshift-storage -o json | jq -r
'.data.AWS_SECRET_ACCESS_KEY|@base64d')
    alias s3='AWS_ACCESS_KEY_ID=$NOOBAA_ACCESS_KEY
AWS_SECRET_ACCESS_KEY=$NOOBAA_SECRET_KEY aws --endpoint https://localhost:10443 --
no-verify-ssl s3'
    s3 ls


  Services:
    Service Mgmt:
      External DNS:
        https://noobaa-mgmt-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
        https://a3406079515be11eaa3b70683061451e-1194613580.us-east-
2.elb.amazonaws.com:443
      Internal DNS:
        https://noobaa-mgmt.openshift-storage.svc:443
      Internal IP:
        https://172.30.235.12:443
      Node Ports:
        https://10.0.142.103:31385
      Pod Ports:
        https://10.131.0.19:8443
    serviceS3:
      External DNS: (3)
        https://s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
        https://a340f4e1315be11eaa3b70683061451e-943168195.us-east-2.elb.amazonaws.com:443
      Internal DNS:
        https://s3.openshift-storage.svc:443
      Internal IP:
        https://172.30.86.41:443
      Node Ports:
        https://10.0.142.103:31011
      Pod Ports:
        https://10.131.0.19:6443
```

(1) 访问密钥（**AWS_ACCESS_KEY_ID** 值）

(2) Secret access key（**AWS_SECRET_ACCESS_KEY** 值）

3 MCG 端点



注意

oc describe nooba 命令的输出列出了可用的内部和外部 DNS 名称。使用内部 DNS 时，流量是空闲的。外部 DNS 使用 Load Balancing 来处理流量，因此会有一个每小时的成本。

2.2. 使用 MCG 命令行界面访问 MULTICLOUD 对象网关

先决条件

- 下载 MCG 命令行界面。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注意

指定使用订阅管理器启用存储库的适当架构。

- 对于 IBM Power，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- 对于 IBM Z，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

流程

运行 **status** 命令访问端点、访问密钥和 secret 访问密钥：

```
noobaa status -n openshift-storage
```

输出结果类似如下：

```
INFO[0000] Namespace: openshift-storage
INFO[0000]
INFO[0000] CRD Status:
INFO[0003]   Exists: CustomResourceDefinition "noobaas.noobaa.io"
INFO[0003]   Exists: CustomResourceDefinition "backingstores.noobaa.io"
INFO[0003]   Exists: CustomResourceDefinition "bucketclasses.noobaa.io"
INFO[0004]   Exists: CustomResourceDefinition "objectbucketclaims.objectbucket.io"
INFO[0004]   Exists: CustomResourceDefinition "objectbuckets.objectbucket.io"
INFO[0004]
INFO[0004] Operator Status:
INFO[0004]   Exists: Namespace "openshift-storage"
INFO[0004]   Exists: ServiceAccount "noobaa"
INFO[0005]   Exists: Role "ocs-operator.v0.0.271-6g45f"
INFO[0005]   Exists: RoleBinding "ocs-operator.v0.0.271-6g45f-noobaa-f9vpj"
```

```
INFO[0006]   Exists: ClusterRole "ocs-operator.v0.0.271-fjhgh"
INFO[0006]   Exists: ClusterRoleBinding "ocs-operator.v0.0.271-fjhgh-noobaa-pdxn5"
INFO[0006]   Exists: Deployment "noobaa-operator"
INFO[0006]
INFO[0006] System Status:
INFO[0007]   Exists: NooBaa "noobaa"
INFO[0007]   Exists: StatefulSet "noobaa-core"
INFO[0007]   Exists: Service "noobaa-mgmt"
INFO[0008]   Exists: Service "s3"
INFO[0008]   Exists: Secret "noobaa-server"
INFO[0008]   Exists: Secret "noobaa-operator"
INFO[0008]   Exists: Secret "noobaa-admin"
INFO[0009]   Exists: StorageClass "openshift-storage.noobaa.io"
INFO[0009]   Exists: BucketClass "noobaa-default-bucket-class"
INFO[0009]   (Optional) Exists: BackingStore "noobaa-default-backing-store"
INFO[0010]   (Optional) Exists: CredentialsRequest "noobaa-cloud-creds"
INFO[0010]   (Optional) Exists: PrometheusRule "noobaa-prometheus-rules"
INFO[0010]   (Optional) Exists: ServiceMonitor "noobaa-service-monitor"
INFO[0011]   (Optional) Exists: Route "noobaa-mgmt"
INFO[0011]   (Optional) Exists: Route "s3"
INFO[0011]   Exists: PersistentVolumeClaim "db-noobaa-core-0"
INFO[0011]   System Phase is "Ready"
INFO[0011]   Exists:  "noobaa-admin"

#------------------#
#- Mgmt Addresses -#
#------------------#

ExternalDNS : [https://noobaa-mgmt-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
https://a3406079515be11eaa3b70683061451e-1194613580.us-east-2.elb.amazonaws.com:443]
ExternalIP  : []
NodePorts   : [https://10.0.142.103:31385]
InternalDNS : [https://noobaa-mgmt.openshift-storage.svc:443]
InternalIP  : [https://172.30.235.12:443]
PodPorts    : [https://10.131.0.19:8443]

#--------------------#
#- Mgmt Credentials -#
#--------------------#

email    : admin@noobaa.io
password : HKLbH1rSuVU0I/souIkSiA==

#----------------#
#- S3 Addresses -#
#----------------#

(1)
ExternalDNS : [https://s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
https://a340f4e1315be11eaa3b70683061451e-943168195.us-east-2.elb.amazonaws.com:443]
ExternalIP  : []
NodePorts   : [https://10.0.142.103:31011]
InternalDNS : [https://s3.openshift-storage.svc:443]
InternalIP  : [https://172.30.86.41:443]
PodPorts    : [https://10.131.0.19:6443]
```

```
#------------------#
#- S3 Credentials -#
#------------------#

2
AWS_ACCESS_KEY_ID     : jVmAsu9FsvRHYmfjTiHV
3
AWS_SECRET_ACCESS_KEY : E//420VNedJfATvVSmDz6FMtsSAzuBv6z180PT5c

#------------------#
#- Backing Stores -#
#------------------#

NAME                          TYPE     TARGET-BUCKET                                      PHASE   AGE
noobaa-default-backing-store   aws-s3   noobaa-backing-store-15dc896d-7fe0-4bed-9349-
5942211b93c9   Ready   141h35m32s

#------------------#
#- Bucket Classes -#
#------------------#

NAME                          PLACEMENT                                                  PHASE   AGE
noobaa-default-bucket-class   {Tiers:[{Placement: BackingStores:[noobaa-default-backing-store]}]}
Ready   141h35m33s

#------------------#
#- Bucket Claims -#
#------------------#

No OBC's found.
```

1 endpoint

2 access key

3 secret access key

您有相关的端点、访问密钥和 secret 访问密钥来连接到应用程序。

例如：

如果 AWS S3 CLI 是应用程序，以下命令将列出 OpenShift Data Foundation 中的存储桶：

```
AWS_ACCESS_KEY_ID=<AWS_ACCESS_KEY_ID>
AWS_SECRET_ACCESS_KEY=<AWS_SECRET_ACCESS_KEY>
aws --endpoint <ENDPOINT> --no-verify-ssl s3 ls
```

2.3. 支持多云对象网关数据存储桶 API

下表列出了 Multicloud 对象网关(MCG)数据存储桶 API 及其支持级别。

数据存储桶	支持	

列出存储桶	支持	
删除存储桶	支持	复制配置是 MCG bucket 类配置的一部分
创建存储桶	支持	不同的一组可处理 ACL
post bucket	不支持	
放置存储桶	部分支持	复制配置是 MCG bucket 类配置的一部分
bucket 生命周期	部分支持	仅限对象过期
策略(Buckets、Objects)	部分支持	支持存储桶策略
bucket 网站	支持	
bucket ACL (Get, Put)	支持	不同的一组可处理 ACL
bucket 位置	Partialy	仅返回默认值
bucket 通知	不支持	
bucket 对象版本	Supoorted	
Get Bucket Info (HEAD)	支持	
bucket 请求支付	部分支持	返回存储桶所有者
放置对象	支持	
删除对象	支持	
获取对象	支持	
对象 ACL (Get, Put)	支持	
Get Object Info (HEAD)	支持	
POST 对象	支持	
复制对象	支持	
多部分上传	支持	
对象标记	支持	

Storage class	不支持	

注意

不支持 **cors, metrics, inventory, analytics, inventory, logging, notifications, accelerate, replication, request payment, locks verbs**

第 3 章 为混合或多云添加存储资源

3.1. 创建新的后备存储

在 OpenShift Data Foundation 中使用此流程创建新的后备存储。

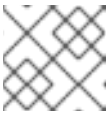
先决条件

- OpenShift Data Foundation 的管理员访问权限。

流程

1. 在 OpenShift Web 控制台中，点 **Storage → Object Storage**。
2. 单击 **Backing Store** 选项卡。
3. 单击 **Create Backing Store**。
4. 在 **Create New Backing Store** 页面中执行以下操作：
   a. 输入**后端存储名称**。
   b. 选择 **Provider**。
   c. 选择 **Region**。
   d. 可选：输入 **端点**。
   e. 从下拉列表中选择一个 **Secret**，或者创建自己的 secret。另外，您也可以**切换到 Credentials** 视图来填写所需的 secret。
   有关创建 OCP secret 的更多信息，请参阅 *Openshift Container Platform* 文档中的 创建 secret 部分。

   每个后备存储都需要不同的机密。有关为特定后备存储创建 secret 的更多信息，请参阅 第 3.3 节 “使用 MCG 命令行界面为混合或多云添加存储资源” 并按照使用 YAML 添加存储资源的步骤进行操作。

   注意

   此菜单与 Google Cloud 和本地 PVC 以外的所有供应商相关。

   f. 输入 **Target bucket**。目标 bucket 是托管在远程云服务的容器存储。它允许您创建一个连接，告诉 MCG 它可以将此存储桶用于系统。
5. 单击 **Create Backing Store**。

验证步骤

1. 在 OpenShift Web 控制台中，点 **Storage → Object Storage**。
2. 单击 **Backing Store** 选项卡，以查看所有后备存储。

3.2. 覆盖默认后备存储

您可以使用 **manualDefaultBackingStore** 标志覆盖默认 NooBaa 后备存储，如果您不想使用默认的后备存储配置，则将其删除。这提供了自定义后备存储配置的灵活性，并根据您的特定需求量身定制。通过利用这个功能，您可以进一步优化系统并增强其性能。

先决条件

- 安装了带有 OpenShift Data Foundation operator 的 OpenShift Container Platform。
- 下载 Multicloud 对象网关 (MCG) 命令行界面：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注意

指定适当的架构，以使用订阅管理器启用存储库。

- 对于 IBM Power，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- 对于 IBM Z，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

另外，您还可以从位于 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/package的 OpenShift Data Foundation RPM 安装 MCG 软件包。

注意

根据您的架构选择正确的产品变体。

流程

1. 检查 **noobaa-default-backing-store** 是否存在：

```
$ oc get backingstore
NAME TYPE PHASE AGE
noobaa-default-backing-store pv-pool Creating 102s
```

2. 修补 NooBaa CR 以启用 **manualDefaultBackingStore** ：

```
$ oc patch noobaa/noobaa --type json --
patch='[{"op":"add","path":"/spec/manualDefaultBackingStore","value":true}]'
```

重要

使用 Multicloud 对象网关 CLI 创建新的后备存储和更新帐户。

3. 创建新的默认后备存储以覆盖默认后备存储。例如：

```
$ noobaa backingstore create pv-pool _NEW-DEFAULT-BACKING-STORE_ --num-volumes
1 --pv-size-gb 16
```

a. 将 ***NEW-DEFAULT-BACKING-STORE*** 替换为您要用于新的默认后备存储的名称。

4. 更新 admin 帐户，以使用新的默认后备存储作为其默认资源：

```
$ noobaa account update admin@noobaa.io --new_default_resource=_NEW-DEFAULT-
BACKING-STORE_
```

a. 将 ***NEW-DEFAULT-BACKING-STORE*** 替换为上一步中的后备存储的名称。
更新 admin 帐户的默认资源可确保在整个系统中使用新配置。

5. 将 default-bucketclass 配置为使用新的默认后备存储：

```
$ oc patch Bucketclass noobaa-default-bucket-class -n openshift-storage --type=json --
patch='[{"op": "replace", "path": "/spec/placementPolicy/tiers/0/backingStores/0", "value":
"NEW-DEFAULT-BACKING-STORE"}]'
```

6. 可选：删除 noobaa-default-backing-store。

a. 删除所有 **noobaa-default-backing-store** 实例以及与它关联的存储桶，并更新使用它作为资源更新的帐户。

b. 删除 noobaa-default-backing-store ：

```
$ oc delete backingstore noobaa-default-backing-store -n openshift-storage | oc patch -n
openshift-storage backingstore/noobaa-default-backing-store --type json --patch='[ {
"op": "remove", "path": "/metadata/finalizers" } ]'
```

在继续操作前，您必须启用 **manualDefaultBackingStore** 标记。此外，更新所有使用默认资源的帐户，并删除与默认后备存储关联的所有 和 bucket 实例，以确保平稳过渡。

3.3. 使用 MCG 命令行界面为混合或多云添加存储资源

多云对象网关 (MCG) 简化了跨云供应商和集群的数据生成过程。

添加 MCG 可以使用的后备存储。

根据部署类型，您可以选择以下步骤之一来创建后备存储：

- 有关创建 AWS 支持的后备存储，请参阅 第 3.3.1 节 “创建 AWS 支持的后备存储”
- 有关创建 AWS-STS 支持的后备存储，请参阅 第 3.3.2 节 “创建 AWS-STS 支持的后备存储”
- 有关创建 IBM COS 支持的后备存储，请参考 第 3.3.3 节 “创建 IBM COS 支持的后备存储”
- 有关创建 Azure 支持的后备存储，请参阅 第 3.3.4 节 “创建 Azure 支持的后备存储”
- 有关创建 GCP 支持的后备存储，请参阅 第 3.3.5 节 “创建由 GCP 支持的后备存储”
- 有关创建本地持久性卷支持的后备存储，请参阅 第 3.3.6 节 “创建由本地持久性卷支持的后备存储”

对于 VMware 部署，请跳至 第 3.4 节 “创建兼容 s3 的多云对象网关后备存储” 以了解更多详细信息。

3.3.1. 创建 AWS 支持的后备存储

先决条件

- 下载多云对象网关(MCG)命令行界面。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注意

指定使用订阅管理器启用存储库的适当架构。例如，如果是 IBM Z，请使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- 另外，您还可以从位于 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages的 OpenShift Data Foundation RPM 安装 MCG 软件包

注意

根据您的架构选择正确的产品变体。

流程

使用 MCG 命令行界面

- 在 MCG 命令行界面中运行以下命令：

```
noobaa backingstore create aws-s3 <backingstore_name> --access-key=<AWS ACCESS KEY> --secret-key=<AWS SECRET ACCESS KEY> --target-bucket <bucket-name> -n openshift-storage
```

<backingstore_name>

后备储存的名称。

<AWS ACCESS KEY> 和 <AWS SECRET ACCESS KEY>

您创建的 AWS 访问密钥 ID 和 secret 访问密钥。

<bucket-name>

现有 AWS 存储桶名称。此参数告知 MCG 将哪一个存储桶用作其后备存储的目标存储桶，以及随后数据存储和管理。
输出结果类似如下：

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "aws-resource"
INFO[0002]   Created: Secret "backing-store-secret-aws-resource"
```

使用 YAML 添加存储资源

1. 使用凭证创建 secret：

```
apiVersion: v1
kind: Secret
metadata:
  name: <backingstore-secret-name>
  namespace: openshift-storage
type: Opaque
data:
  AWS_ACCESS_KEY_ID: <AWS ACCESS KEY ID ENCODED IN BASE64>
  AWS_SECRET_ACCESS_KEY: <AWS SECRET ACCESS KEY ENCODED IN BASE64>
```

<AWS ACCESS KEY>* 和 *<AWS SECRET ACCESS KEY>

使用 Base64 提供并编码您自己的 AWS 访问密钥 ID 和 secret 访问密钥，并使用结果代替 **<AWS ACCESS KEY ID ENCODED IN BASE64>** 和 **<AWS SECRET ACCESS KEY ENCODED IN BASE64>**。

<backingstore-secret-name>

上一步中创建的后备存储 secret 的名称。

2. 为特定的后备存储应用以下 YAML:

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
spec:
  awsS3:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBucket: <bucket-name>
  type: aws-s3
```

<bucket-name>

现有 AWS 存储桶名称。

<backingstore-secret-name>

上一步中创建的后备存储 secret 的名称。

3.3.2. 创建 AWS-STS 支持的后备存储

Amazon Web Services Security Token Service (AWS STS)是一个 AWS 功能，它是一种使用短期凭证进行身份验证的方法。创建 AWS-STS 支持的后备存储涉及以下内容：

- 使用脚本创建 AWS 角色，这有助于获取角色会话的临时安全凭证
- 在 AWS STS OpenShift 集群中安装 OpenShift Data Foundation Operator
- 在 AWS STS OpenShift 集群中创建后备存储

3.3.2.1. 使用脚本创建 AWS 角色

您需要创建一个角色，并在安装 OpenShift Data Foundation 操作器时传递角色 Amazon 资源名称 (ARN)。

先决条件

- 使用 AWS STS 配置 Red Hat OpenShift Container Platform 集群。如需更多信息，请参阅配置 AWS 集群以使用短期凭证。

流程

- 使用与 OpenShift Data Foundation 上 Multicloud Object Gateway (MCG)的 OpenID Connect (OIDC)配置匹配的脚本创建一个 AWS 角色。
  以下示例显示了创建角色所需的详情：

```
{
   “Version”: “2012-10-17",
   “Statement”: [
      {
         “Effect”: “Allow”,
         “Principal”: {
            “Federated”: “arn:aws:iam::123456789123:oidc-provider/mybucket-oidc.s3.us-east-2.amazonaws.com”
         },
         “Action”: “sts:AssumeRoleWithWebIdentity”,
         “Condition”: {
            “StringEquals”: {
               “mybucket-oidc.s3.us-east-2.amazonaws.com:sub”: [
                  “system:serviceaccount:openshift-storage:noobaa”,
                  “system:serviceaccount:openshift-storage:noobaa-endpoint”
               ]
            }
         }
      }
   ]
}
```

其中

123456789123
是 AWS 帐户 ID

MyBucket
是存储桶名称（使用公共存储桶配置）

us-east-2
是 AWS 区域

openshift-storage
是命名空间名称

脚本示例

```
#!/bin/bash
set -x

# This is a sample script to help you deploy MCG on AWS STS cluster.
```

```
# This script shows how to create role-policy and then create the role in AWS.
# For more information see: https://docs.openshift.com/rosa/authentication/assuming-an-aws-iam-role-for-a-service-account.html

# WARNING: This is a sample script. You need to adjust the variables based on your requirement.

# Variables :
# user variables - REPLACE these variables with your values:
ROLE_NAME="<role-name>" # role name that you pick in your AWS account
NAMESPACE="<namespace>" # namespace name where MCG is running. For OpenShift Data Foundation, it is openshift-storage.

# MCG variables
SERVICE_ACCOUNT_NAME_1="<service-account-name-1>" # The service account name of statefulset core and deployment operator (MCG operator)
SERVICE_ACCOUNT_NAME_2="<service-account-name-2>" # The service account name of deployment endpoint (MCG endpoint)

# AWS variables
# Make sure these values are not empty (AWS_ACCOUNT_ID, OIDC_PROVIDER)
# AWS_ACCOUNT_ID is your AWS account number
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
# If you want to create the role before using the cluster, replace this field too.
# The OIDC provider is in the structure:
# 1) <OIDC-bucket>.s3.<aws-region>.amazonaws.com. for OIDC bucket configurations are in an S3 public bucket
# 2) `<characters>.cloudfront.net` for OIDC bucket configurations in an S3 private bucket with a public CloudFront distribution URL
OIDC_PROVIDER=$(oc get authentication cluster -ojson | jq -r .spec.serviceAccountIssuer | sed -e "s/^https:\/\///")
# the permission (S3 full access)
POLICY_ARN_STRINGS="arn:aws:iam::aws:policy/AmazonS3FullAccess"

# Creating the role (with AWS command line interface)

read -r -d '' TRUST_RELATIONSHIP <<EOF
{
 "Version": "2012-10-17",
 "Statement": [
   {
  "Effect": "Allow",
  "Principal": {
    "Federated": "arn:aws:iam::${AWS_ACCOUNT_ID}:oidc-provider/${OIDC_PROVIDER}"
  },
  "Action": "sts:AssumeRoleWithWebIdentity",
  "Condition": {
    "StringEquals": {
     "${OIDC_PROVIDER}:sub": [
       "system:serviceaccount:${NAMESPACE}:${SERVICE_ACCOUNT_NAME_1}",
       "system:serviceaccount:${NAMESPACE}:${SERVICE_ACCOUNT_NAME_2}"
      ]
    }
  }
   }
```

```
 ]
}
EOF

echo "${TRUST_RELATIONSHIP}" > trust.json

aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-document
file://trust.json --description "role for demo"

while IFS= read -r POLICY_ARN; do
   echo -n "Attaching $POLICY_ARN ... "
   aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "${POLICY_ARN}"
   echo "ok."
done <<< "$POLICY_ARN_STRINGS"
```

3.3.2.2. 在 AWS STS OpenShift 集群中安装 OpenShift Data Foundation Operator

先决条件

- 使用 AWS STS 配置 Red Hat OpenShift Container Platform 集群。如需更多信息，请参阅配置 AWS 集群以使用短期凭证。
- 使用与 OpenID Connect (OIDC)配置匹配的脚本创建一个 AWS 角色。如需更多信息，请参阅使用 脚本创建 AWS 角色。

流程

- 从 Operator Hub 安装 OpenShift Data Foundation Operator。
  - 在安装过程中，在 **ARN Details** 字段中添加角色 ARN。
  - 确保 **Update approval** 字段设置为 **Manual**。

3.3.2.3. 创建新的 AWS STS 后备存储

先决条件

- 使用 AWS STS 配置 Red Hat OpenShift Container Platform 集群。如需更多信息，请参阅配置 AWS 集群以使用短期凭证。
- 使用与 OpenID Connect (OIDC)配置匹配的脚本创建一个 AWS 角色。如需更多信息，请参阅使用 脚本创建 AWS 角色。
- 安装 OpenShift Data Foundation Operator。如需更多信息，请参阅在 AWS STS OpenShift 集群上安装 OpenShift Data Foundation Operator。

流程

1. 安装多云对象网关(MCG)。
   它使用简短的凭据与默认后备存储一起安装。

2. 在 MCG 系统就绪后，您可以使用以下 MCG 命令行界面命令创建类型为 **aws-sts-s3** 类型的后备存储：

```
$ noobaa backingstore create aws-sts-s3 <backingstore-name> --aws-sts-arn=<aws-sts-role-arn> --region=<region> --target-bucket=<target-bucket>
```

其中

backingstore-name
后备存储的名称

aws-sts-role-arn
将假定角色的 AWS STS 角色 ARN

region
AWS 存储桶区域

target-bucket
云上的目标存储桶名称

3.3.3. 创建 IBM COS 支持的后备存储

先决条件

- 下载多云对象网关(MCG)命令行界面。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注意

指定使用订阅管理器启用存储库的适当架构。例如，

- 对于 IBM Power，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- 对于 IBM Z，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- 另外，您还可以从位于 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages的 OpenShift Data Foundation RPM 安装 MCG 软件包

注意

根据您的架构选择正确的产品变体。

流程

使用命令行界面

1. 在 MCG 命令行界面中运行以下命令：

```
noobaa backingstore create ibm-cos <backingstore_name> --access-key=<IBM ACCESS
KEY> --secret-key=<IBM SECRET ACCESS KEY> --endpoint=<IBM COS ENDPOINT> --
target-bucket <bucket-name> -n openshift-storage
```

<backingstore_name>

后备储存的名称。

<IBM ACCESS KEY>, <IBM SECRET ACCESS KEY>, 和 <IBM COS ENDPOINT>

IBM 访问密钥 ID、secret 访问密钥和适当的区域端点，对应于现有 IBM 存储桶的位置。
要在 IBM 云中生成上述密钥，您必须在为您的目标存储桶创建服务凭证时包含 HMAC 凭证。

<bucket-name>

现有 IBM 存储桶名称。此参数告知 MCG 将哪一个存储桶用作其后备存储的目标存储桶，以及随后数据存储和管理。
输出结果类似如下：

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "ibm-resource"
INFO[0002]   Created: Secret "backing-store-secret-ibm-resource"
```

使用 YAML 添加存储资源

1. 使用凭证创建 secret：

```
apiVersion: v1
kind: Secret
metadata:
  name: <backingstore-secret-name>
  namespace: openshift-storage
type: Opaque
data:
  IBM_COS_ACCESS_KEY_ID: <IBM COS ACCESS KEY ID ENCODED IN BASE64>
  IBM_COS_SECRET_ACCESS_KEY: <IBM COS SECRET ACCESS KEY ENCODED IN
BASE64>
```

<IBM COS ACCESS KEY ID ENCODED IN BASE64> 和 <IBM COS SECRET ACCESS KEY ENCODED IN BASE64>

使用 Base64 提供并编码您自己的 IBM COS 访问密钥 ID 和 secret 访问密钥，并使用这些属性的结果来代替这些属性。

<backingstore-secret-name>

后备存储 secret 的名称。

2. 为特定的后备存储应用以下 YAML:

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
```

```
  namespace: openshift-storage
spec:
  ibmCos:
    endpoint: <endpoint>
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBucket: <bucket-name>
  type: ibm-cos
```

<bucket-name>

现有的 IBM COS 存储桶名称。此参数告知 MCG 将哪一个存储桶用作其后备存储的目标存储桶，以及随后数据存储和管理。

<endpoint>

与现有 IBM 存储桶名称的位置对应的区域端点。此参数指示 MCG 关于用于其后备存储的端点，然后是数据存储和管理。

<backingstore-secret-name>

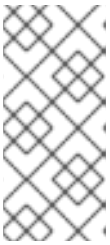
上一步中创建的 secret 的名称。

3.3.4. 创建 Azure 支持的后备存储

先决条件

- 下载多云对象网关(MCG)命令行界面。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```



注意

指定使用订阅管理器启用存储库的适当架构。例如，如果是 IBM Z，请使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- 另外，您还可以从位于 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages的 OpenShift Data Foundation RPM 安装 MCG 软件包

注意

根据您的架构选择正确的产品变体。

流程

使用 MCG 命令行界面

- 在 MCG 命令行界面中运行以下命令：

```
noobaa backingstore create azure-blob <backingstore_name> --account-key=<AZURE ACCOUNT KEY> --account-name=<AZURE ACCOUNT NAME> --target-blob-container <blob container name> -n openshift-storage
```

<backingstore_name>

后备储存的名称。

<AZURE ACCOUNT KEY> 和 <AZURE ACCOUNT NAME>

您为此创建的 AZURE 帐户密钥和帐户名称。

<blob container name>

现有的 Azure blob 容器名称。此参数告知 MCG 将哪一个存储桶用作其后备存储的目标存储桶，以及随后数据存储和管理。
输出结果类似如下：

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "azure-resource"
INFO[0002]   Created: Secret "backing-store-secret-azure-resource"
```

使用 YAML 添加存储资源

1. 使用凭证创建 secret：

```
apiVersion: v1
kind: Secret
metadata:
  name: <backingstore-secret-name>
type: Opaque
data:
  AccountName: <AZURE ACCOUNT NAME ENCODED IN BASE64>
  AccountKey: <AZURE ACCOUNT KEY ENCODED IN BASE64>
```

<AZURE ACCOUNT NAME ENCODED IN BASE64> 和 <AZURE ACCOUNT KEY ENCODED IN BASE64>

使用 Base64 提供并编码您自己的 Azure 帐户名称和帐户密钥，并分别使用这些属性代替这些属性。

<backingstore-secret-name>

唯一的后备存储 secret 名称。

2. 为特定的后备存储应用以下 YAML:

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
spec:
  azureBlob:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBlobContainer: <blob-container-name>
  type: azure-blob
```

<blob-container-name>

现有的 Azure blob 容器名称。此参数告知 MCG 将哪一个存储桶用作其后备存储的目标存储桶，以及随后数据存储和管理。

<backingstore-secret-name>

使用上一步中创建的 secret 的名称。

3.3.5. 创建由 GCP 支持的后备存储

先决条件

- 下载多云对象网关(MCG)命令行界面。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注意

指定使用订阅管理器启用存储库的适当架构。例如，如果是 IBM Z，请使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- 另外，您还可以从位于 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages的 OpenShift Data Foundation RPM 安装 MCG 软件包

注意

根据您的架构选择正确的产品变体。

流程

使用 MCG 命令行界面

- 在 MCG 命令行界面中运行以下命令：

```
noobaa backingstore create google-cloud-storage <backingstore_name> --private-key-json-file=<PATH TO GCP PRIVATE KEY JSON FILE> --target-bucket <GCP bucket name> -n openshift-storage
```

<backingstore_name>

后备储存的名称。

<PATH TO GCP PRIVATE KEY JSON FILE>

为此目的创建的 GCP 私钥的路径。

<GCP bucket name>

现有的 GCP 对象存储存储桶名称。此参数告知 MCG 将哪一个存储桶用作其后备存储的目标存储桶，以及随后数据存储和管理。

输出结果类似如下：

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "google-gcp"
INFO[0002]   Created: Secret "backing-store-google-cloud-storage-gcp"
```

使用 YAML 添加存储资源

1. 使用凭证创建 secret：

```
apiVersion: v1
kind: Secret
metadata:
  name: <backingstore-secret-name>
type: Opaque
data:
  GoogleServiceAccountPrivateKeyJson: <GCP PRIVATE KEY ENCODED IN BASE64>
```

<GCP PRIVATE KEY ENCODED IN BASE64>

使用 Base64 提供并编码您自己的 GCP 服务帐户私钥，并将结果用于此属性。

<backingstore-secret-name>

后备存储 secret 的唯一名称。

2. 为特定的后备存储应用以下 YAML:

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
spec:
  googleCloudStorage:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBucket: <target bucket>
  type: google-cloud-storage
```

<target bucket>

现有的 Google 存储桶。此参数告知 MCG 将哪一个存储桶用作其后备存储的目标存储桶，以及随后数据存储和管理。

<backingstore-secret-name>

上一步中创建的 secret 的名称。

3.3.6. 创建由本地持久性卷支持的后备存储

先决条件

- 下载多云对象网关(MCG)命令行界面。

-

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注意

指定适当的架构，以使用订阅管理器启用存储库。

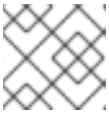
- 对于 IBM Power，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- 对于 IBM Z，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- 另外，您还可以从位于 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages的 OpenShift Data Foundation RPM 安装 MCG 软件包



注意

根据您的架构选择正确的产品变体。

流程

使用 MCG 命令行界面添加存储资源

- 在 MCG 命令行界面中运行以下命令：

注意

此命令必须从 **openshift-storage** 命名空间内运行。

```
$ noobaa -n openshift-storage backingstore create pv-pool <backingstore_name> --num-volumes <NUMBER OF VOLUMES>  --pv-size-gb <VOLUME SIZE> --request-cpu <CPU REQUEST> --request-memory <MEMORY REQUEST> --limit-cpu <CPU LIMIT> --limit-memory <MEMORY LIMIT> --storage-class <LOCAL STORAGE CLASS>
```

使用 YAML 添加存储资源

- 为特定的后备存储应用以下 YAML:

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <backingstore_name>
  namespace: openshift-storage
spec:
   pvPool:
```

```
    numVolumes: <NUMBER OF VOLUMES>
    resources:
      requests:
        storage: <VOLUME SIZE>
        cpu: <CPU REQUEST>
        memory: <MEMORY REQUEST>
      limits:
        cpu: <CPU LIMIT>
        memory: <MEMORY LIMIT>
    storageClass: <LOCAL STORAGE CLASS>
  type: pv-pool
```

<backingstore_name>

后备储存的名称。

<NUMBER OF VOLUMES>

要创建的卷数量。请注意，增加卷数量可向上扩展存储。

<VOLUME SIZE>

每个卷所需的大小（以 GB 为单位）。

<CPU REQUEST>

保证的 CPU 请求，以 CPU 单元 **m** 为单位。

<MEMORY REQUEST>

保证请求的内存量。

<CPU LIMIT>

可消耗的最大 CPU 量，以 CPU 单元 **m** 为单位。

<MEMORY LIMIT>

可消耗的最大内存量。

<LOCAL STORAGE CLASS>

本地存储类名称，建议使用 **ocs-storagecluster-ceph-rbd**。
输出结果类似如下：

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Exists: BackingStore "local-mcg-storage"
```

3.4. 创建兼容 S3 的多云对象网关后备存储

多云对象网关(MCG)可以使用任何 S3 兼容对象存储作为后备存储，例如，Red Hat Ceph Storage 的 RADOS 对象网关(RGW)。以下步骤演示了如何为 Red Hat Ceph Storage 的 RGW 创建 S3 兼容 MCG 后备存储。请注意，部署 RGW 时，OpenShift Data Foundation operator 会自动为 MCG 创建 S3 兼容后备存储。

流程

1. 在 MCG 命令行界面中运行以下命令：

注意

此命令必须从 **openshift-storage** 命名空间内运行。

```
noobaa backingstore create s3-compatible rgw-resource --access-key=<RGW ACCESS
KEY> --secret-key=<RGW SECRET KEY> --target-bucket=<bucket-name> --endpoint=
<RGW endpoint> -n openshift-storage
```

a. 要获取 **<RGW ACCESS KEY>** 和 **<RGW SECRET KEY>**，请使用您的 RGW 用户 secret 名称运行以下命令：

```
oc get secret <RGW USER SECRET NAME> -o yaml -n openshift-storage
```

b. 解码 Base64 中的访问密钥 ID 和访问密钥，并保留它们。

c. 将 **<RGW USER ACCESS KEY>** 和 **<RGW USER SECRET ACCESS KEY>** 替换为上一步中的相应已解码数据。

d. 将 **<bucket-name>** 替换为现有的 RGW 存储桶名称。此参数告知 MCG 将哪一个存储桶用作其后备存储的目标存储桶，以及随后数据存储和管理。

e. 若要获取 &lt **;RGW endpoint&** gt;，请参阅 访问 RADOS 对象网关 S3 端点。
输出结果类似如下：

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "rgw-resource"
INFO[0002]   Created: Secret "backing-store-secret-rgw-resource"
```

您还可以使用 YAML 创建后备存储：

1. 创建 **CephObjectStore** 用户。这还会创建一个包含 RGW 凭证的 secret：

```
apiVersion: ceph.rook.io/v1
kind: CephObjectStoreUser
metadata:
  name: <RGW-Username>
  namespace: openshift-storage
spec:
  store: ocs-storagecluster-cephobjectstore
  displayName: "<Display-name>"
```

a. 将 **<RGW-Username>** 和 **<Display-name>** 替换为唯一的用户名和显示名称。

2. 为 S3-Compatible 后备存储应用以下 YAML:

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <backingstore-name>
  namespace: openshift-storage
spec:
  s3Compatible:
    endpoint: <RGW endpoint>
    secret:
      name: <backingstore-secret-name>
```

```
    namespace: openshift-storage
  signatureVersion: v4
  targetBucket: <RGW-bucket-name>
type: s3-compatible
```

a. 将 **<backingstore-secret-name>** 替换为上一步中使用 **CephObjectStore** 创建的 secret 的名称。

b. 将 **<bucket-name>** 替换为现有的 RGW 存储桶名称。此参数告知 MCG 将哪一个存储桶用作其后备存储的目标存储桶，以及随后数据存储和管理。

c. 若要获取 &lt **;RGW endpoint&** gt;，请参阅 访问 RADOS 对象网关 S3 端点。

3.5. 创建新存储桶类

bucket 类是一个 CRD，代表一组存储桶，用于定义对象 Bucket 类的分层策略和数据放置。

在 OpenShift Data Foundation 中创建存储桶类。

流程

1. 在 OpenShift Web 控制台中，点 **Storage → Object Storage**。
2. 单击 **Bucket Class** 选项卡。
3. 点 **Create Bucket Class**。
4. 在 **Create new Bucket Class** 页面中，执行以下操作：
   a. 选择 bucket 类类型，再输入 bucket 类名称。
      i. 选择 **BucketClass 类型**。选择以下选项之一：
         - **Standard** ：数据将由多云对象网关(MCG)使用，已取消、压缩和加密。
         - **Namespace** ：数据存储在 Namespace 存储中，而无需执行重复数据删除、压缩或加密。
           默认选择 **Standard**。
      ii. 输入 **Bucket Class Name**。
      iii. 点 **Next**。
   b. 在 **放置策略** 中，选择 **Tier 1 - Policy Type** 并单击 **Next**。您可以根据要求选择任一选项。
      - **Spread** 允许在选定资源之间分散数据。
      - **Mirror** 允许在选定资源中完全重复数据。
      - 单击 **Add Tier** 以添加另一个策略层。
   c. 如果您为 **Spread** 选择了 **Tier 1 - Policy Type**，从可用列表中选择至少一个 **Backing Store** 资源并点 **Next**。或者，您也可以 创建新的后备存储。

注意

当在上一步中选择 Policy Type 作为 Mirror 时，至少需要选择 2 个后备存储。

d. 检查并确认 Bucket 类设置。

e. 点 **Create Bucket Class**。

验证步骤

1. 在 OpenShift Web 控制台中，点 **Storage → Object Storage**。
2. 单击 **Bucket Class** 选项卡，再搜索新的 Bucket 类。

3.6. 编辑存储桶类

使用以下步骤，通过 YAML 文件编辑存储桶类组件，方法是点击 Openshift Web 控制台上的 **edit** 按钮。

先决条件

- 管理员对 OpenShift Web 控制台的访问权限。

流程

1. 在 OpenShift Web 控制台中，点 **Storage → Object Storage**。
2. 单击 **Bucket Class** 选项卡。
3. 点击您要编辑的 Bucket 类旁边的 Action Menu (⋮)。
4. 点 **Edit Bucket Class**。
5. 您将被重定向到 **YAML** 文件，在此文件中进行必要的更改并点 **Save**。

3.7. 为存储桶类编辑后备存储

使用以下步骤编辑现有的多云对象网关(MCG)存储桶类，以更改存储桶类中使用的底层后备存储。

先决条件

- 管理员对 OpenShift Web 控制台的访问权限。
- 存储桶类。
- 后备存储。

流程

1. 在 OpenShift Web 控制台中，点 **Storage → Object Storage**。
2. 单击 **Bucket Class** 选项卡。
3. 点击您要编辑的 Bucket 类旁边的 Action Menu (⋮)。
4. 点 **Edit Bucket Class Resources**。
5. 在 **Edit Bucket Class Resources** 页面中，通过添加后备存储到存储桶类或从存储桶类中删除后备存储来编辑存储桶类资源。您还可以编辑使用一或两个分层和不同的放置策略创建的 bucket 类资源。

- 要将后备存储添加到 bucket 类，请选择后备存储的名称。
- 要从存储桶类中删除后备存储，请取消选中后备存储的名称。

6. 点击 **Save**。

第 4 章 管理命名空间存储桶

命名空间存储桶可让您将不同提供程序上的数据存储库连接在一起，以便您可以通过统一视图与所有数据交互。将与各个提供程序关联的对象存储桶添加到命名空间存储桶，并通过命名空间存储桶访问您的数据，以一次性查看所有对象存储桶。这可让您在读取多个其他存储提供商的同时写入您首选的存储供应商，从而显著降低迁移至新存储提供商的成本。

注意

只有其 **write** 目标可用且可正常运行时，才能使用命名空间存储桶。

4.1. 命名空间存储桶中对象的 AMAZON S3 API 端点

您可以使用 Amazon Simple Storage Service(S3) API 与命名空间存储桶中的对象交互。

Red Hat OpenShift Data Foundation 4.6 之后支持以下命名空间存储桶操作：

- ListObjectVersions
- ListObjects
- PutObject
- CopyObject
- ListParts
- CreateMultipartUpload
- CompleteMultipartUpload
- UploadPart
- UploadPartCopy
- AbortMultipartUpload
- GetObjectAcl
- GetObject
- HeadObject
- DeleteObject
- DeleteObjects

有关这些操作及其使用方法的最新信息，请参阅 Amazon S3 API 参考文档。

其他资源

- Amazon S3 REST API 参考
- Amazon S3 CLI 参考

4.2. 使用 MULTICLOUD 对象网关 CLI 和 YAML 添加命名空间存储桶

如需有关命名空间存储桶的更多信息，请参阅管理命名空间存储桶。

根据部署的类型以及是否使用 YAML 或 Multicloud 对象网关 (MCG) CLI，选择以下流程之一来添加命名空间存储桶：

- 使用 YAML 添加 AWS S3 命名空间存储桶
- 使用 YAML 添加 IBM COS 命名空间存储桶
- 使用 Multicloud 对象网关 CLI 添加 AWS S3 命名空间存储桶
- 使用 Multicloud 对象网关 CLI 添加 IBM COS 命名空间存储桶

4.2.1. 使用 YAML 添加 AWS S3 命名空间存储桶

先决条件

- 安装了带有 OpenShift Data Foundation operator 的 OpenShift Container Platform。
- 访问多云对象网关(MCG)。
  如需更多信息，请参阅第 2 章，使用应用程序访问多云对象网关。

流程

1. 使用凭证创建 secret：

```
apiVersion: v1
kind: Secret
metadata:
  name: <namespacestore-secret-name>
  type: Opaque
data:
  AWS_ACCESS_KEY_ID: <AWS ACCESS KEY ID ENCODED IN BASE64>
  AWS_SECRET_ACCESS_KEY: <AWS SECRET ACCESS KEY ENCODED IN BASE64>
```

   其中 **<namespacestore-secret-name>** 是一个唯一的 NamespaceStore 名称。

   您必须使用 **Base64** 提供并编码您自己的 AWS 访问密钥 ID 和 secret 访问密钥，并使用其结果替换 **<AWS ACCESS KEY ID ENCODED IN BASE64>** 和 **<AWS SECRET ACCESS KEY ENCODED IN BASE64>**。

2. 使用 OpenShift 自定义资源定义(CRD)创建 NamespaceStore 资源。
   NamespaceStore 代表底层存储，用作 MCG 命名空间存储桶中数据的**读取**或**写入**目标。

   要创建 NamespaceStore 资源，请应用以下 YAML：

```
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
```

```
  name: <resource-name>
  namespace: openshift-storage
spec:
  awsS3:
    secret:
      name: <namespacestore-secret-name>
      namespace: <namespace-secret>
    targetBucket: <target-bucket>
  type: aws-s3
```

<resource-name>

您要提供给资源的名称。

<namespacestore-secret-name>

上一步中创建的 secret。

<namespace-secret>

包括 secret 的命名空间。

<target-bucket>

为 NamespaceStore 创建的目标存储桶。

3. 创建一个命名空间存储桶类，为命名空间存储桶定义命名空间策略。命名空间策略的类型需要是 **single** 或 **multi**。

- 一个类型为 **single** 的命名空间策略需要以下配置：

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <my-bucket-class>
  namespace: openshift-storage
spec:
  namespacePolicy:
    type:
    single:
      resource: <resource>
```

<my-bucket-class>

唯一的命名空间存储桶类名称。

<resource>

定义命名空间存储桶的读写目标的单个 NamespaceStore 的名称。

- 一个类型为 **multi** 的命名空间策略需要以下配置：

```
apiVersion: noobaa.io/v1alpha1

kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <my-bucket-class>
  namespace: openshift-storage
```

```
spec:
  namespacePolicy:
    type: Multi
    multi:
      writeResource: <write-resource>
      readResources:
      - <read-resources>
      - <read-resources>
```

<my-bucket-class>

唯一的存储桶类名称。

<write-resource>

定义命名空间存储桶 **写入** 目标的单个 NamespaceStore。

<read-resources>

定义命名空间存储桶的**读取**目标的 NamespaceStore 的名称列表。

4. 使用以下 YAML 使用上一步中定义的存储桶类(OBC)资源，创建存储桶：

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: <resource-name>
  namespace: openshift-storage
spec:
  generateBucketName: <my-bucket>
  storageClassName: openshift-storage.noobaa.io
  additionalConfig:
    bucketclass: <my-bucket-class>
```

<resource-name>

您要提供给资源的名称。

<my-bucket>

您要提供给存储桶的名称。

<my-bucket-class>

上一步中创建的 bucket 类。

在 Operator 置备 OBC 后，会在 MCG 中创建存储桶，Operator 会创建一个具有相同名称的 **Secret** 和 **ConfigMap**，并在与 OBC 相同的命名空间中创建 Secret 和 ConfigMap。

4.2.2. 使用 YAML 添加 IBM COS 命名空间存储桶

先决条件

- 安装了带有 OpenShift Data Foundation operator 的 OpenShift Container Platform。
- 访问多云对象网关（MCG），请参阅第 2 章，使用应用程序访问多云对象网关

流程

1. 使用凭证创建 secret：

```
apiVersion: v1
kind: Secret
metadata:
  name: <namespacestore-secret-name>
  type: Opaque
data:
  IBM_COS_ACCESS_KEY_ID: <IBM COS ACCESS KEY ID ENCODED IN BASE64>
  IBM_COS_SECRET_ACCESS_KEY: <IBM COS SECRET ACCESS KEY ENCODED IN
BASE64>
```

<namespacestore-secret-name>

唯一的 NamespaceStore 名称。
您必须使用 **Base64** 提供和编码您自己的 IBM COS 访问密钥 ID 和 secret 访问密钥，并使用其结果替代 **<IBM COS ACCESS KEY ID ENCODED IN BASE64>** 和 **<IBM COS SECRET ACCESS KEY ENCODED IN BASE64>**。

2. 使用 OpenShift 自定义资源定义(CRD)创建 NamespaceStore 资源。
NamespaceStore 代表底层存储，用作 MCG 命名空间存储桶中数据的**读取**或**写入**目标。

要创建 NamespaceStore 资源，请应用以下 YAML：

```
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
spec:
  s3Compatible:
    endpoint: <IBM COS ENDPOINT>
    secret:
      name: <namespacestore-secret-name>
      namespace: <namespace-secret>
    signatureVersion: v2
    targetBucket: <target-bucket>
  type: ibm-cos
```

<IBM COS ENDPOINT>

适当的 IBM COS 端点。

<namespacestore-secret-name>

上一步中创建的 secret。

<namespace-secret>

包括 secret 的命名空间。

<target-bucket>

为 NamespaceStore 创建的目标存储桶。

3. 创建一个命名空间存储桶类，为命名空间存储桶定义命名空间策略。命名空间策略的类型需要是 **single** 或 **multi**。

- 一个类型为 **single** 的命名空间策略需要以下配置：

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <my-bucket-class>
  namespace: openshift-storage
spec:
  namespacePolicy:
    type:
    single:
      resource: <resource>
```

<my-bucket-class>

唯一的命名空间存储桶类名称。

<resource>

定义命名空间存储桶的**读和写**目标的单个 NamespaceStore 的名称。

- 类型为 **multi** 的命名空间策略需要以下配置：

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <my-bucket-class>
  namespace: openshift-storage
spec:
  namespacePolicy:
    type: Multi
    multi:
      writeResource: <write-resource>
      readResources:
      - <read-resources>
      - <read-resources>
```

<my-bucket-class>

唯一的存储桶类名称。

<write-resource>

定义命名空间存储桶写入目标的单个 NamespaceStore。

<read-resources>

NamespaceStores 名称列表，用于定义命名空间存储桶的**读取**目标。

4. 要使用上一步中定义的 bucket 类的 Object Bucket Class (OBC) 资源创建存储桶，请应用以下 YAML：

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: <resource-name>
  namespace: openshift-storage
```

```
spec:
  generateBucketName: <my-bucket>
  storageClassName: openshift-storage.noobaa.io
  additionalConfig:
    bucketclass: <my-bucket-class>
```

<resource-name>

您要提供给资源的名称。

<my-bucket>

您要提供给存储桶的名称。

<my-bucket-class>

上一步中创建的 bucket 类。
在 Operator 置备 OBC 后，会在 MCG 中创建存储桶，Operator 会创建一个具有相同名称的 **Secret** 和 **ConfigMap**，并在与 OBC 相同的命名空间中创建 Secret 和 ConfigMap。

4.2.3. 使用 Multicloud 对象网关 CLI 添加 AWS S3 命名空间存储桶

先决条件

- 安装了带有 OpenShift Data Foundation operator 的 OpenShift Container Platform。
- 访问多云对象网关（MCG），请参阅第 2 章，使用应用程序访问多云对象网关
- 下载 MCG 命令行界面：

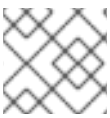
```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注意

指定适当的架构，以使用订阅管理器启用存储库。例如，如果是 IBM Z，请使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

另外，您还可以从位于 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/package的 OpenShift Data Foundation RPM 安装 MCG 软件包。



注意

根据您的架构选择正确的产品变体。

流程

1. 在 MCG 命令行界面中，创建一个 NamespaceStore 资源。
NamespaceStore 代表底层存储，用作 MCG 命名空间存储桶中数据的读取或写入目标。

```
$ noobaa namespacestore create aws-s3 <namespacestore> --access-key <AWS ACCESS KEY> --secret-key <AWS SECRET ACCESS KEY> --target-bucket <bucket-name> -n openshift-storage
```

<namespacestore>

NamespaceStore 的名称。

<AWS ACCESS KEY> 和 <AWS SECRET ACCESS KEY>

您创建的 AWS 访问密钥 ID 和 secret 访问密钥。

<bucket-name>

现有 AWS 存储桶名称。此参数告知 MCG 将哪一个存储桶用作其后备存储的目标存储桶，以及随后数据存储和管理。

2. 创建一个命名空间存储桶类，为命名空间存储桶定义命名空间策略。命名空间策略可以是 **single** 或 **multi**。

- 创建一个命名空间存储桶类，其命名空间策略类型为 **single** ：

```
$ noobaa bucketclass create namespace-bucketclass single <my-bucket-class> --
resource <resource> -n openshift-storage
```

<resource-name>

要为资源指定名称。

<my-bucket-class>

唯一的存储桶类名称。

<resource>

定义命名空间存储桶的**读**和**写**目标的单个 NamespaceStore 的名称。

- 创建一个命名空间存储桶类，其命名空间策略类型为 **multi** ：

```
$ noobaa bucketclass create namespace-bucketclass multi <my-bucket-class> --write-
resource <write-resource> --read-resources <read-resources> -n openshift-storage
```

<resource-name>

要为资源指定名称。

<my-bucket-class>

唯一的存储桶类名称。

<write-resource>

定义命名空间存储桶的**写入**目标的单个 namespace-store。

<read-resources>s

命名空间存储列表，用逗号隔开，用于定义命名空间存储桶的**读取**目标。

3. 使用上一步中定义的 bucket 类的对象 Bucket Class(OBC)资源创建存储桶。

```
$ noobaa obc create my-bucket-claim -n openshift-storage --app-namespace my-app --
bucketclass <custom-bucket-class>
```

<bucket-name>

您选择的存储桶名称。

<custom-bucket-class>

上一步中创建的 bucket 类的名称。

在 Operator 置备 OBC 后，会在 MCG 中创建存储桶，Operator 会创建一个具有相同名称的 **Secret** 和 **ConfigMap**，并在与 OBC 相同的命名空间中创建 Secret 和 ConfigMap。

4.2.4. 使用 Multicloud 对象网关 CLI 添加 IBM COS 命名空间存储桶

先决条件

- 安装了带有 OpenShift Data Foundation operator 的 OpenShift Container Platform。
- 访问多云对象网关（MCG），请参阅第 2 章，使用应用程序访问多云对象网关
- 下载 MCG 命令行界面：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注意

指定适当的架构，以使用订阅管理器启用存储库。

- 对于 IBM Power，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- 对于 IBM Z，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

另外，您还可以从位于 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/package的 OpenShift Data Foundation RPM 安装 MCG 软件包。

注意

根据您的架构选择正确的产品变体。

流程

1. 在 MCG 命令行界面中，创建一个 NamespaceStore 资源。
NamespaceStore 代表底层存储，用作 MCG 命名空间存储桶中数据的**读取**或**写入**目标。

```
$ noobaa namespacestore create ibm-cos <namespacestore> --endpoint <IBM COS ENDPOINT> --access-key <IBM ACCESS KEY> --secret-key <IBM SECRET ACCESS KEY> --target-bucket <bucket-name> -n openshift-storage
```

<namespacestore>

NamespaceStore 的名称。

<IBM ACCESS KEY>, <IBM SECRET ACCESS KEY>, <IBM COS ENDPOINT>

IBM 访问密钥 ID、secret 访问密钥和适当的区域端点，对应于现有 IBM 存储桶的位置。

<bucket-name>

现有 IBM 存储桶名称。此参数告知 MCG 将哪一个存储桶用作其后备存储的目标存储桶，以及随后数据存储和管理。

2. 创建一个命名空间存储桶类，为命名空间存储桶定义命名空间策略。命名空间策略的类型需要是 **single** 或 **multi**。

   - 创建一个命名空间存储桶类，其命名空间策略类型为 **single** ：

     ```
     $ noobaa bucketclass create namespace-bucketclass single <my-bucket-class> --resource <resource> -n openshift-storage
     ```

     <resource-name>

     要为资源指定名称。

     <my-bucket-class>

     唯一的存储桶类名称。

     <resource>

     定义命名空间存储桶的**读**和**写**目标的单个 NamespaceStore 的名称。

   - 创建一个命名空间存储桶类，其命名空间策略类型为 **multi** ：

     ```
     $ noobaa bucketclass create namespace-bucketclass multi <my-bucket-class> --write-resource <write-resource> --read-resources <read-resources> -n openshift-storage
     ```

     <resource-name>

     要为资源指定名称。

     <my-bucket-class>

     唯一的存储桶类名称。

     <write-resource>

     定义命名空间存储桶的**写入**目标的单个 namespace-store。

     <read-resources>

     以逗号分隔的 NamespaceStore 列表，用于定义命名空间存储桶的**读取**目标。

3. 使用以前步骤中定义的 bucket 类的对象 Bucket Class(OBC)资源创建存储桶。

   ```
   $ noobaa obc create my-bucket-claim -n openshift-storage --app-namespace my-app --bucketclass <custom-bucket-class>
   ```

   <bucket-name>

   您选择的存储桶名称。

   <custom-bucket-class>

   上一步中创建的 bucket 类的名称。

在 Operator 置备 OBC 后，会在 MCG 中创建存储桶，Operator 会创建一个具有相同名称的 **Secret** 和 **ConfigMap**，并在与 OBC 相同的命名空间中创建 Secret 和 ConfigMap。

4.3. 使用 OPENSHIFT CONTAINER PLATFORM 用户界面添加命名空间存储桶

您可以使用 OpenShift Container Platform 用户界面添加命名空间存储桶如需有关命名空间存储桶的信息，请参阅管理命名空间存储桶。

先决条件

- 确保已安装带有 OpenShift Data Foundation operator 的 Openshift Container Platform。
- 访问多云对象网关(MCG)。

流程

1. 在 OpenShift Web 控制台中，进入到 **Storage → Object Storage → Namespace Store** 选项卡。
2. 单击 **Create namespace store**，以创建要在命名空间存储桶中使用的命名空间存储资源。

   a. **输入 命名空间存储 名称。**

   b. **选择供应商和地区。**

   c. **选择现有的 secret，或者点击 Swith to credentials 通过输入 secret key 和 secret access key 来创建 secret。**

   d. **输入目标存储桶。**

   e. **点 Create。**

3. **在 Namespace Store 选项卡中，验证新创建的命名空间 存储 是否处于 Ready 状态。**
4. **重复步骤 2 和 3，直到您创建了所有必需的资源量。**
5. **导航到 Bucket Class 选项卡，再单击 Create Bucket Class。**

   a. **选择 Namespace BucketClass type 单选按钮。**

   b. **输入 BucketClass 名称，然后点 Next。**

c. 为您的命名空间存储桶选择一个 Namespace Policy Type，然后点 Next。

- 如果您的命名空间策略类型是 Single，则需要选择一个读取资源。
- 如果您的命名空间策略类型是 Multi，则需要选择读取资源和写入资源。
- 如果命名空间策略类型是 Cache，则需要选择一个定义命名空间存储桶读取和写入目标的 Hub 命名空间存储。

d. 选择一个 Read 和 Write NamespaceStore，用于定义命名空间存储桶的读取和写入目标，然后点 Next。

e. 检查您的新 bucket 类详情，然后单击 Create Bucket Class。

6. 导航到 Bucket Class 选项卡，并验证新创建的资源是否处于 Ready 阶段。

7. 导航到 Object Bucket Claims 选项卡，再点 Create Object Bucket Claim。

a. 为命名空间存储桶输入 ObjectBucketClaim Name。

b. 选择 StorageClass 作为 openshift-storage.noobaa.io。

c. 从列表中选择您之前为命名空间存储创建的 BucketClass。默认情况下，选择 noobaa-default-bucket-class。

d. 点 Create。命名空间存储桶与您的命名空间的 Object Bucket Claim 一起创建。

8. 导航到 Object Bucket Claims 选项卡，并验证创建的 Object Bucket Claim 是否处于 Bound 状态。

9. 导航到 Object Buckets 选项卡，并验证您的命名空间存储桶是否存在于列表中，且处于

Bound 状态。

4.4. 使用 S3 协议在旧应用程序与云原生应用程序间共享

许多传统应用程序会使用文件系统共享数据集。您可以使用 S3 操作访问和共享文件系统中的传统数据。要共享您需要的数据，请执行以下操作：

- 导出预先存在的文件系统数据集，即 RWX 卷，如 Ceph FileSystem(CephFS)，或使用 S3 协议创建新的文件系统数据集。
- 从文件系统和 S3 协议访问文件系统数据集。
- 配置 S3 帐户，并将它们映射到现有文件系统唯一标识符(UID)和组标识符(GID)。

4.4.1. 创建 NamespaceStore 来使用文件系统

先决条件

- 安装了带有 OpenShift Data Foundation operator 的 OpenShift Container Platform。
- 访问多云对象网关(MCG)。

流程

1. 登录 OpenShift Web 控制台。
2. 点 Storage → Object Storage。
3. 点 NamespaceStore 标签来创建在命名空间存储桶中使用的 NamespaceStore 资源。
4. 点 Create namespacestore。

-

5. 为 **NamespaceStore** 输入一个名称。

6. 选择 **Filesystem** 作为供应商。

7. 选择持久性卷声明。

8. 输入文件夹名称。

   如果文件夹名称已存在，则该文件夹会用于创建 **NamespaceStore**，如果不存在则创建新的文件夹。

9. 点 **Create**。

10. 验证 **NamespaceStore** 是否处于 **Ready** 状态。

4.4.2. 使用 NamespaceStore 文件系统配置创建帐户

您可以使用 **NamespaceStore** 文件系统配置创建新帐户，或通过编辑 **YAML** 将现有的普通帐户转换为 **NamespaceStore** 文件系统帐户。

注意

您无法从帐户中删除 **NamespaceStore** 文件系统配置。

先决条件

- 下载 **Multicloud** 对象网关 **(MCG)** 命令行界面：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

流程

- 使用 **MCG** 命令行界面使用 **NamespaceStore** 文件系统配置创建新帐户。

```
$ noobaa account create <noobaa-account-name> [flags]
```

例如：

```
$ noobaa account create testaccount --full_permission --nsfs_account_config --gid 10001 --uid 10001 –default_resource fs_namespacestore
```

allow_bucket_create	指明是否允许帐户创建新存储桶。支持的值是 **true** 或 **false**。默认值为 **true**。
allowed_buckets	以逗号分隔的存储桶名称列表，用户被允许具有访问权限和管理权限。
--default_resource	使用 S3 CreateBucket 操作时，将在其上创建新的存储桶的 NamespaceStore 资源。NamespaceStore 必须使用 RWX(ReadWriteMany)持久性卷声明(PVC)支持。
full_permission	指明是否应该允许完全权限。支持的值是 **true** 或 **false**。默认值为 **false**。
new_buckets_path	创建与新 bucket 对应的目录的文件系统路径。该路径位于 NamespaceStore 文件系统 PVC 中，其中创建新目录充当新创建的对象存储桶类的文件系统映射。
nsfs_account_config	指示该帐户是否用于 NamespaceStore 文件系统的必需字段。
nsfs_only	指明帐户是否仅用于 NamespaceStore 文件系统。支持的值是 true 或 **false**。默认值为 **false**。如果设为 'true'，则会限制您访问其他类型的存储桶。
uid	MCG 帐户要映射到的文件系统 ID，用于访问和管理文件系统中的数据
gid	MCG 帐户要映射到的文件系统的组 ID，用于访问和管理文件系统中的数据

MCG 系统使用帐户配置及其 S3 凭证发送响应：

```
# NooBaaAccount spec:
allow_bucket_creation: true
Allowed_buckets:
  full_permission: true
  permission_list: []
default_resource: noobaa-default-namespace-store
Nsfs_account_config:
  gid: 10001
  new_buckets_path: /
  nsfs_only: true
```

```
  uid: 10001
INFO[0006]   Exists: Secret "noobaa-account-testaccount"
Connection info:
  AWS_ACCESS_KEY_ID      : <aws-access-key-id>
  AWS_SECRET_ACCESS_KEY  : <aws-secret-access-key>
```

您可以使用以下命令列出所有基于自定义资源定义(CRD)的帐户：

```
$ noobaa account list
NAME          ALLOWED_BUCKETS   DEFAULT_RESOURCE             PHASE   AGE
testaccount   [*]               noobaa-default-backing-store   Ready   1m17s
```

如果您对特定帐户感兴趣，您可以直接读取其自定义资源定义(CRD)：

```
$ oc get noobaaaccount/testaccount -o yaml
spec:
  allow_bucket_creation: true
  allowed_buckets:
    full_permission: true
    permission_list: []
  default_resource: noobaa-default-namespace-store
  nsfs_account_config:
    gid: 10001
    new_buckets_path: /
    nsfs_only: true
    uid: 10001
```

4.4.3. 从 openshift-storage 命名空间中访问旧的应用程序数据

当使用 Multicloud Object Gateway(MCG)NamespaceStore 文件系统(NSFS)功能时，您需要具有数据位于 openshift-storage 命名空间中的 PVC。几乎在所有情形中，您需要访问的数据不在 openshift-storage 命名空间中，而是位于传统应用程序使用的命名空间中。

要访问存储在另一个命名空间中的数据，您需要在 openshift-storage 命名空间中创建一个指向传统应用使用的同一 CephFS 卷的 PVC。

流程

1. **使用 scc 显示应用程序命名空间：**

```
$ oc get ns <application_namespace> -o yaml | grep scc
```

<application_namespace>

指定应用程序命名空间的名称。

例如：

```
$ oc get ns testnamespace -o yaml | grep scc

openshift.io/sa.scc.mcs: s0:c26,c5
openshift.io/sa.scc.supplemental-groups: 1000660000/10000
openshift.io/sa.scc.uid-range: 1000660000/10000
```

2. **进入应用程序命名空间：**

```
$ oc project <application_namespace>
```

例如：

```
$ oc project testnamespace
```

3. **确保 ReadWriteMany(RWX) PVC 挂载到您要使用 MCG NSFS 功能从 noobaa S3 端点使用的 pod 上：**

```
$ oc get pvc

NAME                                          STATUS VOLUME
CAPACITY ACCESS MODES STORAGECLASS              AGE
cephfs-write-workload-generator-no-cache-pv-claim  Bound  pvc-aa58fb91-c3d2-475b-bbee-
68452a613e1a
10Gi     RWX          ocs-storagecluster-cephfs 12s
```

```
$ oc get pod

NAME                                          READY   STATUS            RESTARTS   AGE
cephfs-write-workload-generator-no-cache-1-cv892    1/1     Running           0          11s
```

4. **检查 pod 中的持久性卷(PV)的挂载点。**

   a. **从 pod 获取 PV 的卷名称：**

```
$ oc get pods <pod_name> -o jsonpath='{.spec.volumes[]}'
```

<pod_name>

指定 pod 的名称。

例如：

```
$ oc get pods cephfs-write-workload-generator-no-cache-1-cv892 -o
jsonpath='{.spec.volumes[]}'

{"name":"app-persistent-storage","persistentVolumeClaim":{"claimName":"cephfs-
write-workload-generator-no-cache-pv-claim"}}
```

在本例中，PVC 的卷名称为 cephfs-write-workload-generator-no-cache-pv-claim。

b.

列出 pod 中的所有挂载，并检查您在上一步中标识的卷的挂载点：

```
$ oc get pods <pod_name> -o jsonpath='{.spec.containers[].volumeMounts}'
```

例如：

```
$ oc get pods cephfs-write-workload-generator-no-cache-1-cv892 -o
jsonpath='{.spec.containers[].volumeMounts}'

[{"mountPath":"/mnt/pv","name":"app-persistent-storage"},
{"mountPath":"/var/run/secrets/kubernetes.io/serviceaccount","name":"kube-api-access-
8tnc5","readOnly":true}]
```

5.

确认 pod 中的 RWX PV 的挂载点：

```
$ oc exec -it <pod_name> -- df <mount_path>
```

<mount_path>

指定您在上一步中标识的挂载点的路径。

例如：

```
$ oc exec -it cephfs-write-workload-generator-no-cache-1-cv892 -- df /mnt/pv
```

```
main
Filesystem
1K-blocks Used Available  Use%  Mounted on
172.30.202.87:6789,172.30.120.254:6789,172.30.77.247:6789:/volumes/csi/csi-vol-
cc416d9e-dbf3-11ec-b286-0a580a810213/edcfe4d5-bdcb-4b8e-8824-8a03ad94d67c
10485760  0    10485760   0%    /mnt/pv
```

6. **确保 UID 和 SELinux 标签与旧命名空间使用的标签相同：**

```
$ oc exec -it <pod_name> -- ls -latrZ <mount_path>
```

例如：

```
$ oc exec -it cephfs-write-workload-generator-no-cache-1-cv892 -- ls -latrZ /mnt/pv/

total 567
drwxrwxrwx. 3 root       root system_u:object_r:container_file_t:s0:c26,c5      2 May 25 06:35 .
-rw-r--r--. 1 1000660000 root system_u:object_r:container_file_t:s0:c26,c5 580138 May 25
06:35 fs_write_cephfs-write-workload-generator-no-cache-1-cv892-data.log
drwxrwxrwx. 3 root       root system_u:object_r:container_file_t:s0:c26,c5     30 May 25 06:35
..
```

7. **获取您要从 openshift-storage 命名空间访问的旧应用程序 RWX PV 的信息：**

```
$ oc get pv | grep <pv_name>
```

<pv_name>

指定 PV 的名称。

例如：

```
$ oc get pv | grep pvc-aa58fb91-c3d2-475b-bbee-68452a613e1a

pvc-aa58fb91-c3d2-475b-bbee-68452a613e1a   10Gi       RWX            Delete
Bound    testnamespace/cephfs-write-workload-generator-no-cache-pv-claim   ocs-
storagecluster-cephfs             47s
```

8. **确保来自 openshift-storage 命名空间中的 PVC 可以被访问，以便一个或多个 noobaa-endpoint pod 可以访问 PVC。**

a.

从 volumeAttributes 找到 subvolumePath 和 volumeHandle 的值。您可以从传统应用程序 PV 的 YAML 描述中获取这些值：

```
$ oc get pv <pv_name> -o yaml
```

例如：

```
$ oc get pv pvc-aa58fb91-c3d2-475b-bbee-68452a613e1a -o yaml

apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: openshift-storage.cephfs.csi.ceph.com
  creationTimestamp: "2022-05-25T06:27:49Z"
  finalizers:
  - kubernetes.io/pv-protection
  name: pvc-aa58fb91-c3d2-475b-bbee-68452a613e1a
  resourceVersion: "177458"
  uid: 683fa87b-5192-4ccf-af2f-68c6bcf8f500
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: cephfs-write-workload-generator-no-cache-pv-claim
    namespace: testnamespace
    resourceVersion: "177453"
    uid: aa58fb91-c3d2-475b-bbee-68452a613e1a
  csi:
    controllerExpandSecretRef:
      name: rook-csi-cephfs-provisioner
      namespace: openshift-storage
    driver: openshift-storage.cephfs.csi.ceph.com
    nodeStageSecretRef:
      name: rook-csi-cephfs-node
      namespace: openshift-storage
    volumeAttributes:
      clusterID: openshift-storage
      fsName: ocs-storagecluster-cephfilesystem
      storage.kubernetes.io/csiProvisionerIdentity: 1653458225664-8081-openshift-
storage.cephfs.csi.ceph.com
      subvolumeName: csi-vol-cc416d9e-dbf3-11ec-b286-0a580a810213
      subvolumePath: /volumes/csi/csi-vol-cc416d9e-dbf3-11ec-b286-
0a580a810213/edcfe4d5-bdcb-4b8e-8824-8a03ad94d67c
    volumeHandle: 0001-0011-openshift-storage-0000000000000001-cc416d9e-dbf3-
11ec-b286-0a580a810213
  persistentVolumeReclaimPolicy: Delete
  storageClassName: ocs-storagecluster-cephfs
```

```
  volumeMode: Filesystem
status:
  phase: Bound
```

b. **使用您在上一步中确定的 subvolumePath 和 volumeHandle 值在 openshift-storage 命名空间中创建一个新的 PV 和 PVC 对象，指向与旧应用程序 PV 相同的 CephFS 卷：**

***YAML 文件示例*：**

```
$ cat << EOF >> pv-openshift-storage.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cephfs-pv-legacy-openshift-storage
spec:
  storageClassName: ""
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi     (1)
  csi:
    driver: openshift-storage.cephfs.csi.ceph.com
    nodeStageSecretRef:
      name: rook-csi-cephfs-node
      namespace: openshift-storage
    volumeAttributes:
    # Volume Attributes can be copied from the Source testnamespace PV
      "clusterID": "openshift-storage"
      "fsName": "ocs-storagecluster-cephfilesystem"
      "staticVolume": "true"
    # rootpath is the subvolumePath: you copied from the Source testnamespace PV
      "rootPath": /volumes/csi/csi-vol-cc416d9e-dbf3-11ec-b286-0a580a810213/edcfe4d5-
bdcb-4b8e-8824-8a03ad94d67c
    volumeHandle: 0001-0011-openshift-storage-0000000000000001-cc416d9e-dbf3-
11ec-b286-0a580a810213-clone     (2)
  persistentVolumeReclaimPolicy: Retain
  volumeMode: Filesystem
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cephfs-pvc-legacy
  namespace: openshift-storage
spec:
  storageClassName: ""
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi     (3)
  volumeMode: Filesystem
```

```
  # volumeName should be same as PV name
  volumeName: cephfs-pv-legacy-openshift-storage
EOF
```

1 **在 openshift-storage 命名空间中创建的 PV 的存储容量必须与原始 PV 相同。**

2 **您在 openshift-storage 中创建的目标 PV 的卷处理需要与原始应用程序 PV 不同，例如，在卷句柄末尾添加 -clone。**

3 **在 openshift-storage 命名空间中创建的 PVC 的存储容量必须与原始 PVC 相同。**

c. **使用上一步中指定的 YAML 文件，在 openshift-storage 命名空间中创建 PV 和 PVC：**

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

<YAML_file>

指定 YAML 文件的名称。

例如：

```
$ oc create -f pv-openshift-storage.yaml

persistentvolume/cephfs-pv-legacy-openshift-storage created
persistentvolumeclaim/cephfs-pvc-legacy created
```

d. **确保 PVC 在 openshift-storage 命名空间中可用：**

```
$ oc get pvc -n openshift-storage

NAME                          STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS                AGE
cephfs-pvc-legacy             Bound    cephfs-pv-legacy-openshift-storage         10Gi
RWX                                    14s
```

e.

进入 **openshift-storage** 项目：

```
$ oc project openshift-storage

Now using project "openshift-storage" on server "https://api.cluster-
5f6ng.5f6ng.sandbox65.opentlc.com:6443".
```

f.

创建 **NSFS** 命名空间存储：

```
$ noobaa namespacestore create nsfs <nsfs_namespacestore> --pvc-
name='<cephfs_pvc_name>' --fs-backend='CEPH_FS'
```

<nsfs_namespacestore>

指定 NSFS 命名空间存储的名称。

<cephfs_pvc_name>

在 openshift-storage 命名空间中指定 CephFS PVC 的名称。

例如：

```
$ noobaa namespacestore create nsfs legacy-namespace --pvc-name='cephfs-pvc-
legacy' --fs-backend='CEPH_FS'
```

g.

确保 noobaa-endpoint pod 重启，并在 NSFS 命名空间存储中成功挂载 PVC，例如 /nsfs/legacy-namespace 挂载点：

```
$ oc exec -it <noobaa_endpoint_pod_name> -- df -h /nsfs/<nsfs_namespacestore>
```

<noobaa_endpoint_pod_name>

指定 noobaa-endpoint pod 的名称。

例如：

```
$ oc exec -it noobaa-endpoint-5875f467f5-546c6 -- df -h /nsfs/legacy-namespace

Filesystem
Size  Used Avail Use% Mounted on
```

```
172.30.202.87:6789,172.30.120.254:6789,172.30.77.247:6789:/volumes/csi/csi-vol-cc416d9e-dbf3-11ec-b286-0a580a810213/edcfe4d5-bdcb-4b8e-8824-8a03ad94d67c
10G     0   10G   0% /nsfs/legacy-namespace
```

h. **创建一个 MCG 用户帐户：**

```
$ noobaa account create <user_account> --full_permission --allow_bucket_create=true --new_buckets_path='/' --nsfs_only=true --nsfs_account_config=true --gid <gid_number> --uid <uid_number> --default_resource='legacy-namespace'
```

<user_account>

指定 MCG 用户帐户的名称。

<gid_number>

指定 GID 号。

<uid_number>

指定 UID 号。

重要

使用与传统应用的相同 UID 和 GID。您可以从前面的输出中找到它。

例如：

```
$ noobaa account create leguser --full_permission --allow_bucket_create=true --new_buckets_path='/' --nsfs_only=true --nsfs_account_config=true --gid 0 --uid 1000660000 --default_resource='legacy-namespace'
```

i. **创建一个 MCG 存储桶。**

i. **在传统应用 pod 的 CephFS PV 和 PVC 的 NSFS 共享中为 S3 创建一个专用文件夹：**

```
$ oc exec -it <pod_name> -- mkdir <mount_path>/nsfs
```

例如：

```
$ oc exec -it cephfs-write-workload-generator-no-cache-1-cv892 -- mkdir
/mnt/pv/nsfs
```

ii.

使用 nsfs/ 路径创建 MCG 存储桶：

```
$ noobaa api bucket_api create_bucket '{
  "name": "<bucket_name>",
  "namespace":{
    "write_resource": { "resource": "<nsfs_namespacestore>", "path": "nsfs/" },
    "read_resources": [ { "resource": "<nsfs_namespacestore>", "path": "nsfs/" }]
  }
}'
```

例如：

```
$ noobaa api bucket_api create_bucket '{
  "name": "legacy-bucket",
  "namespace":{
    "write_resource": { "resource": "legacy-namespace", "path": "nsfs/" },
    "read_resources": [ { "resource": "legacy-namespace", "path": "nsfs/" }]
  }
}'
```

j.

检查位于传统应用程序和 openshift-storage 命名空间中的 PVC 中的文件夹的 SELinux 标签：

```
$ oc exec -it <noobaa_endpoint_pod_name> -n openshift-storage -- ls -ltraZ
/nsfs/<nsfs_namespacstore>
```

例如：

```
$ oc exec -it noobaa-endpoint-5875f467f5-546c6 -n openshift-storage -- ls -ltraZ
/nsfs/legacy-namespace

total 567
drwxrwxrwx. 3 root       root system_u:object_r:container_file_t:s0:c0,c26      2 May 25
06:35 .
-rw-r--r--. 1 1000660000 root system_u:object_r:container_file_t:s0:c0,c26 580138 May
25 06:35 fs_write_cephfs-write-workload-generator-no-cache-1-cv892-data.log
drwxrwxrwx. 3 root       root system_u:object_r:container_file_t:s0:c0,c26     30 May 25
06:35 ..
```

```
$ oc exec -it <pod_name> -- ls -latrZ <mount_path>
```

例如：

```
$ oc exec -it cephfs-write-workload-generator-no-cache-1-cv892 -- ls -latrZ /mnt/pv/

total 567
drwxrwxrwx. 3 root       root system_u:object_r:container_file_t:s0:c26,c5      2 May 25
06:35 .
-rw-r--r--. 1 1000660000 root system_u:object_r:container_file_t:s0:c26,c5 580138 May
25 06:35 fs_write_cephfs-write-workload-generator-no-cache-1-cv892-data.log
drwxrwxrwx. 3 root       root system_u:object_r:container_file_t:s0:c26,c5     30 May 25
06:35 ..
```

在这些示例中，您可以看到 SELinux 标签不是不是导致权限被拒绝或访问问题的结果。

9. **确保传统应用程序和 openshift-storage pod 在文件中使用相同的 SELinux 标签。**

   您可以使用以下方法之一进行此操作：

   - **第 4.4.3.1 节 “更改传统应用程序项目中的默认 SELinux 标签，使其与 openshift-storage 项目中的默认 SELinux 标签匹配”.**
   - **第 4.4.3.2 节 “仅修改具有挂载传统应用程序 PVC 的 pod 的部署配置的 SELinux 标签”.**

10. **删除 NSFS 命名空间存储：**

    a. **删除 MCG 存储桶：**

    ```
    $ noobaa bucket delete <bucket_name>
    ```

    例如：

    ```
    $ noobaa bucket delete legacy-bucket
    ```

    '

b.

删除 MCG 用户帐户：

```
$ noobaa account delete <user_account>
```

例如：

```
$ noobaa account delete leguser
```

c.

删除 NSFS 命名空间存储：

```
$ noobaa namespacestore delete <nsfs_namespacestore>
```

例如：

```
$ noobaa namespacestore delete legacy-namespace
```

11.

删除 PV 和 PVC：

重要

在删除 PV 和 PVC 前，请确保 PV 配置了保留策略。

```
$ oc delete pv <cephfs_pv_name>
```

```
$ oc delete pvc <cephfs_pvc_name>
```

<cephfs_pv_name>

指定传统应用的 CephFS PV 名称。

<cephfs_pvc_name>

指定传统应用的 CephFS PVC 名称。

例如：

```
$ oc delete pv cephfs-pv-legacy-openshift-storage
```

```
$ oc delete pvc cephfs-pvc-legacy
```

4.4.3.1. 更改传统应用程序项目中的默认 SELinux 标签，使其与 openshift-storage 项目中的默认 SELinux 标签匹配

1. **使用 sa.scc.mcs 显示当前的 openshift-storage 命名空间：**

```
$ oc get ns openshift-storage -o yaml | grep sa.scc.mcs

openshift.io/sa.scc.mcs: s0:c26,c0
```

2. **编辑传统应用程序命名空间，并使用 openshift-storage 命名空间的 sa.scc.mcs 中的值修改 sa.scc.mcs ：**

```
$ oc edit ns <appplication_namespace>
```

例如：

```
$ oc edit ns testnamespace
```

```
$ oc get ns <application_namespace> -o yaml | grep sa.scc.mcs
```

例如：

```
$ oc get ns testnamespace -o yaml | grep sa.scc.mcs

openshift.io/sa.scc.mcs: s0:c26,c0
```

3. **重启旧的应用容器集。重新标记所有文件，现在 SELinux 标签与 openshift-storage 部署相匹配。**

4.4.3.2. 仅修改具有挂载传统应用程序 PVC 的 pod 的部署配置的 SELinux 标签

1. **使用 MustRunAs 和 seLinuxOptions 选项创建一个新的 scc，使用 openshift-storage 项目使用的 Multi Category Security(MCS)：**

YAML 文件示例：

```
$ cat << EOF >> scc.yaml
allowHostDirVolumePlugin: false
allowHostIPC: false
allowHostNetwork: false
allowHostPID: false
allowHostPorts: false
allowPrivilegeEscalation: true
allowPrivilegedContainer: false
allowedCapabilities: null
apiVersion: security.openshift.io/v1
defaultAddCapabilities: null
fsGroup:
  type: MustRunAs
groups:
- system:authenticated
kind: SecurityContextConstraints
metadata:
  annotations:
  name: restricted-pvselinux
priority: null
readOnlyRootFilesystem: false
requiredDropCapabilities:
- KILL
- MKNOD
- SETUID
- SETGID
runAsUser:
  type: MustRunAsRange
seLinuxContext:
  seLinuxOptions:
    level: s0:c26,c0
  type: MustRunAs
supplementalGroups:
  type: RunAsAny
users: []
volumes:
- configMap
- downwardAPI
- emptyDir
- persistentVolumeClaim
- projected
- secret
EOF
```

```
$ oc create -f scc.yaml
```

2. **为部署创建一个服务帐户，并将其添加到新创建的 scc 中。**

   a. **创建服务帐户：**

   ```
   $ oc create serviceaccount <service_account_name>
   ```

<service_account_name>`

指定服务帐户的名称。

例如：

```
$ oc create serviceaccount testnamespacesa
```

b.

将服务帐户添加到新创建的 scc 中：

```
$ oc adm policy add-scc-to-user restricted-pvselinux -z <service_account_name>
```

例如：

```
$ oc adm policy add-scc-to-user restricted-pvselinux -z testnamespacesa
```

3.

对传统应**用程序部署进行补丁，使其使用新创建的服务帐户。这可让您在部署中指定 SELinux 标签：**

```
$ oc patch dc/<pod_name> '{"spec":{"template":{"spec":{"serviceAccountName":
"<service_account_name>"}}}}'
```

例如：

```
$ oc patch dc/cephfs-write-workload-generator-no-cache --patch '{"spec":{"template":{"spec":
{"serviceAccountName": "testnamespacesa"}}}}'
```

4.

编辑部署，以指定要在部署配置的 SELinux 标签中使用的安全上下文：

```
$ oc edit dc <pod_name> -n <application_namespace>
```

添加以下行：

```
spec:
 template:
   metadata:
```

```
    securityContext:
      seLinuxOptions:
        Level: <security_context_value>
```

<security_context_value>

当您执行命令以在 NSFS 共享内为 S3 创建专用文件夹时，可以在 legacy 应用 Pod 的 CephFS PV 和 PVC 上创建一个专用文件夹时，即可找到这个值。

例如：

```
$ oc edit dc cephfs-write-workload-generator-no-cache -n testnamespace
```

```
spec:
 template:
   metadata:
     securityContext:
       seLinuxOptions:
         level: s0:c26,c0
```

5. **确保正确在部署配置中的 SELinux 标签中使用安全上下文：**

```
$ oc get dc <pod_name> -n <application_namespace> -o yaml | grep -A 2 securityContext
```

例如"

```
$ oc get dc cephfs-write-workload-generator-no-cache -n testnamespace -o yaml | grep -A 2
securityContext

    securityContext:
      seLinuxOptions:
        level: s0:c26,c0
```

传统应用程序重启并开始使用与 openshift-storage 命名空间相同的 SELinux 标签。

第 5 章 保护多云对象网关

5.1. 更改默认帐户凭证以确保在 MULTICLOUD 对象网关中提高安全性

使用命令行界面更改并轮转 Multicloud 对象网关(MCG)帐户凭证，以防止应用程序出现问题，并确保帐户安全性。

5.1.1. 重置 noobaa 帐户密码

先决条件

- 正在运行的 OpenShift Data Foundation 集群。
- 下载 Multicloud 对象网关(MCG)命令行界面以便更轻松地管理。具体步骤请参阅 使用应用程序访问多云对象网关。

流程

- 要重置 noobaa 帐户密码，请运行以下命令：

```
$ noobaa account passwd <noobaa_account_name> [options]
```

```
$ noobaa account passwd
FATA[0000]   Missing expected arguments: <noobaa_account_name>

Options:
   --new-password='': New Password for authentication - the best practice is to omit this flag, in that
   case the CLI will prompt to prompt and read it securely from the terminal to avoid leaking secrets in t
    he shell history
   --old-password='': Old Password for authentication - the best practice is to omit this flag, in that
   case the CLI will prompt to prompt and read it securely from the terminal to avoid leaking secrets in
   the shell history
   --retype-new-password='': Retype new Password for authentication - the best practice is to omit this flag, in that case the CLI will prompt to prompt and read it securely from the terminal to avoid
   leaking secrets in the shell history


Usage:
```

```
noobaa account passwd <noobaa-account-name> [flags] [options]

Use "noobaa options" for a list of global command-line options (applies to all commands).
```

Example:

```
$ noobaa account passwd admin@noobaa.io
```

输出示例：

```
Enter old-password: [got 24 characters]
Enter new-password: [got 7 characters]
Enter retype-new-password: [got 7 characters]
INFO[0017]   Exists: Secret "noobaa-admin"
INFO[0017]   Exists: NooBaa "noobaa"
INFO[0017]   Exists: Service "noobaa-mgmt"
INFO[0017]   Exists: Secret "noobaa-operator"
INFO[0017]   Exists: Secret "noobaa-admin"
INFO[0017] ✈   RPC: account.reset_password() Request: {Email:admin@noobaa.io
VerificationPassword:* Password:*}
WARN[0017] RPC: GetConnection creating connection to wss://localhost:58460/rpc/
0xc000402ae0
INFO[0017] RPC: Connecting websocket (0xc000402ae0) &{RPC:0xc000501a40
Address:wss://localhost:58460/rpc/ State:init WS:<nil> PendingRequests:map[]
NextRequestID:0
Lock:{state:1 sema:0} ReconnectDelay:0s cancelPings:<nil>}
INFO[0017] RPC: Connected websocket (0xc000402ae0) &{RPC:0xc000501a40
Address:wss://localhost:58460/rpc/ State:init WS:<nil> PendingRequests:map[]
NextRequestID:0
Lock:{state:1 sema:0} ReconnectDelay:0s cancelPings:<nil>}
INFO[0020]   RPC: account.reset_password() Response OK: took 2907.1ms
INFO[0020]   Updated:  "noobaa-admin"
INFO[0020]   Successfully reset the password for the account "admin@noobaa.io"
```

重要

要从终端访问 admin 帐户凭证，请运行 noobaa status 命令：

```
---------------------
- Mgmt Credentials -
---------------------

email    : admin@noobaa.io
password : ***
```

5.1.2. 为帐户重新生成 S3 凭证

先决条件

- 正在运行的 **OpenShift Data Foundation** 集群。
- 下载 **Multicloud** 对象网关(MCG)命令行界面以便更轻松地管理。具体步骤请参阅 使用应用程序访问多云对象网关。

流程

1. 获取帐户名称。

   要列出帐户，请运行以下命令：

   ```
   $ noobaa account list
   ```

   输出示例：

   ```
   NAME          ALLOWED_BUCKETS   DEFAULT_RESOURCE            PHASE   AGE
   account-test  [*]               noobaa-default-backing-store  Ready   14m17s
   test2         [first.bucket]    noobaa-default-backing-store  Ready   3m12s
   ```

   或者，从终端运行 **oc get noobaaaccount** 命令：

   ```
   $ oc get noobaaaccount
   ```

   输出示例：

   ```
   NAME          PHASE   AGE
   account-test  Ready   15m
   test2         Ready   3m59s
   ```

2. 要重新生成 **noobaa** 帐户 **S3** 凭证，请运行以下命令：

   ```
   $ noobaa account regenerate <noobaa_account_name> [options]
   ```

   ```
   $ noobaa account regenerate
   FATA[0000]  Missing expected arguments: <noobaa-account-name>

   Usage:
   ```

```
noobaa account regenerate <noobaa-account-name> [flags] [options]

Use "noobaa options" for a list of global command-line options (applies to all commands).
```

3. 运行 **noobaa account regenerate** 命令后，它会提示您一个警告，**"This will invalidate all connections between S3 clients and NooBaa which are connected using the current credentials."**，并要求确认：

   Example:

```
$ noobaa account regenerate account-test
```

   输出示例：

```
INFO[0000] You are about to regenerate an account's security credentials.
INFO[0000] This will invalidate all connections between S3 clients and NooBaa which are connected using the current credentials.
INFO[0000] are you sure? y/n
```

4. **批准后，它会重新生成凭证并最终打印它们：**

```
INFO[0015]   Exists: Secret "noobaa-account-account-test"
Connection info:
AWS_ACCESS_KEY_ID      : ***
AWS_SECRET_ACCESS_KEY  : ***
```

5.1.3. 为 OBC 重新生成 S3 凭证

先决条件

- **正在运行的 OpenShift Data Foundation 集群。**
- **下载 Multicloud 对象网关(MCG)命令行界面以便更轻松地管理。具体步骤请参阅 使用应用程序访问多云对象网关。**

流程

1. **要获取 OBC 名称，请运行以下命令：**

```
$ noobaa obc list
```

输出示例：

```
NAMESPACE   NAME       BUCKET-NAME                                    STORAGE-CLASS
BUCKET-CLASS                PHASE
default     obc-test   obc-test-35800e50-8978-461f-b7e0-7793080e26ba   default.noobaa.io
noobaa-default-bucket-class   Bound
```

或者，从终端运行 oc get obc 命令：

```
$ oc get obc
```

输出示例：

```
NAME       STORAGE-CLASS       PHASE   AGE
obc-test   default.noobaa.io   Bound   38s
```

2. **要重新生成 noobaa OBC S3 凭证，请运行以下命令：**

```
$ noobaa obc regenerate <bucket_claim_name> [options]
```

```
$ noobaa obc regenerate
FATA[0000]   Missing expected arguments: <bucket-claim-name>

Usage:
  noobaa obc regenerate <bucket-claim-name> [flags] [options]

Use "noobaa options" for a list of global command-line options (applies to all commands).
```

3. 运行 **noobaa obc regenerate 命令后，它将提示警告，"This will invalidate all connections between the S3 clients and noobaa which are connected using the current credentials."，并要求确认：**

Example:

```
$ noobaa obc regenerate obc-test
```

输出示例：

```
INFO[0000] You are about to regenerate an OBC's security credentials.
INFO[0000] This will invalidate all connections between S3 clients and NooBaa which are
```

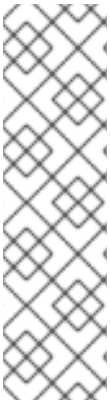
```
connected using the current credentials.
INFO[0000] are you sure? y/n
```

4. **批准后，它会重新生成凭证并最终打印它们：**

```
INFO[0022]   RPC: bucket.read_bucket() Response OK: took 95.4ms

ObjectBucketClaim info:
  Phase                : Bound
  ObjectBucketClaim    : kubectl get -n default objectbucketclaim obc-test
  ConfigMap            : kubectl get -n default configmap obc-test
  Secret               : kubectl get -n default secret obc-test
  ObjectBucket         : kubectl get objectbucket obc-default-obc-test
  StorageClass         : kubectl get storageclass default.noobaa.io
  BucketClass          : kubectl get -n default bucketclass noobaa-default-bucket-class

Connection info:
 BUCKET_HOST           : s3.default.svc
 BUCKET_NAME           : obc-test-35800e50-8978-461f-b7e0-7793080e26ba
   BUCKET_PORT         : 443
   AWS_ACCESS_KEY_ID   : ***
   AWS_SECRET_ACCESS_KEY  : ***

Shell commands:
  AWS S3 Alias         : alias s3='AWS_ACCESS_KEY_ID=***
AWS_SECRET_ACCESS_KEY=*** aws s3 --no-verify-ssl --endpoint-url ***'

Bucket status:
  Name                 : obc-test-35800e50-8978-461f-b7e0-7793080e26ba
  Type                 : REGULAR
  Mode                 : OPTIMAL
  ResiliencyStatus     : OPTIMAL
  QuotaStatus          : QUOTA_NOT_SET
  Num Objects          : 0
  Data Size            : 0.000 B
  Data Size Reduced    : 0.000 B
  Data Space Avail     : 13.261 GB
  Num Objects Avail    : 9007199254740991
```

5.2. 为多云对象网关启用安全模式部署

您可以指定允许连接到 Multicloud Object Gateway (MCG)负载均衡器服务启用安全模式部署的 IP 地址的范围。这有助于控制可以访问 MCG 服务的 IP 地址。



注意

您可以在使用命令行界面部署 OpenShift Data Foundation 时，通过在 storagecluster 自定义资源定义(CRD)中设置 disableLoadBalancerService 变量来禁用 MCG 负载均衡器的使用。这有助于限制 MCG 为私有集群创建任何公共资源，并禁用 MCG 服务 EXTERNAL-IP。如需更多信息，请参阅 Red Hat 知识库文章，使用命令行界面以内部模式安装 Red Hat OpenShift Data Foundation 4.X。有关在部署 OpenShift Data Foundation 后禁用 MCG 负载均衡器服务的详情，请参考 部署 OpenShift Data Foundation 后禁用多云对象网关外部服务。

先决条件

- 正在运行的 OpenShift Data Foundation 集群。
- 如果是裸机部署，请确保负载均衡器控制器支持在 Kubernetes 服务中设置 loadBalancerSourceRanges 属性。

流程

- 编辑 NooBaa 自定义资源 (CR)，以指定在部署 OpenShift Data Foundation 后可以访问 MCG 服务的 IP 地址范围。

```
$ oc edit noobaa -n openshift-storage noobaa
```

noobaa

控制 NooBaa 系统部署的 NooBaa CR 类型。

noobaa

NooBaa CR 的名称。

例如：

```
...
spec:
  ...
  loadBalancerSourceSubnets:
    s3: ["10.0.0.0/16", "192.168.10.0/32"]
    sts:
      - "10.0.0.0/16"
      - "192.168.10.0/32"
...
```

loadBalancerSourceSubnets

可以在 NooBaa CR 的 spec 下添加一个新的字段，以指定应该可以访问 NooBaa 服务的 IP 地址。

在本例中，位于子网 10.0.0.0/16 或 192.168.10.0/32 中的所有 IP 地址都可以访问 MCG S3 和安全令牌服务(STS)，而不允许其他 IP 地址访问。

验证步骤

- 要验证是否设置了指定的 IP 地址，在 OpenShift Web 控制台中运行以下命令，检查输出是否与 MCG 提供的 IP 地址匹配：

```
$ oc get svc -n openshift-storage <s3 | sts> -o=go-template='{{
.spec.loadBalancerSourceRanges }}'
```

第 6 章 混合和多云存储桶的镜像数据

您可以使用 Multicloud Object Gateway(MCG)简化流程跨云供应商和集群跨数据。在创建反映数据管理策略和镜像的存储桶类前，您必须添加可以被 MCG 使用的后备存储。如需更多信息，请参阅第 4 章 *第 3 章 为混合或多云添加存储资源*。

您可以使用 OpenShift UI、YAML 或 MCG 命令行界面设置镜像数据。

请参见以下部分：

- 第 6.1 节 “使用 MCG 命令行创建存储桶类来镜像数据”
- 第 6.2 节 “使用 YAML 创建存储桶类来镜像数据”

6.1. 使用 MCG 命令行创建存储桶类来镜像数据

先决条件

- 确保下载 Multicloud 对象网关(MCG)命令行界面。

流程

1. 在 Multicloud Object Gateway(MCG)命令行界面中，运行以下命令来创建带有镜像策略的存储桶类：

```
$ noobaa bucketclass create placement-bucketclass mirror-to-aws --backingstores=azure-resource,aws-resource --placement Mirror
```

2. 将新创建的存储桶类设置为一个新的存储桶声明，生成一个新的存储桶，该存储桶将在两个位置之间进行镜像：

```
$ noobaa obc create  mirrored-bucket --bucketclass=mirror-to-aws
```

6.2. 使用 YAML 创建存储桶类来镜像数据

1. **应用以下 YAML。此 YAML 是一个混合示例，在本地 Ceph 存储和 AWS 之间镜像数据：**

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <bucket-class-name>
  namespace: openshift-storage
spec:
  placementPolicy:
    tiers:
    - backingStores:
      - <backing-store-1>
      - <backing-store-2>
      placement: Mirror
```

2. **将以下行添加到标准 Object Bucket Claim (OBC) 中：**

```
additionalConfig:
  bucketclass: mirror-to-aws
```

有关 OBCs 的更多信息，请参阅 **第 9 章 *对象 Bucket 声明*。**

第 7 章 MULTICLOUD 对象网关中的存储桶策略

OpenShift Data Foundation 支持 AWS S3 存储桶策略。bucket 策略允许您为用户授予存储桶及其对象的访问权限。

7.1. BUCKET 策略简介

bucket 策略是一个访问策略选项，可供您向 AWS S3 存储桶和对象授予权限。bucket 策略使用基于 JSON 的访问策略语言。有关访问策略语言的更多信息，请参阅AWS 访问策略语言概述。

7.2. 在 MULTICLOUD 对象网关中使用存储桶策略

先决条件

- 正在运行的 OpenShift Data Foundation 平台。
- 访问 Multicloud 对象网关(MCG)，请参阅 第 2 章 *使用应用程序访问多云对象网关*

流程

在 MCG 中使用存储桶策略：

1. 以 JSON 格式创建 bucket 策略。

   例如：

```
{
    "Version": "NewVersion",
    "Statement": [
        {
            "Sid": "Example",
            "Effect": "Allow",
            "Principal": [
                    "john.doe@example.com"
            ],
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::john_bucket"
            ]
```

```
        }
    ]
}
```

2. **使用 AWS S3 客户端，使用 put-bucket-policy 命令将存储桶策略应用到 S3 存储桶：**

```
# aws --endpoint ENDPOINT --no-verify-ssl s3api put-bucket-policy --bucket MyBucket --policy BucketPolicy
```

a. **将 *ENDPOINT* 替换为 S3 端点。**

b. **将 *MyBucket* 替换为 bucket，以设置策略。**

c. **将 *BucketPolicy* 替换为 bucket 策略 JSON 文件。**

d. **如果您使用默认的自签名证书，请添加 --no-verify-ssl。**

例如：

```
# aws --endpoint https://s3-openshift-storage.apps.gogo44.noobaa.org --no-verify-ssl s3api put-bucket-policy -bucket MyBucket --policy file://BucketPolicy
```

如需有关 put-bucket-policy 命令的更多信息，请参阅有关 put-bucket-policy 的 AWS CLI 命令参考。

注意

principal 元素指定允许或拒绝访问某一资源的用户，如存储桶。目前，只有 NooBaa 帐户才能用作主体。对于对象存储桶声明，NooBaa 会自动创建一个帐户 obc-account.<generated bucket name>@noobaa.io。

注意

不支持 bucket 策略条件。

其他资源

- bucket 策略有许多可用元素，与访问权限有关。

- 有关这些元素的详细信息，以及如何使用它们控制访问权限的示例，请参阅 AWS 访问策略语言概述。

- 如需存储桶策略的更多示例，请参阅 AWS Bucket 策略示例。

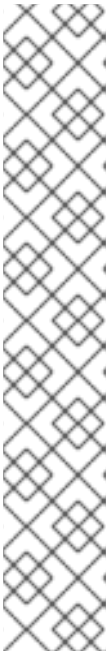
7.3. 在 MULTICLOUD 对象网关中创建用户

先决条件

- 正在运行的 OpenShift Data Foundation 平台。

- 下载 MCG 命令行界面以简化管理。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```



注意

指定使用订阅管理器启用存储库的适当架构。

- 对于 IBM Power，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- 对于 IBM Z，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- 或者，您也可以从 下载红帽 OpenShift Data Foundation 页面上的 OpenShift Data Foundation RPM 安装 MCG 软件包。

注意

根据您的架构选择正确的产品变体。

流程

执行以下命令创建一个 **MCG** 用户帐户：

```
noobaa account create <noobaa-account-name> [--allow_bucket_create=true] [--allowed_buckets=[]]
[--default_resource=''] [--full_permission=false]
```

<noobaa-account-name>

指定新 MCG 用户帐户的名称。

--allow_bucket_create

允许用户创建新存储桶。

--allowed_buckets

设置用户的允许存储桶列表（使用逗号或多个标记）。

--default_resource

设置默认资源。新存储桶在此默认资源（包括将来的资源）上创建新的 bucket。

--full_permission

允许此帐户访问所有现有和将来的存储桶。

重要

您需要提供访问最少一个存储桶的访问权限，或完整权限来来访问所有存储桶。

第 8 章 多云对象网关存储桶复制

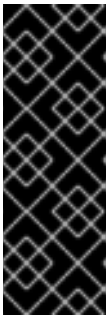
从一个多云对象网关(MCG)存储桶向另一个 MCG 存储桶的数据复制提供了更高的弹性和更好的协作选项。这些存储桶可以是由任何受支持的存储解决方案(AWS S3、Azure 等)支持的数据存储桶或命名空间存储桶。

复制策略由复制规则列表组成。每条规则定义目的地桶，并可基于对象密钥前缀指定过滤器。在第二个 bucket 上配置补充复制策略会导致双向复制。

先决条件

- 正在运行的 OpenShift Data Foundation 平台。
- 访问多云对象网关，请参阅第使用应用程序访问多云对象网关。
- 下载 Multicloud 对象网关 (MCG) 命令行界面：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

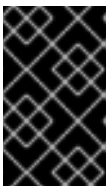


重要

指定使用订阅管理器启用存储库的适当架构。例如，对于 IBM Power，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- 另外，您还可以从位于 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages 的 OpenShift Data Foundation RPM 安装 mcg 软件包



重要

根据您的架构选择正确的产品变体。

注意

某些特定的 MCG 功能仅在某些 MCG 版本中可用，必须使用适当的 MCG CLI 工具版本来充分利用 MCG 的功能。

要复制存储桶，请参阅将存储桶复制到另一个存储桶。

要设置存储桶类复制策略，请参阅设置存储桶类复制策略。

8.1. 将存储桶复制到其他存储桶

您可以通过两种方式设置存储桶复制策略：

- 使用 MCG 命令行界面将存储桶复制到另一个存储桶。
- 使用 YAML 将存储桶复制到其他存储桶。

8.1.1. 使用 MCG 命令行界面将存储桶复制到另一个存储桶

在创建对象存储桶声明(OBC)时，您可以为多云对象网关(MCG)数据存储桶设置复制策略。您必须在 JSON 文件中定义复制策略参数。

流程

在 MCG 命令行界面中，运行以下命令来创建具有特定复制策略的 OBC：

```
noobaa obc create <bucket-claim-name> -n openshift-storage --replication-policy /path/to/json-file.json
```

<bucket-claim-name>

指定存储桶声明的名称。

/path/to/json-file.json

是定义复制策略的 JSON 文件的路径。

JSON 文件示例：

```
[{ "rule_id": "rule-1", "destination_bucket": "first.bucket", "filter": {"prefix": "repl"}}]
```

"prefix"

是可选的。它是要复制的对象键的前缀，甚至可以留空，例如 {"prefix": ""}。

例如：

```
noobaa obc create my-bucket-claim -n openshift-storage --replication-policy /path/to/json-file.json
```

8.1.2. 使用 YAML 将存储桶复制到另一个存储桶

您可以在创建对象存储桶声明(OBC)时为多云对象网关(MCG)数据存储桶设置复制策略，也可以在以后编辑 YAML。您必须将策略作为 JSON 兼容字符串提供，以遵循以下流程中显示的格式。

流程

- **应用以下 YAML：**

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: <desired-bucket-claim>
  namespace: <desired-namespace>
spec:
  generateBucketName: <desired-bucket-name>
  storageClassName: openshift-storage.noobaa.io
  additionalConfig:
    replicationPolicy: {"rules": [{ "rule_id": "", "destination_bucket": "", "filter": {"prefix": ""}}]}
```

<desired-bucket-claim>

指定存储桶声明的名称。

<desired-namespace>

指定命名空间。

<desired-bucket-name>

指定存储桶名称的前缀。

"rule_id"

指定规则的 ID 号，例如 {"rule_id": "rule-1"}。

"destination_bucket"

指定目标存储桶的名称，例如 {"destination_bucket": "first.bucket"}。

"prefix"

是可选的。它是要复制的对象键的前缀，甚至可以留空，例如 {"prefix": ""}。

附加信息

- 如需有关 OBC 的更多信息，请参阅 Object Bucket Claim。

8.2. 设置存储桶类复制策略

可以设置一个复制策略，它自动应用到在特定存储桶类下创建的所有存储桶。您可以通过两种方式执行此操作：

- 使用 MCG 命令行界面来设置存储桶类复制策略。
- 使用 YAML 设置存储桶类复制策略。

8.2.1. 使用 MCG 命令行界面设置存储桶类复制策略

您可以在创建存储桶类时为多云对象网关(MCG)数据存储桶设置复制策略。您必须在 JSON 文件中定义 replication-policy 参数。您可以为 Placement 和 Namespace bucket 类设置存储桶类复制策略。

您可以为 Placement 和 Namespace bucket 类设置存储桶类复制策略。

流程

- 在 MCG 命令行界面中运行以下命令：

```
noobaa -n openshift-storage bucketclass create placement-bucketclass <bucketclass-name>
--backingstores <backingstores> --replication-policy=/path/to/json-file.json
```

<bucketclass-name>

指定存储桶类的名称。

<backingstores>

指定后备存储的名称。您可以传递多个以逗号分开的后备存储。

/path/to/json-file.json

是定义复制策略的 JSON 文件的路径。

JSON 文件示例：

```
[{ "rule_id": "rule-1", "destination_bucket": "first.bucket", "filter": {"prefix": "repl"}}]
```

"prefix"

是可选的。对象密钥的前缀会被复制。您可以将其留空，例如 {"prefix": ""}。

例如：

```
noobaa -n openshift-storage bucketclass create placement-bucketclass bc --
backingstores azure-blob-ns --replication-policy=/path/to/json-file.json
```

本例使用 JSON 文件中定义的特定复制策略创建放置存储桶类。

8.2.2. 使用 YAML 设置存储桶类复制策略

您可以在创建存储桶类时为多云对象网关(MCG)数据存储桶设置复制策略，也可以在以后编辑其 YAML。您必须将策略作为 JSON 兼容字符串提供，以遵循以下流程中显示的格式。

流程

1. **应用以下 YAML：**

```
apiVersion: noobaa.io/v1alpha1
```

```
kind: BucketClass
metadata:
  labels:
    app: <desired-app-label>
  name: <desired-bucketclass-name>
  namespace: <desired-namespace>
spec:
  placementPolicy:
    tiers:
    - backingstores:
      - <backingstore>
      placement: Spread
  replicationPolicy: [{ "rule_id": "<rule id>", "destination_bucket": "first.bucket", "filter": {"prefix":
"<object name prefix>"}}]
```

此 YAML 是一个创建放置存储桶类的示例。每个上传到存储桶的对象存储桶声明(OBC)对象根据前缀进行过滤，并将其复制到 first.bucket。

<desired-app-label>

为 **app 指定一个标签。**

<desired-bucketclass-name>

指定存储桶类名称。

<desired-namespace>

指定创建存储桶类的命名空间。

<backingstore>

指定后备存储的名称。您可以传递多个后备存储。

"rule_id"

指定规则的 ID 号，如 '{"rule_id": "rule-1"}。

"destination_bucket"

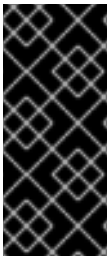
指定目标存储桶的名称，例如 {"destination_bucket": "first.bucket"}。

"prefix"

是可选的。对象密钥的前缀会被复制。您可以将其留空，例如 {"prefix": ""}。

8.3. 启用基于日志的存储桶复制

在创建存储桶复制策略时，您可以使用日志，以便更快地复制最新的数据，而默认的基于扫描的复制则适用于复制其余数据。



重要

此功能需要在 AWS 或 Azure 上设置存储桶日志。有关设置 AWS 日志的更多信息，请参阅启用 Amazon S3 服务器访问日志记录。AWS 日志存储桶需要在与源 NamespaceStore AWS 存储桶相同的区域中创建。

注意

此功能仅支持由 NamespaceStore 支持的存储桶。由 BackingStores 支持的存储桶无法使用基于日志的复制。

8.3.1. 在 Amazon Web Service 环境中使用 OpenShift Web 控制台为新命名空间存储桶启用基于日志的存储桶复制

您可以使用 Amazon Web Service (AWS)云环境的事件日志来优化复制。在创建命名空间存储桶期间，您可以使用 Web 控制台为新命名空间存储桶启用基于日志的存储桶复制。

先决条件

- 确保 AWS 中启用了对象日志记录。如需更多信息，请参阅启用 Amazon S3 服务器访问日志记录 中的"使用 S3 控制台"部分。
- 管理员对 OpenShift Web 控制台的访问权限。

流程

1. 在 OpenShift Web 控制台中，导航到 Storage → Object Storage → Object Bucket Claims。
2. 点 Create ObjectBucketClaim。
3. 输入 ObjectBucketName 的名称，然后选择 StorageClass 和 BucketClass。

4. 选择 **Enable replication** 复选框来启用复制。

5. 在 **Replication policy** 部分中，选择 **Optimize replication using event logs** 复选框。

6. 输入包含 **Event log Bucket** 下的日志的存储桶名称。

   如果日志没有存储在存储桶的根目录中，请提供不带 **s3://**的完整路径。

7. 输入前缀来仅复制名称以给定前缀开头的对象。

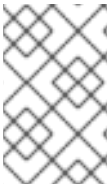
8.3.2. 使用 YAML 为现有命名空间存储桶启用基于日志的存储桶复制

您可以使用命令行界面或应用 **YAML** 而不是使用 **AWS S3** 命令创建的存储桶，为现有存储桶启用基于日志的存储桶复制。

流程

- 编辑存储桶的 **OBC** 的 **YAML**，以启用基于日志的存储桶复制。在 **spec** 下添加以下内容：

```
replicationPolicy: '{"rules":[{"rule_id":"<RULE ID>", "destination_bucket":"<DEST>", "filter":
{"prefix": "<PREFIX>"}}], "log_replication_info": {"logs_location": {"logs_bucket": "
<LOGS_BUCKET>"}}}'
```



注意

也可以在创建 **OBC** 前将其添加到 **OBC** 的 **YAML** 中。

rule_id

指定您选择的 **ID** 来识别规则

destination_bucket

指定对象要复制到的目标 **MCG** 存储桶的名称

(optional) {"filter": {"prefix": <>}}

指定您可以设置的前缀字符串来过滤复制的对象

log_replication_info

指定一个对象，其中包含与基于日志的复制优化相关的数据。{"logs_location": {"logs_bucket": <>}} 设置为 AWS S3 服务器访问日志的位置。

8.3.3. 在 Microsoft Azure 中启用基于日志的存储桶复制

先决条件

- 请参阅 Microsoft Azure 文档，并确保您已在 Microsoft Azure 门户中完成以下任务：

a. 确保已创建了新应用程序，并记下名称、应用程序（客户端）ID 和目录（租户）ID。

如需更多信息，请参阅 注册应用程序。

b. 确保创建了新的客户端 secret，并记下应用程序 secret。

c. 确保创建了新的 Log Analytics 工作区，并且记下了其名称和工作区 ID。

如需更多信息，请参阅创建日志分析工作区。

d. 确保在 Access control 和 members 下分配了 Reader 角色，并且提供了您在上一步中注册的应用的名称。

如需更多信息，请参阅使用 Azure 门户分配 Azure 角色。

e. 确保创建了新的存储帐户，并且记录 Access 密钥。

f. 在创建的存储帐户的 Monitoring 部分中，选择一个 blob，在 Diagnostic settings 屏幕中，仅选择 StorageWrite 和 StorageDelete，在目标详情中添加之前创建的 Log Analytics 工作区。确保在所创建存储帐户的 Monitoring 部分的 Diagnostic settings 屏幕中选择了 blob。另外，请确保只选择 StorageWrite 和 StorageDelete，目标详情中会添加之前创建的 Log Analytics 工作区。

如需更多信息，请参阅 Azure Monitor 中的诊断设置。

g. 确保创建了用于对象源和对象目的地的两个新容器。

- 管理员对 OpenShift Web 控制台的访问权限。

流程

1. 使用命名空间 存储 要使用的凭证创建一个 secret。

```
apiVersion: v1
kind: Secret
metadata:
  name: <namespacestore-secret-name>
type: Opaque
data:
    TenantID: <AZURE TENANT ID ENCODED IN BASE64>
    ApplicationID: <AZURE APPLICATIOM ID ENCODED IN BASE64>
    ApplicationSecret: <AZURE APPLICATION SECRET ENCODED IN BASE64>
    LogsAnalyticsWorkspaceID: <AZURE LOG ANALYTICS WORKSPACE ID ENCODED IN
BASE64>
    AccountName: <AZURE ACCOUNT NAME ENCODED IN BASE64>
    AccountKey: <AZURE ACCOUNT KEY ENCODED IN BASE64>
```

2. 创建由 Azure 中创建的容器支持的 NamespaceStore。

如需更多信息，请参阅使用 OpenShift Container Platform 用户界面添加命名空间存储桶。

3. 创建一个新的 Namespace-Bucketclass 和 OBC 来使用它。

4. 通过查找目标 OBC 的 YAML 或列出所有 S3 存储桶，如 - s3 ls 来检查对象存储桶名称。

5. 通过在 YAML .spec 下添加以下内容，使用以下模板在源 OBC 上应用 Azure 复制策略：

```
replicationPolicy:'{"rules":[ {"rule_id":"ID goes here", "sync_deletions": "<true or false>"",
"destination_bucket":object bucket name"}
 ], "log_replication_info":{"endpoint_type":"AZURE"}}'
```

sync_deletion

指定布尔值 **true** 或 **false**。

destination_bucket

确保使用对象存储桶的名称，而不是声明。可以使用 **s3 ls** 命令或通过在 **OBC** 的 **YAML** 中查找值来检索名称。

验证步骤

1. 将对象写入源存储桶。
2. 等待 **MCG** 复制它们。
3. 从源存储桶中删除对象。
4. 验证对象已从目标存储桶中删除。

8.3.4. 启用基于日志的存储桶复制删除

先决条件

- 管理员对 **OpenShift Web** 控制台的访问权限。
- 为所需存储桶配置的 **AWS Server Access Logging**。

流程

1. 在 **OpenShift Web** 控制台中，导航到 **Storage → Object Storage → Object Bucket Claims**。
2. 点 **Create new Object bucket claim**。
3. （可选）在 **Replication rules** 部分中，为每个规则分别选中 **Sync deletion** 复选框。

4. 输入包含 Event log Bucket 下的日志的存储桶名称。

   如果日志没有存储在存储桶的根目录中，请提供不带 s3:// 的完整路径。

5. 输入前缀来仅复制名称以给定前缀开头的对象。

第 9 章 对象 BUCKET 声明

Object Bucket Claim 可以用来为您的工作负载请求 S3 兼容存储桶后端。

您可以通过三种方式创建对象 Bucket 声明：

- 第 9.1 节 “动态对象 Bucket 声明”
- 第 9.2 节 “使用命令行界面创建对象 Bucket 声明”
- 第 9.3 节 “使用 OpenShift Web 控制台创建对象 Bucket 声明”

对象 bucket 声明在 NooBaa 中创建一个新 bucket 和应用帐户，其具有存储桶的权限，包括新的 access key 和 secret access key。应用程序帐户仅允许访问单个存储桶，默认情况下无法创建新的存储桶。

9.1. 动态对象 BUCKET 声明

与持久卷类似，您可以将 Object Bucket 声明（OBC）的详细信息添加到应用的 YAML 中，并获取配置映射和机密中可用的对象服务端点、访问密钥和 secret 访问密钥。可在应用程序的环境变量中动态阅读此信息。

注意

只有在 OpenShift 使用自签名证书时，多云对象网关端点才会使用自签名证书。在 OpenShift 中使用签名证书自动将 Multicloud Object Gateway 端点证书替换为签名证书。通过浏览器访问端点，获取多云对象网关当前使用的证书。如需更多信息，请参阅使用应用程序访问多云对象网关。

流程

1. 在应用程序 YAML 中添加以下行：

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
```

```
  name: <obc-name>
spec:
  generateBucketName: <obc-bucket-name>
  storageClassName: openshift-storage.noobaa.io
```

这些线是 **OBC** 本身。

a. 将 **<obc-name>** 替换为唯一的 **OBC** 名称。

b. 将 **<obc-bucket-name>** 替换为 **OBC** 的唯一存储桶名称。

2. 若要自动使用 **OBC** 将更多行添加到 **YAML** 文件。

例如：

```
apiVersion: batch/v1
kind: Job
metadata:
  name: testjob
spec:
  template:
    spec:
      restartPolicy: OnFailure
      containers:
        - image: <your application image>
          name: test
          env:
            - name: BUCKET_NAME
              valueFrom:
                configMapKeyRef:
                  name: <obc-name>
                  key: BUCKET_NAME
            - name: BUCKET_HOST
              valueFrom:
                configMapKeyRef:
                  name: <obc-name>
                  key: BUCKET_HOST
            - name: BUCKET_PORT
              valueFrom:
                configMapKeyRef:
                  name: <obc-name>
                  key: BUCKET_PORT
            - name: AWS_ACCESS_KEY_ID
              valueFrom:
                secretKeyRef:
                  name: <obc-name>
                  key: AWS_ACCESS_KEY_ID
```

```
        - name: AWS_SECRET_ACCESS_KEY
          valueFrom:
            secretKeyRef:
              name: <obc-name>
              key: AWS_SECRET_ACCESS_KEY
```

以下示例是存储桶声明结果之间的映射，这是一个带有凭证数据和 secret 的配置映射。此特定作业从 NooBaa 声明 Object Bucket，它将创建一个存储桶和帐户。

a. **将 <obc-name> 的所有实例替换为您的 OBC 名称。**

b. **将 <your application image> 替换为您的应用程序镜像。**

3. **应用更新的 YAML 文件：**

```
# oc apply -f <yaml.file>
```

将 <yaml.file> 替换为 YAML 文件的名称。

4. **要查看新配置映射，请运行以下命令：**

```
# oc get cm <obc-name> -o yaml
```

将 obc-name 替换为您的 OBC 的名称。

您可以在输出中预期以下环境变量：

- **BUCKET_HOST - 应用程序中使用的端点。**
- **BUCKET_PORT - 可供应用使用的端口。**
  - **端口与 BUCKET_HOST 相关。例如，如果 BUCKET_HOST 是 https://my.example.com，BUCKET_PORT 为 443，则对象服务的端点将是 https://my.example.com:443。**

- **BUCKET_NAME -** 请求或生成的存储桶名称。

- **AWS_ACCESS_KEY_ID -** 作为凭据一部分的访问密钥。

- **AWS_SECRET_ACCESS_KEY - 属于凭据的 Secret** 访问密钥。

重要

获取 **AWS_ACCESS_KEY_ID** 和 **AWS_SECRET_ACCESS_KEY**。使用名称以便与 **AWS S3 API** 兼容。您需要在执行 **S3** 操作时指定密钥，特别是从多云对象网关(MCG)存储桶中读取、写入或列表时。密钥以 **Base64** 编码。解码密钥，然后才能使用它们。

```
# oc get secret <obc_name> -o yaml
```

<obc_name>

指定对象存储桶声明的名称。

9.2. 使用命令行界面创建对象 BUCKET 声明

在使用命令行界面创建对象 **Bucket** 声明（**OBC**）时，您会获得一个配置映射和一个 **Secret**，其中包含应用使用对象存储服务所需的所有信息。

先决条件

- 下载多云对象网关(MCG)命令行界面。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注意

指定使用订阅管理器启用存储库的适当架构。

- **对于 IBM Power，使用以下命令：**

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- **对于 IBM Z，使用以下命令：**

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

流程

1. **使用命令行界面生成新 bucket 和凭据的详细信息。**

   运行以下命令：

   ```
   # noobaa obc create <obc-name> -n openshift-storage
   ```

   将 <obc-name> 替换为唯一的 OBC 名称，如 myappobc。

   另外，您可以使用 --app-namespace 选项指定创建 OBC 配置映射和 secret 的命名空间，如 myapp-namespace。

   例如：

   ```
   INFO[0001]  Created: ObjectBucketClaim "test21obc"
   ```

   MCG 命令行界面已创建了必要的配置，并已向 OpenShift 告知新的 OBC。

2. **运行以下命令来查看 OBC：**

   ```
   # oc get obc -n openshift-storage
   ```

例如：

```
NAME       STORAGE-CLASS                 PHASE   AGE
test21obc  openshift-storage.noobaa.io   Bound   38s
```

3. **运行以下命令查看新 OBC 的 YAML 文件：**

```
# oc get obc test21obc -o yaml -n openshift-storage
```

例如：

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  creationTimestamp: "2019-10-24T13:30:07Z"
  finalizers:
  - objectbucket.io/finalizer
  generation: 2
  labels:
    app: noobaa
    bucket-provisioner: openshift-storage.noobaa.io-obc
    noobaa-domain: openshift-storage.noobaa.io
  name: test21obc
  namespace: openshift-storage
  resourceVersion: "40756"
  selfLink: /apis/objectbucket.io/v1alpha1/namespaces/openshift-
storage/objectbucketclaims/test21obc
  uid: 64f04cba-f662-11e9-bc3c-0295250841af
spec:
  ObjectBucketName: obc-openshift-storage-test21obc
  bucketName: test21obc-933348a6-e267-4f82-82f1-e59bf4fe3bb4
  generateBucketName: test21obc
  storageClassName: openshift-storage.noobaa.io
status:
  phase: Bound
```

4. **在 openshift-storage 命名空间内，您可以找到配置映射和 secret 来使用此 OBC。CM 和 secret 的名称与 OBC 相同。**

运行以下命令来查看 secret：

```
# oc get -n openshift-storage secret test21obc -o yaml
```

例如：

```
apiVersion: v1
data:
  AWS_ACCESS_KEY_ID: c0M0R2xVanF3ODR3bHBkVW94cmY=
  AWS_SECRET_ACCESS_KEY:
Wi9kcFluSWxHRzlWaFlzNk1hc0xma2JXcjM1MVhqa051SlBleXpmOQ==
kind: Secret
metadata:
  creationTimestamp: "2019-10-24T13:30:07Z"
  finalizers:
  - objectbucket.io/finalizer
  labels:
    app: noobaa
    bucket-provisioner: openshift-storage.noobaa.io-obc
    noobaa-domain: openshift-storage.noobaa.io
  name: test21obc
  namespace: openshift-storage
  ownerReferences:
  - apiVersion: objectbucket.io/v1alpha1
    blockOwnerDeletion: true
    controller: true
    kind: ObjectBucketClaim
    name: test21obc
    uid: 64f04cba-f662-11e9-bc3c-0295250841af
  resourceVersion: "40751"
  selfLink: /api/v1/namespaces/openshift-storage/secrets/test21obc
  uid: 65117c1c-f662-11e9-9094-0a5305de57bb
type: Opaque
```

该机密为您提供了 S3 访问凭据。

5. **运行以下命令来查看配置映射：**

```
# oc get -n openshift-storage cm test21obc -o yaml
```

例如：

```
apiVersion: v1
data:
  BUCKET_HOST: 10.0.171.35
  BUCKET_NAME: test21obc-933348a6-e267-4f82-82f1-e59bf4fe3bb4
  BUCKET_PORT: "31242"
  BUCKET_REGION: ""
  BUCKET_SUBREGION: ""
kind: ConfigMap
metadata:
```

```
creationTimestamp: "2019-10-24T13:30:07Z"
finalizers:
- objectbucket.io/finalizer
labels:
  app: noobaa
  bucket-provisioner: openshift-storage.noobaa.io-obc
  noobaa-domain: openshift-storage.noobaa.io
name: test21obc
namespace: openshift-storage
ownerReferences:
- apiVersion: objectbucket.io/v1alpha1
  blockOwnerDeletion: true
  controller: true
  kind: ObjectBucketClaim
  name: test21obc
  uid: 64f04cba-f662-11e9-bc3c-0295250841af
resourceVersion: "40752"
selfLink: /api/v1/namespaces/openshift-storage/configmaps/test21obc
uid: 651c6501-f662-11e9-9094-0a5305de57bb
```

配置映射包含应用的 S3 端点信息。

9.3. 使用 OPENSHIFT WEB 控制台创建对象 BUCKET 声明

您可以使用 OpenShift Web 控制台创建对象 BucketClaim (OBC)。

先决条件

- 对 **OpenShift Web 控制台的管理访问权限.**
- 为了让您的应用程序与 **OBC 通信，您需要使用 configmap 和 secret。有关此信息的详情请参考 第 9.1 节 “动态对象 Bucket 声明”。**

流程

1. **登录 OpenShift Web 控制台。**
2. **在左侧导航栏中，点击 Storage → Object Storage → Object Bucket Claims → Create Object Bucket Claim。**
   a. **输入对象存储桶声明的名称，并根据您的部署（内部或外部）从下拉菜单中选择适当的存储类：**

内部模式

以下存储类是在部署后创建的，可供使用：

- **OCS-storagecluster-ceph-rgw 使用 Ceph 对象网关 (RGW)**
- **openshift-storage.noobaa.io 使用 Multicloud 对象网关(MCG)**

外部模式

以下存储类是在部署后创建的，可供使用：

- **ocs-external-storagecluster-ceph-rgw 使用 RGW**
- **openshift-storage.noobaa.io 使用 MCG**

注意

RGW OBC 存储类仅可用于全新安装的 OpenShift Data Foundation 版本 4.5。它不适用于从以前的 OpenShift 数据基础版本升级的集群。

b.

点 Create。

创建 OBC 后，您会被重定向到其详情页面。

9.4. 将对象 BUCKET 声明附加到部署

创建后，对象 Bucket 声明 (OBC) 可以附加到特定的部署。

先决条件

- 对 OpenShift Web 控制台的管理访问权限.

流程

1. 在左侧导航栏中，点击 Storage → Object Storage → Object Bucket Claims。
2. 单击您创建的 OBC 旁边的操作菜单 (⋮)。
   a. 从下拉菜单中选择 Attach to Deployment。
   b. 从 Deployment Name 列表中选择所需的部署，然后单击 Attach。

9.5. 使用 OPENSHIFT WEB 控制台查看对象存储桶

您可以使用 OpenShift Web 控制台查看为对象 Bucket 声明 (OBC) 创建的对象存储桶详情。

先决条件

- 对 OpenShift Web 控制台的管理访问权限.

流程

1. 登录 OpenShift Web 控制台。
2. 在左侧导航栏中，点击 Storage → Object Storage → Object Buckets。

   可选：您还可以进入到特定 OBC 的详细信息页面，再点 Resource 链接来查看该 OBC 的对象存储桶。
3. 选择您要查看详情的对象存储桶。选择后，您将进入 Object Bucket Details 页面。

9.6. 删除对象 BUCKET 声明

先决条件

先决条件

- 对 **OpenShift Web 控制台的管理**访问权限.

流程

1. 在左侧导航栏中，点击 **Storage → Object Storage → Object Bucket Claims**。
2. 点击您要删除的 **Object Bucket Claim(OBC)** 旁边的 **Action** 菜单 (⋮)。
   a. 选择 **Delete Object Bucket Claim**。
   b. 点 **Delete**。

第 10 章 对象存储桶的缓存策略

缓存存储桶是带有 hub 目标和缓存目标的命名空间存储桶。hub 目标是一个 S3 兼容的大型对象存储桶。缓存存储桶是本地多云对象网关(MCG)存储桶。您可以创建一个缓存存储桶来缓存 AWS 存储桶或 IBM COS 存储桶。

- AWS S3
- IBM COS

10.1. 创建 AWS 缓存存储桶

先决条件

- 下载多云对象网关(MCG)命令行界面。

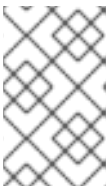
```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注意

指定使用订阅管理器启用存储库的适当架构。如果是 IBM Z，请使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

另外，您还可以从位于 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/package的 OpenShift Data Foundation RPM 安装 MCG 软件包。



注意

根据您的架构选择正确的产品变体。

流程

1. 创建 NamespaceStore 资源。NamespaceStore 代表底层存储，用作 MCG 命名空间存储桶中数据的读取或写入目标。在 MCG 命令行界面中运行以下命令：

```
noobaa namespacestore create aws-s3 <namespacestore> --access-key <AWS ACCESS
KEY> --secret-key <AWS SECRET ACCESS KEY> --target-bucket <bucket-name>
```

a. **将 <namespacestore> 替换为命名空间存储的名称。**

b. **将 <AWS ACCESS KEY> 和 <AWS SECRET ACCESS KEY> 替换为您为此创建的 AWS 访问密钥 ID 和 secret 访问密钥。**

c. **将 <bucket-name> 替换为现有的 AWS 存储桶名称。此参数告知 MCG 将哪一个存储桶用作其后备存储的目标存储桶，以及随后数据存储和管理。**

您还可以通过应用 YAML 来添加存储资源。首先使用凭证创建 secret：

```
apiVersion: v1
kind: Secret
metadata:
  name: <namespacestore-secret-name>
type: Opaque
data:
  AWS_ACCESS_KEY_ID: <AWS ACCESS KEY ID ENCODED IN BASE64>
  AWS_SECRET_ACCESS_KEY: <AWS SECRET ACCESS KEY ENCODED IN
BASE64>
```

您必须使用 Base64 提供并编码您自己的 AWS 访问密钥 ID 和 secret 访问密钥，并使用结果代替 <AWS ACCESS KEY ID ENCODED IN BASE64> 和 <AWS SECRET ACCESS KEY ENCODED IN BASE64>。

使用一个唯一的名称替换 <namespacestore-secret-name>。

然后应用以下 YAML:

```
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <namespacestore>
  namespace: openshift-storage
```

```
spec:
  awsS3:
    secret:
      name: <namespacestore-secret-name>
      namespace: <namespace-secret>
    targetBucket: <target-bucket>
  type: aws-s3
```

d. **将 <namespacestore> 替换为唯一的名称。**

e. **将 <namespacestore-secret-name> 替换为上一步中创建的 secret。**

f. **将 <namespace-secret> 替换为用于在上一步中创建 secret 的命名空间。**

g. **将 <target-bucket> 替换为您为命名空间存储创建的 AWS S3 存储桶。**

2. **运行以下命令来创建存储桶类：**

```
noobaa bucketclass create namespace-bucketclass cache <my-cache-bucket-class> --
backingstores <backing-store> --hub-resource <namespacestore>
```

a. **将 <my-cache-bucket-class> 替换为唯一的存储桶类名称。**

b. **将 <backing-store> 替换为相关的后备存储。您可以在此字段中列出一个或多个以逗号分开的后备存储。**

c. **将 <namespacestore> 替换为上一步中创建的命名空间存储。**

3. **运行以下命令，以使用 Object Bucket Claim (OBC) 资源创建 bucket，该资源使用第 2 步中定义的 bucket 类。**

```
noobaa obc create <my-bucket-claim> my-app --bucketclass <custom-bucket-class>
```

a. **将 <my-bucket-claim> 替换为唯一名称。**

b. 将 <custom-bucket-class> 替换为在第 2 步中创建的 bucket 类的名称。

10.2. 创建 IBM COS 缓存存储桶

先决条件

- 下载多云对象网关(MCG)命令行界面。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注意

指定使用订阅管理器启用存储库的适当架构。

- 对于 IBM Power，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- 对于 IBM Z，使用以下命令：

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

另外，您还可以从位于 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/package的 OpenShift Data Foundation RPM 安装 MCG 软件包。

注意

根据您的架构选择正确的产品变体。

流程

1. 创建 NamespaceStore 资源。NamespaceStore 代表底层存储，用作 MCG 命名空间存储桶中数据的读取或写入目标。在 MCG 命令行界面中运行以下命令：

```
noobaa namespacestore create ibm-cos <namespacestore> --endpoint <IBM COS
ENDPOINT> --access-key <IBM ACCESS KEY> --secret-key <IBM SECRET ACCESS
KEY> --target-bucket <bucket-name>
```

a. **将 <namespacestore> 替换为 NamespaceStore 的名称。**

b. **将 <IBM ACCESS KEY>、、<IBM SECRET ACCESS KEY>、<IBM COS ENDPOINT> 替换为 IBM 访问密钥 ID、机密访问密钥和对应于现有 IBM 存储桶位置的适当区域端点。**

c. **将 <bucket-name> 替换为现有的 IBM 存储桶名称。此参数告知 MCG 将哪一个存储桶用作其后备存储的目标存储桶，以及随后数据存储和管理。**

您还可以通过应用 YAML 来添加存储资源。首先，使用凭证创建一个 secret：

```
apiVersion: v1
kind: Secret
metadata:
  name: <namespacestore-secret-name>
type: Opaque
data:
  IBM_COS_ACCESS_KEY_ID: <IBM COS ACCESS KEY ID ENCODED IN BASE64>
  IBM_COS_SECRET_ACCESS_KEY: <IBM COS SECRET ACCESS KEY ENCODED
IN BASE64>
```

您必须使用 Base64 提供和编码您自己的 IBM COS 访问密钥 ID 和 secret 访问密钥，并使用结果代替 <IBM COS ACCESS KEY ID ENCODED IN BASE64> 和 <IBM COS SECRET ACCESS KEY ENCODED IN BASE64>。

使用一个唯一的名称替换 <namespacestore-secret-name>。

然后应用以下 YAML：

```
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <namespacestore>
  namespace: openshift-storage
```

```
spec:
  s3Compatible:
    endpoint: <IBM COS ENDPOINT>
    secret:
      name: <backingstore-secret-name>
      namespace: <namespace-secret>
    signatureVersion: v2
    targetBucket: <target-bucket>
  type: ibm-cos
```

d. **将 <namespacestore> 替换为唯一的名称。**

e. **将 <IBM COS ENDPOINT> 替换为适当的 IBM COS 端点。**

f. **将 <backingstore-secret-name> 替换为上一步中创建的 secret。**

g. **将 <namespace-secret> 替换为用于在上一步中创建 secret 的命名空间。**

h. **将 <target-bucket> 替换为您为命名空间存储创建的 AWS S3 存储桶。**

2. **运行以下命令来创建存储桶类：**

```
noobaa bucketclass create namespace-bucketclass cache <my-bucket-class> --
backingstores <backing-store> --hubResource <namespacestore>
```

a. **将 <my-bucket-class> 替换为唯一的存储桶类名称。**

b. **将 <backing-store> 替换为相关的后备存储。您可以在此字段中列出一个或多个以逗号分开的后备存储。**

c. **将 <namespacestore> 替换为上一步中创建的命名空间存储。**

3. **运行以下命令，以使用 Object Bucket Claim 资源创建 bucket，该资源使用第 2 步中定义的 bucket 类。**

```
noobaa obc create <my-bucket-claim> my-app --bucketclass <custom-bucket-class>
```

a.

将 <my-bucket-claim> 替换为唯一名称。

b.

将 <custom-bucket-class> 替换为在第 2 步中创建的 bucket 类的名称。

第 11 章 MULTICLOUD 对象网关中的生命周期存储桶配置

Multicloud Object Gateway (MCG)生命周期提供了一种降低存储成本的方法，因为累积了数据对象。

删除已过期的对象是简化处理未使用的数据的方法。数据过期是 Amazon Web Services (AWS) 生命周期管理的一部分，并为自动删除设置过期日期。生命周期过期时间的最短时间仅为一天。如需更多信息，请参阅 过期对象。

AWS S3 API 用于在 MCG 中配置生命周期存储桶。有关数据存储桶 API 及其 支持级别的详情，请参考 对多云对象网关数据存储桶 API 的支持。

在与 AWS 共同映射时 MCG 的 expiratation 规则 API 有一些限制：

- ExpiredObjectDeleteMarker 被接受，但没有处理。
- 没有定义特定非当前版本的过期条件的选项

第 12 章 扩展多云对象网关性能

多云对象网关(MCG)的性能可能会因环境而异。在某些情况下，特定的应用需要更快的性能，这可以通过扩展 S3 端点来轻松解决。

MCG 资源池是一组 NooBaa 守护进程容器，默认启用两种类型的服务：

- 存储服务
- S3 端点服务

S3 端点服务

S3 端点是每个多云对象网关(MCG)默认提供的服务，用于处理 MCG 中的繁重数据摘要。端点服务处理内联数据块、重复数据删除、压缩和加密，它接受来 MCG 的数据放置指令。

12.1. 自动扩展 MULTICLOUD OBJECT GATEWAY 端点

当 MCG S3 服务的负载增加或减少时，MultiCloud Object Gateway(MCG)端点的数量会自动缩放。OpenShift 数据基础集群使用一个活跃的 MCG 端点进行部署。每个 MCG 端点 pod 都默认配置有 1 个 CPU 和 2Gi 内存请求，其限值与请求匹配。当端点上的 CPU 负载在固定时间段内超过 80% 用量阈值时，部署第二个端点会降低第一个端点的负载。当两个端点的平均 CPU 负载在固定时间段内低于 80% 阈值时，会删除其中一个端点。此功能提高了 MCG 的性能和可服务性。

您可以使用以下 oc patch 命令扩展 noobaa-endpoint 的 Horizontal Pod Autoscaler (HPA)，例如：

```
# oc patch -n openshift-storage storagecluster ocs-storagecluster \
    --type merge \
    --patch '{"spec": {"multiCloudGateway": {"endpoints": {"minCount": 3,"maxCount": 10}}}}'
```

上面的示例将 minCount 设为 3，maxCount 设为 '10。

12.2. 为 PV 池资源增加 CPU 和内存

MCG 默认配置支持低资源消耗。但是，当您需要提高 CPU 和内存以适应特定工作负载并增加工作负载的 MCG 性能时，您可以在 OpenShift Web 控制台中为 CPU 和内存配置所需的值。

流程

1. **在 OpenShift Web 控制台中，导航到 Storage → Object Storage → Backing Store。**

2. **选择相关的后备存储并点击 YAML。**

3. **向下滚动，直到找到 spec: 并使用 CPU 和内存更新 pvPool。添加限制的新属性，然后添加 cpu 和 memory。**

   参考示例：

```
spec:
  pvPool:
    resources:
      limits:
        cpu: 1000m
        memory: 4000Mi
      requests:
        cpu: 800m
        memory: 800Mi
        storage: 50Gi
```

4. **点 Save。**

验证步骤

- **要验证，您可以检查 PV 池 pod 的资源值。**

第 13 章 访问 RADOS 对象网关 S3 端点

用户可以直接访问 RADOS 对象网关 (RGW) 端点。

在以前的 Red Hat OpenShift Data Foundation 版本中，需要手动公开的 RGW 服务来创建 RGW 公共路由。自 OpenShift Data Foundation 版本 4.7 起，默认情况下会创建 RGW 路由，并命名为 rook-ceph-rgw-ocs-storagecluster-cephobjectstore。

第 14 章 使用 TLS 证书用于应用程序来访问 RGW

大多数 S3 应用程序都需要 TLS 证书，如 Deployment 配置文件中包含的选项、作为请求中的文件传递，或者存储在 /etc/pki 路径中。

RADOS 对象网关 (RGW) 的 TLS 证书被存储为 Kubernetes 的 secret，您需要从 secret 中获取详细信息。

先决条件

正在运行的 OpenShift Data Foundation 集群。

流程

- 对于内部 RGW 服务器
  - 从 kubernetes secret 获取 TLS 证书和密钥：

```
$ oc get secrets/<secret_name> -o jsonpath='{.data..tls\.crt}' | base64 -d

$ oc get secrets/<secret_name> -o jsonpath='{.data..tls\.key}' | base64 -d
```

<secret_name>

默认 kubernetes secret 的名称为 **<objectstore_name>-cos-ceph-rgw-tls-cert**。指定对象存储的名称。

- 对于外部 RGW 服务器
  - 从 kubernetes secret 获取 TLS 证书：

```
$ oc get secrets/<secret_name> -o jsonpath='{.data.cert}' | base64 -d
```

<secret_name>

默认 kubernetes secret 的名称为 ceph-rgw-tls-cert，它是一个不透明类型的 secret。用于存储 TLS 证书的键值是 cert。