Red Hat OpenShift Data Foundation 4.9

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: 2023-02-14

Red Hat OpenShift Data Foundation 4.9 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。

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Managing_hybrid_and_multicloud_resources.ent file |.

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 4

对红帽文档提供反馈 5

第 1 章 关于 MULTICLOUD 对象网关 6

第 2 章 使用应用程序访问多云对象网关 7
- 2.1. 从终端访问 MULTICLOUD 对象网关 8
- 2.2. 使用 MCG 命令行界面访问 MULTICLOUD 对象网关 10

第 3 章 允许用户访问多云对象网关控制台 13

第 4 章 为混合或多云添加存储资源 15
- 4.1. 创建新的后备存储 15
- 4.2. 使用 MCG 命令行界面为混合或多云添加存储资源 15
  - 4.2.1. 创建 AWS 支持的后备存储 16
  - 4.2.2. 创建 IBM COS 支持的后备存储 18
  - 4.2.3. 创建 Azure 支持的后备存储 19
  - 4.2.4. 创建由 GCP 支持的后备存储 21
  - 4.2.5. 创建由本地持久性卷支持的后备存储 23
- 4.3. 创建兼容 S3 的多云对象网关后备存储 24
- 4.4. 使用用户界面为混合和多云添加存储资源 25
- 4.5. 创建新存储桶类 27
- 4.6. 编辑存储桶类 28
- 4.7. 为存储桶类编辑后备存储 28

第 5 章 管理命名空间存储桶 30
- 5.1. 命名空间存储桶中对象的 AMAZON S3 API 端点 30
- 5.2. 使用 MULTICLOUD 对象网关 CLI 和 YAML 添加命名空间存储桶 31
  - 5.2.1. 使用 YAML 添加 AWS S3 命名空间存储桶 31
  - 5.2.2. 使用 YAML 添加 IBM COS 命名空间存储桶 33
  - 5.2.3. 使用 Multicloud 对象网关 CLI 添加 AWS S3 命名空间存储桶 35
  - 5.2.4. 使用 Multicloud 对象网关 CLI 添加 IBM COS 命名空间存储桶 37
- 5.3. 使用 OPENSHIFT CONTAINER PLATFORM 用户界面添加命名空间存储桶 39

第 6 章 混合和多云存储桶的镜像数据 41
- 6.1. 使用 MCG 命令行创建存储桶类来镜像数据 41
- 6.2. 使用 YAML 创建存储桶类来镜像数据 41
- 6.3. 使用用户界面将存储桶配置为镜像数据 42

第 7 章 MULTICLOUD 对象网关中的存储桶策略 44
- 7.1. 关于存储桶策略 44
- 7.2. 使用存储桶策略 44
- 7.3. 在 MULTICLOUD 对象网关中创建 AWS S3 用户 45

第 8 章 MULTICLOUD OBJECT GATEWAY 存储桶和存储桶类复制 48
- 8.1. 将存储桶复制到其他存储桶 48
  - 8.1.1. 使用 MCG 命令行界面将存储桶复制到另一个存储桶 49
  - 8.1.2. 使用 YAML 将存储桶复制到另一个存储桶 49
- 8.2. 设置存储桶类复制策略 50
  - 8.2.1. 使用 MCG 命令行界面设置存储桶类复制策略 50
  - 8.2.2. 使用 YAML 设置存储桶类复制策略 51

第 9 章 对象 BUCKET 声明 53

9.1. 动态对象 BUCKET 声明 53
9.2. 使用命令行界面创建对象 BUCKET 声明 55
9.3. 使用 OPENSHIFT WEB 控制台创建对象 BUCKET 声明 58
9.4. 将对象 BUCKET 声明附加到部署 60
9.5. 使用 OPENSHIFT WEB 控制台查看对象存储桶 61
9.6. 删除对象 BUCKET 声明 62

第 10 章 对象存储桶的缓存策略 **63**
10.1. 创建 AWS 缓存存储桶 63
10.2. 创建 IBM COS 缓存存储桶 65

第 11 章 通过添加端点扩展多云对象网关性能 **68**
11.1. 自动扩展 MULTICLOUD OBJECT GATEWAY 端点 68
11.2. 使用存储节点扩展 MULTICLOUD 对象网关 68

第 12 章 访问 RADOS 对象网关 S3 端点 **69**

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。提供反馈：

- 关于特定内容的简单评论：
  1. 请确定您使用 *Multi-page HTML* 格式查看文档。另外，确定 **Feedback** 按钮出现在文档页的右上方。
  2. 用鼠标指针高亮显示您想评论的文本部分。
  3. 点在高亮文本上弹出的 **Add Feedback**。
  4. 按照显示的步骤操作。
- 要提交更复杂的反馈，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 基础架构，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：

- 第 2.1 节 “从终端访问 Multicloud 对象网关”
- 第 2.2 节 “使用 MCG 命令行界面访问 Multicloud 对象网关”

例 2.1. 示例

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，以便使用虚拟主机样式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

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 基础架构，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

现在，您有相关的端点、访问密钥和 secret 访问密钥来连接到您的应用。

例 2.2. 示例

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
```

第 3 章 允许用户访问多云对象网关控制台

要允许用户访问 Multicloud 对象网关(MCG)控制台，请确保用户满足以下条件：

- 用户位于 **cluster-admins** 组中。
- 用户位于 **system:cluster-admins** 虚拟组中.

先决条件

- 正在运行的 OpenShift Data Foundation 平台。

流程

1. 启用对 MCG 控制台的访问。
   在集群中执行以下步骤：

   a. 创建 **cluster-admins** 组。

   ```
   # oc adm groups new cluster-admins
   ```

   b. 将组绑定到 **cluster-admin** 角色。

   ```
   # oc adm policy add-cluster-role-to-group cluster-admin cluster-admins
   ```

2. 从 **cluster-admins** 组中添加或删除用户，以控制对 MCG 控制台的访问。

   - 将一组用户添加到 **cluster-admins** 组：

     ```
     # oc adm groups add-users cluster-admins <user-name> <user-name> <user-name>...
     ```

     其中 **<user-name>** 是要添加的用户的名称。

     注意

     如果您要向 **cluster-admins** 组添加一组用户，您不需要将新添加的用户绑定到 cluster-admin 角色，以允许访问 OpenShift Data Foundation 仪表板。

   - 从 **cluster-admins** 组中删除一组用户：

     ```
     # oc adm groups remove-users cluster-admins <user-name> <user-name> <user-name>...
     ```

     其中 **<user-name>** 是要删除的用户的名称。

验证步骤

1. 在 OpenShift Web 控制台中，以有权访问多云对象网关控制台的用户身份登录。
2. 导航到 **Storage → OpenShift Data Foundation**。
3. 在 **Storage Systems** 选项卡中，选择 storage 系统，然后点 **Overview → Object** 选项卡。

4. 选择 **Multicloud Object Gateway** 链接。
5. 点击 **Allow selected permissions**。

第 4 章 为混合或多云添加存储资源

4.1. 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** → **OpenShift Data Foundation**。
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. 输入 **端点**.这是可选的。
   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 4.2 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** → **OpenShift Data Foundation**。
2. 单击 **Backing Store** 选项卡，以查看所有后备存储。

4.2. 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 4.2.1 节 “创建 AWS 支持的后备存储”
- 有关创建 IBM COS 支持的后备存储，请参考 第 4.2.2 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 4.2.3 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 4.2.4 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 4.2.5 节 “创建由本地持久性卷支持的后备存储”

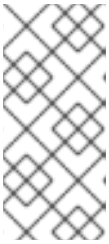
对于 VMware 部署，请跳至 第 4.3 节 “创建兼容 s3 的多云对象网关后备存储” 以了解更多详细信息。

4.2.1. 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 基础架构，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

1. 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
```

a. 将 **<backingstore_name>** 替换为后备存储的名称。

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。
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"
```

您还可以使用 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>
```

a. 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>**。

b. 将 **<backingstore-secret-name>** 替换为唯一名称。

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:
  awsS3:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBucket: <bucket-name>
  type: aws-s3
```

a. 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。

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

4.2.2. 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 基础架构，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

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
```

   a. 将 **<backingstore_name>** 替换为后备存储的名称。

   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。
   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。

   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。
   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"
```

您还可以使用 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>
```

   a. 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>**。

   b. 将 **<backingstore-secret-name>** 替换为唯一名称。

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

   a. 将 **<bucket-name>** 替换为现有的 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。

   b. 将 **<endpoint>** 替换为与现有 IBM 存储桶名称位置对应的区域端点。这个参数告诉 Multicloud Object Gateway 使用哪个端点进行后备存储，然后再将哪一端点用于数据存储和管理。

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

4.2.3. 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 基础架构，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

1. 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>
```

   a. 将 **<backingstore_name>** 替换为后备存储的名称。

   b. 将 **<AZURE ACCOUNT KEY>** 和 **<AZURE ACCOUNT NAME>** 替换为您为此创建的 AZURE 帐户密钥和帐户名称。

   c. 将 **<blob container name>** 替换为现有的 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"
```

您还可以使用 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>
```

a. 您必须使用 Base64 提供和编码您自己的 Azure 帐户名称和帐户密钥，并使用结果代替 **<AZURE ACCOUNT NAME ENCODED IN BASE64>** 和 **<AZURE ACCOUNT KEY ENCODED IN BASE64>**。

b. 将 **<backingstore-secret-name>** 替换为唯一名称。

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:
  azureBlob:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBlobContainer: <blob-container-name>
  type: azure-blob
```

a. 将 **<blob-container-name>** 替换为现有的 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。

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

4.2.4. 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 基础架构，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

1. 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>
```

   a. 将 **<backingstore_name>** 替换为后备存储的名称。

   b. 将 **<PATH TO GCP PRIVATE KEY JSON FILE>** 替换为为此创建的 GCP 私钥的路径。

   c. 将 **<GCP bucket name>** 替换为现有的 GCP 对象存储桶名称。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"
```

您还可以使用 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>
```

   a. 您必须使用 Base64 提供并编码您自己的 GCP 服务帐户私钥，并使用结果代替 **<GCP PRIVATE KEY ENCODED IN BASE64>**。

   b. 将 **<backingstore-secret-name>** 替换为唯一名称。

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
```

a. 将 **<target bucket>** 替换为现有的 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。

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

4.2.5. 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

指定使用订阅管理器启用存储库的适当架构。例如，如果是 IBM Z 基础架构，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

1. 在 MCG 命令行界面中运行以下命令：

```
noobaa backingstore create  pv-pool <backingstore_name> --num-volumes=<NUMBER OF
VOLUMES>  --pv-size-gb=<VOLUME SIZE> --storage-class=<LOCAL STORAGE CLASS>
```

a. 将 **<backingstore_name>** 替换为后备存储的名称。

b. 将 **<NUMBER OF VOLUMES>** 替换为您要创建的卷数。请注意，增加卷数量可向上扩展存储。

c. 将 **<VOLUME SIZE>** 替换为每个卷所需的大小（以 GB 为单位）。

d. 将 **<LOCAL STORAGE CLASS>** 替换为本地存储类，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"
```

您还可以使用 YAML 添加存储资源：

1. 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>
    storageClass: <LOCAL STORAGE CLASS>
  type: pv-pool
```

a. 将 **<backingstore_name>** 替换为后备存储的名称。

b. 将 **<NUMBER OF VOLUMES>** 替换为您要创建的卷数。请注意，增加卷数量可向上扩展存储。

c. 将 **<VOLUME SIZE>** 替换为每个卷所需的大小（以 GB 为单位）。请注意，字母 **G** 应保留。

d. 将 **<LOCAL STORAGE CLASS>** 替换为本地存储类，建议使用 **ocs-storagecluster-ceph-rbd**。

4.3. 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：

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

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. 要获取 **<RGW endpoint>**，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。

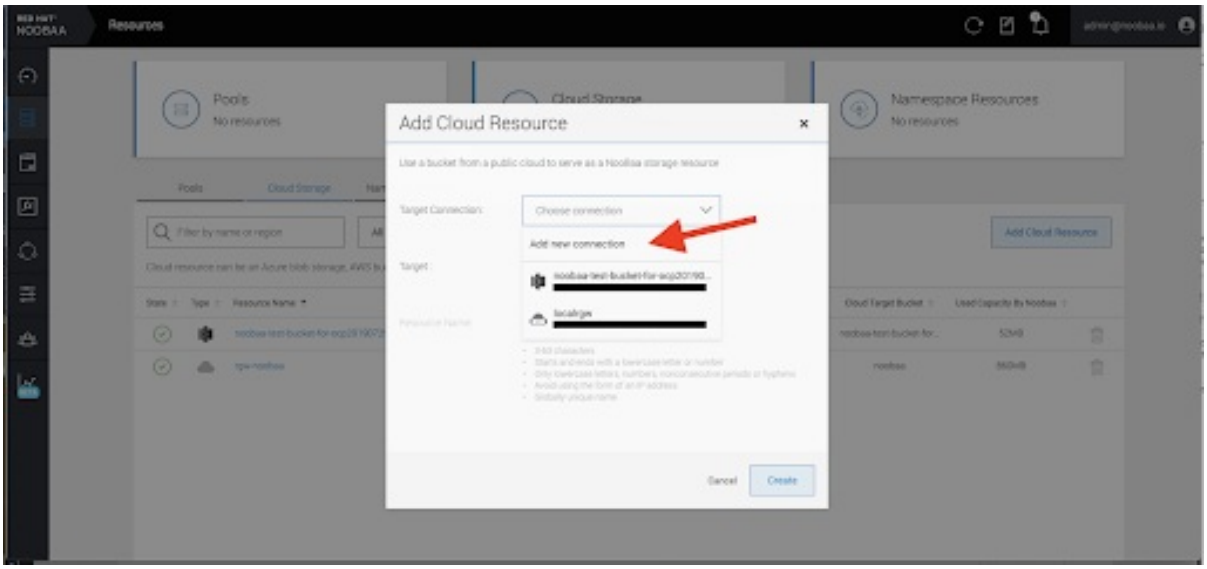
   c. 要获取 **<RGW endpoint>**，请参阅访问 RADOS 对象网关 S3 端点。

4.4. 使用用户界面为混合和多云添加存储资源

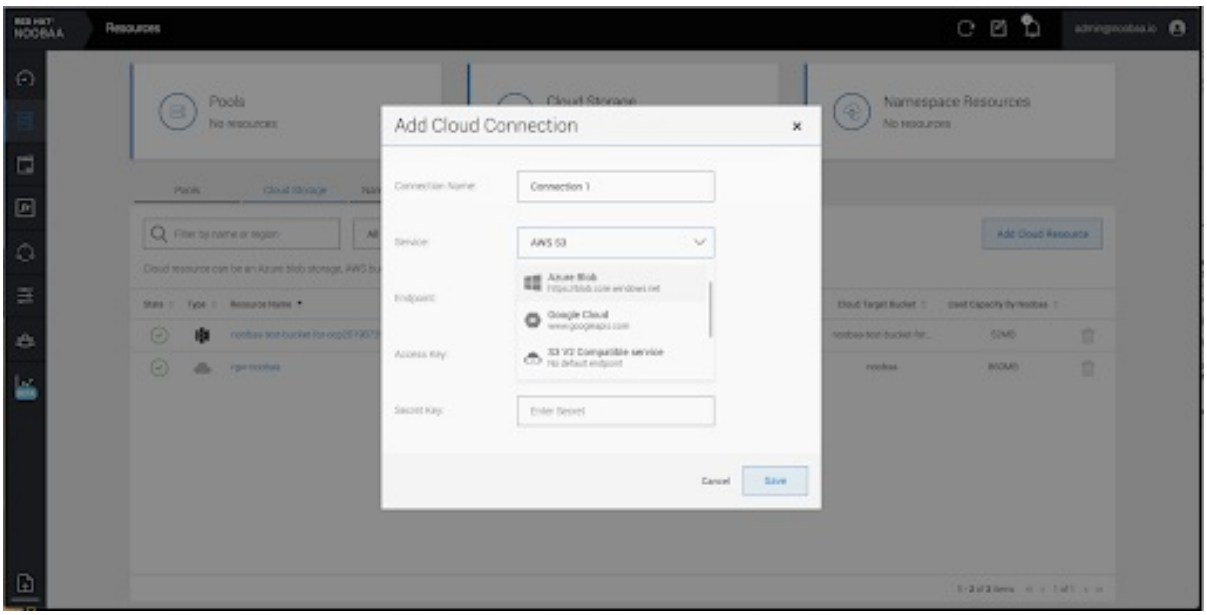
流程

1. 在 OpenShift Web 控制台中，点 **Storage → OpenShift Data Foundation**。
2. 在 **Storage Systems** 选项卡中，选择 storage 系统，然后点 **Overview → Object** 选项卡。
3. 选择 **Multicloud Object Gateway** 链接。
1. 选择左侧的**资源**选项卡，突出显示下方。从填充的列表中，选择 **Add Cloud Resource**。

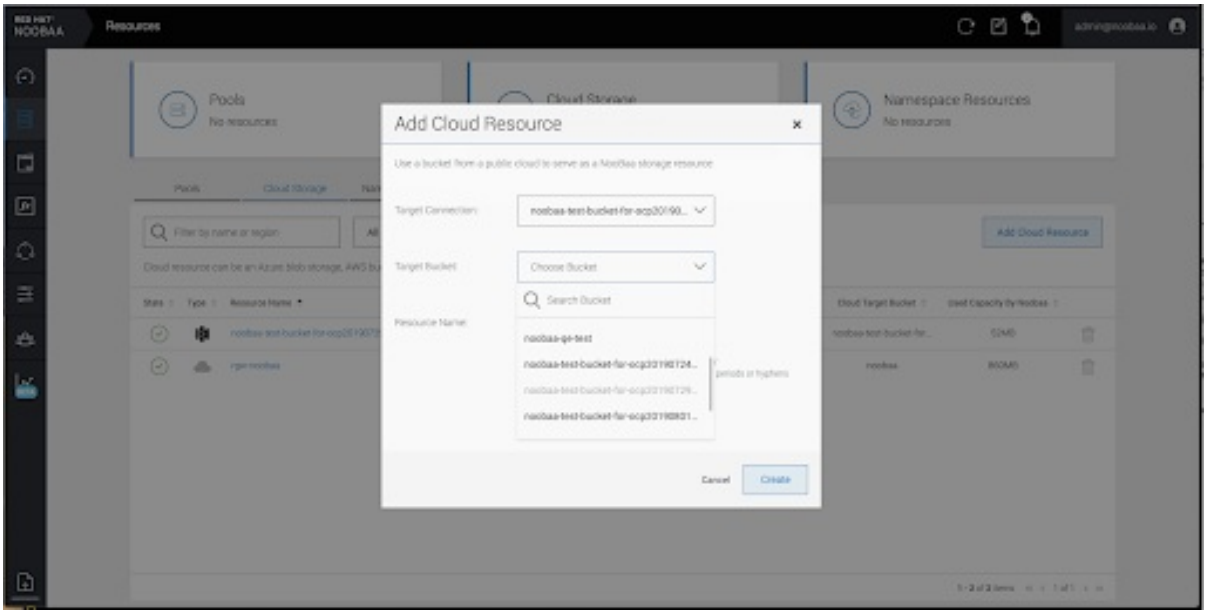
2. 选择 **Add new connection**。



3. 选择相关的原生云供应商或 S3 兼容选项并填写详情。



4. 选择新创建的连接并将其映射到现有存储桶。



5. 重复这些步骤，根据需要创建任意数量的后备存储。

注意

在 NooBaa UI 中创建的资源不能由 OpenShift UI 或 MCG CLI 使用。

4.5. 创建新存储桶类

bucket 类是一个 CRD，代表一种存储桶类别，用于定义对象 Bucket 类 (OBC) 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 → OpenShift Data Foundation**。
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 → OpenShift Data Foundation**。
2. 单击 **Bucket Class** 选项卡，再搜索新的 Bucket 类。

4.6. 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 → OpenShift Data Foundation**。
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**。

4.7. 为存储桶类编辑后备存储

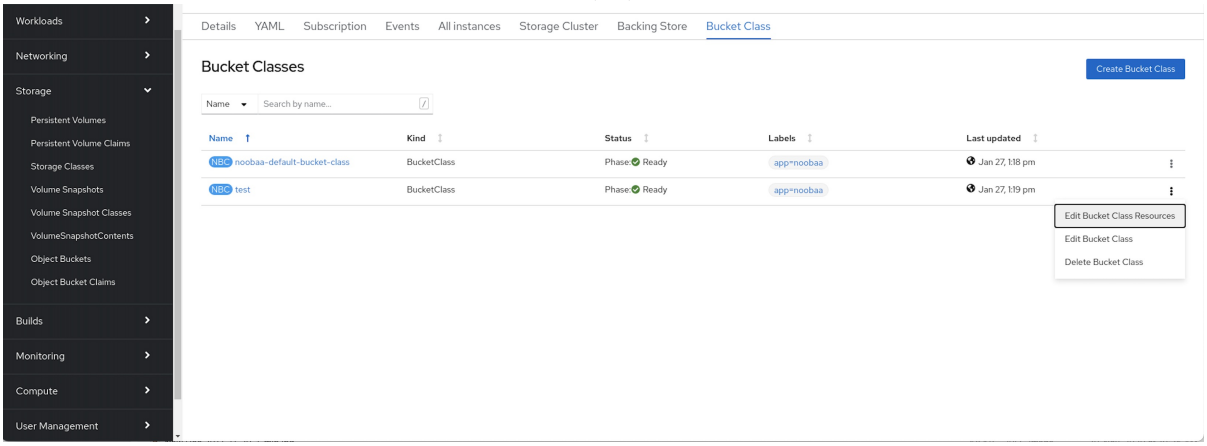
使用以下步骤编辑现有的多云对象网关(MCG)存储桶类，以更改存储桶类中使用的底层后备存储。

先决条件

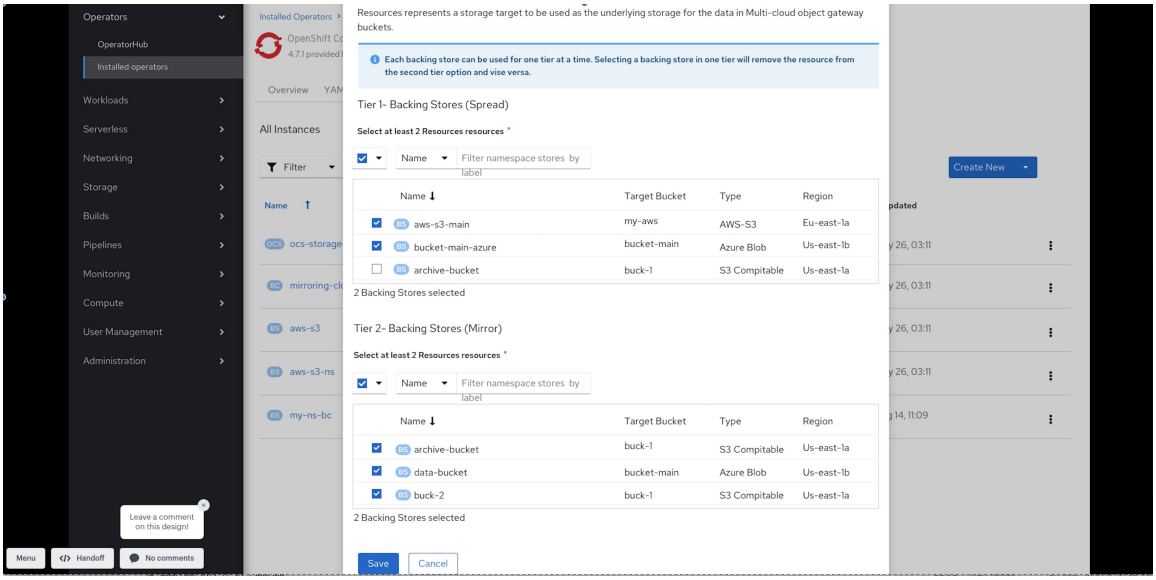
- 管理员对 OpenShift Web 控制台的访问权限。
- 存储桶类。
- 后备存储。

流程

1. 在 OpenShift Web 控制台中，点 **Storage → OpenShift Data Foundation**。
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，请清除后备存储的名称。



6. 点击 **Save**。

第 5 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

只有其写入目标可用且可正常运行时，才能使用命名空间存储桶。

5.1. 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。

其它资源

- Amazon S3 REST API 参考
- Amazon S3 CLI 参考

5.2. 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

5.2.1. 使用 YAML 添加 AWS S3 命名空间存储桶

先决条件

- 正在运行的 OpenShift Data Foundation
- 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:
  AWS_ACCESS_KEY_ID: <AWS ACCESS KEY ID ENCODED IN BASE64>
  AWS_SECRET_ACCESS_KEY: <AWS SECRET ACCESS KEY ENCODED IN BASE64>
```

   a. 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>**。

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

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
```

a. 将 **<resource-name>** 替换为您要提供给资源的名称。

b. 将 **<namespacestore-secret-name>** 替换为在第 1 步中创建的 secret。

c. 将 **<namespace-secret>** 替换为可找到 secret 的命名空间。

d. 将 **<target-bucket>** 替换为您为 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>
```

  - 将 **<my-bucket-class>** 替换为唯一的命名空间存储桶类名称。
  - 将 **<resource>** 替换为单个 namespace-store 的名称，该存储将定义命名空间存储桶的读取和写入目标。

- 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>
```

- 将 **<my-bucket-class>** 替换为唯一的存储桶类名称。
- 将 **<write-resource>** 替换为定义命名空间存储桶写入目标的单一命名空间存储名称。
- 将 **<read-resources>** 替换为定义命名空间存储桶读取目标的 namespace-store 的名称列表。

4. 应用以下 YAML，以使用对象 Bucket Class (OBC) 资源创建 bucket，该资源使用第 2 步中定义的 bucket 类。

```
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>
```

   a. 将 **<resource-name>** 替换为您要提供给资源的名称。
   b. 将 **<my-bucket>** 替换为您要提供给存储桶的名称。
   c. 将 **<my-bucket-class>** 替换为上一步中创建的存储桶类。

当 Operator 置备 OBC 后，会在 MCG 中创建存储桶，Operator 会创建一个名称相同且位于 OBC 同一命名空间上的 Secret 和 ConfigMap。

5.2.2. 使用 YAML 添加 IBM COS 命名空间存储桶

先决条件

- 正在运行的 OpenShift Data Foundation 平台。
- 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>
```

a. 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>**。

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

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
```

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

b. 将 **<namespacestore-secret-name>** 替换为在第 1 步中创建的 secret。

c. 将 **<namespace-secret>** 替换为可找到 secret 的命名空间。

d. 将 **<target-bucket>** 替换为您为 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>
```

  - 将 **<my-bucket-class>** 替换为唯一的命名空间存储桶类名称。

  - 将 **<resource>** 替换为单个 namespace-store 的名称，该存储将定义命名空间存储桶的读取和写入目标。

- 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>
```

  - 将 **<my-bucket-class>** 替换为唯一的存储桶类名称。
  - 将 **<write-resource>** 替换为定义命名空间存储桶写入目标的单一命名空间存储名称。
  - 将 **<read-resources>** 替换为定义命名空间存储桶读取目标的 namespace-store 的名称列表。

4. 应用以下 YAML，以使用对象 Bucket Class (OBC) 资源创建 bucket，该资源使用第 2 步中定义的 bucket 类。

```
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>
```

   a. 将 **<resource-name>** 替换为您要提供给资源的名称。
   b. 将 **<my-bucket>** 替换为您要提供给存储桶的名称。
   c. 将 **<my-bucket-class>** 替换为上一步中创建的存储桶类。

当 Operator 置备 OBC 后，会在 MCG 中创建存储桶，Operator 会创建一个名称相同且位于 OBC 同一命名空间上的 Secret 和 ConfigMap。

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

先决条件

- 正在运行的 OpenShift Data Foundation 平台。
- 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 基础架构，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> -n openshift-storage
```

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

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**。

   - 运行以下命令，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
```

     - 将 **<resource-name>** 替换为您要为其提供资源的名称。
     - 将 **<my-bucket-class>** 替换为唯一的存储桶类名称。
     - 将 **<resource>** 替换为定义命名空间存储桶读写目标的单个 namespace-store。

- 运行以下命令，创建一个命名空间存储桶类，其命名空间策略为 **multi**：

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

  - 将 **<resource-name>** 替换为您要为其提供资源的名称。
  - 将 **<my-bucket-class>** 替换为唯一的存储桶类名称。
  - 将 **<write-resource>** 替换为定义命名空间存储桶写入目标的单个 namespace-store。
  - 将 **<read-resources>** 替换为一个以逗号分开的命名空间存储列表，该存储将定义命名空间存储桶的读取目标。

3. 运行以下命令，以使用对象 Bucket Class (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 -n openshift-storage --app-namespace my-app --bucketclass <custom-bucket-class>
```

   a. 将 **<bucket-name>** 替换为您选择的存储桶名称。

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

当 Operator 置备 OBC 后，会在 MCG 中创建存储桶，Operator 会创建一个名称相同且位于 OBC 同一命名空间上的 Secret 和 ConfigMap。

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

先决条件

- 正在运行的 OpenShift Data Foundation 平台。
- 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 基础架构，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> -n openshift-storage
```

   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。

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**。

   - 运行以下命令，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
```

     - 将 **<resource-name>** 替换为您要为其提供资源的名称。
     - 将 **<my-bucket-class>** 替换为唯一的存储桶类名称。
     - 将 **<resource>** 替换为定义命名空间存储桶读写目标的单个 namespace-store。

   - 运行以下命令，创建一个命名空间存储桶类，其命名空间策略为 **multi**：

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

     - 将 **<resource-name>** 替换为您要为其提供资源的名称。
     - 将 **<my-bucket-class>** 替换为唯一的存储桶类名称。
     - 将 **<write-resource>** 替换为定义命名空间存储桶写入目标的单个 namespace-store。
     - 将 **<read-resources>** 替换为一个以逗号分开的命名空间存储列表，该存储将定义命名空间存储桶的读取目标。

3. 运行以下命令，以使用对象 Bucket Class (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 -n openshift-storage --app-namespace my-app --
bucketclass <custom-bucket-class>
```

a. 将 **<bucket-name>** 替换为您选择的存储桶名称。

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

当 Operator 置备 OBC 后，会在 MCG 中创建存储桶，Operator 会创建一个名称相同且位于 OBC 同一命名空间上的 Secret 和 ConfigMap。

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

随着 OpenShift Data Foundation 4.8 的发布，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更多信息，请参阅管理命名空间存储桶。

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)。

流程

1. 登录 OpenShift Web 控制台。
2. 点 Storage → OpenShift Data Foundation。
3. 点 **Namespace Store** 选项卡创建要在命名空间存储桶中使用的 **namespacestore** 资源。
   a. 单击 **Create namespace store**。
   b. 输入命名空间存储名称。
   c. 选择一个供应商。
   d. 选择一个地区。
   e. 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。
   f. 选择目标存储桶。
   g. 点击 **Create**。
   h. 验证命名空间存储是否处于 **Ready** 状态。
   i. 重复这些步骤，直到您拥有所需的资源量。
4. 点击 **Bucket Class** 选项卡 → **Create a new Bucket Class**。
   a. 选择 **Namespace** 单选按钮。
   b. 输入 Bucket 类名称。
   c. 添加描述（可选）。

   d. 点 **Next**。

5. 为您的命名空间存储桶选择一个命名空间策略类型，然后单击 **Next**。
6. 选择目标资源。
   - 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。
7. 点 **Next**。
8. 检查您的新 bucket 类，然后点击 **Create Bucketclass**。
9. 在 **BucketClass** 页面中，验证新创建的资源是否处于 **Created** 阶段。
10. 在 OpenShift Web 控制台中，点 **Storage** → **OpenShift Data Foundation**。
11. 在 **Status** 卡中，单击 **Storage System**，再单击弹出窗口中的 storage 系统链接。
12. 在 **Object** 选项卡中，点 **Multicloud Object Gateway** → **Buckets** → **Namespace Buckets** 选项卡。
13. 点 **Create Namespace Bucket**。
    a. 在 **Choose Name** 选项卡中，指定命名空间存储桶的**名称**，再单击 **Next**。
    b. 在 **Set Placement** 标签页中：
       i. 在 **Read Policy** 下，选中在第 5 步中创建的每个命名空间资源应从中读取数据的复选框。
       ii. 如果您使用的命名空间策略类型是 **Multi**，在 **Write Policy** 下指定要将数据写入的命名空间资源。
       iii. 点 **Next**。
    c. 点击 **Create**。

验证

- 验证命名空间存储桶是否在 **State** 列中带有绿色勾号、预期的读取资源和预期的写入资源名称。

第 6 章 混合和多云存储桶的镜像数据

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。

先决条件

- 您必须首先添加 MCG 使用的后备存储，请参阅 第 4 章 *为混合或多云添加存储资源*。

然后，您创建一个 bucket 类来反映数据管理策略镜像。

流程

您可以通过三种方式设置镜像数据：

- 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”
- 第 6.3 节 “使用用户界面将存储桶配置为镜像数据”

6.1. 使用 MCG 命令行创建存储桶类来镜像数据

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
```

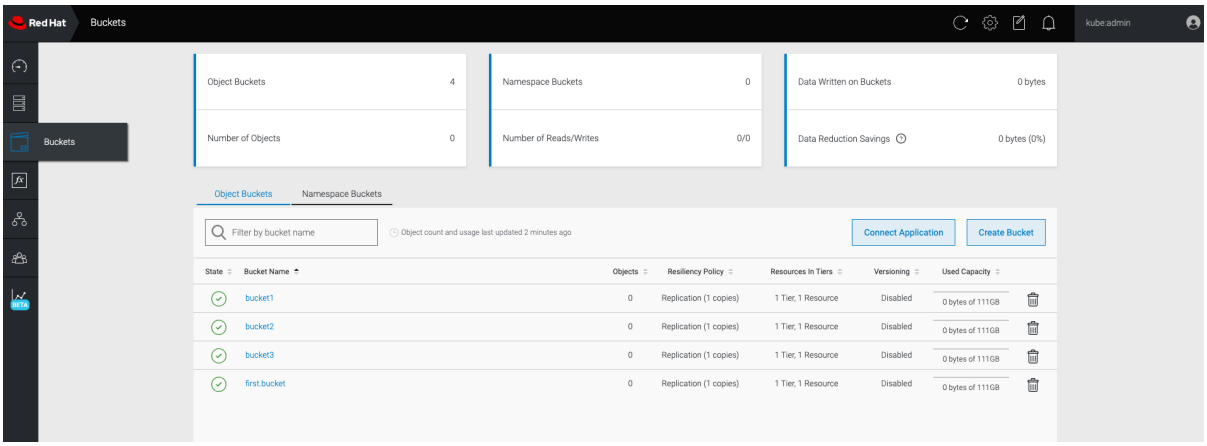
2. 将以下行添加到标准 Object Bucket Claim (OBC) 中：

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

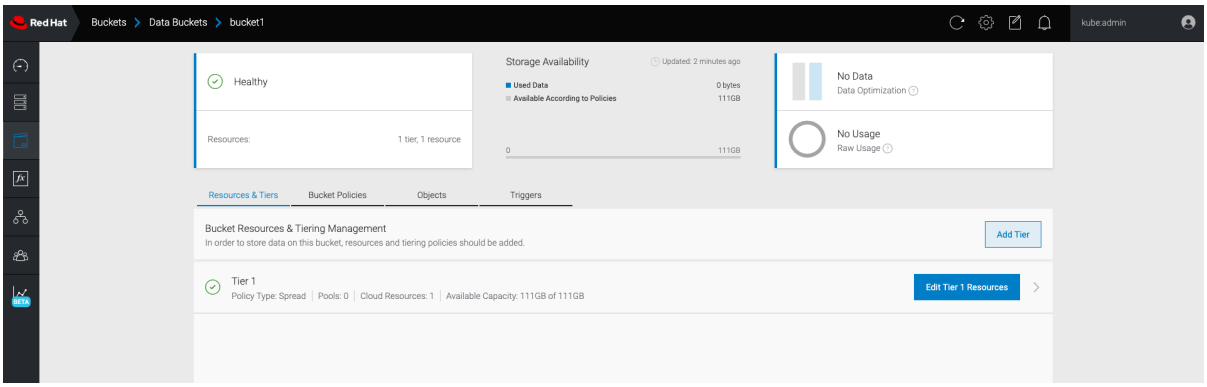
有关 OBCs 的更多信息，请参阅 第 9 章 *对象 Bucket 声明*。

6.3. 使用用户界面将存储桶配置为镜像数据

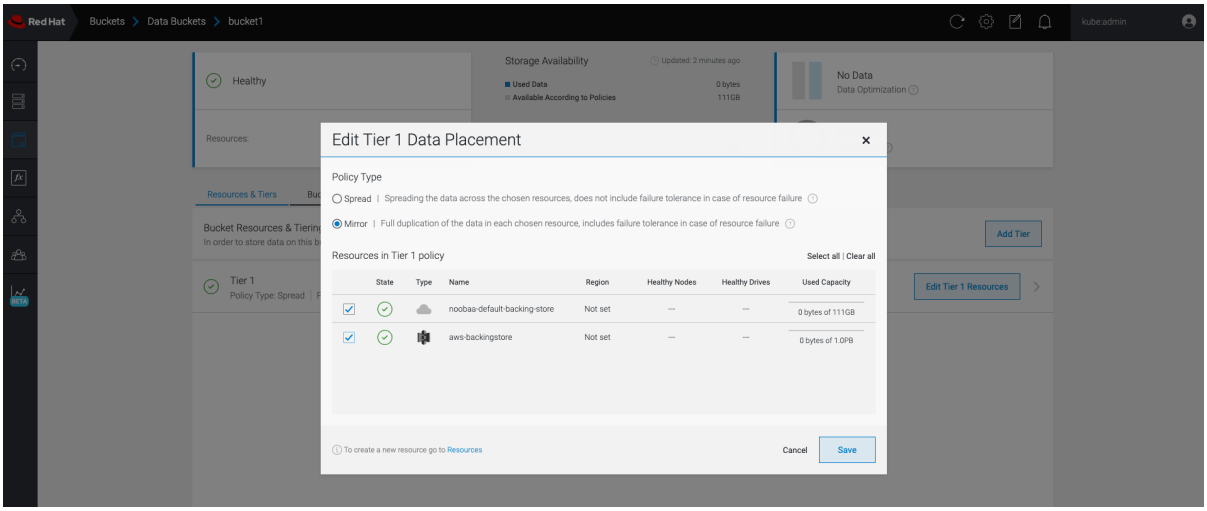
1. 在 OpenShift Web 控制台中，点 **Storage → OpenShift Data Foundation**。
2. 在 **Status** 卡中，单击 **Storage System**，再单击弹出窗口中的 storage 系统链接。
3. 在 **Object** 选项卡中，点 **Multicloud Object Gateway** 链接。
4. 在 **NooBaa** 页面中，单击左侧的存储 **桶** 图标。您可以看到存储桶列表：



5. 点击您要更新的存储桶。
6. 点击 **Edit Tier 1 Resources**：



7. 选择 **Mirror** 并检查您要用于这个存储桶的相关资源。在以下示例中，在 RGW 和 **AWS-backingstore** 中被镜像的 **noobaa-default-backing-store** 间的数据会被镜像：



8. 点击 **Save**。

注意

在 NooBaa UI 中创建的资源不能被 OpenShift UI 或多云对象网关(MCG)CLI 使用。

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. 关于存储桶策略

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. 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。请参见以下示例：

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

   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。

   有关创建 S3 用户的说明，请参阅 第 7.3 节 “在 Multicloud 对象网关中创建 AWS S3 用户”。

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。

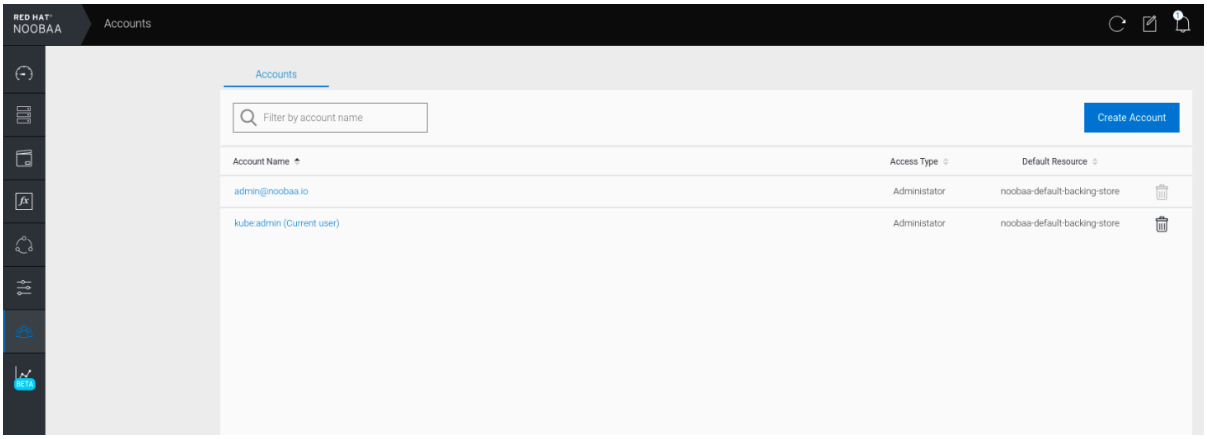
7.3. 在 MULTICLOUD 对象网关中创建 AWS S3 用户

先决条件

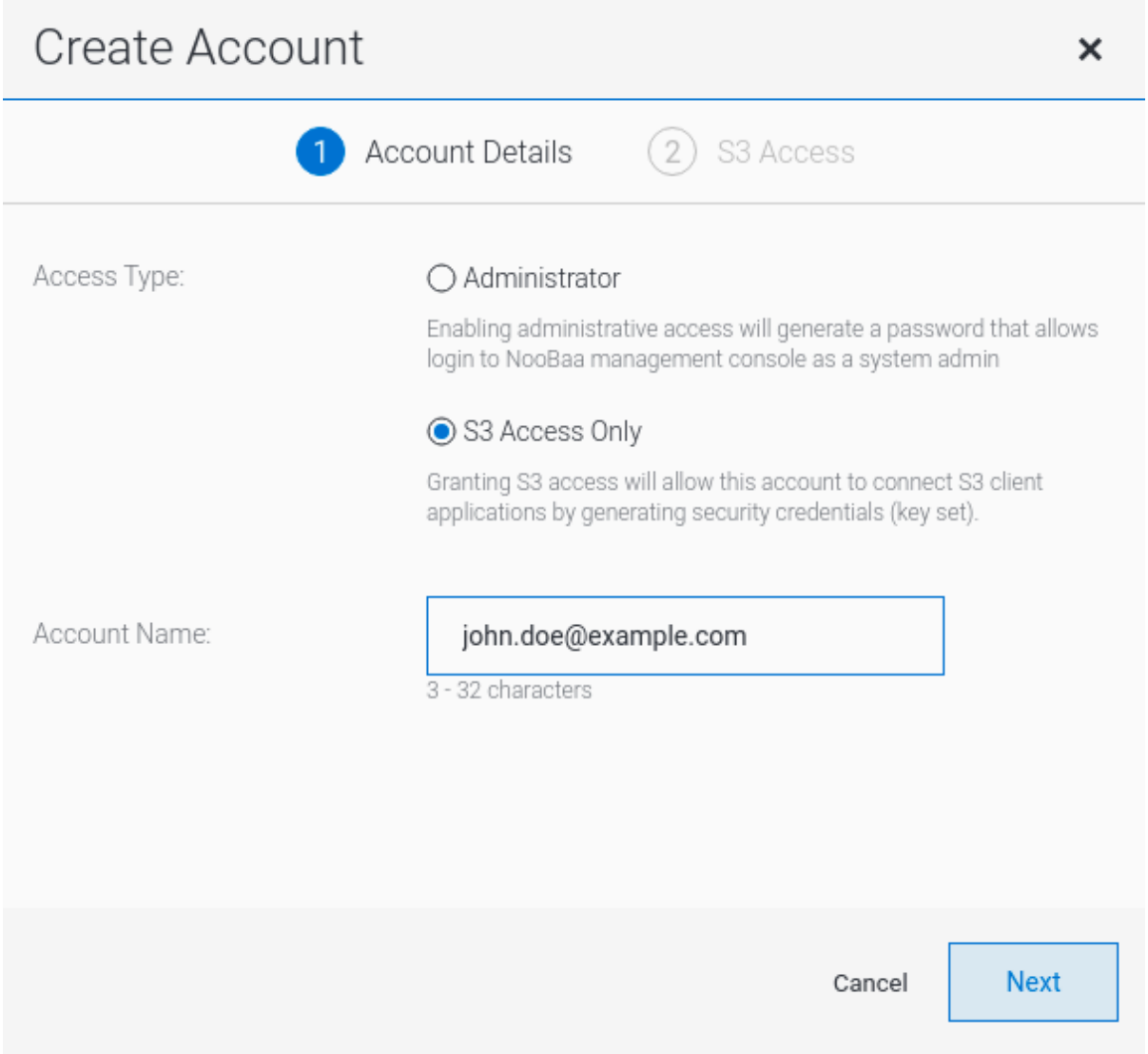
- 正在运行的 OpenShift Data Foundation 平台。
- 访问 Multicloud 对象网关(MCG)，请参阅 第 2 章 *使用应用程序访问多云对象网关*

流程

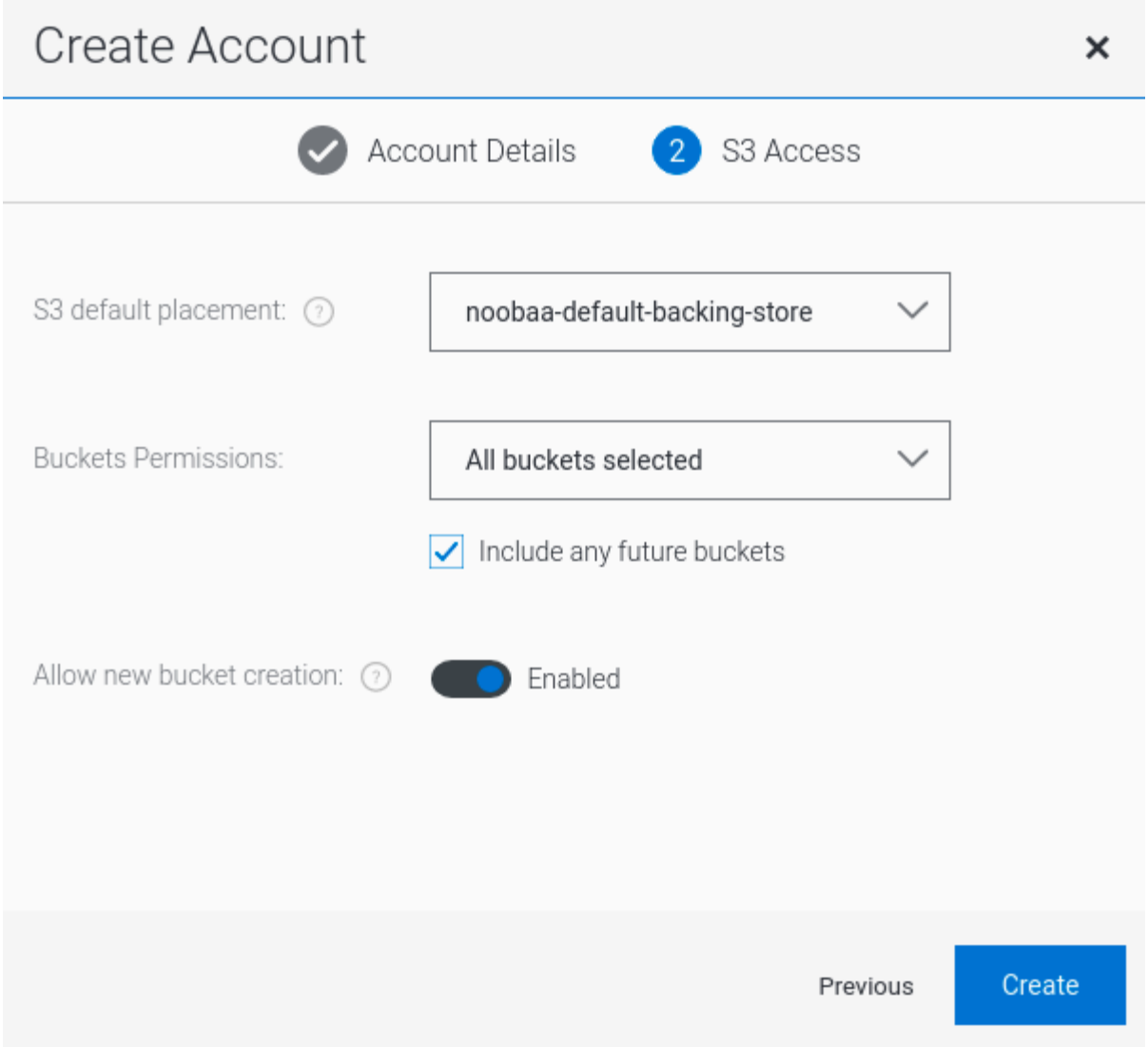
1. 在 OpenShift Web 控制台中，点 **Storage → OpenShift Data Foundation**。
2. 在 **Status** 卡中，单击 **Storage System**，再单击弹出窗口中的 storage 系统链接。
3. 在 **Object** 选项卡中，点 **Multicloud Object Gateway** 链接。
4. 在 **Accounts** 选项卡下，单击 **Create Account**。



5. 选择 **S3 Access Only**，提供 **帐户名称**，例如 john.doe@example.com。点 **Next**。



6. 选择 **S3 默认放置**，例如 **noobaa-default-backing-store**。选择 **Buckets Permissions**。可以选择特定的存储桶或所有存储桶。点击 **Create**。


Create Account
Account Details
2
S3 Access
S3 default placement:
noobaa-default-backing-store
Buckets Permissions:
All buckets selected
Include any future buckets
Allow new bucket creation:
Enabled
Previous
Create

第 8 章 MULTICLOUD OBJECT GATEWAY 存储桶和存储桶类复制

bucket 间的数据复制提供了更高的弹性和更好的协作选项。这些 bucket 可以是由任何受支持的存储解决方案支持的数据存储桶（S3、Azure 等）或命名空间存储桶（不支持 PV Pool 和 GCP）。

- 有关如何创建后备存储的更多信息，请参阅使用 MCG 命令行界面为混合或多云添加存储资源。
- 有关如何创建命名空间存储的更多信息，请参阅使用 Multicloud Object Gateway CLI 和 YAML 添加命名空间存储桶。

bucket 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。在第二个存储桶中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 Multicloud Object Gateway(MCG)，请参阅使用应用程序访问 Multicloud 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
```

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

需要多云对象网关(MCG)存储桶具有特定复制策略的应用程序可以创建一个对象 Bucket 声明(OBC)，并在 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 8.1. 示例

```
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

需要 Multicloud Object Gateway(MCG)存储桶具有特定复制策略的应用程序可以创建一个对象 Bucket Claim(OBC)，并将 **spec.additionalConfig.replication-policy** 参数添加到 OBC。

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:
  replication-policy: [{ "rule_id": "<rule id>", "destination_bucket": "first.bucket", "filter":
{"prefix": "<object name 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

要求多云对象网关(MCG)存储桶类的应用程序具有特定的复制策略，可以创建**存储桶类**并在 JSON 文件中定义 **replication-policy** 参数。

可以为两种类型的存储桶类复制策略设置存储桶类复制策略：

- Placement
- 命名空间

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。可以使用逗号分隔来传递多个后备储存。

/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 8.2. 示例

```
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

需要 Multicloud Object Gateway(MCG)存储桶类的应用程序具有特定的复制策略，可以使用 **spec.replicationPolicy** 字段创建存储桶类。

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。可以传递多个后备储存。

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

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。

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. 您可以在 YAML 文件中添加更多行来自动使用 OBC。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。

```
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
```

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：

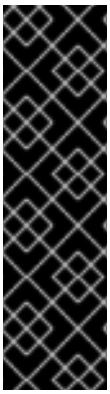
```
# oc get cm <obc-name> -o yaml
```

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 基础架构，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**。

输出示例：

```
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
```

输出示例：

```
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
```

输出示例：

```
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
```

输出示例：

```
Example output:
apiVersion: v1
data:
  AWS_ACCESS_KEY_ID: c0M0R2xVanF3ODR3bHBkVW94cmY=
  AWS_SECRET_ACCESS_KEY:
```

```
Wi9kcFluSWxHRzIWaFlzNk1hc0xma2JXcjM1MVhqa051SlBleXpmOQ==
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
```

输出示例：

```
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)。

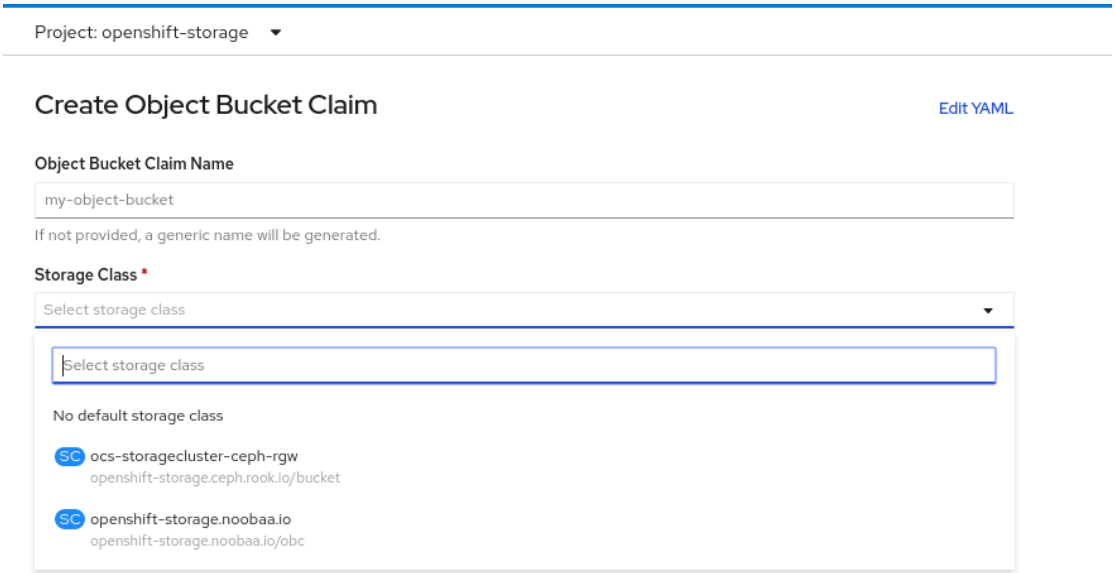
先决条件

- 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 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：

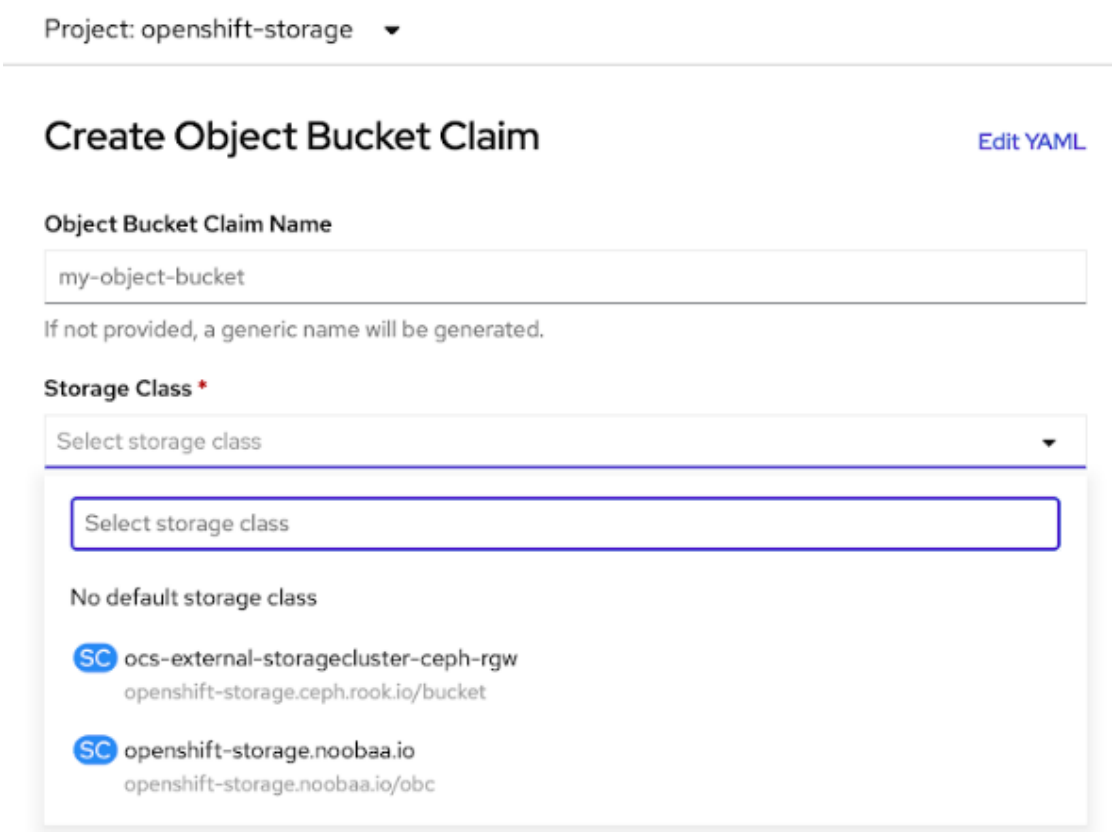
   内部模式



   以下存储类是在部署后创建的，可供使用：

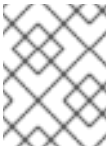
   - **OCS-storagecluster-ceph-rgw** 使用 Ceph 对象网关 (RGW)
   - **openshift-storage.noobaa.io** 使用 Multicloud 对象网关(MCG)

   外部模式



以下存储类是在部署后创建的，可供使用：

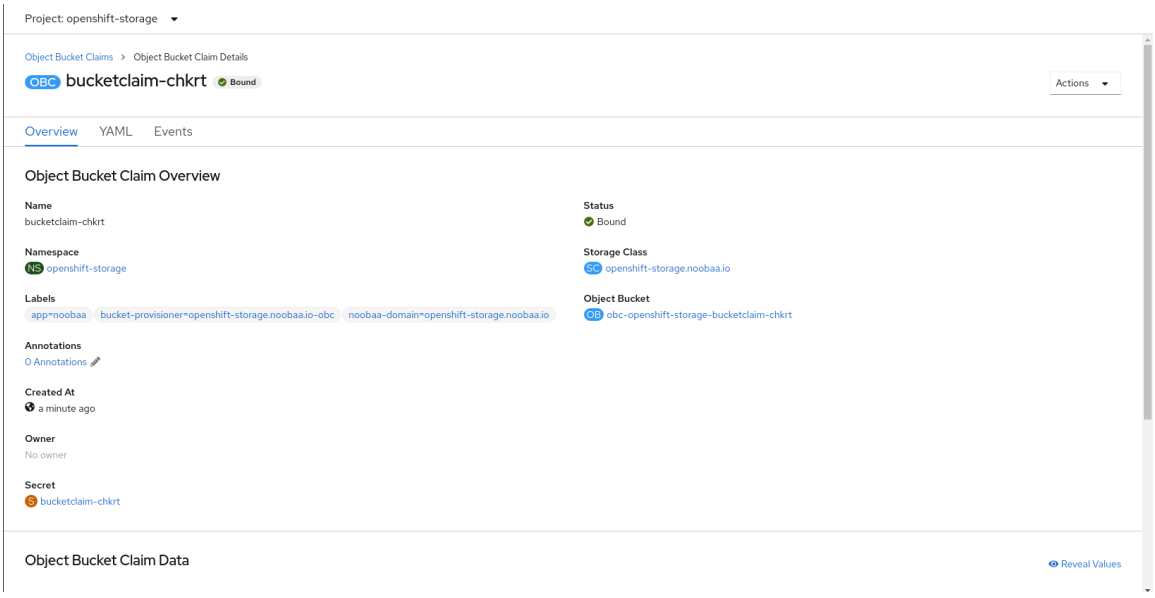
- **ocs-external-storagecluster-ceph-rgw** 使用 RGW
- **openshift-storage.noobaa.io** 使用 MCG



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. 点击 **Create**。
创建 OBC 后，您会被重定向到其详情页面：



其它资源

- *第 9 章 对象 Bucket 声明*

9.4. 将对象 BUCKET 声明附加到部署

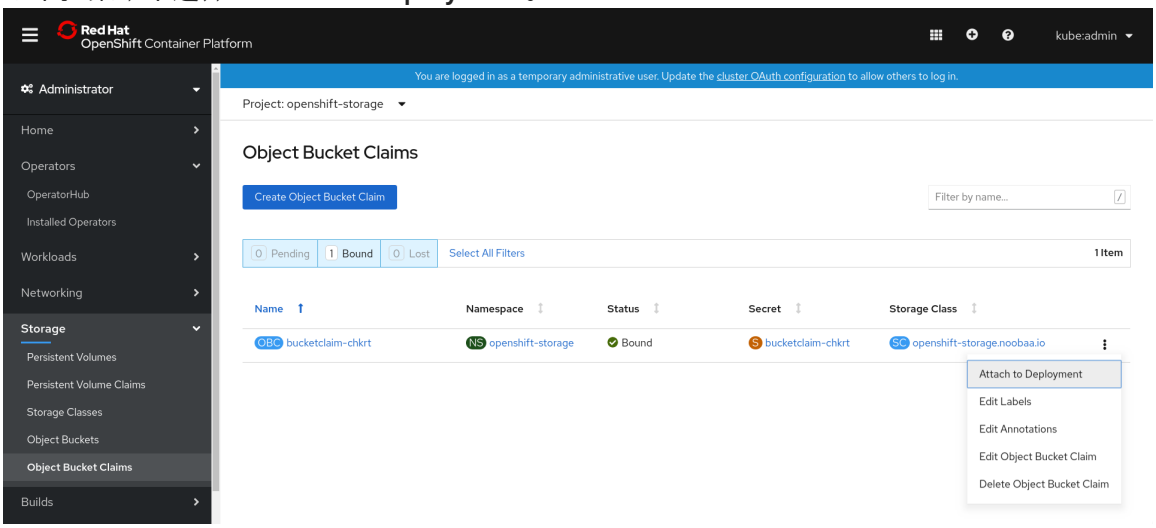
创建后，对象 Bucket 声明 (OBC) 可以附加到特定的部署。

先决条件

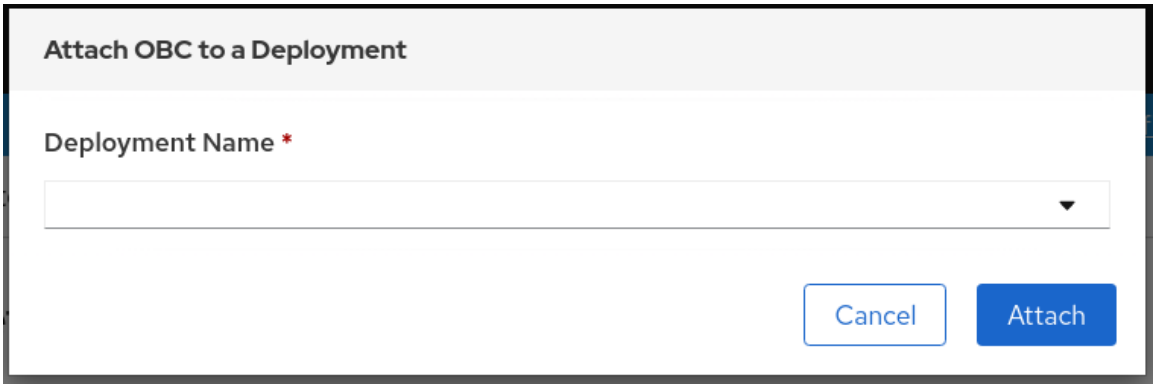
- 对 OpenShift Web 控制台的管理访问权限.

流程

1. 在左侧导航栏中，点击 **Storage → Object 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**。



其它资源

- *第 9 章 对象 Bucket 声明*

9.5. 使用 OPENSHIFT WEB 控制台查看对象存储桶

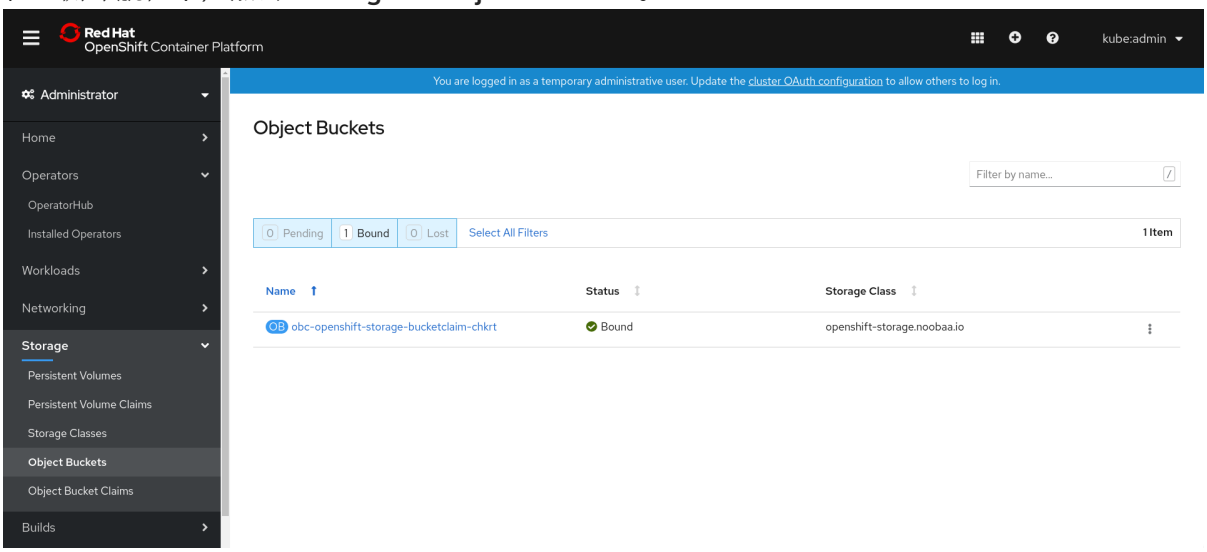
您可以使用 OpenShift Web 控制台查看为对象 Bucket 声明 (OBC) 创建的对象存储桶详情。

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 Buckets**。



   或者，您还可以导航到特定 OBC 的详细信息页面，再单击 **Resource** 链接来查看该 OBC 的对象存储桶。

3. 选择您要查看详细信息的对象 bucket。您可以导航到 **Object Bucket Details** 页面。

其它资源

- *第 9 章 对象 Bucket 声明*

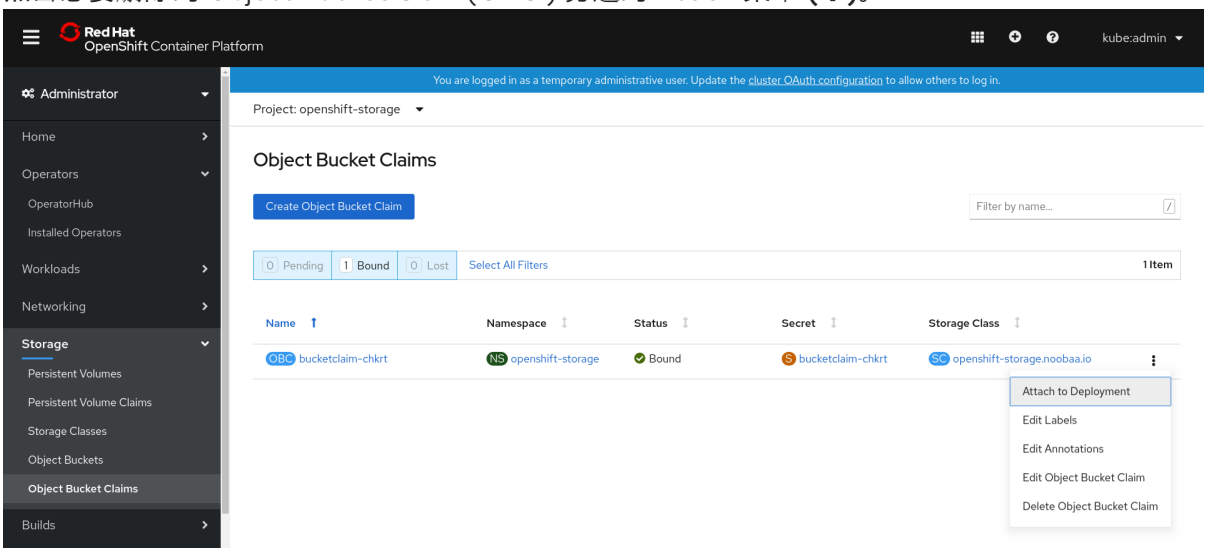
9.6. 删除对象 BUCKET 声明

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 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. 点击 **Delete**。

其它资源

- *第 9 章 对象 Bucket 声明*

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 基础架构，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 基础架构，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 通过添加端点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。

11.1. 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。

11.2. 使用存储节点扩展 MULTICLOUD 对象网关

先决条件

- 在 OpenShift Container Platform 上运行 OpenShift 数据基础集群，可访问多云对象网关(MCG)。

MCG 中的存储节点是一个 NooBaa 守护进程容器，附加到一个或多个持久性卷(PV)，用于本地对象存储。NooBaa 守护进程可以在 Kubernetes 节点上部署。这可以通过创建一个由 StatefulSet pod 组成的 Kubernetes 池来完成。

流程

1. 登录 **OpenShift Web 控制台**。
2. 在 MCG 用户界面中，点击 **Overview → Add Storage Resources**。
3. 在窗口中，单击 **Deploy Kubernetes Pool**。
4. 在 **Create Pool** 步骤中，为将来安装的节点创建目标池。
5. 在 **Configure** 步骤中，配置请求的 pod 数以及每个 PV 的大小。对于每个新 pod，将创建一个 PV。
6. 在 **Review** 步骤中，您可以找到新池的详细信息，再选择要使用的部署方法：本地或外部部署。如果选择了本地部署，Kubernetes 节点将在集群内部署。如果选择了外部部署，您将获得 YAML 文件，供外部运行。
7. 所有节点都会分配给您在第一步中选择的池，并可在 **Resources → Storage resources→ Resource name** 下找到。

第 12 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。

先决条件

- 正在运行的 OpenShift Data Foundation

流程

1. 运行 **oc get service** 命令以获取 RGW 服务名称。

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

NAME                                                  TYPE      CLUSTER-IP     EXTERNAL-IP  PORT(S)
AGE

(...)

rook-ceph-rgw-ocs-storagecluster-cephobjectstore  ClusterIP  172.30.145.254   <none>
80/TCP,443/TCP   5d7h

(...)
```

2. 运行 **oc expose** 命令，以公开 RGW 服务。

```
$ oc expose svc/<RGW service name> --hostname=<route name>
```

   a. 将 **<RGW-service name>** 替换为上一步中的 RGW 服务名称。

   b. 将 **<route name>** 替换为您要为 RGW 服务创建的路由。
   例如：

```
$ oc expose svc/rook-ceph-rgw-ocs-storagecluster-cephobjectstore --hostname=rook-
ceph-rgw-ocs.ocp.host.example.com
```

3. 运行 **oc get route** 命令，以确认 **oc expose** 成功并且存在 RGW 路由。

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

   输出示例：

```
NAME                                                  HOST/PORT                          PATH
rook-ceph-rgw-ocs-storagecluster-cephobjectstore   rook-ceph-rgw-
ocs.ocp.host.example.com

SERVICES                                              PORT        TERMINATION   WILDCARD
rook-ceph-rgw-ocs-storagecluster-cephobjectstore   http        <none>
```

验证

- 要验证 **ENDPOINT**，请运行以下命令：

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

将 **<ENDPOINT>** 替换为在第 3 步中从命令中获得的路由。

例如：

```
$ aws s3 --no-verify-ssl --endpoint http://rook-ceph-rgw-ocs.ocp.host.example.com ls
```

重要

要获取默认用户 **ocs-storagecluster-cephobjectstoreuser** 的访问密钥和 secret，请运行以下命令：

- 访问密钥：

```
$ oc get secret rook-ceph-object-user-ocs-storagecluster-cephobjectstore-ocs-storagecluster-cephobjectstoreuser -n openshift-storage -o yaml | grep -w "AccessKey:" | head -n1 | awk '{print $2}' | base64 --decode
```

- secret 密钥：

```
$ oc get secret rook-ceph-object-user-ocs-storagecluster-cephobjectstore-ocs-storagecluster-cephobjectstoreuser -n openshift-storage -o yaml | grep -w "SecretKey:" | head -n1 | awk '{print $2}' | base64 --decode
```