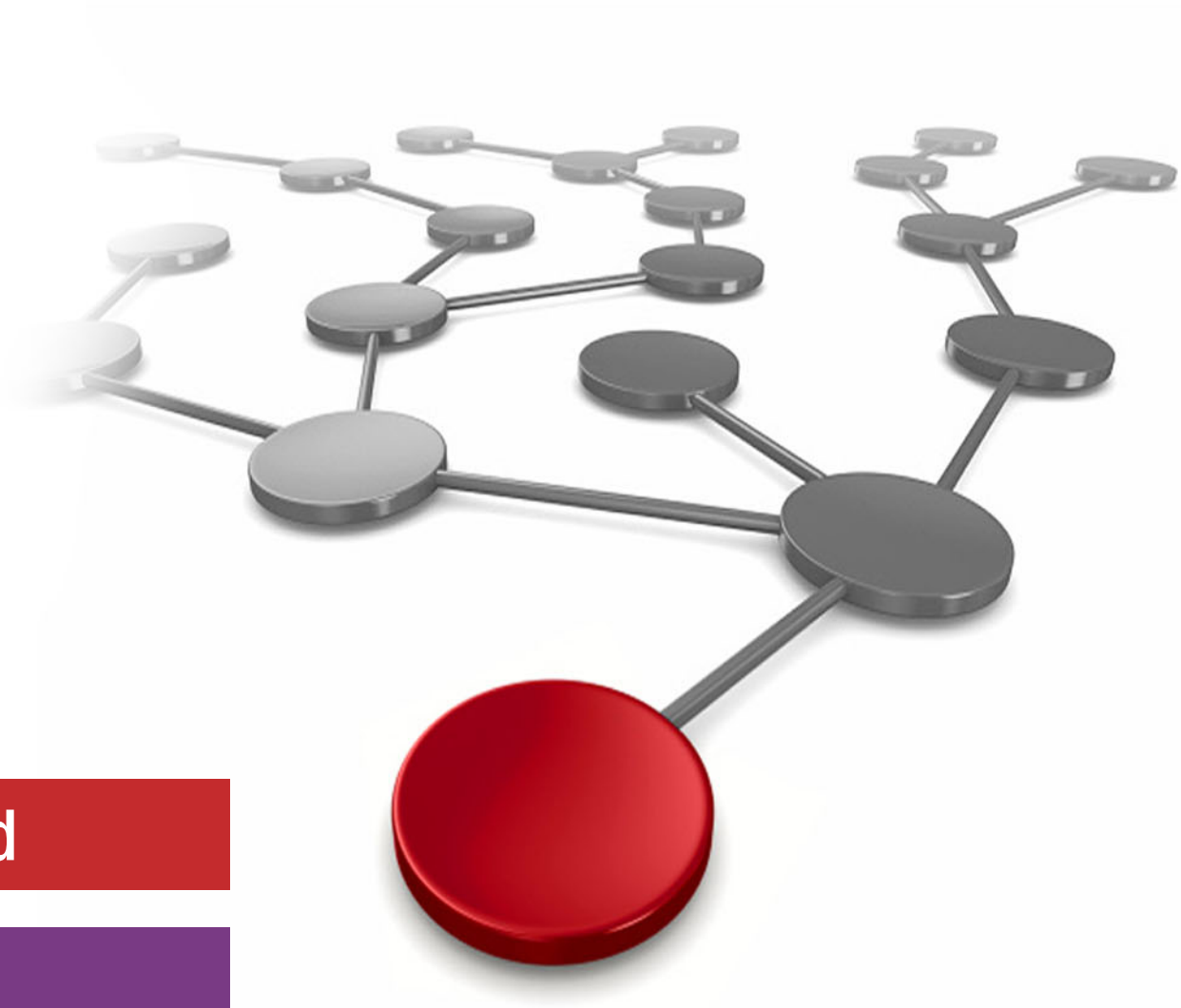
Redbooks
ibm.com/redbooks

IBM Storage Fusion Multicloud Object Gateway

Eyal Abraham

Shawn Houston



Cloud

Storage

IBM

Redpaper

Introduction

This Redpaper provides an overview of IBM® Storage Fusion Multicloud Object Gateway (MCG) and can be used as a quick reference guide for the most common use cases. The intended audience is cloud and application administrators and other technical staff members who want to learn how MCG works, how to set it up, how to use a Backing Store or Namespace Store, and object caching.

MCG, based on the open-source community NooBaa project, is a cloud-native storage platform that is designed to address the evolving needs of modern applications. MCG was developed with a focus on scalability, flexibility, and efficiency. It offers a unified, software-defined, storage solution that seamlessly integrates with public, private, and hybrid cloud environments. MCG provides advanced data management capabilities and intelligent automation to allow organizations to optimize data storage infrastructure and operations.

MCG is a highly customizable data gateway for object storage. It provides data services such as caching, tiering, mirroring, deduplication, encryption, and compression over many storage resources including private and public cloud object storage, file systems, and container PV/PVC. MCG can simplify data flows by linking any of the previously mentioned storage sources into a single, scalable data service that provides a single S3 API and management tool.

MCG is part of IBM Storage Fusion Data Foundation. This document refers to MCG as it is integrated within any Red Hat OpenShift installation.

The section “Multicloud Object Gateway use cases” on page 4 provides detailed descriptions of the following use cases:

- “Backing Store for an IBM Storage Fusion backup and restore location” on page 5
- “Backing Store for object read caching” on page 19
- “Backing Store for object mirroring” on page 23

Audience

This technical Redpaper is targeted for cloud and application administrators, and other technical staff members who want to learn how MCG works, how to set it up, and which use cases it serves.

This document can be used as a quick reference guide for the most common use cases for MCG, including setup and usage of a Backing Store or Namespace Store, mirroring, and

object caching. Only the basic use cases are covered in this document release. For more information see, IBM Storage Fusion Data Foundation.

IBM Multicloud Object Gateway (MCG)

MCG is a highly customizable data gateway for objects. It defines resources by using Custom Resource Definitions (CRDs). The following list describes the CRDs used throughout this paper:

- Backing Store

  The Backing Store CRD represents a storage target to be used as the underlying storage for data in an MCG object bucket. These storage targets are used to store deduplicated, compressed, and encrypted data. Backing Stores store objects in an MCG-specific format and can be accessed by only the MCG instance that created them. When a Bucket Class is defined, the Backing Stores are referred to by name.

- Namespace Store

  The Namespace Store CRD represents a storage target to be used as the underlying storage for data in an MCG object bucket. These storage targets are used to store and read unaltered objects. Unlike Backing Stores, Namespace Stores store objects in their native form without deduplication or compression. Objects that are stored by using a Namespace Store can be accessed by external tools without MCG. When a Namespace Store is defined, the Backing Stores are referred to by name.

- Bucket Class

  Represents a class for object buckets that defines policies for data placement, namespace, replication policy, and more. A Bucket Class is similar in concept to a Storage Class used to define a Persistent Volume Claim (PVC).

When an application uses MCG to access stored objects, it references an Object Bucket Claims associated with a Bucket Class to request the allocation of a new Object Bucket, much the same way an application uses PVCs and Persistent Volumes (PVs). Figure 1 shows the resource hierarchy from storage provider through MCG to the application. Figure 1 also provides hints for the various areas that each resource governs.

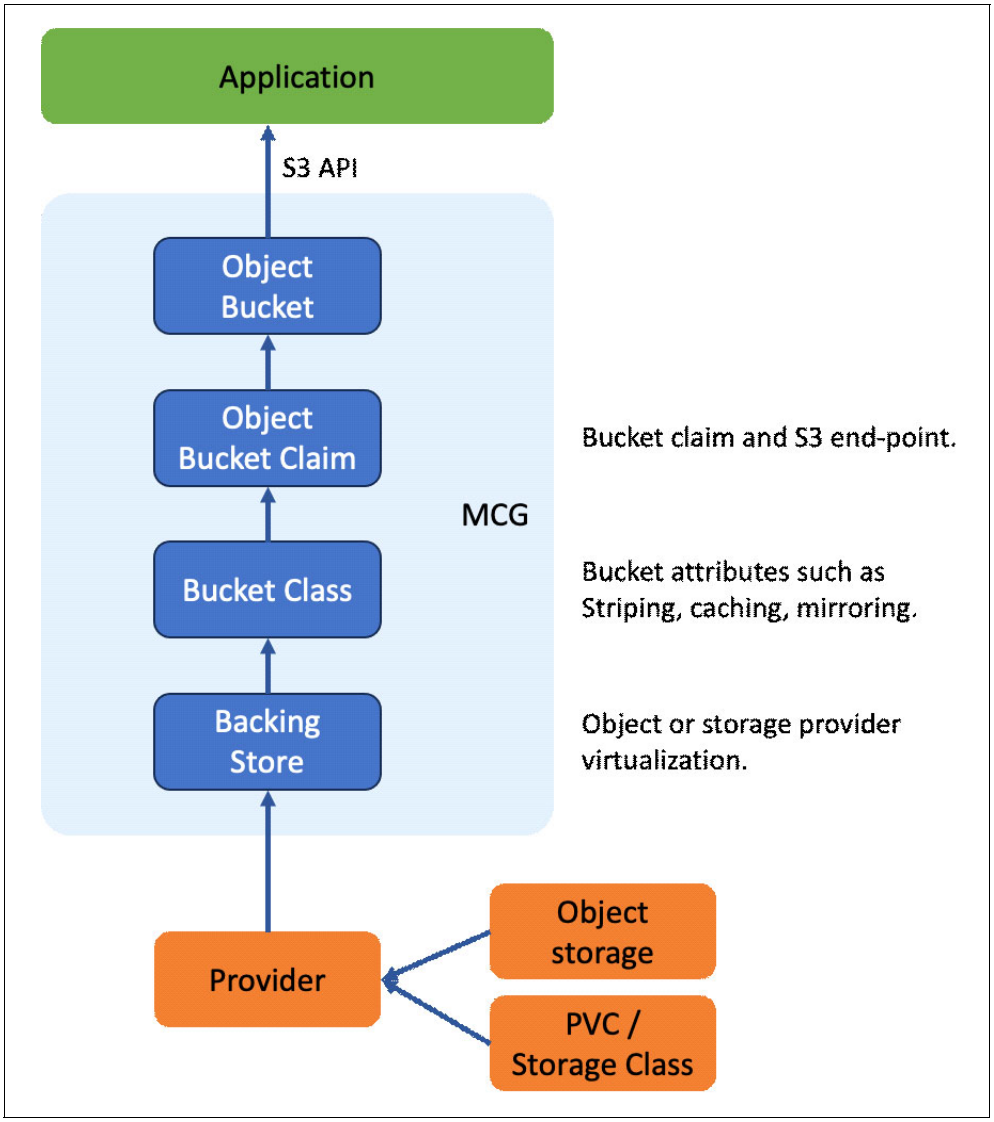


Figure 1 MCG resource hierarchy

MCG as part of Data Foundation with Storage Fusion installation

To better understand the examples that are contained in this technical paper, it is expected that the reader has basic experience and working knowledge of Red Hat OpenShift, IBM Storage Fusion, and IBM Storage Scale. It is suggested that a working setup of Red Hat OpenShift, IBM Storage Fusion, and IBM Storage Scale be available before proceeding with the use cases described in this document.

MCG is part of Data Foundation that is installed by the IBM Storage Fusion operator. MCG can be installed as a stand-alone service. This technical paper discusses only the case where MCG is configured with Data Foundation. A minimum set of services includes installation of IBM Storage Fusion with Data Foundation, Global Data Platform, and Backup services.

The screen capture that is shown in Figure 2 describes the minimum versions for IBM Storage Fusion services.

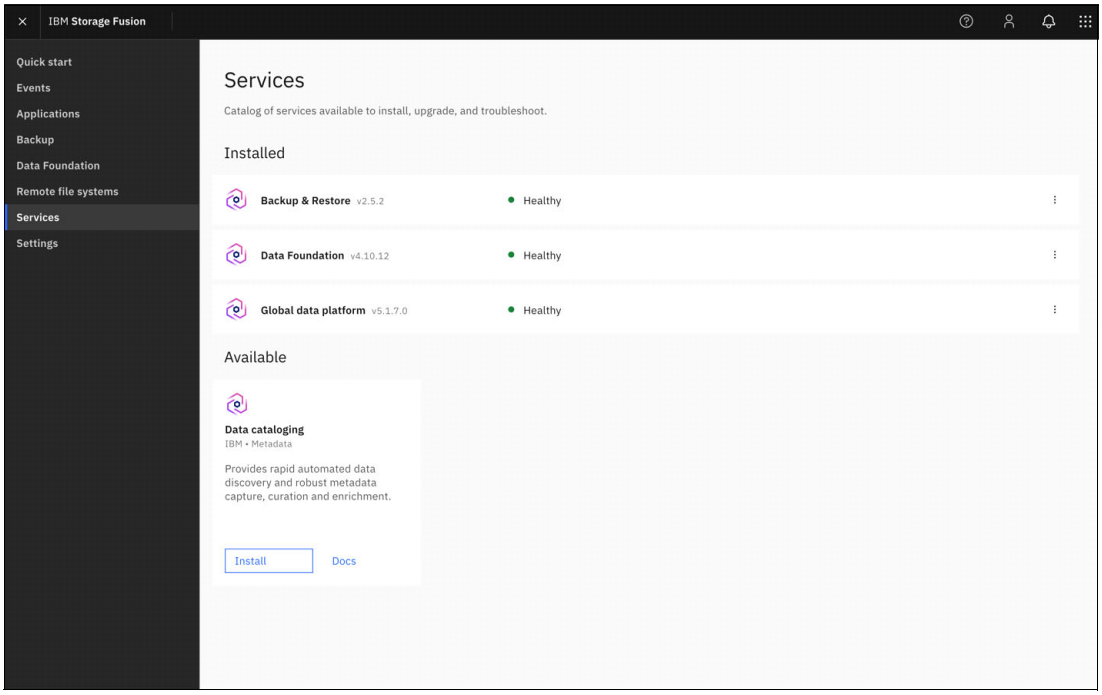


Figure 2 Required IBM Storage Fusion services.

Multicloud Object Gateway use cases

This section describes three use cases that demonstrate MCG capabilities and solutions. All activities are based on operating within a Red Hat OpenShift 4.10 or later and IBM Fusion 2.5.x or later.

The first use case creates an S3 interface to an Object Bucket. The Object Bucket uses a Backing Store that serves as a front end to a Storage Scale file system. The Object Bucket can be used as a general-purpose object storage and as an off-cluster backup location for Storage Fusion backups.

The second use case demonstrates MCG's read-cache feature. MCG is configured to cache objects locally on the Red Hat OpenShift cluster. This setup can be used if an application is expected to perform high-volume object reads from external object storage. By caching the

frequently read objects locally, you can reduce the latency costs of reading the objects from the external object storage location.

The third use case demonstrates MCG's object mirroring feature by using a mirrored Bucket Class. In this configuration, MCG automatically duplicates any object that is written to the mirrored Bucket Class into two object storage locations, one local to the cluster and one external to the cluster. This solution can create multiple copies of an object for enhanced data protection.

Backing Store for an IBM Storage Fusion backup and restore location

The first use case is an object storage gateway to a file system. See Figure 3. The object bucket is backed by an external IBM Storage Scale cluster with MCG providing a gateway service to the Storage Scale cluster. This is useful if your Storage Fusion target location for backup objects is a file system. MCG can provide a front end to the file system with a Backing Store that can then be used to create an Object Bucket Class. The new Bucket Class can then be used to create an S3-compliant object bucket for storing the backup objects. This solution demonstrates a useful setup for storing backup objects that are created by the IBM Storage Fusion Backup and Restore service.

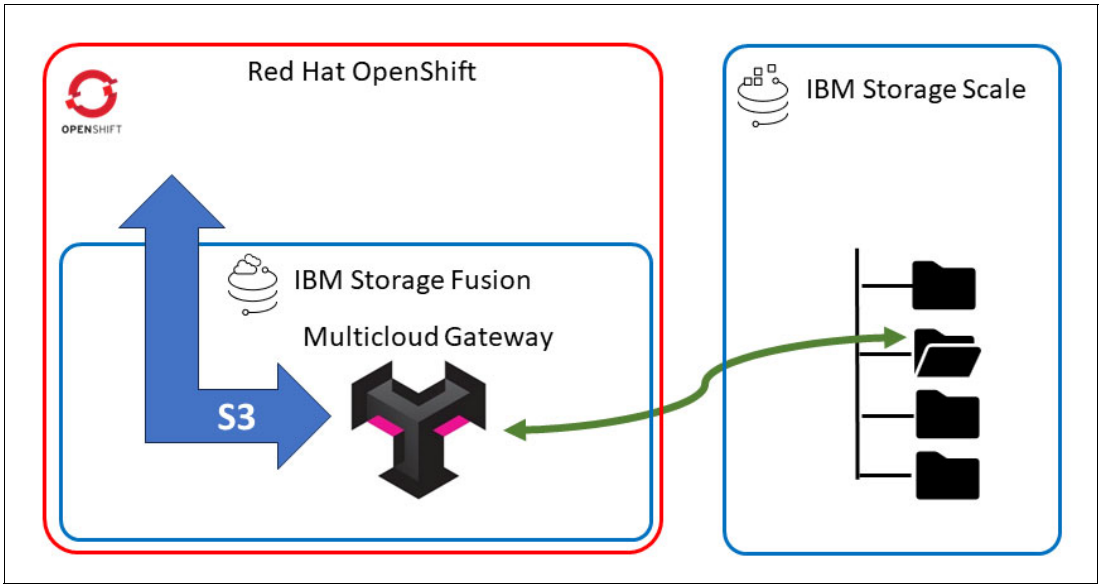


Figure 3 MCG providing an object gateway to IBM Storage Scale

Using a Backing Store stores the object in an MCG-specific object format that applies deduplication, compression, and encryption. It is recommended to back up your MCG configuration so that MCG can be rebuilt, and objects can be restored after cluster and MCG loss.

Fusion Global Data Platform

The preparation step for this use case, which is shown in Figure 4, is to provide access to an external file system. This use case uses an external IBM Storage Scale cluster and mounts it to the Red Hat OpenShift cluster by using the IBM Fusion Global Data Platform (GDP) service. For more information about GDP configuration, see Global Data Platform. Other file system types can be mounted and defined as file storage classes to the Red Hat OpenShift cluster.

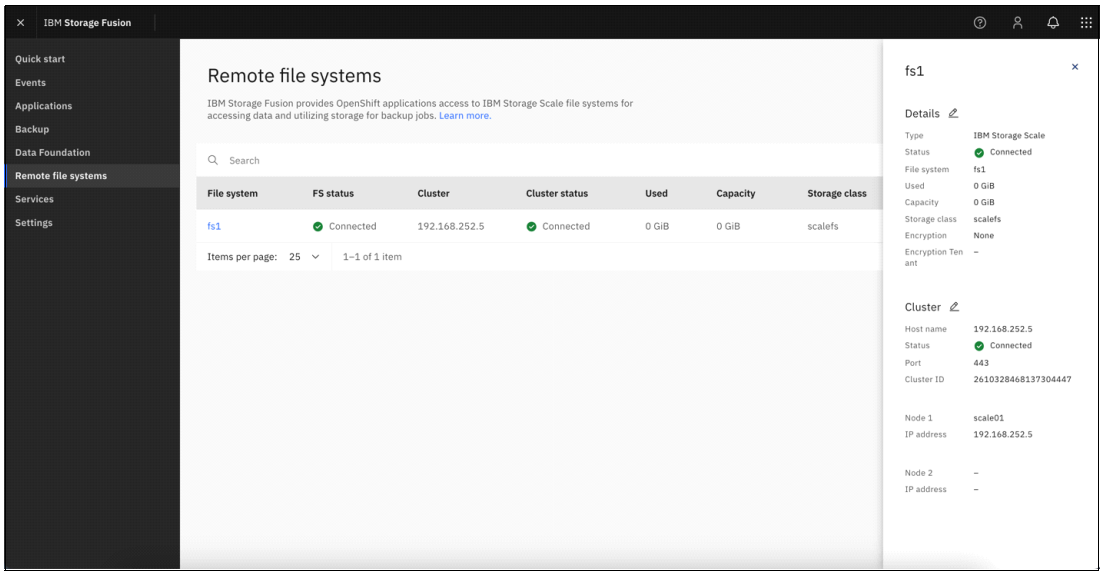


Figure 4 IBM Storage Scale mounted with Fusion Global Data Platform service

Figure 4 shows an external IBM Storage Scale cluster mounted to the Red Hat OpenShift cluster as a remote file system, using Fusion Global Data Platform service. Note the storage class name **`scalefs`** assigned to this file system for later use in MCG.

Defining a Backing Store

The next step is to define and create a Backing Store that uses the `scalefs` file system storage class. In the Red Hat OpenShift GUI, select **Storage** → **Data Foundation** and select the **Backing Store** tab. See Figure 5.

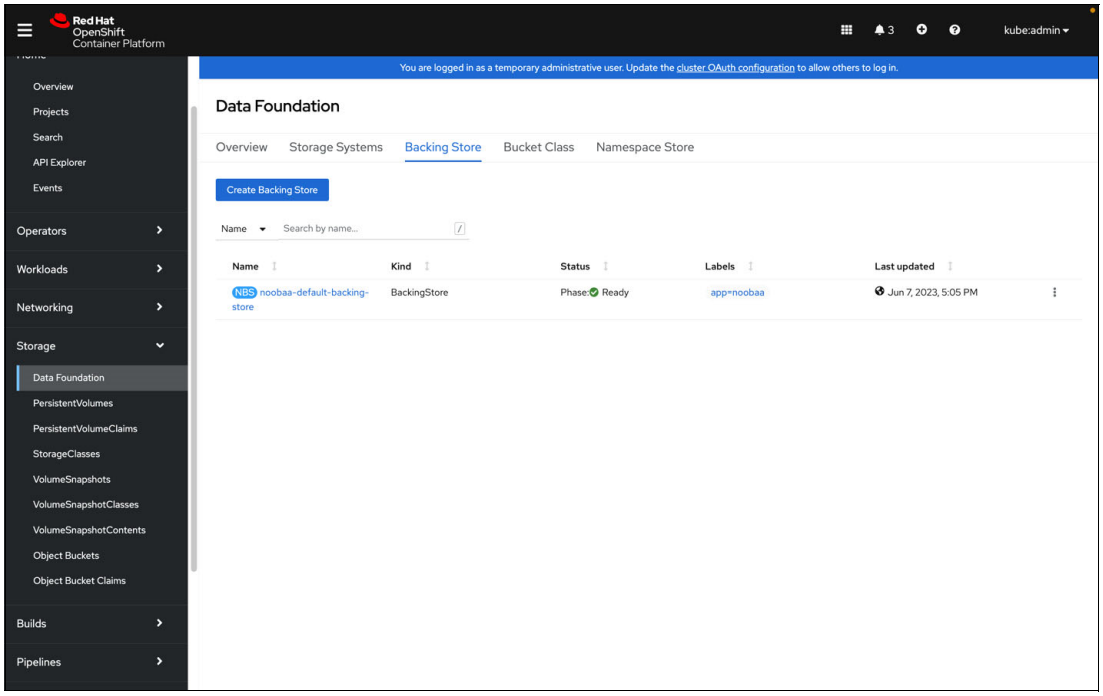


Figure 5 Backing Store tab under Data Foundation

Create a new Backing Store by selecting **Create Backing Store** and completing the entries in the form. Figure 6 on page 8 shows an example of a Backing Store called `backing-store-scalefs`.

This Backing Store is based on a single 50 GB PVC created against the `scalefs` file system that was defined in section “Fusion Global Data Platform”.

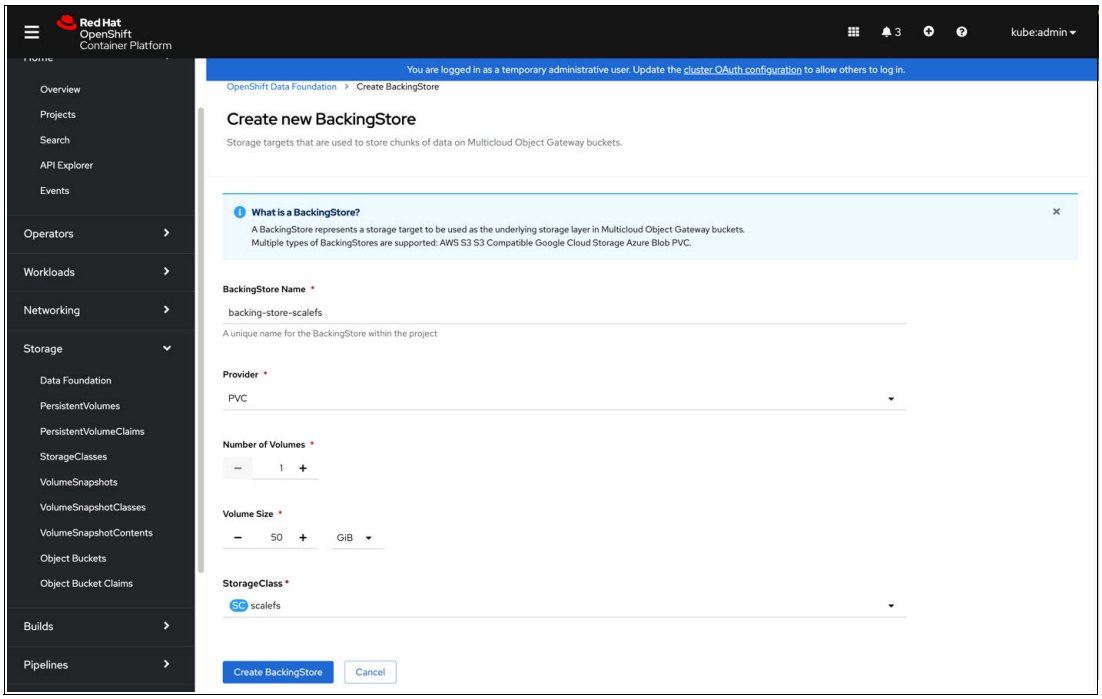


Figure 6 Create a Backing Store

When the data is entered, select **Create Backing Store** and wait for the Backing Store creation status to show a Ready state as depicted in Figure 7.

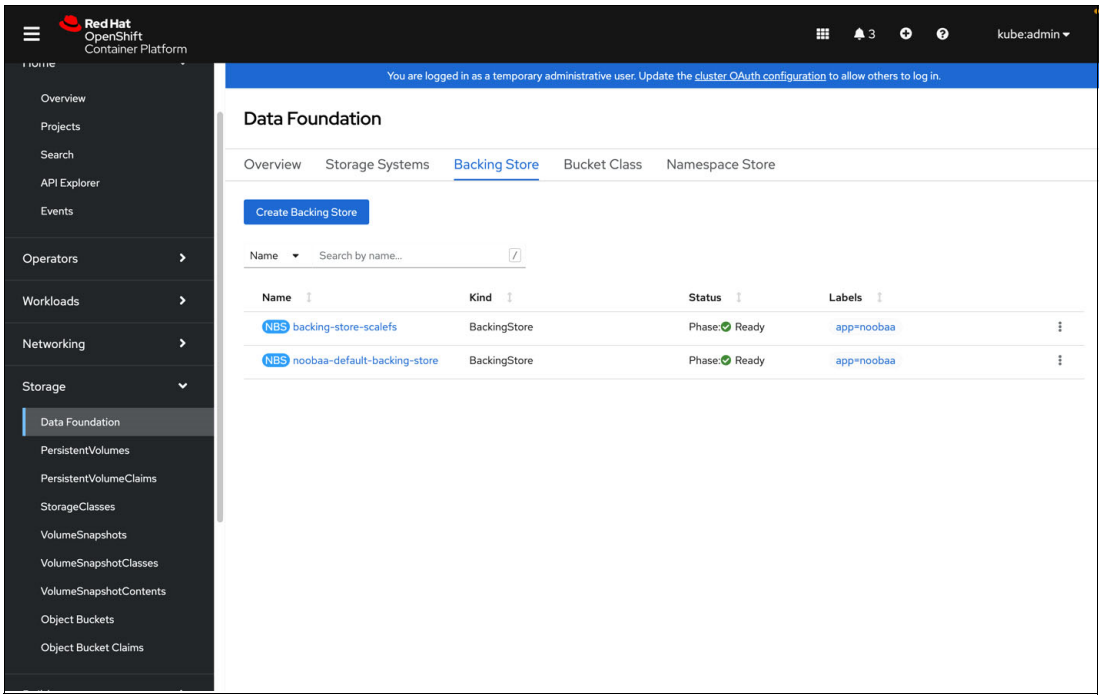


Figure 7 Backing Store created and in Ready state

Defining a Bucket Class

A Bucket Class is similar in principle to a Storage Class. It defines the attributes and behavior of the underlying object storage. A Bucket Class allows the user to define multiple classes of object storage behaviors that can then be selected to fit an application's requirements. The following list provides some object class examples:

- An object class that provides caching
- Object classes that aggregate multiple object stores to be used as one
- Classes that define mirroring of object stores.

Define a Bucket Class by using the Red Hat OpenShift GUI. Select **Storage** → **Data Foundation** and then select the **Bucket Class** tab. See Figure 8.

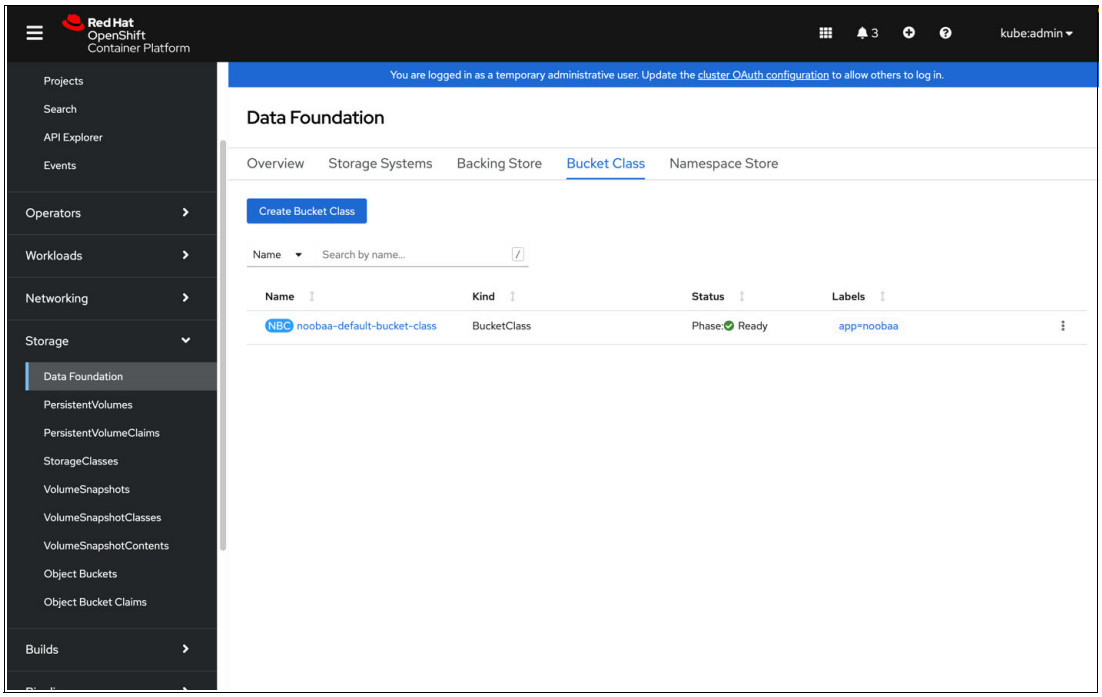


Figure 8 Bucket Class tab under Data Foundation

The Data Foundation operator provides a wizard to define the Bucket Class attributes. Use the wizard to begin the 4-step process and select **Create Bucket Class**. See Figure 8.

In step 1, the first window in the wizard defines the type and name of the Bucket Class. Select **Standard** to create a class that uses the characteristics of the Backing Store to deduplicate, compress, and encrypt the object data. See Figure 9.

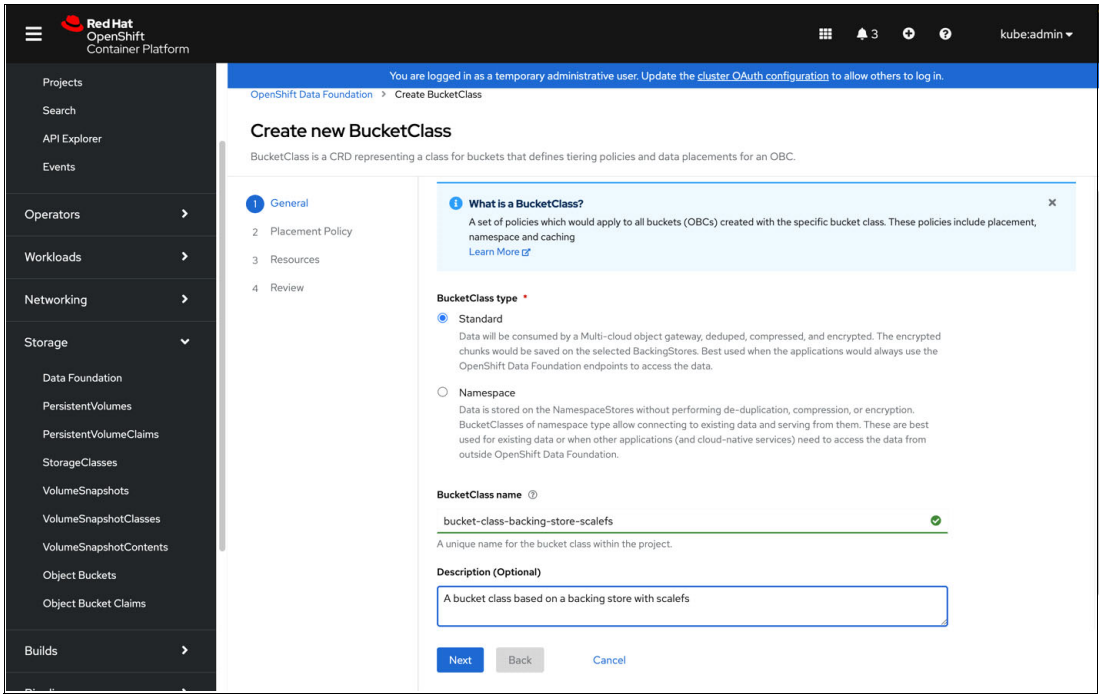


Figure 9 Bucket Class type

In step 2, define the Bucket Class data placement policy. See Figure 10. The policy selection also affects the way MCG protects the stored object data, such that the data is either spread across available backing stores or mirrored between them. Select the **Spread** option for the single Backing Store.

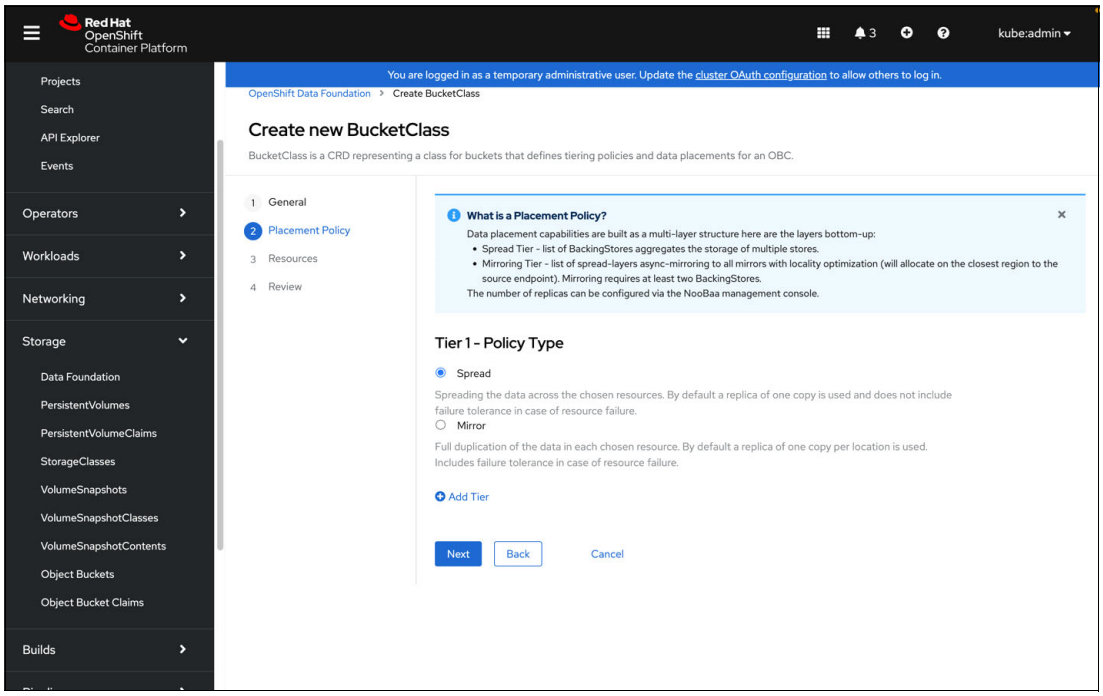


Figure 10 Bucket Class data placement policy

In step 3, the wizard provides the option to select the Backing Stores that support the Bucket Class. See Figure 11. Multiple Backing Stores can be used by MCG according to the data spread policy selected in step 2. For this use case, select the single Backing Store created to provide the front end for the Storage Scale cluster.

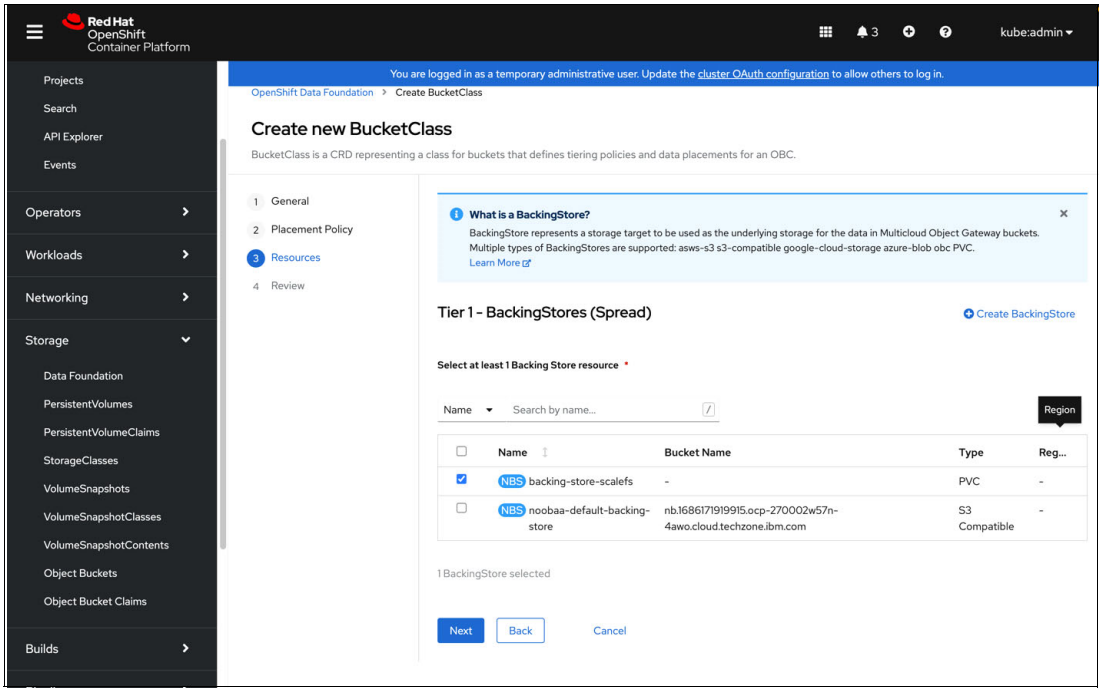


Figure 11 Step 3: Select Backing Stores for the Bucket Class

In step 4 of the wizard, review the selections and approve the creation of the Bucket Class. See Figure 12. Select **Create BucketClass**.

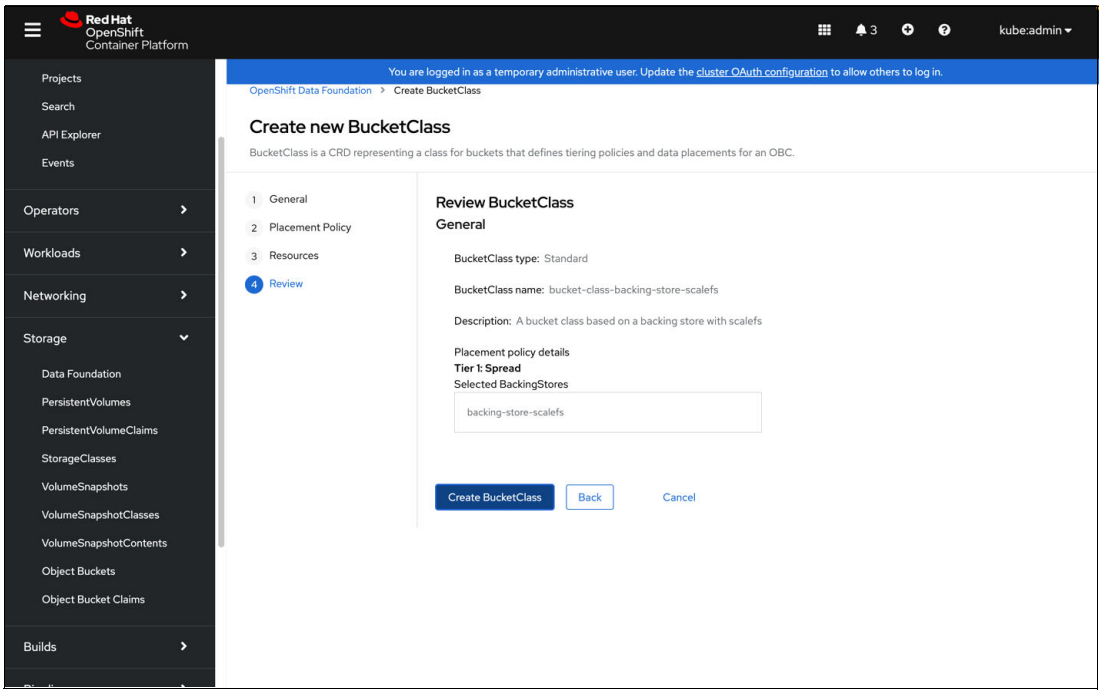


Figure 12 Step 4: review and approve Bucket Class attributes before creating

After the Bucket Class is created, a new Bucket Class is displayed in the Data Foundation **Bucket Class** tab. See Figure 13.

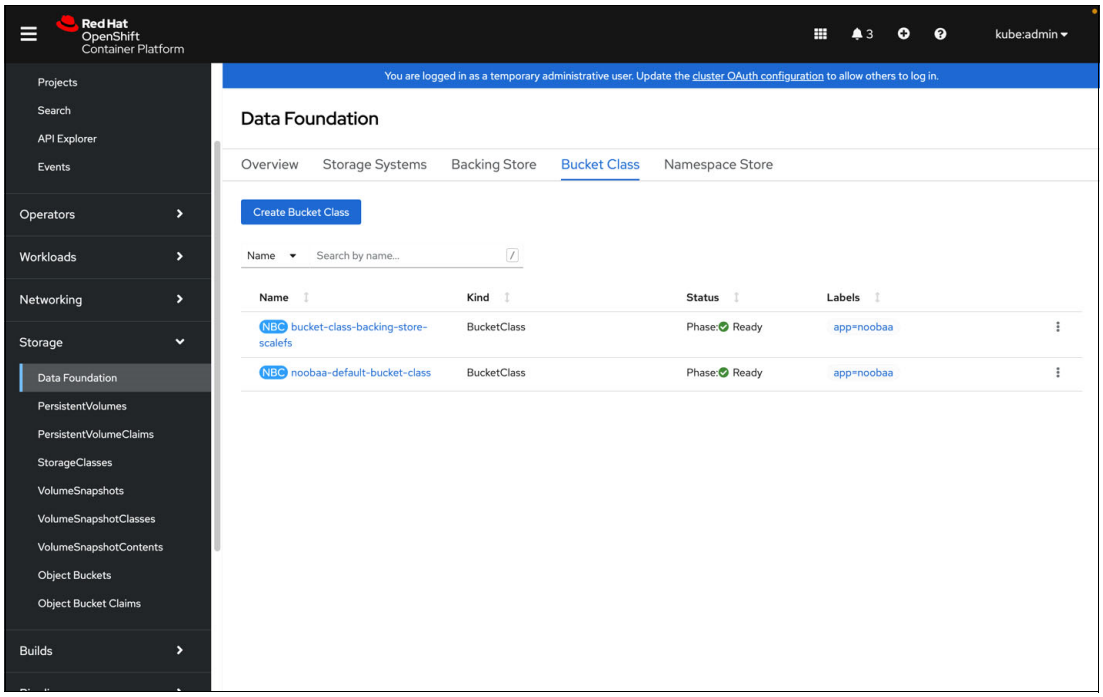


Figure 13 Bucket Class ready for use

Creating an Object Bucket Claim

The final step for creating a backup target for Storage Fusion is creating an Object Bucket Claim. An Object Bucket Claim results in an S3 endpoint with credentials to be used to create a general S3 backup location in the Storage Fusion UI.

In the Red Hat OpenShift GUI, select **Storage** → **Object Bucket Claim**. See Figure 14. Select the **Project: file-browser** pull-down menu to set the selected project namespace to **All projects**.

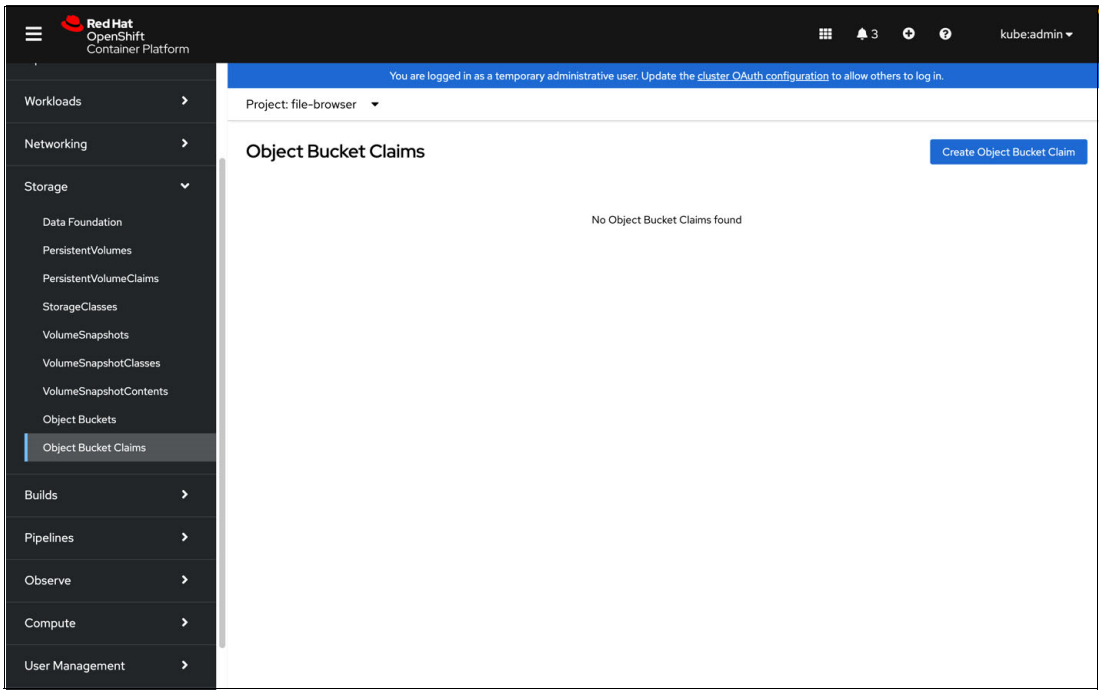


Figure 14 Object Bucket Claims page

Select **Create Object Bucket Claim** and select the parameters for the bucket claim. See Figure 14.

To select the Bucket Class defined in the previous steps, click the **openshift-storage.noobaa.io storage** class and then the **bucket-class-backing-store-scalefs** Bucket Class. See Figure 15.

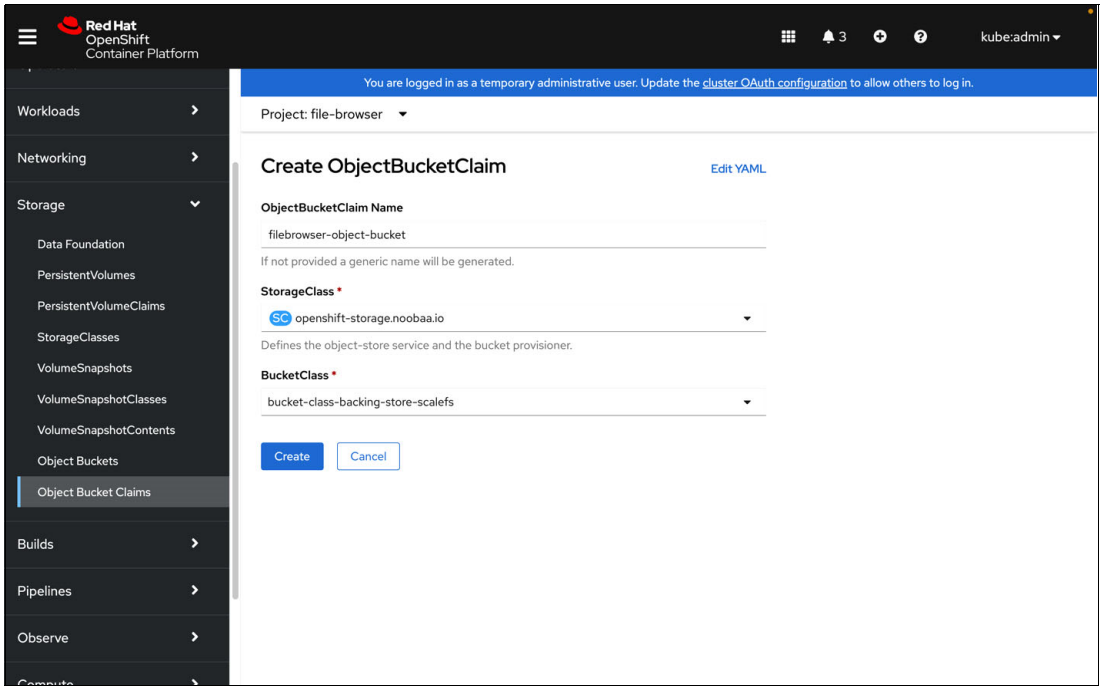


Figure 15 Object Bucket Claim parameters

After selecting **Create,** the **Object Bucket Claim** is displayed. See Figure 16.

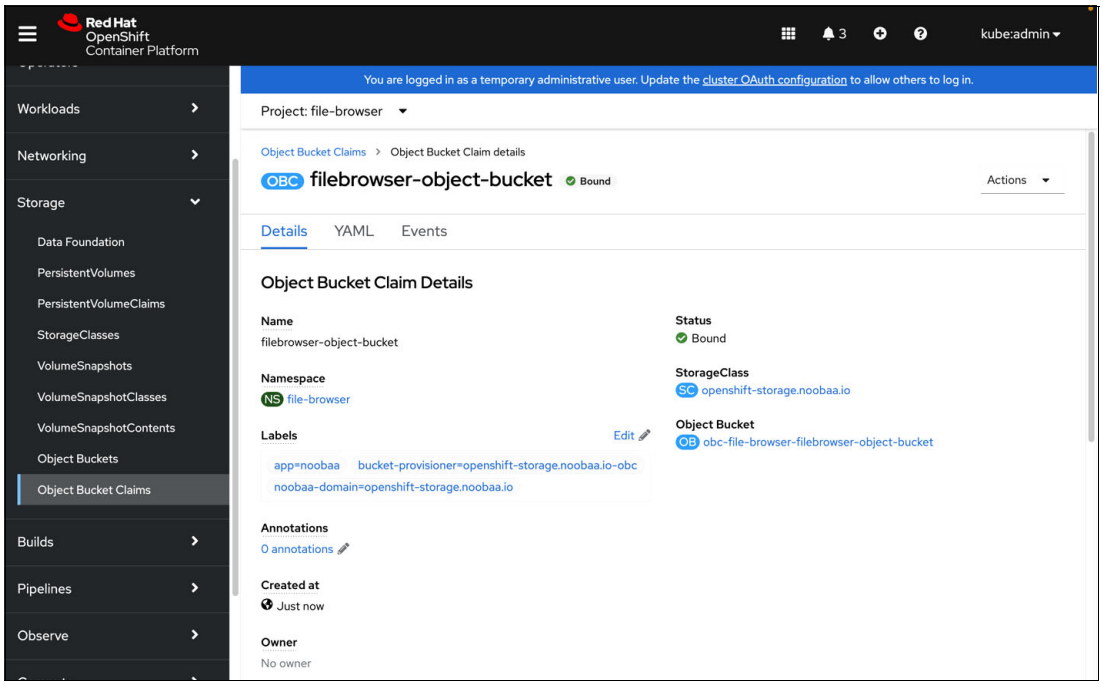


Figure 16 Bound Object Bucket Claim page

Scroll down the **Object Bucket Claim** page to view the parameters needed to access the object bucket. See Figure 17 and Figure 18. The parameters include the S3 endpoint and endpoint credentials to use in Storage Fusion to define the backup location for the sample application.

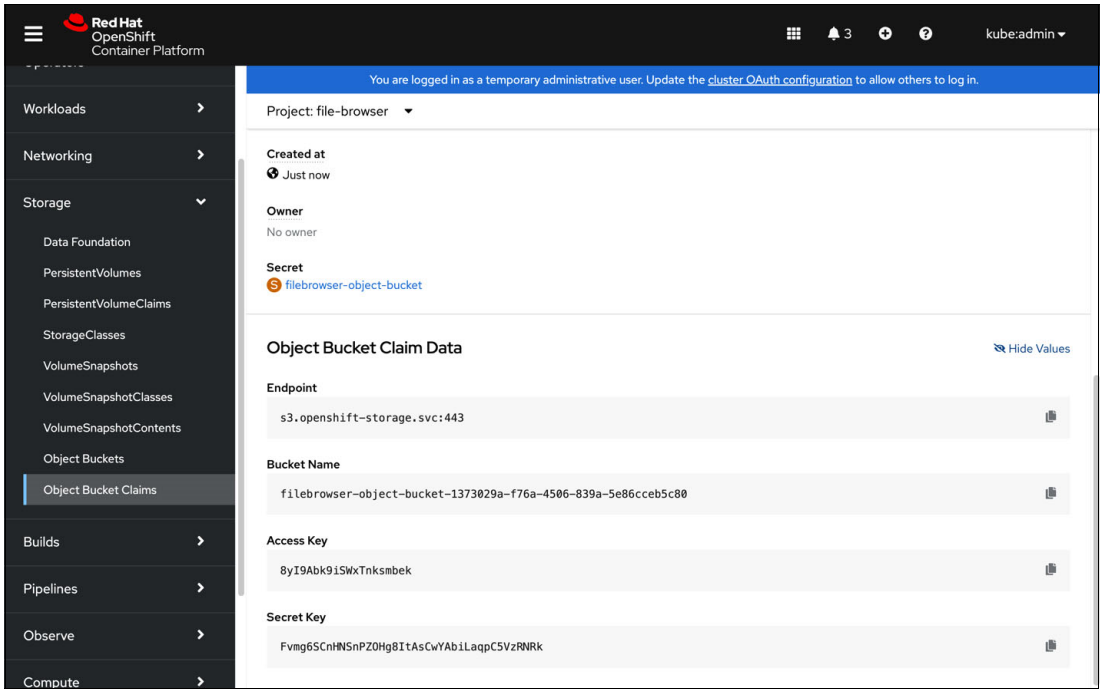


Figure 17 Object Bucket Claim endpoint credentials

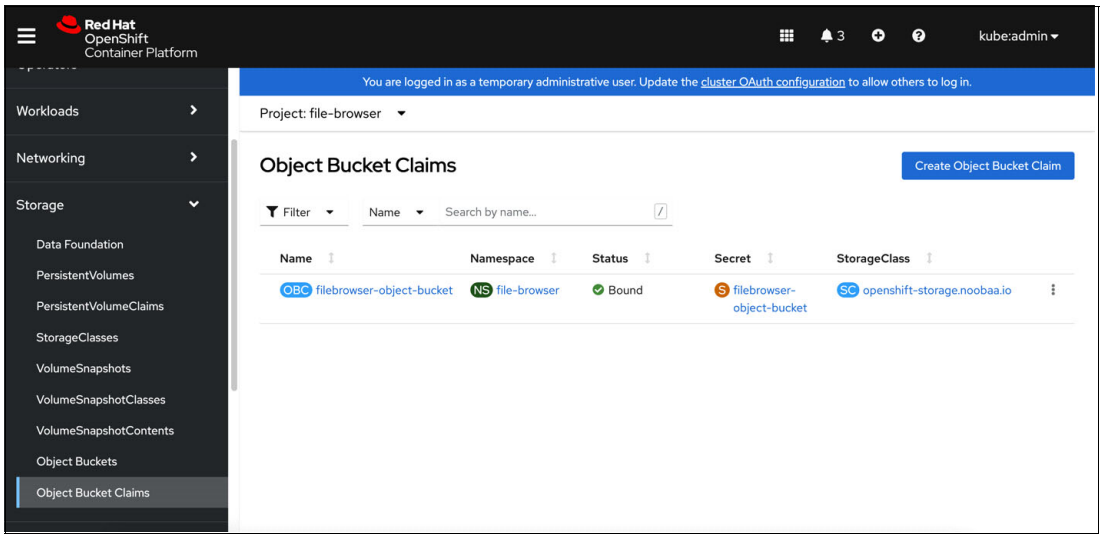


Figure 18 Object Bucket Claim page with newly created claim

Creating a Fusion Backup location with the Object Bucket Claim

After creating the Object Bucket Claim, switch to the Fusion GUI and define the backup location and backup policy. The backups are directed at the Storage Scale cluster through the MCG Backing Store.

The location type is an S3-compliant object store. See Figure 19. The endpoint and credentials are provided by the Object Bucket Claim defined in MCG.

Note: IBM Storage Fusion does not accept a self-signed certificate generated by OpenShift Container Platform, including private CAs. You might need to alter the endpoint URL to specifically include an "http://" prefix and remove the ":443" suffix.

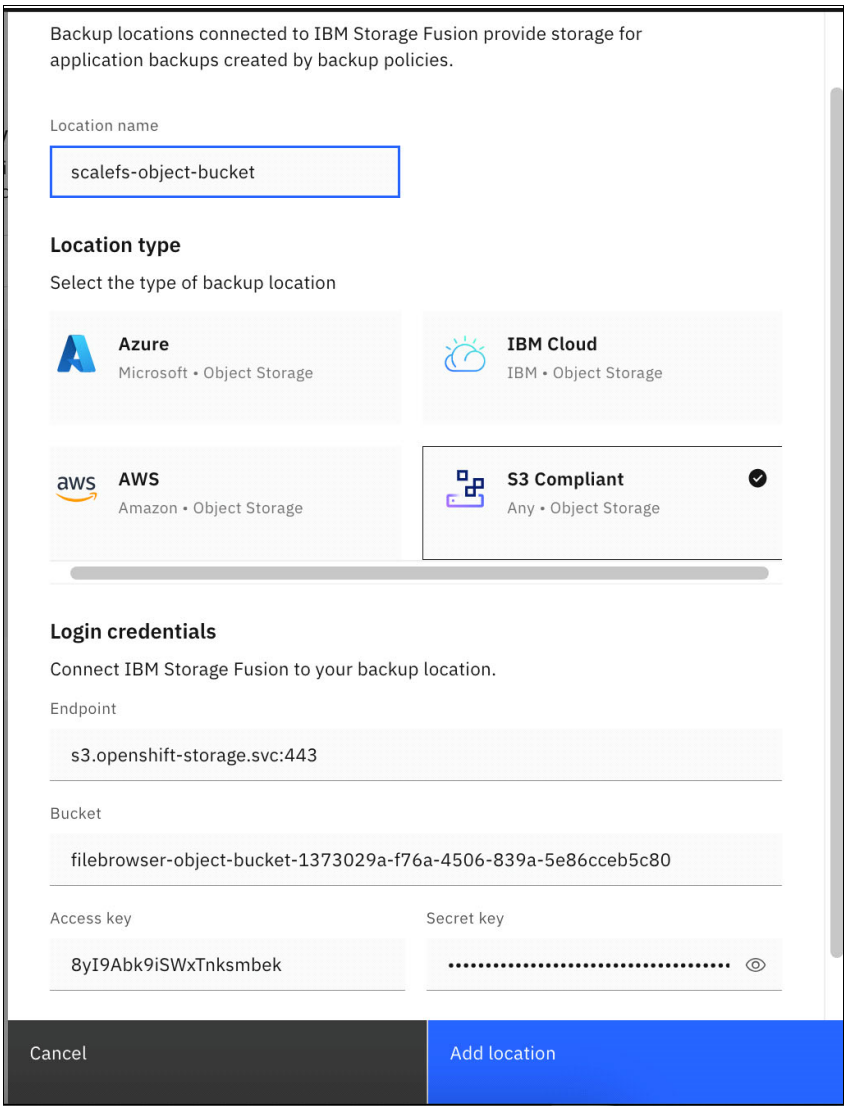


Figure 19 Defining a backup location in the Fusion UI

After confirming the added location, an active backup location will be available as shown in Figure 20.

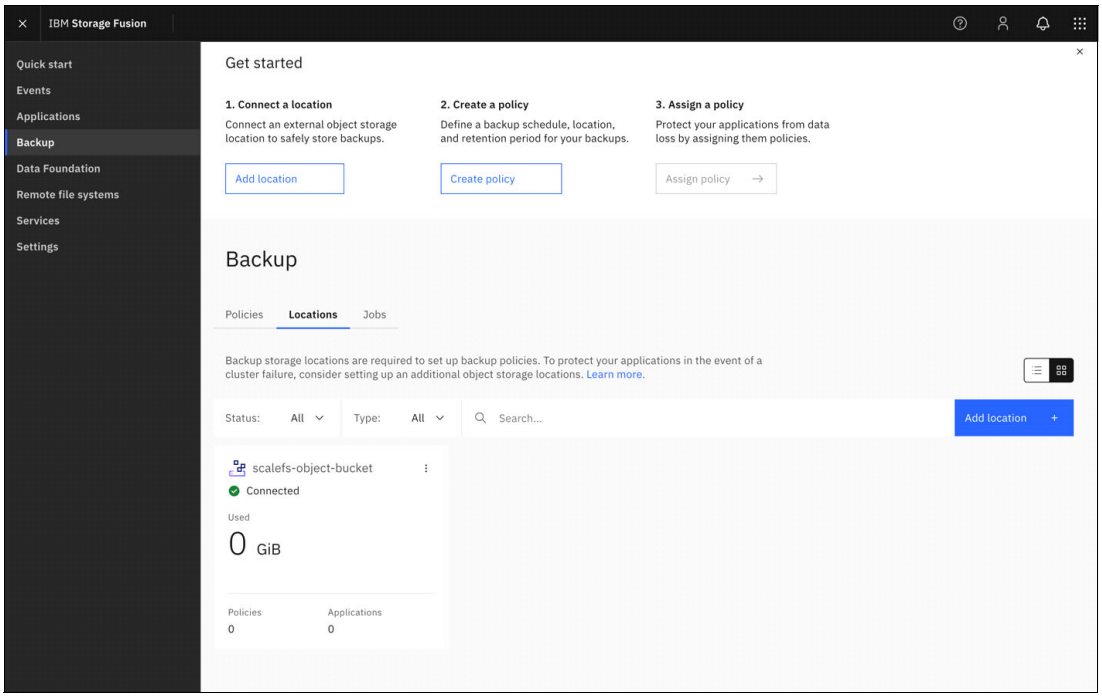


Figure 20 An active and connected backup location

The backup location can now be assigned to any backup policy that performs a backup to an object store. The policy can be assigned to an application and automated backups run per policy. See Figure 21.

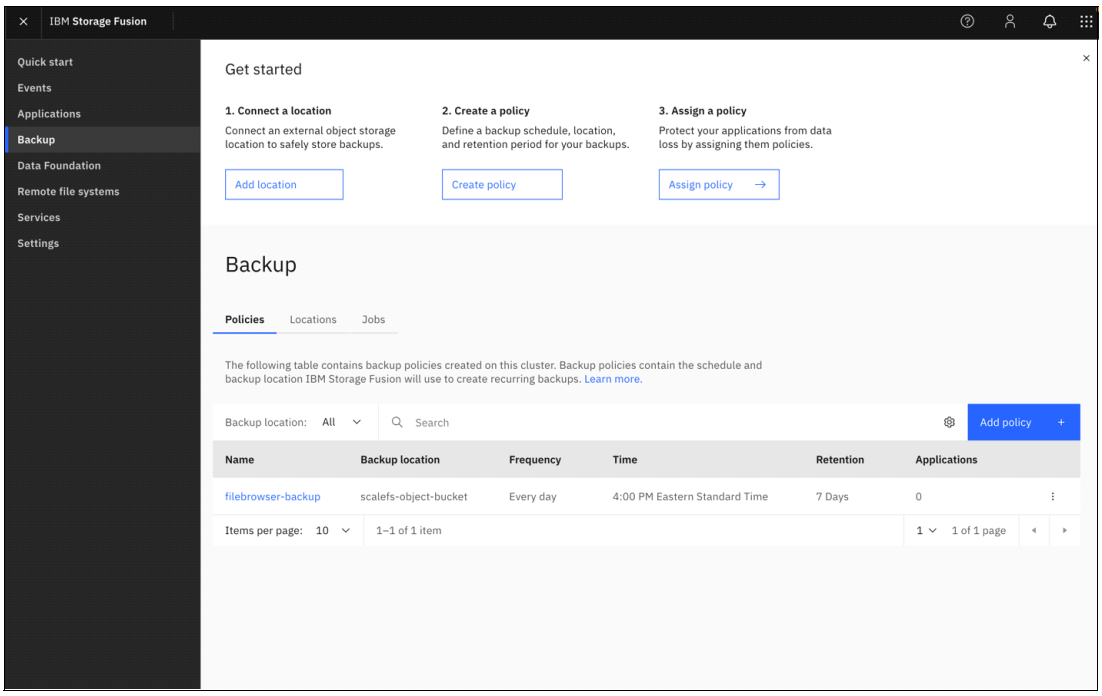


Figure 21 Fusion backup policy that uses the 'scalefs-object-bucket' backup location

Figure 22 shows the summary and details of the completed backup.

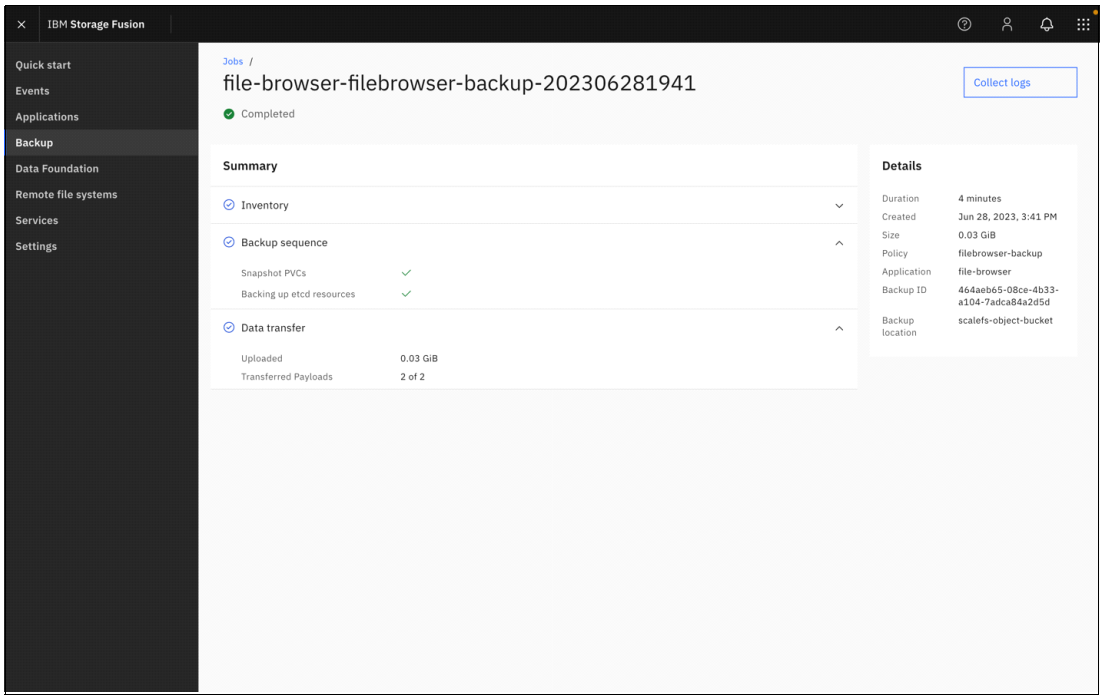


Figure 22 Completed backup with data transfer to the 'scalefs-object-bucket'

Selecting the **Applications** tab shows the completed backups list as shown in Figure 23.

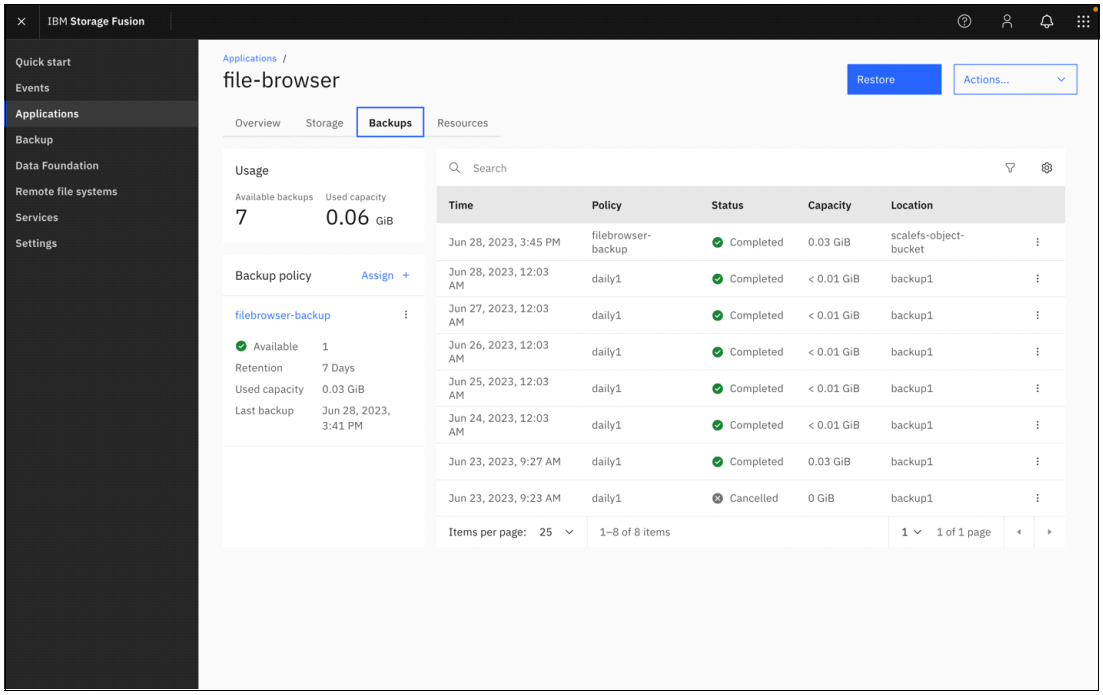


Figure 23 Completed backups list

The objects that are stored by MCG on the Storage Scale cluster can be viewed on the Storage Scale file system by logging in to the Storage Scale NSD and traversing the file system to the backing store directory location. Figure 24 shows an example that corresponds to the case described in the previous sections.

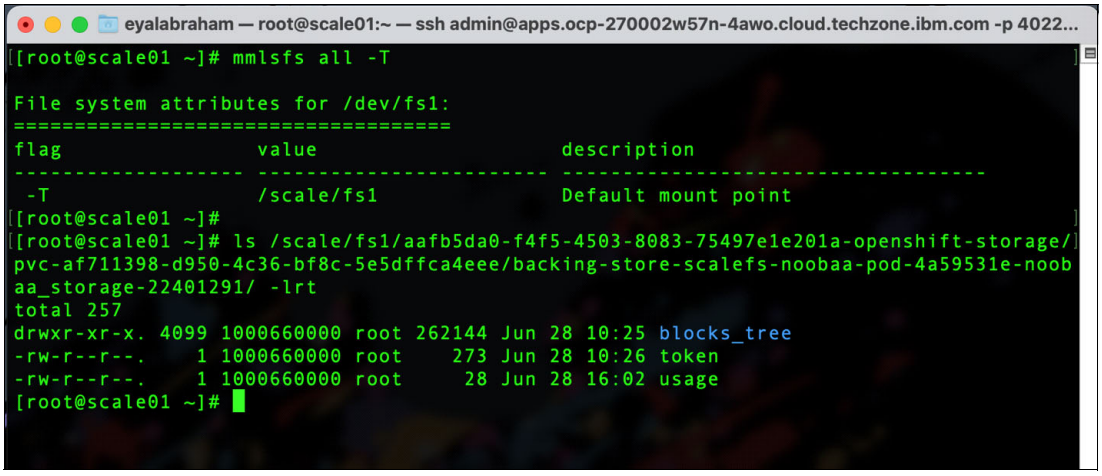


Figure 24 Object storage location in the Storage Scale file system

Backing Store for object read caching

The second use case is object read caching. The implementation of read caching includes write through cache semantics. As objects are read, they are checked for location. If they exist in the cache, then they are read locally. Otherwise, if the objects are not in the cache, then they are retrieved from the remote object store. A copy is kept in the cache until the cache dwell time is reached or cache full pressure moves the least recently used (LRU) objects.

Objects are written through the cache to the remote object store immediately. However, a local copy is stored in the cache at the same time as it is assumed that recently written objects will be read again soon. The architecture for read caching is optimized for object stores that might, for example, have a high-cost link, ingress and egress costs, or very high latency. A local cache provides latency hiding. Having the local read cache reduces cost or latency for reading the objects multiple times, and for slow connections it can speed up subsequent reads to the same objects.

Creating a cache-only Bucket Class is a two-part operation. First, a Namespace Store is created pointing to the remote object store endpoint. Second, a Bucket Class is created that points to the remote Namespace Store and to a second Backing Store, which is usually backed by local storage. The Bucket Class backed by local storage is designated for Caching.

Defining a remote Namespace Store with local cache

First, create a Namespace Store pointing to the remote object store. Figure 25 on page 20 shows the progression to create a read cache Bucket Class using a generic S3 data source and the default MCG Backing Store for the local cache space.

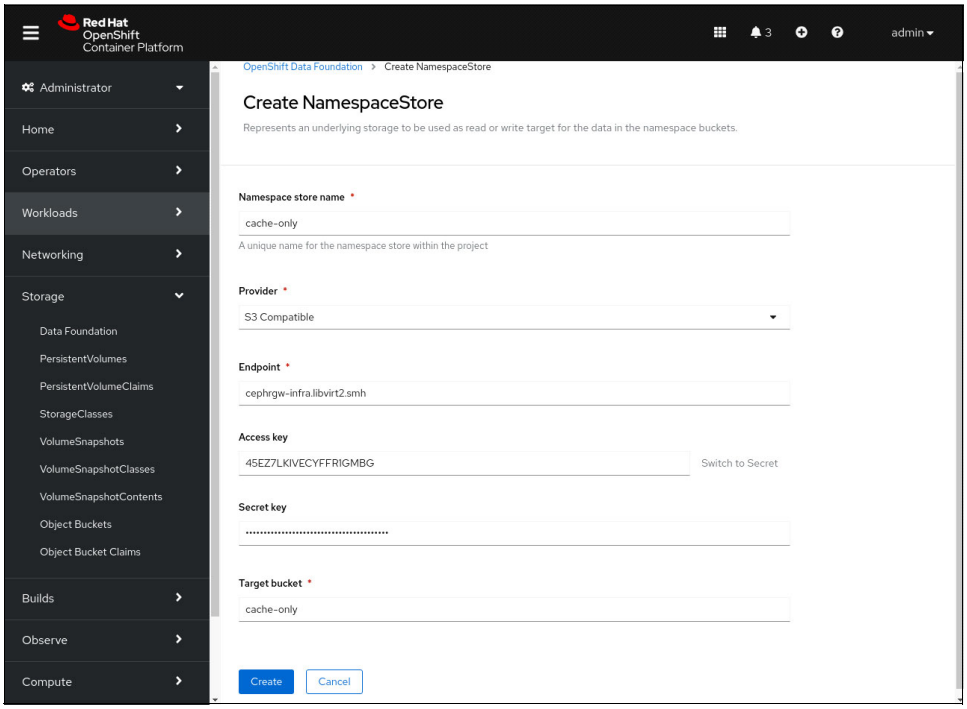


Figure 25 Create a Namespace Store for the object read caching use case

Creating a caching Bucket Class

Next create a new Bucket Class that uses the Namespace Store and a Backing Store to create the local read cache. See Figure 26. The appropriate selection for the placement policy is Cache NamespaceStore. The Data Store selected for this example is the default Backing Store, noobaa-default-backing-store. The default Backing Store is provided by local storage resources. Note that it is possible to input a time to live value. For this example, leave the value at the default value of zero, which means the time to live is 24 hours. The final step is a review page. After verifying the input, click **Create Bucket Class**.

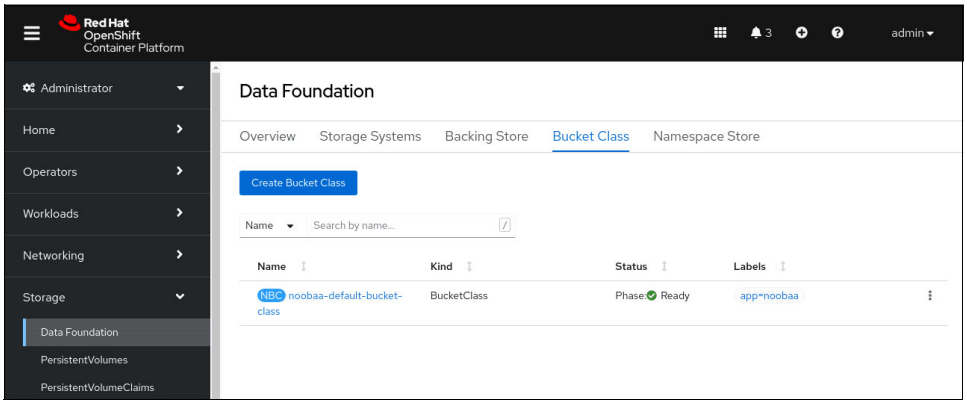


Figure 26 Creating a new Bucket Class selection screen

After clicking **Create Bucket Class**, choose the Namespace Bucket Class type, choose a unique name for your Bucket Class, and optionally add a description. Select **Next.** See Figure 27.

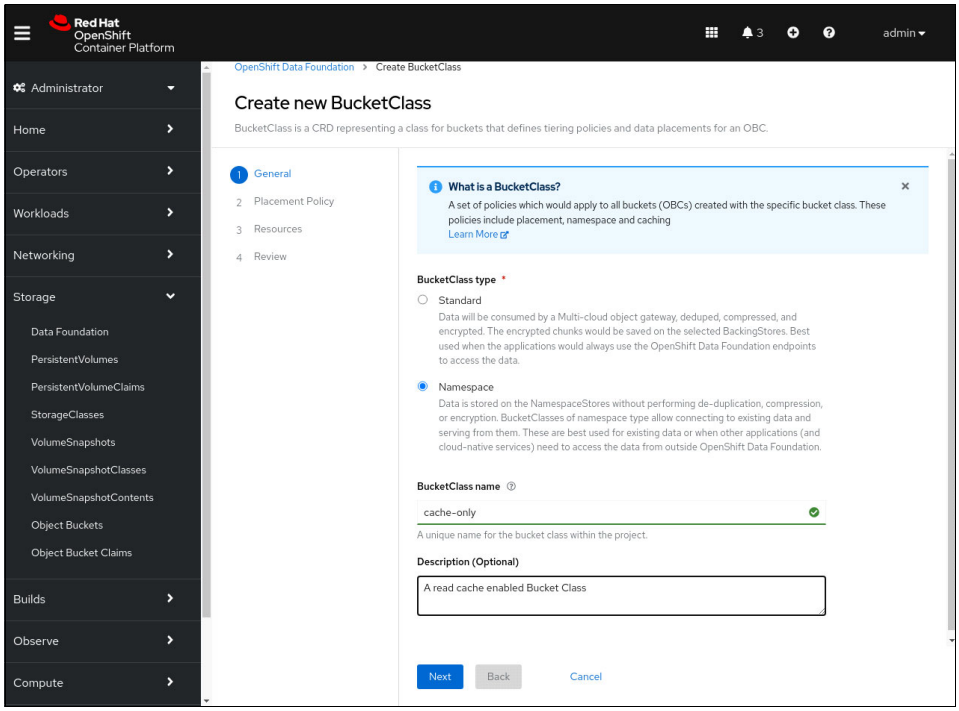


Figure 27 Creating a new Bucket Class, create screen

Choose **Cache NamespaceStore** as the Placement Policy Type. See Figure 28. Select **Next**.

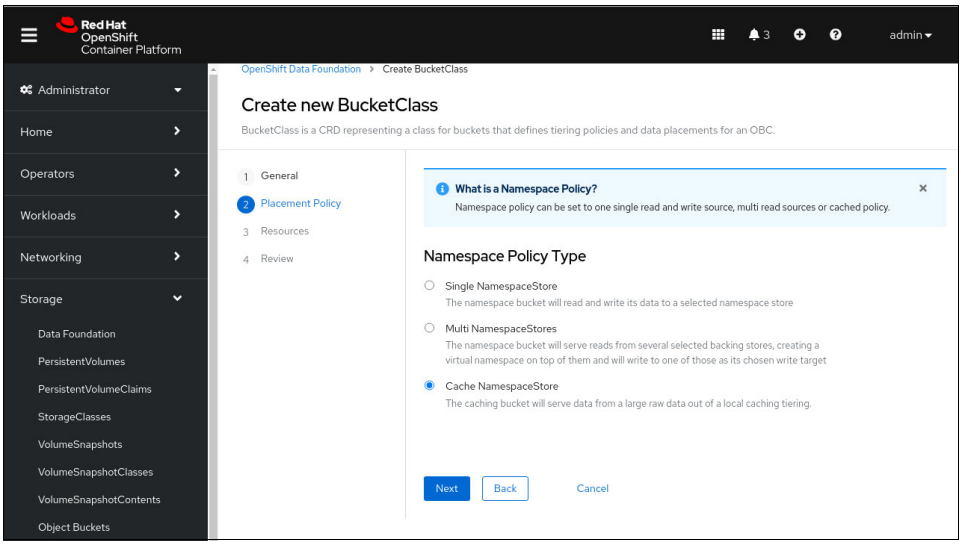


Figure 28 Creating a new Bucket Class placement screen

Select the Namespace Store created earlier from the NamespaceStore drop-down list and a Backing Store to use from the Backing store drop-down list. You can use a Backing Store you created or as in this example, use the default Backing Store. If you want a **Time to live** other than the default 24 hours, represented by the zero entry here, you can set it in this window. See Figure 29. Select **Next** when done.

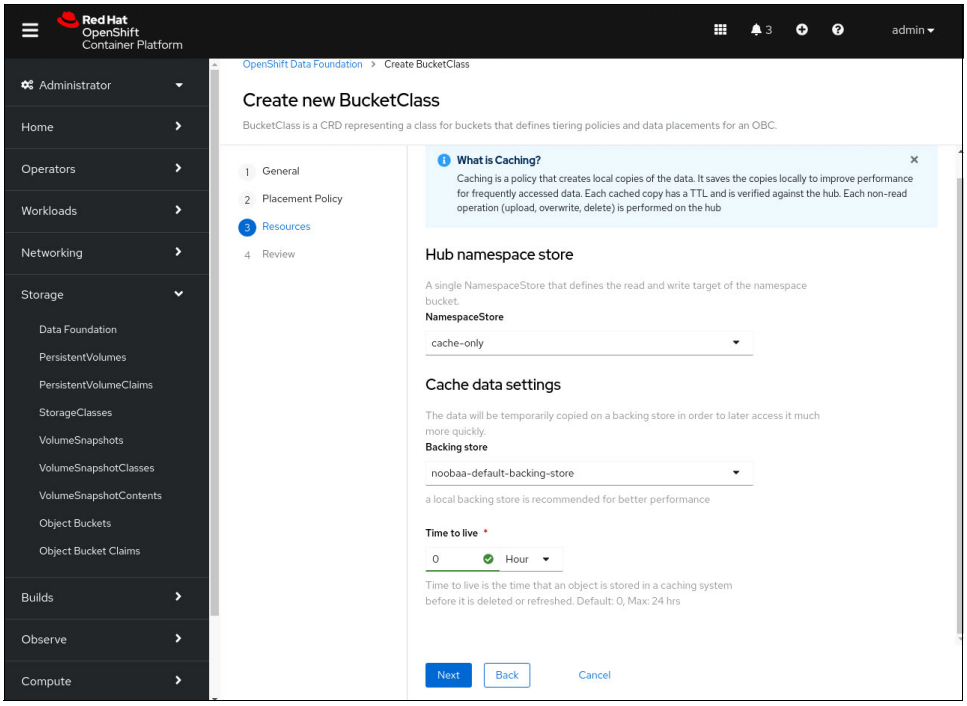


Figure 29 Resource selection screen

Review the settings. See Figure 30. Click **Create BucketClass**.

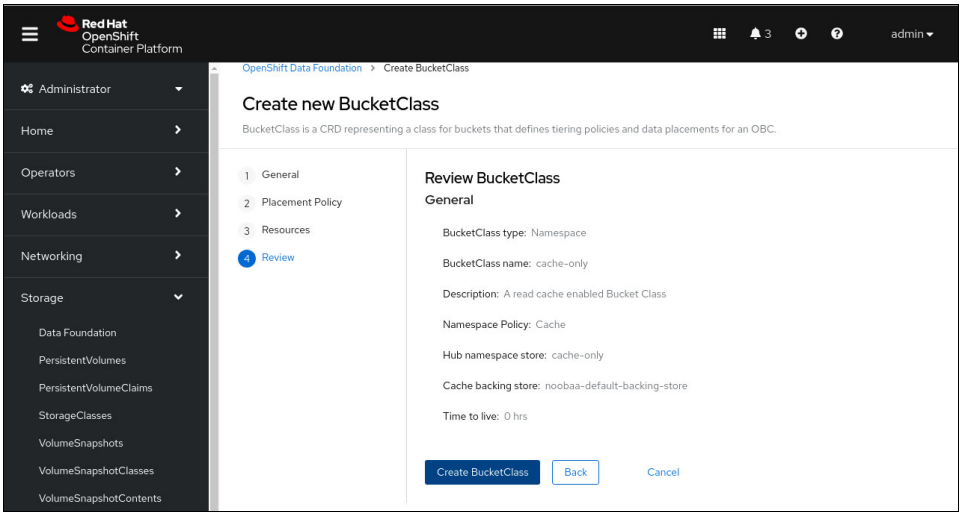


Figure 30 Review the new Bucket Class

Creating an ObjectBucketClaim backed by a cached object store

To demonstrate the read cache Bucket Class, Figure 31 shows an Object Bucket Claim using the cache-only Bucket Class previously created. Select the **Create** button to create the Object Bucket Claim (OBC). Then use the endpoint and credentials to connect to the cached bucket.

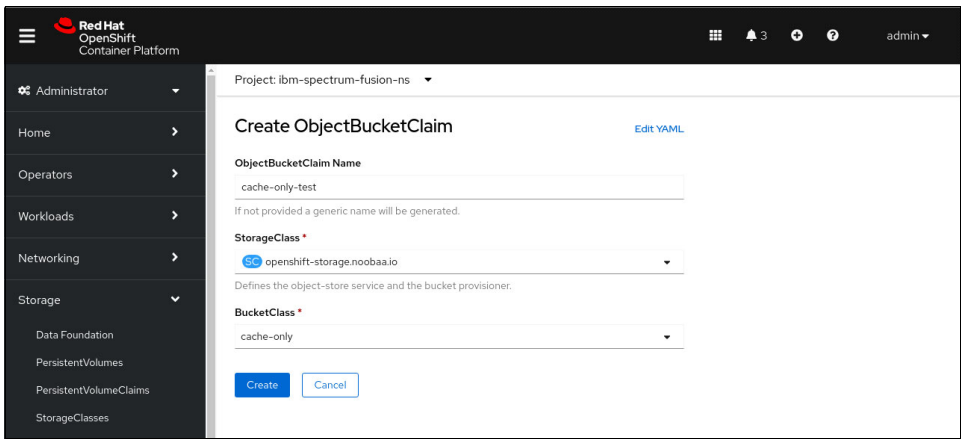


Figure 31 Using a read cache enabled Bucket Class in an Object Bucket Claim (OBC)

Backing Store for object mirroring

The third use case is a mirroring object storage implemented with MCG. See Figure 32 on page 24. The implementation defines two backing stores: one through a gateway to an external Storage Scale cluster, and a second backing store on a Data Foundation Rados Block Device (RBD), internal to the Red Hat OpenShift cluster. MCG provides options for creating a Backing Store. The types selected are convenient for this basic demonstration.

The Bucket Class is a mirrored type that uses MCG's mirroring feature to write the object to two Backing Stores. The object is stored on an external Storage Scale cluster and on the internal RDB-backed storage. More than two copies can be created, this section demonstrates a basic case of mirroring two copies.

Note: Using a Backing Store stores the object in an MCG-specific object format that applies deduplication, compression, and encryption. It is recommended to back up your MCG configuration so that MCG can be rebuilt, and objects can be restored after a cluster and MCG loss.

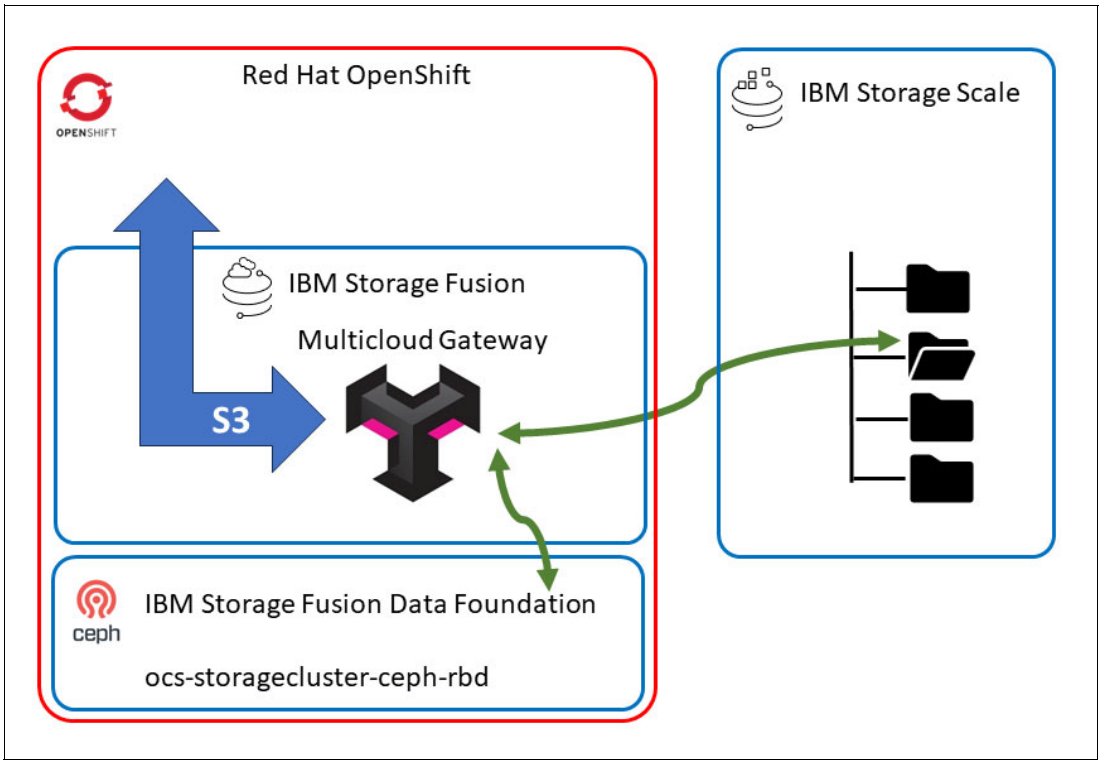


Figure 32 MCG mirrored object storage use case

Prepare a set of Backing Stores that will be used to store the mirrored objects. Backing Stores can be defined for several different targets including AWS S3, generic S3, PVC, Google Cloud Storage, Azure Blob, and IBM Cloud® Object Storage. See Figure 33.

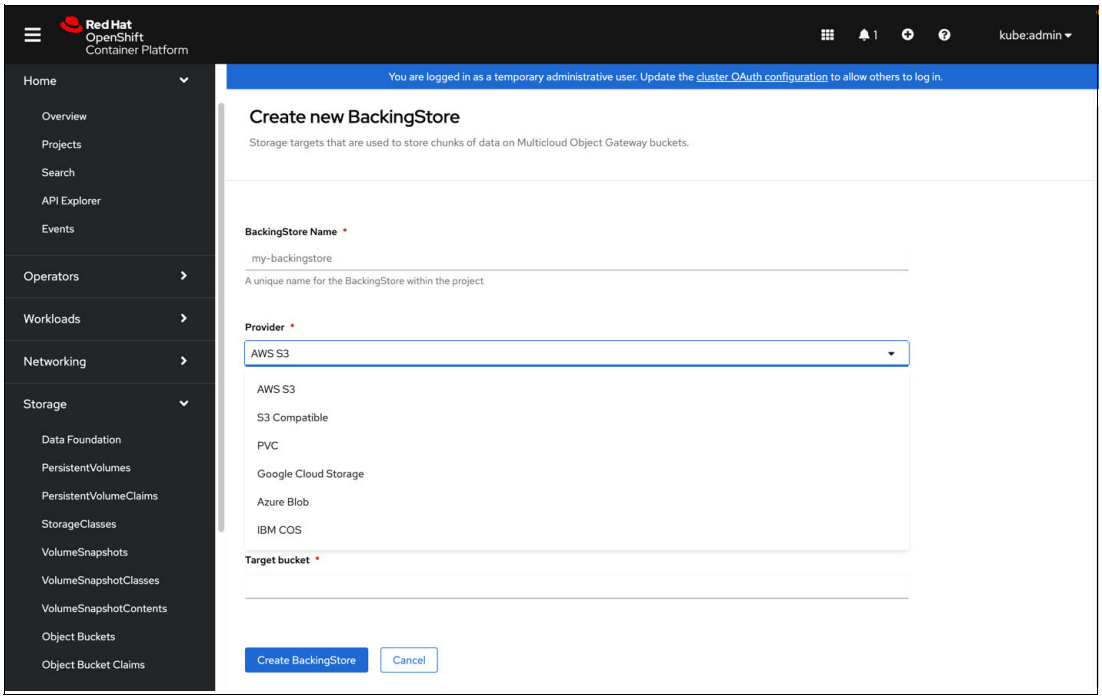


Figure 33 Available targets for creating a Backing Store

For the basic setup described in this paper, two backing stores were created. One is defined on the external Storage Scale cluster, and another is defined on the internal Data Foundation cluster using its RBD devices. See Figure 34.

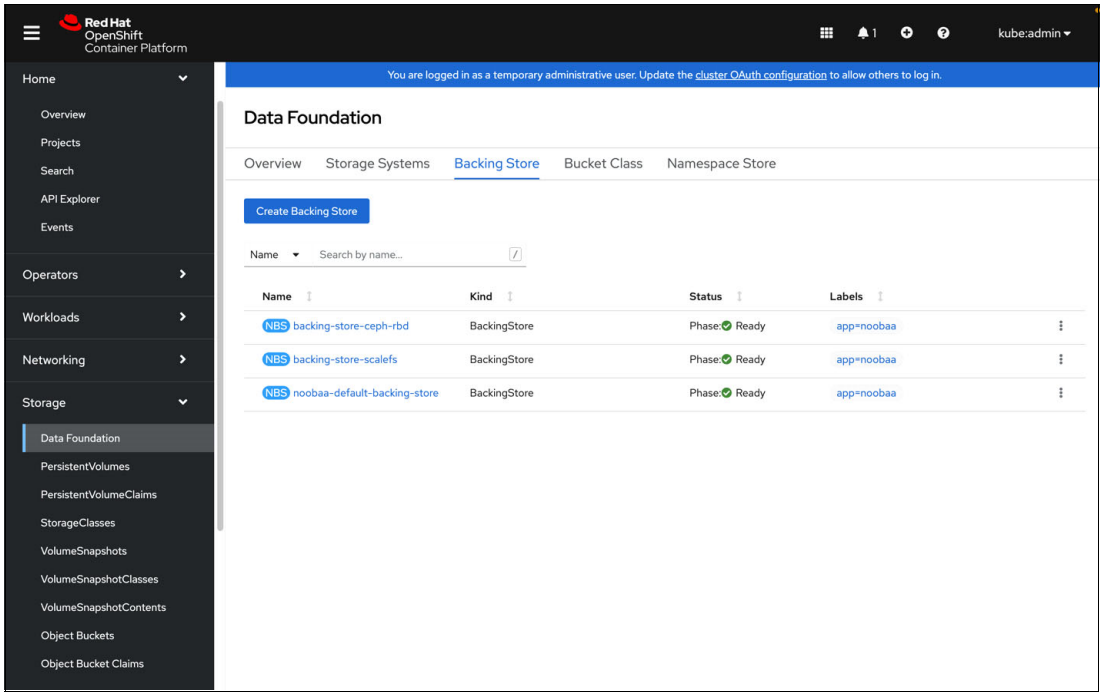


Figure 34 Ready Backing Stores available for a mirrored Bucket Class

Defining a mirrored Bucket Class

A Bucket Class is similar in principle to a Storage Class. It defines the attributes and behavior of the underlying object storage. A Bucket Class allows the user to define multiple classes of object storage behaviors that can then be selected to fit an application's requirements. The following list provides examples of Bucket Class behaviors:

- An object class that provides caching
- Object classes that aggregate multiple object stores to be used as one
- Classes that define mirroring of object stores.

Use the OpenShigt GUI to define a Bucket Class by selecting **Storage** → **Data Foundation**. Select the Bucket Class tab. The Data Foundation operator provides a wizard for defining the Bucket Class attributes. Start the wizard by clicking **Create BucketClass** and advance through the four setup steps.

In the wizard, define the type and name of the Bucket Class. Select the **Standard** type, which creates a class that uses the Backing Store's properties to deduplicate, compress, and encrypt the object data. See Figure 35. Click **Next**.

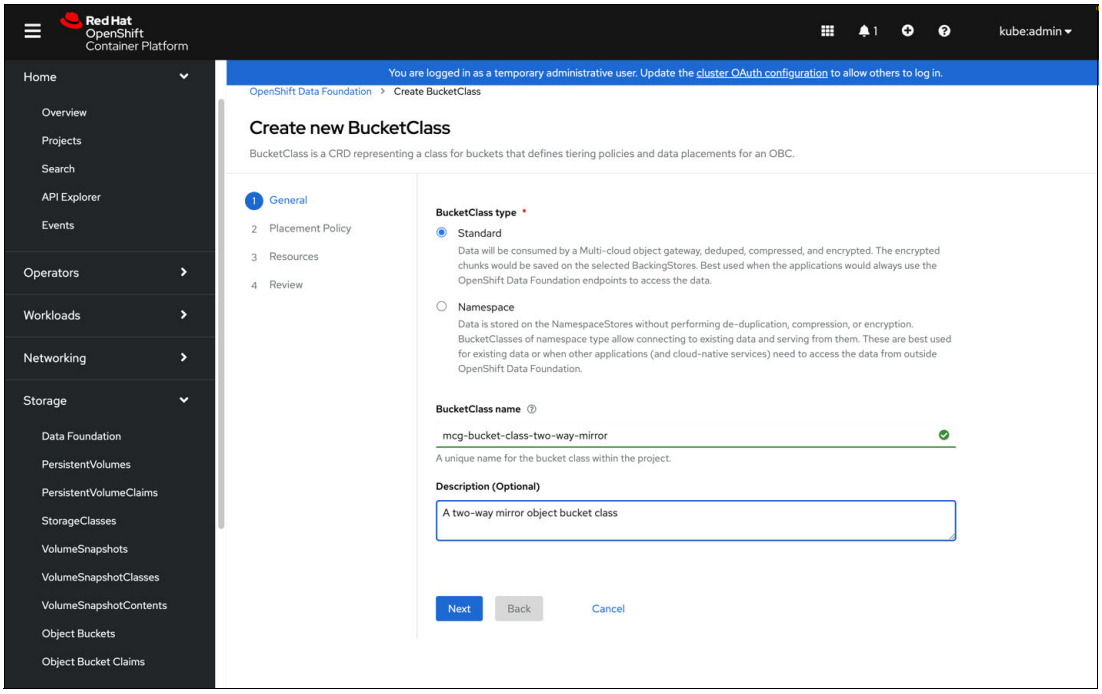


Figure 35 Mirrored Bucket Store creation wizard

On the next page of the wizard, select the 'Mirror' attribute for the Bucket Class. See Figure 36.

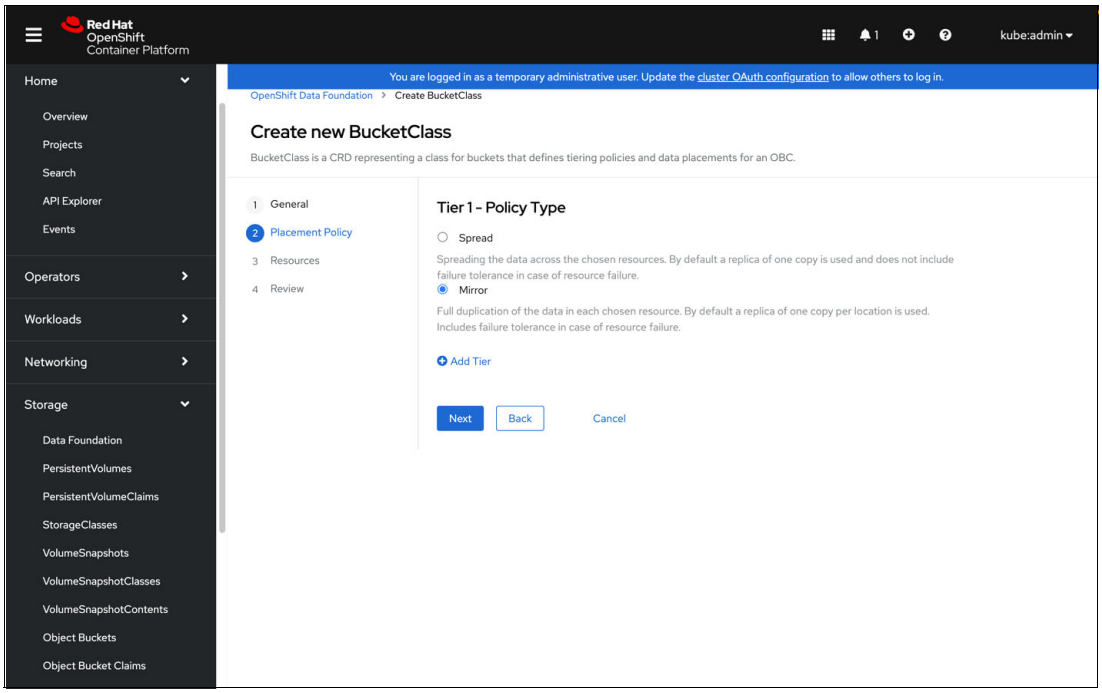


Figure 36 Selecting the Mirroring attribute for a Bucket Class

After you select the 'Mirror' attribute for the Bucket Class, you see a list of the available Backing Stores. On this wizard page, select which Backing Store to use for the mirrored objects. See Figure 37. A copy of the objects are stored on each of the selected stores. Click **Next**.

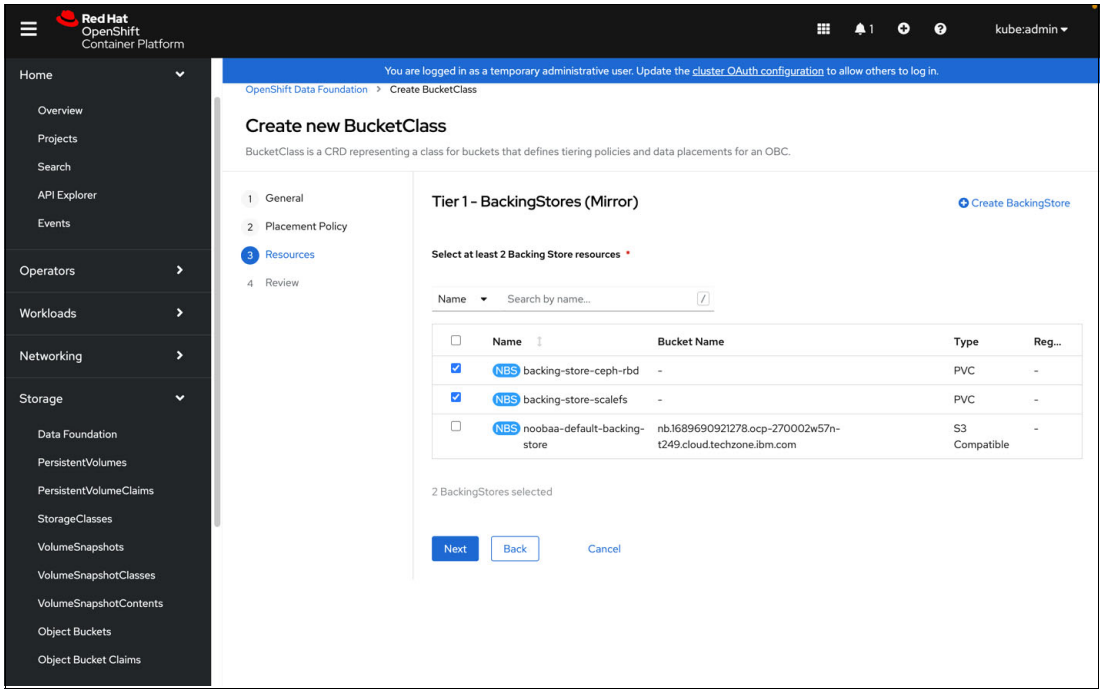


Figure 37 Target Backing Stores for mirrored object storage

Before finalizing the Bucket Class creation, the wizard provides a summary page with the class's attributes. See Figure 38.

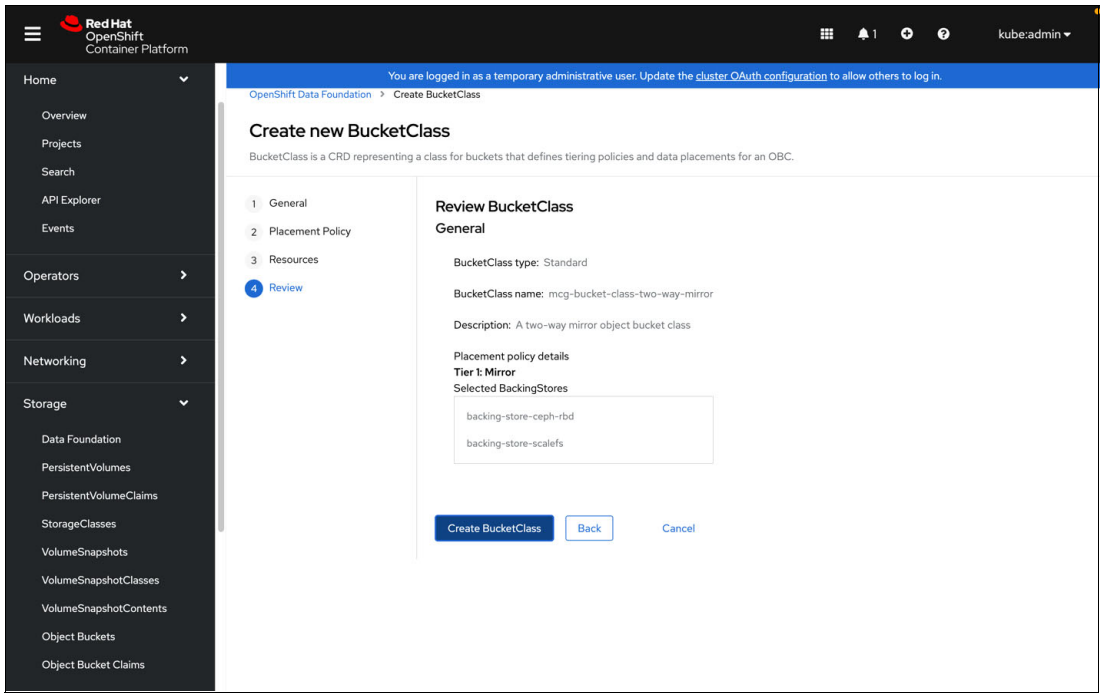


Figure 38 Mirrored Bucket Class wizard summary view

The new mirrored Bucket Class will be listed in the **Bucket Class** tab of Data Foundation and can be used after the status field shows a **Ready** state as shown in Figure 39.

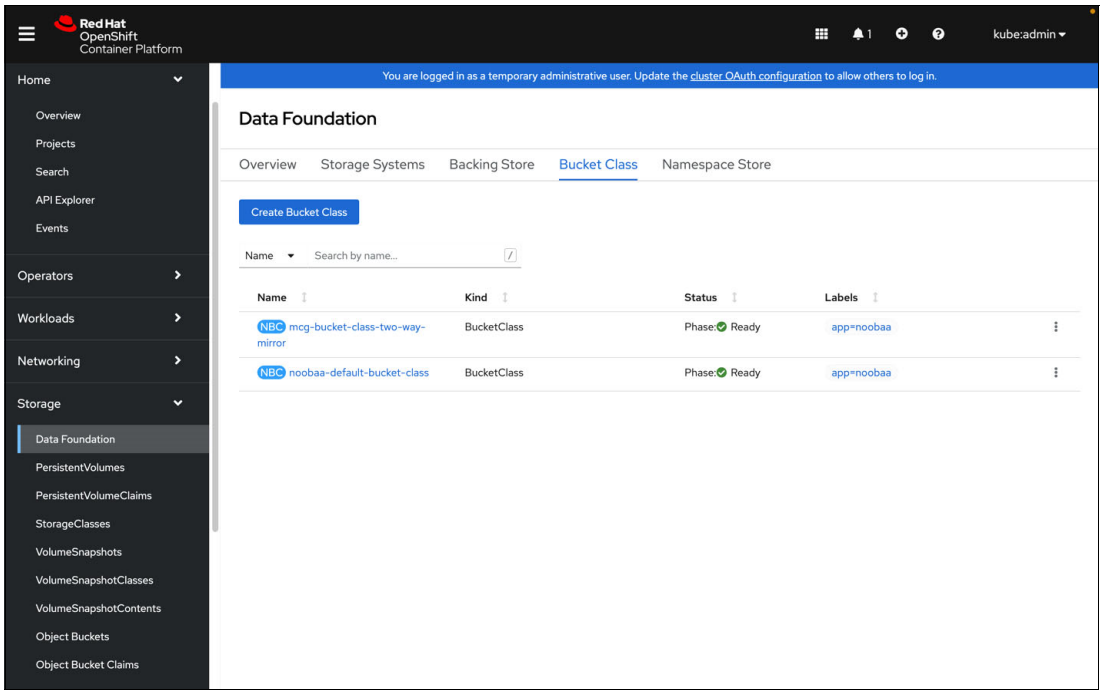


Figure 39 A two-way mirror Bucket Class defined and ready for use

Creating a two-way mirror Object Bucket Claim

For the final step, create an Object Bucket Claim. An Object Bucket Claim results in an S3 endpoint with credentials that can be used as a general S3 location for storing objects. This location can also be used as a backup location for Storage Fusion backups.

In the Red Hat OpenShift GUI, select **Storage** → **Object Bucket Claim**. Verify that the selected project namespace is set to **All projects**. See Figure 40. Provide a name for the claim and select the **openshift-storage.noobaa.io** class. Select the previously created mirrored Bucket Class.

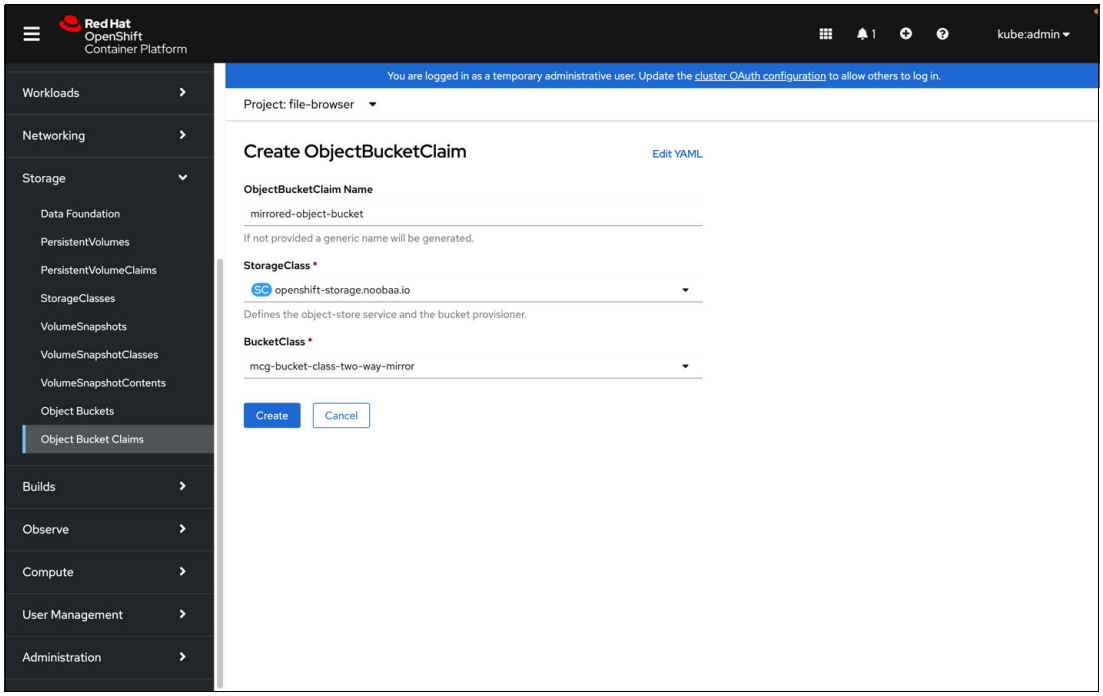


Figure 40 Create a mirrored Object Bucket Claim

The Object Bucket details are displayed after selecting **Create**. See Figure 41. Scroll down the page to reveal the parameters needed to access the object bucket. These include the S3 endpoint and endpoint credentials.

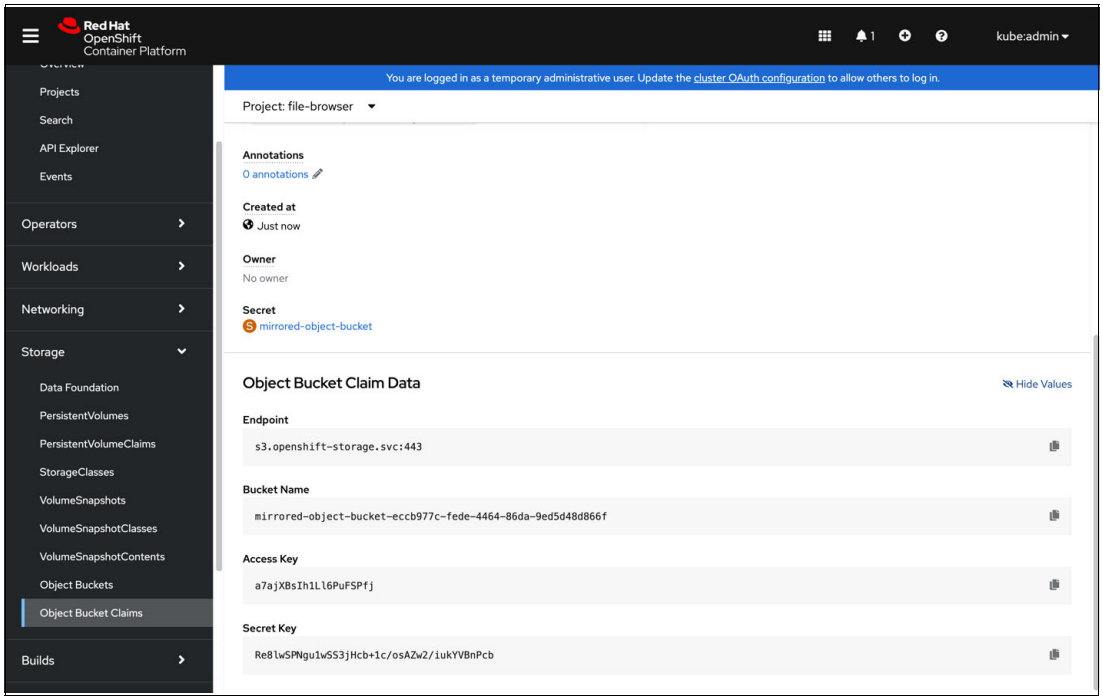


Figure 41 Mirrored Object Bucket Claim access credentials

Appendix A: Backing up Multicloud Object Gateway configuration

The MCG Backing Store represents storage targets that are used to store deduplicated, compressed, and encrypted data. Backing Stores store objects in an MCG-specific format that can be accessed by only the MCG instance that created them. An internal database holds parameters that are necessary for data access such as encryption keys, and it is recommended to back up the MCG database. This database can be restored after a cluster failure to renew access to objects stored using a Backing Store.

For more information, see Backup and Restore for Multicloud Object Gateway database (NooBaa DB). A Red Hat Customer Portal ID required to access this link.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *IBM Storage Fusion Product Guide*, REDP-5688
- *Accelerating IBM watsonx.data with IBM Storage Fusion HCI System*, REDP-5720

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

- IBM Storage Fusion documentation for Data Foundation and Multicloud Gateway

  `https://www.ibm.com/docs/en/storage-fusion/2.6?topic=storage-fusion-data-foundation`

- Red Hat documentation

  `https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.6/html/managing_hybrid_and_multicloud_resources/about-the-multicloud-object-gateway`

- Open-source community NooBaa project on GitHub

  `https://github.com/noobaa/noobaa-operator`

Authors

This paper was produced by a team of specialists from around the world working with the IBM Redbooks, Tucson Center.

Eyal Abraham is a Solutions Architect at IBM Corp. Eyal began his 30-year career as a Systems Engineer at EMC Corp. At EMC, Eyal established the EMC eLab for Fibre Channel switch technology, managed Professional Services development and operations for EMC Centera, and held project management responsibilities for EMC mid-range storage systems. Eyal joined IBM in 2009 where he has managed the world-wide technical sales team for IBM XIV®, and IBM FlashSystem®. He is now responsible for IBM Storage Fusion technical sales in the US National Market. Eyal holds a B.Sc in Electrical Engineering from Tel-Aviv University, and an MBA from Northeastern University.

Shawn Houston is a programmer and Software Defined Storage Specialist with over 30 years of professional experience. Shawn's career so far has included Electrical Engineering, Space Communications, High Performance Computing, Web Security, and Bioinformatics prior to his career's current Software Defined Storage trajectory. Shawn is currently a Senior Solution Architect at IBM.

Thanks to the following people for their contributions to this project:

Larry Coyne
IBM Redbooks®, Tucson Center

Ben Randall
IBM Infrastructure

Andrew Beattie
IBM Australia & New Zealand Global Sales - Infrastructure Sales (Mkt)

Sanjay Patel, JC Lopez
IBM Americas Global Sales - Infrastructure Sales (Mkt)

Nikica Gulin
IBM Infrastructure, Technology Lifecycle Services

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

- Find us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

IBM®	IBM FlashSystem®	Redbooks (logo)  ®
IBM Cloud®	Redbooks®	XIV®

The following terms are trademarks of other companies:

Red Hat, OpenShift, are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

REDP-5718-00

ISBN 073846144x

Printed in U.S.A.