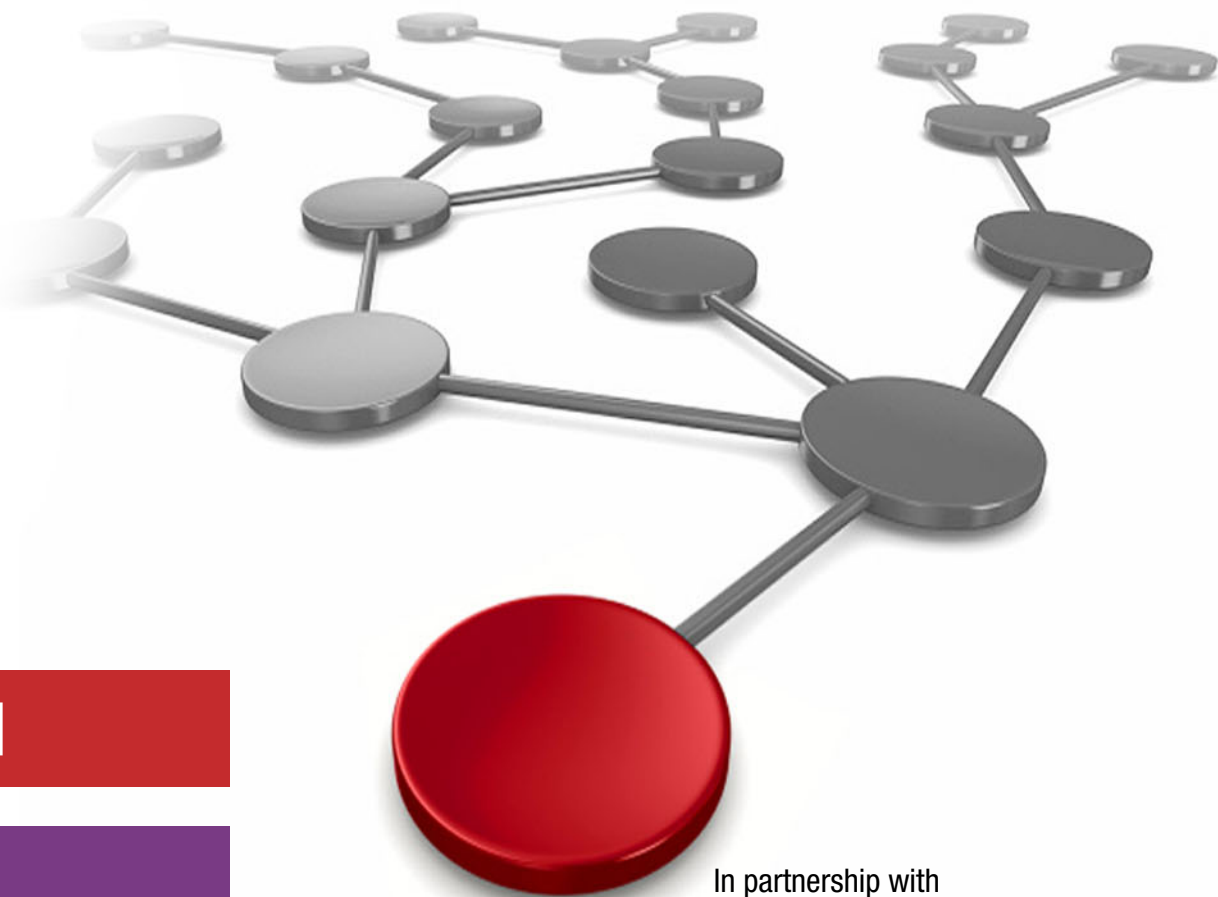# A Deployment Guide for IBM Spectrum Scale Unified File and Object Storage

Sandeep Bazar

Larry Coyne

Deepak Ghuge

Dean Hildebrand

Lata Jagwani

Simon Lorenz

Varun Mittal

Brian Nelson

Bill Owen

Smita Raut

Cloud

Storage

In partnership with
IBM **Academy of Technology**

IBM®

Redpaper

IBM

International Technical Support Organization

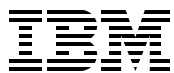# A Deployment Guide for IBM Spectrum Scale Unified File and Object Storage

February 2017

Note: Before using this information and the product it supports, read the information in "Notices" on page v.

Fourth Edition (February 2017)

This edition applies to Version 4, Release 2, Modification 2 of IBM Spectrum Scale.

This document was created or updated on May 24, 2017.

Contents

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.

v

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.

AIX®	IBM Spectrum Protect™	Redbooks (logo) ®
GPFS™	IBM Spectrum Scale™	Storwize®
IBM®	Redbooks®	Tivoli®
IBM Spectrum™	Redpaper™	

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

Preface

Because of the explosion of unstructured data that is generated by individuals and organizations, a new storage paradigm that is called *object storage* has been developed. Object storage stores data in a flat namespace that scales to trillions of objects. The design of object storage also simplifies how users access data, supporting new types of applications and allowing users to access data by using various methods, including mobile devices and web applications. Data distribution and management are also simplified, allowing greater collaboration across the globe.

OpenStack Swift is an emerging open source object storage software platform that is widely used for cloud storage. IBM® Spectrum Scale, which is based on IBM General Parallel File System (IBM GPFS™) technology, is a high-performance and proven product that is used to store data for thousands of mission-critical commercial installations worldwide. Throughout this IBM Redpaper™ publication, IBM Spectrum™ Scale is used to refer to GPFS. The examples in this paper are based on IBM Spectrum Scale™ V4.2.2.

IBM Spectrum Scale also automates common storage management tasks, such as tiering and archiving at scale. Together, IBM Spectrum Scale and OpenStack Swift provide an enterprise-class object storage solution that efficiently stores, distributes, and retains critical data.

This paper provides instructions about setting up and configuring IBM Spectrum Scale Object Storage that is based on OpenStack Swift. It also provides an initial set of preferred practices that ensure optimal performance and reliability. This paper is intended for administrators who are familiar with IBM Spectrum Scale and OpenStack Swift components.

Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Sandeep Bazar is a Software Engineer with the IBM Spectrum Scale protocols test team. He is responsible for testing and automating the functions of the object protocol in the IBM Spectrum Scale environment. He has been working with IBM for over five years. Sandeep has previously held roles of building and testing for IBM Scale Out Network Attached Storage and IBM Storwize® V7000 Unified products. Sandeep holds a master degree in Computer Science from the KIIT University, Bhubaneswar.

Larry Coyne is a Project Leader at the International Technical Support Organization (ITSO), Tucson, Arizona, center. He has 34 years of IBM experience, with 23 years in IBM storage software management. He holds degrees in Software Engineering from the University of Texas at El Paso, and Project Management from George Washington University. His areas of expertise include client relationship management, quality assurance, development management, and support management for IBM Storage Management Software.

Deepak Ghuge is a Master Inventor and a Software Engineer with the IBM Spectrum Scale development team. He is responsible for providing the OpenStack Keystone integration into IBM Spectrum Scale. He looks after the authentication and security aspects of the IBM Spectrum Scale Object Storage. He has been working with IBM for seven years. Deepak previously worked with the IBM Scale Out Network Attached Storage and Storwize V7000 Unified teams. Deepak holds a Bachelor of Tech degree in Computer Engineering from the College of Engineering, Pune.

Dean Hildebrand is a Master Inventor and Principal Research Staff Member in the Cloud Storage Software Research group at the IBM Almaden Research Center. He is a member of the IBM Academy of Technology, has authored numerous scientific publications, created over 50 patents, and chaired and sat on the program committee of numerous conferences. Dr. Hildebrand pioneered pNFS, demonstrating the feasibility of providing standard and scalable access to any parallel file system. He received a Bachelor of Science degree in Computer Science from the University of British Columbia in 1998 and Master of Science and PhD. Degrees in Computer Science from the University of Michigan in 2003 and 2007.

Lata Jagwani is an Information Developer with the IBM ISDL ID team. She is responsible for writing and editing the customer-facing documentation for various storage products, such as IBM Spectrum Scale and IBM DeepFlash 150. She has been with IBM for a little over a year and holds a bachelor degree in Computer Science and a master degree in English Literature from the University of Pune.

Simon Lorenz is an IT Architect at IBM Research and Development in Mainz, Germany. He joined IBM Germany in 1993 and worked on productivity and manufacturing quality improvements within the IBM Disk Drive Manufacturing Management software. During international assignments, he helped to improve fully automated chip factories in Asia and the US. Simon has held various positions within IBM Research and Development. Since 2009, he worked on Storage Systems Management software and has been responsible for subcomponents, such as system health reporting, cluster configuration management, and recovery. Simon joined the IBM General Parallel File System development team in 2014.

Varun Mittal is a Software Engineer with IBM Storage Labs in Pune, India. He works with the IBM Spectrum Scale development team on Object protocol support. Varun has been working with IBM for eight years. He previously worked as NAS protocols authentication developer with IBM Scale Out Network Attached Storage and Storwize V7000 Unified offerings. Varun holds a master degree in Computer Applications from Indian Institute of Technology, Roorkee.

Brian Nelson is a Software Engineer with the IBM Spectrum Scale development team. He is responsible for providing the OpenStack Swift integration into IBM Spectrum Scale and Elastic Storage Server (ESS). He worked within IBM for over 18 years on projects, such as IBM AIX®, IBM Director, and IBM PowerVC. Brian holds a Bachelor of Science degree in Computer Science from Trinity University.

Bill Owen is a Senior Technical Staff Member with the IBM Spectrum Scale development team. He has led the effort for the integration of OpenStack with IBM Spectrum Scale, focusing on the Swift object, Cinder block, and Manila file storage components of OpenStack. He has worked in various development roles within IBM for over 15 years. Before joining IBM, Bill developed and deployed grid management systems for electric utilities. Bill holds Bachelor of Science and Master of Science degrees in Electrical Engineering from New Mexico State University.

Smita Raut is an Advisory Software Engineer with IBM Storage Labs in Pune, India. She works with the IBM Spectrum Scale development team on the object protocol support. In her six years with IBM, she has worked on various products, such as IBM Scale Out Network Attached Storage and IBM V7000 Unified. She has an overall industry experience of 15 years with organizations such as Persistent Systems, TIBCO Inc, and BMC Software.

Thanks to the IBM Spectrum Scale team for their hard work and tireless effort through all phases of development, test, and deployment.

Thanks to the authors of the previous editions of this paper.

Dean Hildebrand
Bill Owen
Simon Lorenz
Brian Nelson
Andreas Luengen
James Wormwell
Martha Burns

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 ITSO 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

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks® publications in one of the following ways:

► Use the online **Contact us** review Redbooks form found at:

 ibm.com/redbooks

► Send your comments in an email to:

 redbooks@us.ibm.com

► Mail your comments to:

 IBM Corporation, International Technical Support Organization
 Dept. HYTD Mail Station P099
 2455 South Road
 Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ► Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- ► Follow us on Twitter:

  http://twitter.com/ibmredbooks

- ► Look for 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

Summary of changes

This section describes the technical changes that were made in this edition of the paper and in previous editions. This edition might also include minor corrections and editorial changes that are not identified.

Summary of Changes
for A Deployment Guide for IBM Spectrum Scale Unified File and Object Storage
as created or updated on May 24, 2017.

May 2017, update

Removed several online urls in the "Related publications" on page 55.

February 2017, Fourth Edition

Updated to describe IBM Spectrum Scale integration and many new features. Also, the installation and maintenance descriptions were updated and are consistent with IBM Spectrum Scale V4.2.2.

September 2016, update

Note: The September 2016 update added a note that configuration steps were out of date. This warning is no longer needed.

March 2015, Third Edition

This revision reflects the addition, deletion, and modification of the new and changed information that has been listed below.

Changed information

► Added the new IBM Spectrum Scale product name, which is based on IBM General Parallel File System (GPFS) and was formerly known as ESS
► Added the new IBM Spectrum Protect™ product name (based on IBM Tivoli® Storage Manager)

December 2014, Second Edition

This revision reflects the addition, deletion, and modification of the new and changed information that is listed below.

New information

► Introduced a new automated installation procedure by using the Chef configuration management toolset

► Described in more detail the steps that are required to back up your object store configuration and data to protect against disaster scenarios and how to recover from that saved data

Changed information

► Added numerous clarifications and tips that we added based on our own and customer experience by using Swift in a GPFS environment

► Removed the installation appendixes and made installation information available for download here.

IBM Spectrum Scale Unified File and Object Storage

This chapter introduces IBM Spectrum Scale Unified File and Object Storage, describes its high-level architecture, and describes the benefits that it brings to organizations and businesses. In this paper, the terms *IBM Spectrum Scale Unified File and Object Storage* and *IBM Spectrum Scale Object Storage* are synonymous.

1.1 Introduction

Object storage is rapidly growing as a secure and cost-effective way for organizations to archive their data. When deployed, many organizations soon realize that the benefits of using REST APIs for data access extend to much more than just archiving, thereby allowing direct access by its organization. If the workload profile fits, it can even replace NFS and SMB.

This paper explains how IBM Spectrum Scale Object Storage targets this broader view of the capabilities of object storage by providing an industry-unique architecture that can simultaneously support both file and object workloads.

Data centers are struggling to efficiently and cost-effectively store and manage vast amounts of data. The increasing number of application domains, such as analytics, online transaction processing (OLTP), and high-performance computing (HPC), have created silos of storage within data centers. With each new application, a new storage system can be required, forcing system administrators to become experts in numerous storage management tools. With the emergence of object storage, administrators must avoid this pattern of deploying a new island of storage.

Object storage offers multiple benefits beyond today's existing distributed data access protocols. Initially motivated by mobile and web-based applications, the RESTful HTTP interface is web-friendly and typically includes wire encryption (HTTPS) and scalable role-based authentication and authorization. Further, data access semantics are relaxed to support eventual consistency, and objects cannot be updated after they are created, although they can be replaced, upgraded, or removed. Changes to typical file access semantics allow for highly scalable data access, both within and across data centers. These types of semantics are ideal for images, videos, text documents, virtual machine (VM) images, and other similar files.

Unfortunately, not all applications can leverage object storage or their APIs, such as S3 and Swift. The result is that the dominant object storage solutions on the market today force administrators into the same old paradigm of creating an isolated island of storage in their data center. IBM Spectrum Scale Object Storage breaks this pattern by allowing for high-performance data access to the same data through both file and object protocols. This allows applications to use the correct protocol for their performance, security, data access semantics, and scalability requirements.

This paper details everything that an administrator needs to deploy and manage IBM Spectrum Scale Object Storage, focusing on the following topics:

► Advantages of files and objects in IBM Spectrum Scale while building an object storage solution.

► Installation of OpenStack Swift on IBM Spectrum Scale, providing preferred practices for configuring and tuning both Swift and IBM Spectrum Scale. This version describes the use of the IBM spectrumscale installation toolkit to automate the installation steps.

► A discussion about how to back up your object store configuration and data to protect against disaster scenarios and about recovering the saved data.

► Clarifications and tips based on our own and customer experience of using Swift in an IBM Spectrum Scale environment. The Swift and IBM Spectrum Scale features were tested in our labs as part of IBM Spectrum Scale Object Storage.

1.2 Assumptions

The purpose of this paper is to describe the benefits of using IBM Spectrum Scale Object Storage to guide the administrator through the installation and configuration of IBM Spectrum Scale Object Storage, and to describe the general set of configurations and scenarios that have been validated.

The following assumptions are made:

► The audience is familiar with IBM Spectrum Scale because this paper is not intended to serve as a comprehensive IBM Spectrum Scale overview. For a comprehensive understanding of IBM Spectrum Scale, see "Related publications" on page 55.

► The paper focuses on the benefits of IBM Spectrum Scale Object Storage and does not attempt to make any direct comparison to other object storage solutions.

1.3 Key concepts and terminology

This section defines the key concepts and terminology that are associated with IBM Spectrum Scale and OpenStack Swift as they relate to IBM Spectrum Scale Object Storage.

IBM Spectrum Scale

The following terms are associated with IBM Spectrum Scale:

► IBM Spectrum Scale Object Storage: The software that is used to access data in an IBM Spectrum Scale file system by using Swift or S3 RESTful APIs. In this paper, it is assumed that this software is installed on every IBM Spectrum Scale protocol node, and IBM Spectrum Scale is the file system that underpins OpenStack Swift. For more information, see IBM Knowledge Center.

► IBM Spectrum Scale protocol node: Also known as the Cluster Export Services (CES) node. These are the nodes from which the non-IBM Spectrum Scale clients can access data in and managed by IBM Spectrum Scale by using the appropriate protocol artifacts, such as exports, shares, and containers.

► Elastic Storage Server (ESS): The ESS combines the IBM Spectrum Scale software with storage enclosures, drives, and networking components to provide a high-capacity and highly scalable storage solution. ESS includes the unique and innovative IBM Spectrum Scale RAID capability, which provides extreme data integrity and reduced latency with faster rebuild times and enhanced data protection.

For more information, see the following resources:

– See Elastic Storage Server planning and service information here.

– See IBM Knowledge Center here, here, and here.

► Network Shared Disk (NSD): A logical unit number (LUN) that is provided by a storage subsystem, such as an ESS, to use in an IBM Spectrum Scale file system.

► Storage pool: A collection of NSDs. IBM Spectrum Scale storage pools in IBM Spectrum Scale allow the grouping of storage devices within a file system based on performance, locality, and reliability characteristics. Storage pools provide a method to partition file system storage to offer several benefits, including improved price-performance, reduced contention on premium resources, seamless archiving, and advanced failure containment.

For example, one pool can be an enterprise-class storage system that uses high-performance flash devices, and another pool can consist of numerous disk controllers that host a large set of economical Serial ATA (SATA) or Near-Line SAS disks.

There are two types of IBM Spectrum Scale storage pools: internal and external. For internal storage pools, IBM Spectrum Scale manages the data storage. However, for external storage pools, IBM Spectrum Scale handles the policy processing but leaves data management to an external application, such as IBM Spectrum Protect (formerly known as IBM Tivoli Storage Manager). Storage pools are declared as an attribute of the disk and file system.

► Fileset: A subtree of a file system namespace. Place the Swift object store in independent filesets to allow fileset-level administrative operations, such as snapshot and backup, to apply only to the files in the object store rather than to the entire file system. The fileset is identified as an attribute of each file and can be specified in a policy to control the initial data placement, migration, and replication of the file data. A Swift Storage policy maps to a fileset so that all object data that is defined for the policy is contained by the fileset. For more information, see Storage policy on page 6.

► Replication: A feature of IBM Spectrum Scale that enables multiple copies of data and metadata for failure containment and disaster recovery. IBM Spectrum Scale supports replication of both metadata and data independently and provides fine-grained control of the replication. Replication can incur costs in terms of disk capacity and performance and must be used carefully.

► IBM Spectrum Scale quotas: The amount of disk space and the number of inodes that are assigned as upper limits for a specified user, group of users, or fileset. With OpenStack Swift, IBM Spectrum Scale user quotas are not used. Instead, the system relies on OpenStack Swift quotas to provide a similar type of service. However, IBM Spectrum Scale fileset quotas can still be defined, such as for inodes to limit the resources that are consumed by a fileset.

► IBM Spectrum Scale access control lists (ACLs): Fine-grained access control mechanisms. With OpenStack Swift, IBM Spectrum Scale ACLs are not used. Instead, the system relies on OpenStack Swift ACLs to set access permissions.

► Cluster Export Services (CES): CES provides highly available file and object services to an IBM Spectrum Scale cluster. Network File System (NFS), Object, or Server Message Block (SMB) protocols are supported today.

► CES IP: Protocol services are made available through CES IP addresses (CES IPs). These addresses are separate from the IP addresses that are used internally by the cluster. The pool of CES IPs is considered floating because each IP address can move independently among all protocol nodes. In the event of a protocol node failure, accessibility to all protocols is maintained because the CES IPs automatically move from *the failed protocol node to a healthy protocol node*.

► Active File Management (AFM): A scalable, high-performance, file system caching layer that is integrated with the IBM Spectrum Scale cluster file system. With AFM, you can create associations from a local IBM Spectrum Scale cluster to a remote cluster or storage and define the location and the flow of file data to automate the management of the data, which allows you to implement a single namespace view across sites around the world.

OpenStack Swift

The following terms are associated with the OpenStack Swift Object Storage open source layer:

► Account: The top-level element in the IBM Spectrum Scale Object Storage system hierarchy. An account contains a list of the containers in the account. In the OpenStack environment, account is synonymous with project or tenant, as used by Keystone.

► Project: The Keystone entity that maps to a Swift account. In the past, the term tenant was also used in Keystone. Tenant was deprecated and replaced by project.

► Container: The second-level element in the hierarchy under accounts. A container maintains a list of objects that belong to the container. The account and container provide a namespace for objects, analogous to files in a directory path. Many features, such as ACLs, versioning, and quotas, are controlled at the container level. A container in Swift is equivalent to a bucket when using the S3 protocol to access object data.

► Object: The third-level element in the hierarchy under containers. An object stores actual data content and metadata that describes the object. In IBM Spectrum Scale, objects are stored as files, and object metadata (as shown in Example 1-1) is stored as file-extended attributes.

Example 1-1   Object metadata

```
Content-Length = 14540
name = /AUTH_39150cd50dfb47e7be85280735174691/bill/testobj ETag =
2cd0ae668a585a14e07c2ea4f264d79b
X-Timestamp = 1402267969.60475
Content-Type = application/octet-stream
```

► Object metadata: Key-value pairs that are stored by Swift and Swift clients on an object through the `POST` command. These key-value pairs are stored as file-extended attributes.

► Keystone: The OpenStack service that provides identity, token, catalog, and policy services. In this paper, Keystone is used as the Swift authentication service that provides role-based management of Swift accounts and containers. Keystone can be integrated with the existing Lightweight Directory Access Protocol (LDAP) and Active Directory (AD) systems. IBM Spectrum Scale Object Storage can be configured with an instance of the Keystone service running on each protocol node, or configured to use a pre-existing Keystone in your environment.

► Proxy service: The proxy service runs on the IBM Spectrum Scale protocol nodes and accepts requests from clients regarding accounts, containers, or objects, and directs each request to the appropriate service within the cluster.

► Account service: The account service runs on the IBM Spectrum Scale protocol nodes and accepts requests to perform account operations.

► Container service: The container service runs on the IBM Spectrum Scale protocol nodes and accepts requests to perform container operations.

► Object service: The object service runs on the IBM Spectrum Scale protocol nodes and *accepts requests to store or retrieve objects in IBM Spectrum Scale*.

► Virtual devices: In traditional Swift deployments, devices map with physical disks. In IBM Spectrum Scale Object Storage, virtual devices map with a top-level directory in which the object store hierarchy is created. IBM Spectrum Scale configures 128 virtual devices on initial configuration. This number is sufficient for most deployments. Because IBM Spectrum Scale is a parallel file system, a virtual device can be accessed from any protocol node.

► Partitions: These are used to group and evenly distribute objects in the file system. The number of partitions is specified during Swift configuration when the Swift rings are created. The number of partitions determines the expected percentage of objects in each one. The IBM Spectrum Scale installation procedure uses a partition power of 14 so that 2^{14} or 16384 partitions are created. These are evenly distributed across the 128 virtual devices under normal circumstances.

► Pseudo-hierarchical folders and directories: Objects that are stored in a flat namespace but whose names include one or more forward slash (/) characters to imply levels in a hierarchy. Each forward slash (/) character creates another level in the hierarchy.

Pseudo-directories do not affect how objects are placed on disk, but they can be useful while listing objects within a specific pseudo-directory.

► SQLite: A lightweight database in which account and container listings are stored. Each account database contains a list of containers for that account, and each container database contains a list of associated objects.

► Memcached: A distributed caching service that caches items in memory for fast retrieval. Account and container database information and authentication tokens are examples of the cached items.

► Swift access control list (ACL): Controls access to a container. Swift provides role-based authentication that is mixed with fine-grained ACLs. Swift ACLs apply to containers and not to individual objects.

> **Note:** IBM Spectrum Scale ACLs are not used with Swift.

► Swift quotas: Allows specification of the amount of disk space or number of objects that can be consumed by either an account with its containers or to an individual container. The interaction between Swift Quotas and IBM Spectrum Scale quotas are described in Chapter 6, "Swift feature overview" on page 47.

► Swift ring: Determines the partition and protocol node that are used to access accounts, containers, and object data. There is one ring for account data and one ring for container data. There is one object ring for each storage policy that is defined.

► Eventual consistency: All Swift items, including accounts, containers, and objects, are ensured to eventually reach a consistent state, thereby allowing scalable storage, and providing clients access to the most up-to-date data. For more information about eventual consistency in IBM Spectrum Scale Object Storage, see IBM Knowledge Center.

► Storage policy: Storage policies allow for some level of segmenting of the Swift cluster for various purposes through the creation of multiple object rings. In IBM Spectrum Scale Object Storage, the supported policy types are enable-file-access, enable compression, enable-encryption, and default.

► Objectizer service: The objectizer service converts files that are ingested from the file interface on unified file and object access container path to be available from the object interface. When new files are added from the file interface, they must be visible to the Swift container database to show the correct container listing and container or account statistics. This service ensures synchronization between the file metadata and the object metadata at the predefined time interval to provide accurate container and account listing.

► ID Mapping modes: The behavior of user ID mapping between file and object protocols in unified file and object access is governed by ID mapping modes or Identity Management modes. There are two ID mapping modes in unified file and object access: *local mode*, which is a separate identity between an object and file (also the default mode), and *unified mode*, which is a shared identity between an object and file.

► Multi-region: Multi-region is active-active replication support in OpenStack Swift across multiple geographically dispersed regions. It is used to provide a higher level of resiliency over single-site configurations, and improve response time for clients by creating a replica of the data in a cluster closer to the clients. Multi-region can also be used to distribute the object load over several clusters to reduce contention in the file system.

1.4 Introduction to IBM Spectrum Scale Unified File and Object Storage

This section describes the features, the use cases, the high-level architecture, and the benefits of IBM Spectrum Scale Object Storage.

1.4.1 IBM Spectrum Scale features

IBM Spectrum Scale enables virtualization, analytics, file, and object use cases to be unified into a single scale-out data plane for the entire data center. As shown in Figure 1-1, IBM Spectrum Scale provides a single point of management and a single namespace for all data, which means that applications can leverage the POSIX, NFS, and SMB file access protocols with an HDFS connector plug-in, and the Swift and S3 object protocols, all to a single data set. Data can then be tiered in differentiated classes of storage and accessed from around the globe, ensuring that data is always available in the correct place at the correct time.
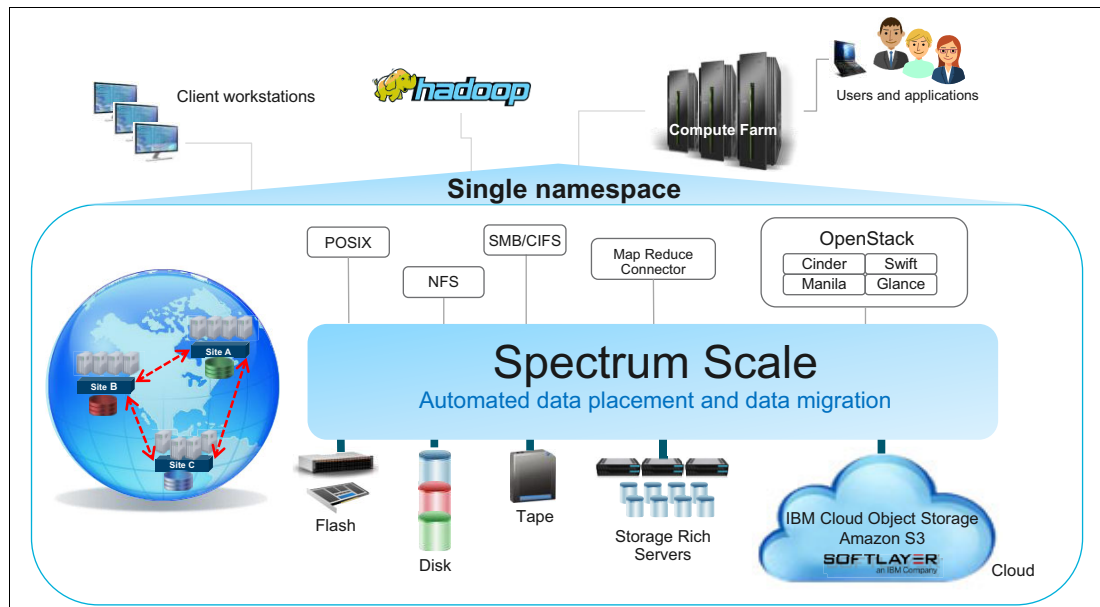

Figure 1-1   IBM Spectrum Scale overview

1.4.2 Use cases

This section describes the types of use cases and the potential users who can benefit from IBM Spectrum Scale Object Storage:

Data Lake   A unified file and object offering that provides a single place for all of your data storage needs.

Enterprise archiving
> Scalable, cost-effective, and active data archiving.

Backup   High-performance and high-capacity backup solution through file or object protocols.

Disaster recovery
> High-performance and high-capacity backup solution through file or object protocols.

Analytics and deep learning
> In-place analytics of object data by using the integrated high-performance HDFS connector.

Content storage
> Low-cost, high-performance, and high-capacity repository for unstructured data.

Content distribution
> Highly scalable data access with an integrated caching tier.

Video production
> High-performance streaming of multi-media content.

Worldwide collaboration
> Share data in a secure and scalable manner by using the S3 or Swift protocols. Object data can also be replicated across multiple sites to provide a highly available active-active cloud storage solution.

An unstructured data landing zone for the output of sensors, devices, and applications
> Support for multiple streams of data into a secure data repository for immediate analysis.

IBM Spectrum Scale enables a single unified namespace for data storage while also allowing data to be physically placed on different storage devices (IBM Spectrum Scale storage pools), which can range from solid-state drives (SSDs) and flash devices to external disk subsystems, internal disks within servers, and tape.

IBM Spectrum Scale Object Storage is targeted toward two types of potential users:

► Sites that already use IBM Spectrum Scale and are seeking to benefit from the enhanced features that are provided by S3 and Swift Object Storage protocols in a single data plane.

► Sites that are seeking an enterprise-ready, cost-efficient, and high-performing file and object solution. Users can start with a set of data access protocols and use additional or fewer protocols as their applications change over time.

IBM Spectrum Scale Object targets workloads that demand the following capabilities:

► Enterprise-ready security, data protection, and management features, such as encryption, snapshots, and backup to prevent data loss.

► Low-cost storage offering that supports commodity disks and tape.

- ► High throughput and scalable access to a virtually unlimited number of immutable objects, which includes seamless integration of SSDs, flash, and other fast storage devices with auto-tiering to low-cost storage.

- ► High capacity, dense storage.

- ► Capability of scaling from terabytes to hundreds of petabytes of data.

- ► Role-based authentication and authorization for administrators and users.

- ► Support for S3 and Swift Object Storage protocols.

- ► Active-active multi-site and disaster recovery support.

- ► High-speed multi-site replication.

- ► Unified file and object protocol access to a single or separate data set, that is, a single data lake that can ingest and access data by using various protocols and perform in-place analytics.

1.4.3 High-level architecture

IBM Spectrum Scale Object Storage combines the benefits of IBM Spectrum Scale with the best pieces of OpenStack Swift, the most widely used open source object store today, OpenStack Swift is bundled and managed as part of IBM Spectrum Scale Object Storage, hiding all the complexities that otherwise would be exposed in the raw open source project.

The OpenStack open source object storage project, OpenStack Swift, is emerging as a dominant object storage solution due to its extreme scalability, extensibility, and resilience. OpenStack Swift provides a robust object layer with an active community that is continuously adding innovative and new features. To ensure compatibility with the Swift packages over time, no code changes are required to either IBM Spectrum Scale or Swift to build the solution.

OpenStack Swift supports Swift and S3 APIs.

For more information about the Swift API, see this website.

For more information about the Amazon Simple Storage Services (S3) API, see this website.

An example of IBM Spectrum Scale Object Storage architecture is shown in Figure 1-2 on page 10. In this example, applications perform RESTful (HTTP) requests to the object store through (at least two, for high availability) IBM Spectrum Scale protocol nodes. All objects in this example are stored in an ESS.

For more information about ESS planning and service information, see IBM Knowledge Center and this website.

For more information about IBM Spectrum Scale RAID, see IBM Knowledge Center here and here.

In the example configuration in Figure 1-2 on page 10, the ESS Model GL6 solution has redundant servers and uses IBM Spectrum Scale RAID (8+2P) across six-hundred ninety-six 2, 4, 6, or 8 TB drives. With 8 TB drives, this configuration offers over 4.5 PB of accessible storage capacity with 174 disks per server and 80% storage capacity efficiency.

Applications can access objects by using the full 40 Gbps of the available network bandwidth with storage network bandwidth to spare.

Note: This solution is an example showing one of many possible configurations. Shared storage configurations that use ESS or a different storage controller can be used. Also, storage-rich server configurations that use Shared Nothing or File Placement Optimizer (FPO) configuration are possible. The requirements of your workload dictate which configuration is best for you.
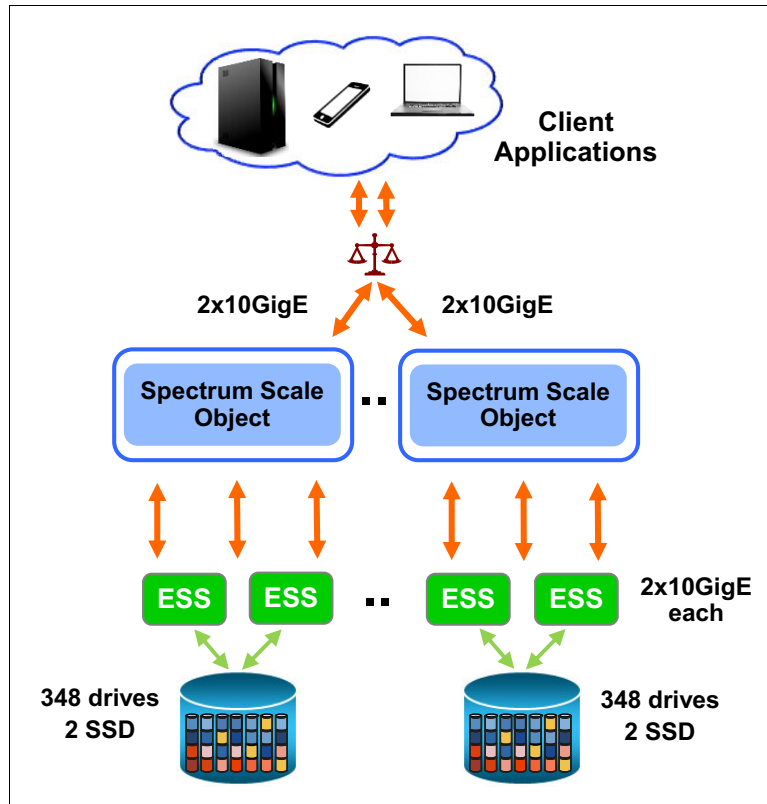
Figure 1-2   IBM Spectrum Scale Object Storage example that uses Elastic Storage Servers

Consider the following key points in the example architecture:

► Data protection is handled by the ESS, and IBM Spectrum Scale protocol nodes provide the front-end service that handles data access.

► Additional IBM Spectrum Scale protocol nodes provide both additional object access bandwidth and an additional level of fault tolerance in case of an IBM Spectrum Scale protocol node failure.

► Each IBM Spectrum Scale protocol node can independently access all objects.

► The only logical networks that are required are an application network that applications use to access the object store (IP-based) and storage network (IP-based or InfiniBand) that IBM Spectrum Scale uses to store data.

1.4.4 Benefits

As described in the example architecture in Figure 1-2, IBM Spectrum Scale Object Storage can provide an efficient storage solution for both cost and space, which is built on commodity parts that offer high throughput. The density and performance varies with each storage solution.

The following summary lists the general benefits of IBM Spectrum Scale Object Storage:

► Use of IBM Spectrum Scale data protection: Delegating the responsibility of protecting data to IBM Spectrum Scale and not using the Swift three-way replication or the relatively slow erasure coding increases the efficiency and the performance of the system in the following ways:

  – With IBM Spectrum Scale RAID, storage efficiency rises from 33% to up to 80%.

  – Disk failure recovery does not cause data to flow over the storage network. Recovery is handled transparently and with minimal impact on applications. For more information, see "GPFS-based implementation of a hyper-converged system for a software defined infrastructure" by Azagury, et al.

  – Applications now realize the full bandwidth of the storage network because IBM Spectrum Scale writes only a single copy of each object to the storage servers.

  – Increased maximum object size, up to a configurable value of 5 TB. With IBM Spectrum Scale data striping, large objects do not cause capacity imbalances or server hotspots, and they do not inefficiently use available network bandwidth.

  – No separate replication network is required to replicate data within a single cluster.

  – Capacity growth is seamless because the storage capacity can be increased without requiring rebalancing of the objects across the cluster.

  – IBM Spectrum Scale protocol nodes can be added or removed without failure requiring recovery operations or movement of data between nodes and disks.

► Integration of file and object in a single system: As described in IBM Spectrum Scale features, applications can store various application data in a single file system. For storage policies that have file access that is enabled, the same data can be accessed in place from both object and file interfaces so that there is no need to move data between storage pillars for different kinds of processing.

► Energy saving: High per-server storage density and efficient use of network resources reduces energy costs.

► Enterprise storage management features: IBM Spectrum Scale Object Storage leverages all IBM Spectrum Scale features, such as global namespace, compression, encryption, backup, disaster recovery, ILM (auto-tiering), tape integration, transparent cloud tiering (TCT), and remote caching.

1.4.5 Detailed architecture

A more detailed view of the architecture is shown in Figure 1-3. The IBM Spectrum Scale protocol nodes run the IBM Spectrum Scale client and all the Swift services. Clients that use the Swift or S3 API (users or applications) first obtain a token from the Keystone authorization service. The token is included in all requests that are made to the Swift proxy service, which verifies the token by comparing it with cached tokens or by contacting the authorization service. For more information, see this website.
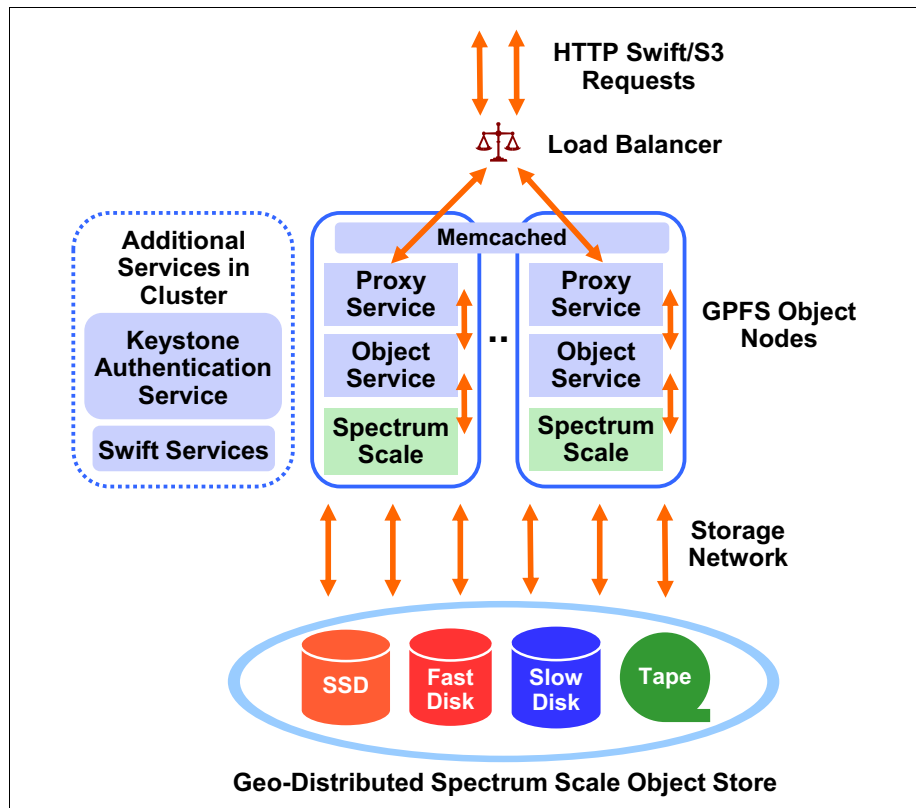
Figure 1-3   IBM Spectrum Scale Object Store architecture

As shown in Figure 1-3, all of the IBM Spectrum Scale protocol nodes are both active and provide a front end for the entire object store. The Load Balancer, which distributes HTTP requests across the IBM Spectrum Scale protocol nodes, can be based on software or hardware.

After applications are authenticated, they perform all object store operations, such as storing and retrieving objects and metadata, or listing account and container information through any of the proxy service daemons (possibly by using an HTTP Load Balancer, as shown in Figure 1-3). For object requests, the proxy service then contacts the object service for the object, which in turn performs file system-related operations to the IBM Spectrum Scale client. Account and container information requests are handled in a similar manner.

This paper focuses on the use of IBM Spectrum Scale shared storage architectures, such as the ESS or the combination of IBM Spectrum Scale NSD servers with a storage controller. IBM Spectrum Scale Object Storage can also be configured with storage-rich server clusters, including those that use FPO. The tuning guidelines that are provided in this paper do not apply to FPO configurations.

2

Planning for an IBM Spectrum Scale Unified File and Object Storage deployment

This chapter describes the key issues that must be addressed while deploying IBM Spectrum Scale Unified File and Object Storage. The purpose of this chapter is to help administrators understand the best configuration for their environment. This chapter also lists the various software configurations to consider while deploying IBM Spectrum Scale Unified File and Object Storage.

If you have questions or comments regarding the information in this chapter, contact IBM at scale@us.ibm.com. For current IBM Spectrum Scale (based on General Parallel File System) product support information or to open a service request, contact IBM Support at this website.

2.1 IBM Spectrum Scale architecture

In Release 4.1.1 and later, IBM Spectrum Scale includes the object protocol as an integral component, and one of the supported CES protocols. IBM Spectrum Scale deployment automation allows you to easily install an IBM Spectrum Scale cluster that includes object storage that is configured in an efficient and highly available way. The object storage features are a superset of OpenStack Swift Object Storage, with extensions built on the enterprise features of IBM Spectrum Scale. IBM Spectrum Scale provides performance and health monitoring capabilities to help you understand the health of the environment easily and a graphical user interface to manage your cluster, including the object protocol.

With IBM Spectrum Scale Object Storage, the physical storage architecture is independent, but affects the level of efficiency, fault tolerance, and performance. Because the storage also constitutes the bulk of storage costs, the choice of storage hardware also determines the cost and performance of the system. The current storage architecture focuses on shared storage deployments, which can be configured to cover a wide range of workload requirements (see Figure 1-2 on page 10).

In Version 4.1.1 and later, IBM Spectrum Scale includes the object protocol as an integral component, and one of the supported CES protocols. IBM Spectrum Scale deployment automation allows you to easily install an IBM Spectrum Scale cluster that includes object storage that is configured in an efficient and highly available way. The object storage features are a superset of OpenStack Swift Object Storage, with extensions built on the enterprise features of IBM Spectrum Scale. IBM Spectrum Scale provides performance and health monitoring capabilities to help you understand the health of the environment easily and a graphical user interface to manage your cluster, including the object protocol.

2.1.1 Networking

IBM Spectrum Scale Object Storage simplifies the network configuration of typical Swift deployments, but most of the same considerations must be made.

The client network that applications use to connect to IBM Spectrum Scale protocol nodes is typically a 1 or 10 GigE network with one or two network interface cards (NICs) that are installed on each server. Because requests are balanced across IBM Spectrum Scale protocol nodes, every server typically has the same network configuration. The network bandwidth of the client network can scale with additional protocol nodes and the network load balancer can be used to balance requests across the servers. Of course, scaling this network beyond the maximum bandwidth of the storage subsystem might not yield substantial benefits unless a significant amount of data is cached on the IBM Spectrum Scale protocol nodes.

The storage network connects the IBM Spectrum Scale protocol nodes to the storage subsystem. This network is typically 10 or 40 GigE or InfiniBand and is not visible to the protocol node clients. The storage network architecture layout must be carefully planned based upon the available physical storage bandwidth and must follow the guidelines of the chosen storage subsystem.

The Object services (proxy, account, and container) must be able to communicate with each other. To enable this communication, the client network is used with its CES IP addresses. CES IP addresses are assigned to all protocol nodes. Object services need constant network access between all the configured CES IP addresses. The standard configuration uses all the available CES IP addresses.

IBM Spectrum Scale can create node groups. If node groups are configured, CES IP addresses are assigned to those node groups. Depending on the physical network configuration, communication between the node groups might not be ensured. It is still possible to configure the object protocol for isolated node and network groups by selecting a node group that can be used to host the object services. For an explanation of the configuration of objects in such an environment, see IBM Knowledge Center.

Also, see 2.1.6, "High availability and CES monitoring" on page 18.

> **Note:** No additional Swift Replication Network is required because all object data protection management is handled by IBM Spectrum Scale.

2.1.2 Placement of object store components in IBM Spectrum Scale

Because IBM Spectrum Scale can store data for many different applications, object store data can be placed in a separate IBM Spectrum Scale management entity that is called an independent fileset, or it can be placed in multiple independent filesets in an IBM Spectrum Scale file system. By doing so, it allows IBM Spectrum Scale information lifecycle management (ILM), which includes snapshots and backups to uniquely identify and manage the object store data.

Objects

The location of the objects must be placed in the IBM Spectrum Scale storage pool that stores most of the data. Most IBM Spectrum Scale deployments have a single storage pool for data, but the fileset that contains the object data can be mapped to a separate storage pool to provide performance isolation. Object data with similar storage requirements can be grouped by using Swift storage policies and their associated filesets. For more information about how object data can be organized by policy for efficient storage management, see "Swift storage policies" on page 16.

Account and container

Store the account and container information in the same fileset as the object data to ease the management of the data by having all the data in a single data management entity. In cases where there is a very large number of objects per container, performance can be improved by grouping account and container data into a dedicated fileset that can be backed by fast storage, such as SSDs. For more information about configuring account and container data policies, see 5.4, "Account and container data placement policies" on page 43.

Snapshots and backup/restore

Snapshots and backups are critical tools to protect data against software and user errors and catastrophic hardware failures. Snapshots must be used for all critical data sets. By having account, container, and object information in a single independent fileset, IBM Spectrum Scale can then snapshot the entire object store without affecting any other workload that might be running against data in other filesets or file systems. After a snapshot is created, it can be backed up to an external storage pool, such as a tape, and then restored if required. When multiple filesets are used, more thought must be given on how to create a consistent set of snapshots. For suggestions about how to handle this scenario, see Chapter 7, "Data protection: Backup, restore, and disaster recovery" on page 49.

Swift storage policies

Using Swift storage policies, it is possible to link a container to a storage policy. All objects that are stored in this container are automatically treated as given by the storage policy function. IBM Spectrum Scale Object Storage enhances the storage policy function by providing unified file and object access, encryption (provided by the storage system), and compression functions. For more information about creating and administering storage policies, see IBM Knowledge Center.

For every object storage policy that is created, IBM Spectrum Scale updates the Swift configuration with a new object ring and creates its own independent fileset that is used to store object data for that policy. For more information, see 2.1.7, "IBM Spectrum Scale and Swift interaction" on page 19. Because each storage policy has a dedicated independent fileset, features such as compression and encryption can easily be applied to all of the objects in that policy (fileset). This in turn means that those additional filesets must be considered for snapshots and backup/restore.

2.1.3 Authentication

IBM Spectrum Scale Object Storage relies on the Keystone service for validating an incoming user that is associated with a request before processing the object access request. Keystone is the identity validation service that is used by different OpenStack services.

Figure 2-1 represents the high-level flow of the IBM Spectrum Scale object interface access.
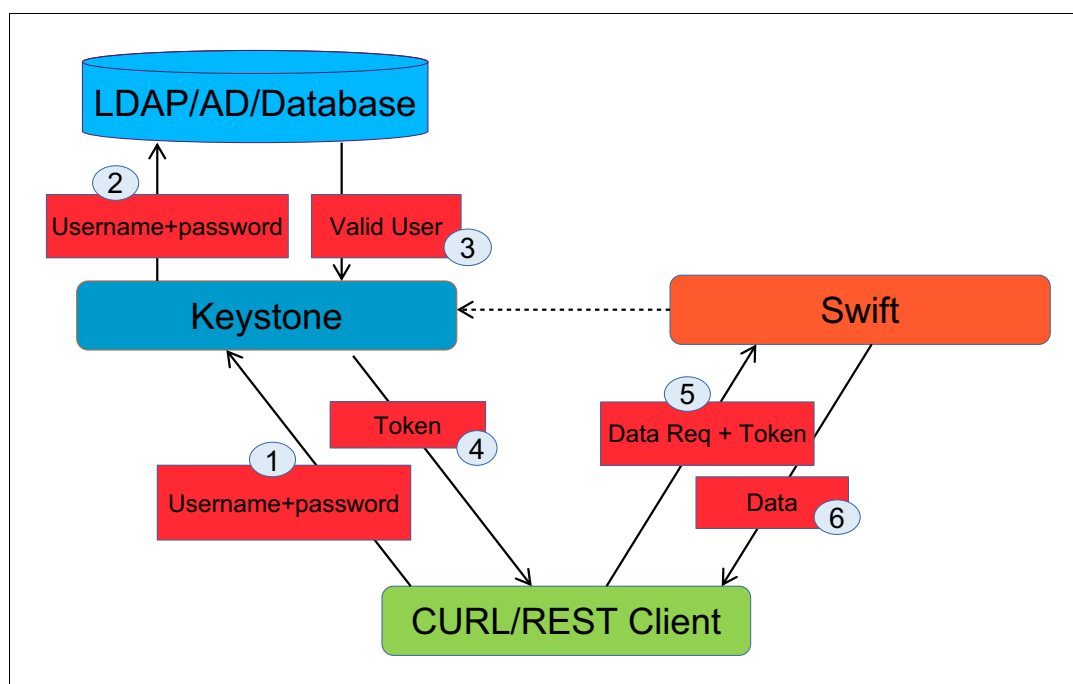

Figure 2-1   High-level flow of the IBM Spectrum Scale object interface access

IBM Spectrum Scale supports configuring Keystone with different identity back ends, such as:

► Microsoft Active Directory (AD) server
► Lightweight Directory Access Protocol (LDAP) server
► Postgres database

The Identity back end is the registry of user name and password. Before sending a request to the IBM Spectrum Scale object store, the user sends a request to the Keystone service to obtain a token that is required for accessing the object service. The request to the Keystone service contains the user name and password. The Keystone service validates the user name and password with the configured Identity back end. On successful validation of the user name and password, Keystone returns a token to the user. The obtained token is used for subsequent object store access requests.

If Keystone is configured outside of the IBM Spectrum Scale environment, IBM Spectrum Scale Object Storage can be configured with the same external Keystone server. The external Keystone setup is usually done for a multi-region setup or a setup in which different and advanced Keystone functions are required but not supported by Keystone, which is configured in IBM Spectrum Scale.

2.1.4 Data protection

IBM Spectrum Scale Object Storage relies on the storage subsystem for data protection and realizes the same level of protection against physical storage failures, such as RAID 5 and RAID 6. IBM Spectrum Scale can provide an additional level of protection by means of the following data protection features:

► Snapshots: These are point-in-time copies of the data in filesets that can also be stored in an external mass capacity storage system by using backups. Object data can be protected by taking snapshots and performing backup operations on object filesets.

► Active File Management (AFM)-based Asynchronous Disaster Recovery: IBM Spectrum Scale disaster recovery is based on AFM asynchronous DR that provides a solution for an IBM Spectrum Scale cluster to fail over to another IBM Spectrum Scale cluster and also to fail back again when the first cluster recovers. Setting up AFM asynchronous DR between object filesets across sites provides an additional level of data protection.

► Swift multiregion active-active replication: This feature provides active-active replication capability for object data across multiple sites. For more information, see Chapter 7, "Data protection: Backup, restore, and disaster recovery" on page 49.

2.1.5 Performance

The system design is crafted to ensure that it meets the following requirements:

► Individual object throughput: Each object is accessed through a single protocol node only. Therefore, the robustness, which includes maximum network throughput, number of CPUs, and system memory, of each node determines the performance of reading and writing objects (assuming that the storage subsystem can at least saturate a single node). In addition, for smaller objects, it is possible that an object is stored on a single storage device, in which the latency of the storage device might be the limiting performance factor.

► Aggregate object throughput: The overall performance is determined less by the capability of a single node or individual storage device than by other factors, such as the number of protocol nodes, the aggregate throughput of the storage subsystem, the client network bandwidth, and the runtime performance of the SQLite databases that are used to store account and container information.

► Container/account access performance: Swift uses `memcached` to improve the performance of retrieving account and container information by caching the data across the memory of the protocol nodes. In addition, the internal SQLite databases perform small I/O accesses to IBM Spectrum Scale to retrieve information. Increasing the amount of available memory on the servers improves account and container access performance.

In addition, several other factors affect performance:

► Tuning of IBM Spectrum Scale is important. Although more information about IBM Spectrum Scale tuning is provided in 4.3, "IBM Spectrum Scale installation and configuration" on page 31, a general description of tuning is available at this website.

► If performance is an issue, it is important to understand which other workloads are running against the same disk subsystem that might be reducing performance.

► As the number of objects in a container increases, the performance of the associated container database is impacted. This can cause longer delays for tasks that involve container database lookups or changes. In general, it is better to distribute a large number of objects among more containers rather than having a few containers with a large number of objects.

2.1.6 High availability and CES monitoring

To ensure the availability of IBM Spectrum Scale Object Storage, it is important to ensure a correct configuration that accounts for the level of fault tolerance that your use case requires.

The IBM Spectrum Scale Object Storage High Availability (HA) capability is built on the automatic network addresses reassignment function of the CES framework. The set of CES protocol nodes uses a specific network address pool of CES addresses. If one of the protocol nodes fails, the CES framework reassigns the CES addresses on the failing node to one or more of the remaining active nodes. Because the full set of CES addresses are always assigned among the protocol nodes, the object storage configuration does not have to be changed when a node fails. The CES framework moves addresses and restarts the required object services. Although the failover is automatic, clients of object storage might experience a temporary outage as the network addresses are moved from one node to another.

The account, container, and object ring layouts are each configured with 128 virtual devices. Each virtual device is assigned to a single CES address. Depending on the number of addresses in the CES pool, this often implies that many virtual devices across the ring layouts are assigned to the same CES address, and only available through the node that is configured to have that address. A larger number of protocol nodes implies that the object server workload on each node is less, and the impact of a node going down has less of an effect as the CES addresses are reassigned.

In addition to checking for failing nodes, the CES framework also monitors the underlying set of object storage services. If one of the services becomes unavailable, the CES monitoring facility detects the failure and attempts to restart the service. If the service cannot be successfully restarted, the CES framework reassigns the CES addresses on that node to one of the other protocol nodes. This ensures that the object services continue to be available even when a specific service on a node is not operating properly.

The HA capability of the embedded Keystone server in object storage is also managed by the CES framework. The Keystone server continues to be available even if the protocol nodes stop functioning. However, the HA capability of Keystone is not managed if the object protocol is configured to use an external Keystone service. The embedded Keystone server can be configured to use an external authentication service, or the Keystone server can be external to object storage. In these cases, it is expected that the HA capability is implemented by the administrator of those external services.

The availability of the IBM Spectrum Scale cluster must also be considered in the system design. For more information, see the following resources:

- *IBM Spectrum Scale Version 4 Release 2.2 Administration Guide*, SA23-1455, found here.
- *IBM Spectrum Scale Version 4 Release 2.2 Command and Programming Reference*, SA23-1456, found here.
- *IBM Spectrum Scale Version 4 Release 2.2 Concepts, Planning, and Installation Guide*, GA76-0441, found here.
- *Configuring GPFS for Reliability*, found here.

2.1.7 IBM Spectrum Scale and Swift interaction

Swift accesses data in an IBM Spectrum Scale file system by using the POSIX interface, but it places limits on the amount of time that it waits to store or retrieve information. Depending on the load of the system, these timeouts might need to be increased in the Swift configuration files.

As explained in 2.1.2, "Placement of object store components in IBM Spectrum Scale" on page 15, IBM Spectrum Scale creates an independent fileset for every object storage policy that is created. The Swift services can work with just one data path, which is configured in the Swift service configuration file. As the Swift code is unchanged, there is a need to link the root data path to each additional fileset data path. Symbolic links are used to create the link from the original root fileset data path to the newly created storage policy fileset. As the Swift services follow the link, requests can be served by the Swift services. For more information about how such links are created, see IBM Knowledge Center.

2.1.8 Unified file and object access

The unified file and object access capability is useful for use cases where the same data is accessed through the file and object interfaces. The data can be read from and written through either of the interfaces and the authentication modes govern the ownership and authorization of the data.

Data that is imported from the file interface is available for object access after the objectizer service has processed that data.

Figure 2-2 shows an overview of unified file and object access.
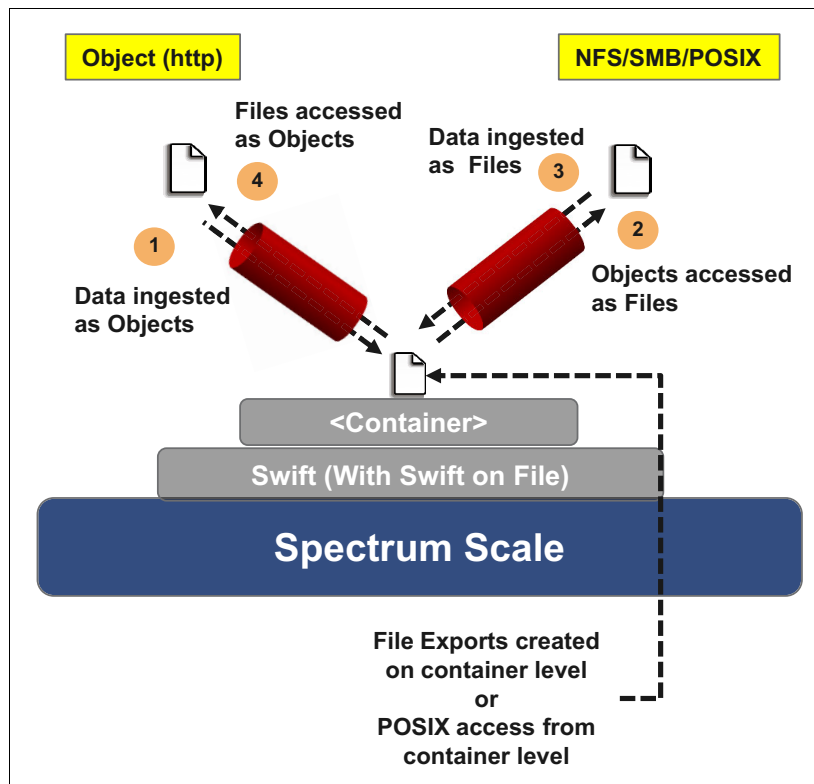
Figure 2-2   Unified file and object access

Containers that are created with a unified file and object access policy that are exposed as export points need appropriate ACLs set as required by SMB, NFS, and POSIX.

Identity management nodes

The identity management mode defines how the ownership of data is assigned and access is authorized.

The mode is represented by the `id_mgmt` configuration parameter in the `object-server-sof.conf` file:

```
id_mgmt = local_mode | unified_mode
```

You can list the currently configured `id_mgmt` mode by running the following command:

```
mmobj config list --ccrfile object-server-sof.conf --section DEFAULT --property id_mgmt
```

You can change `id_mgmt` in the `object-server-sof.conf` file by running the following command:

```
mmobj config change --ccrfile object-server-sof.conf --section DEFAULT
--property id_mgmt --value <mode>
```

For example, to set the identity management mode to `unified`, run the following command:

```
mmobj config change --ccrfile object-server-sof.conf --section DEFAULT
--property id_mgmt --value unified_mode
```

Unified file and object access consists of the following two modes:

► Local mode: Separate identity between object and file. This is the default mode. See Figure 2-3.

– Objects that are created or updated by using the object interface are owned by the Swift user. Applications processing the object data from file interface need the required file ACL to access the object data.

– To address this use case, the object authentication setup is independent of the file authentication setup. However, you can still set up the object and file authentication from a common authentication server in the case of AD or LDAP.
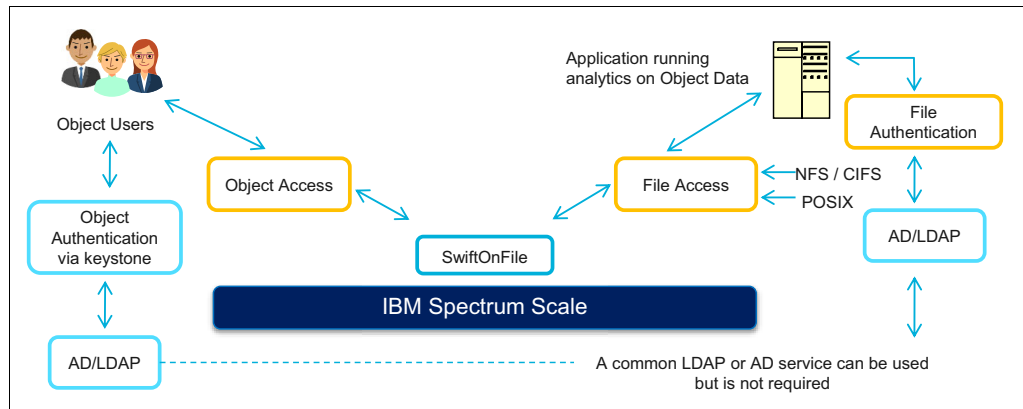
Figure 2-3   Local mode

► Unified mode: Shared identity between object and file. See Figure 2-4.

– Object that is created from the object interface is owned by the user performing the object PUT operation.

– Users from the object and file are expected to be common and from the same directory service (only AD+RFC 2307 or LDAP).
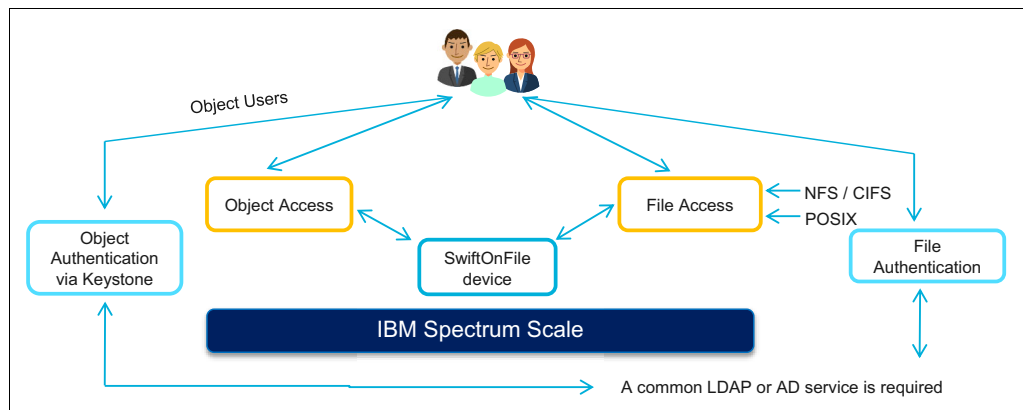
Figure 2-4   Unified mode

> **Note:** If your deployment uses only an SMB-based file interface for file access and file authentication is configured with AD with Automatic ID mapping, unified file and object access can be used if the object is configured with the same AD domain.

– If the object exists, the existing ownership of the corresponding file is retained if `retain_owner` is set to `yes` in `object-server-sof.conf`. For more information, see IBM Knowledge Center.

– Retaining ACL, extended attributes (xattrs), and Windows attributes (winattrs): If the object is created or updated over an existing file, the existing file ACL, xattrs, and winattrs are retained if `retain_acl`, `retain_xattr`, and `retain_winattr` are set to `yes` in `object-server-sof.conf`. For more information, see IBM Knowledge Center.

For more information, see IBM Knowledge Center.

Use cases

This section describes some of the use cases for IBM Spectrum Scale Object Storage when unified file and object access is enabled:

► In-place analytics for object data and publishing results by using the object interface. This feature enables data analytics of object data that is hosted on IBM Spectrum Scale, where you can leverage in-place object data analytics. IBM Spectrum Scale supports Hadoop connectors that you can use to run analytics on the object data that is accessible from the file interface and generates in-place results that are directly accessible from the object interface. This prevents any unnecessary movement of data across object interfaces and proves to be a suitable platform for object storage and integrated in-place analytics for the data that is hosted by it, as shown in Figure 2-5.
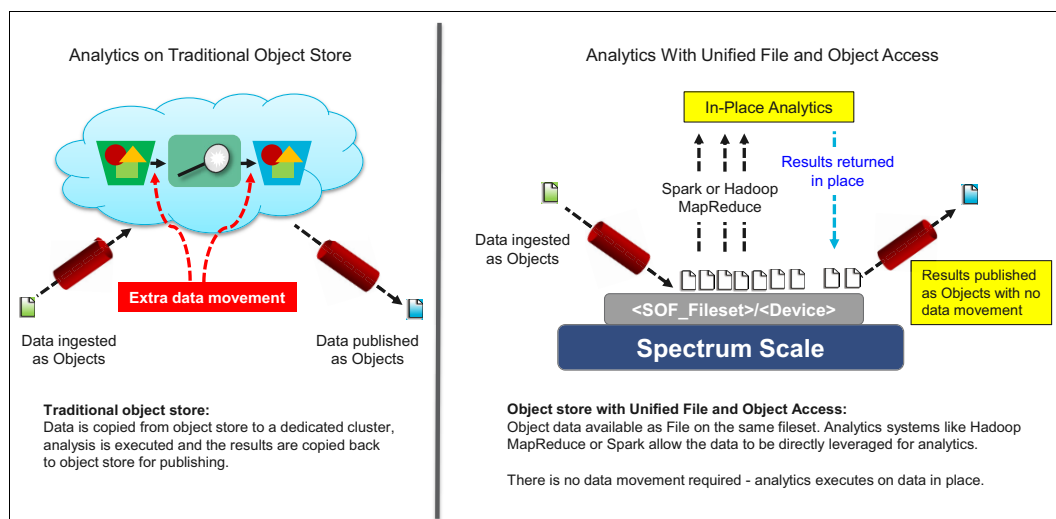
*Figure 2-5 Traditional object store versus unified file and object access*

► Data that is created from the file interface is accessible from the object interface after the objectization of the files.

► Import data by using the object interface, process object data with file-oriented applications, and, optionally, publish outcomes as objects.

► Users read and write data through file and object with common user authentication and identity.

For more information, see the OpenStack summit presentations that are found here, here, here, and here.

2.1.9 Multi-region support

In a multi-region configuration, IBM Spectrum Scale Object Storage provides a unified interface to the object data that is distributed over up to three independent CES protocol clusters. Each cluster is considered as a region, and each has a full replica of the object data-account and container databases and the actual objects themselves. This type of configuration can reduce latency for geographically distant clients by enabling them to access object data in a local region rather than having all clients access a single cluster. It can also provide an active-active disaster recovery solution because clients can access object data from a secondary region if their preferred region goes down.

Clients have the flexibility to select the region that they use. Each object store endpoint in Keystone has a region that is associated with it. In a multi-region configuration, there are multiple object store endpoints and the client selects the one with the appropriate region setting. Clients can be written to be fault-tolerant by first trying the endpoint in the preferred region and then trying the subsequent regions. If a region is unavailable or under high stress, the client can automatically try a different region.

The procedure to set up a multi-region configuration involves setting up an initial object storage deployment and then installing subsequent object storage deployments as additional regions. The ability to join two or more existing object storage deployments is not supported. There is no special requirement on the initial deployment. It is a typical deployment and can support multi-region later. The installation of subsequent object storage regions must have a multi-region data file that is obtained from the initial region. The `mmobj multiregion` command is used to produce the necessary data file for installation, and for performing other multi-region configuration tasks. A full overview of the multi-region capability, and the installation and configuration steps can be found in IBM Knowledge Center.

The multi-region capability is built by using the region support in OpenStack Swift. For more information about the underlying support, see the *OpenStack Swift Administration Guide*, found here.

2.1.10 S3 access

IBM Spectrum Scale Object Storage supports both Swift and Amazon S3 access to object data. The support for the S3 API is focused data access requests, such as GET, PUT, POST, and DELETE. The complete list of supported S3 commands is documented here.

Authentication of S3 requests uses a different set of credentials than normal Swift requests. You must generate these special credentials and use them when accessing the IBM Spectrum Scale Object Storage with the Amazon S3 API. For more information about creating the credentials that are used for S3 access, see IBM Knowledge Center.

2.1.11 Administration tools

The integration of OpenStack Swift with IBM Spectrum Scale includes commands for the management of Swift configuration and services and the management of specific object features. You can manage services across the cluster by using these commands from a single command line.

The `mmces` commands can be used to administer Swift configuration on all protocol nodes in an IBM Spectrum Scale cluster. You can modify Swift endpoint addresses by using the `mmces address` command, start and stop object services by using the `mmces service` command, and make node-level modifications by using the `mmces node` command.

For more information about the `mmces` command, see IBM Knowledge Center.

The `mmobj` commands can be used to modify the object protocol services configuration on all nodes in an IBM Spectrum Scale cluster. They can be used to configure and manage:

► Storage policies
► File access
► S3 access
► Multiregion

For information about the `mmobj` command, see IBM Knowledge Center.

For object administration tasks, see IBM Knowledge Center.

2.1.12 Tested configurations

The following list shows the software and the configurations and software that have been tested and shown to work:

► IBM Spectrum Scale Version 4.1.1 through 4.2.2
► RHEL7 on x86, Power BE, and Power LE
► Security-Enhanced Linux (SELinux) in enforcing, permissive, and disabled modes
► OpenStack Liberty with Swift Version 2.5
► Keystone authentication
► Maximum object size: 5 TB
► Shared storage configurations, including ESS, NSD servers, or traditional block controllers
► Shared nothing storage configuration with up to 32 nodes
► Single IBM Spectrum Scale cluster
► Active-active object storage, with up to three IBM Spectrum Scale clusters that use Swift multi-region
► Active-passive disaster recovery enabled object storage with AFM DR

3

IBM Spectrum Scale Unified File and Object Storage configuration settngs overview

This chapter summarizes the configuration settings to optimize Swift while taking advantage of IBM Spectrum Scale features wherever appropriate.

The setup requires no changes to the Swift software. Optimization is provided through the configuration settings that are made when creating Swift rings and also in Swift configuration files.

3.1 Swift replication

Object storage does not replicate data within a region. The combination of having all ring devices available on all nodes and the dynamic network address management of CES ensures that all ring devices are always accessible. Data protection of the underlying file system can be provided through the IBM Spectrum Scale RAID capabilities, storage controller RAID, or IBM Spectrum Scale replication.

A standard OpenStack Swift deployment depends on replication because the deployment is made up of disparate nodes with private storage. The loss of a node means that the ring devices and the object data that are contained on that node are no longer available. A standard Swift deployment replicates the data so that it can get it from another node if the primary is unavailable. Object storage is built upon IBM Spectrum Scale and the CES framework. Therefore, it does not need replication to ensure data access.

However, replication is required for object storage in a multi-region environment. Each region has one copy of the data. If there are two or three regions, there will be two or three copies of the data respectively. However, the data is not duplicated within a region.

3.2 Swift rings and storage policies

Every IBM Spectrum Scale protocol node can access all of the data in the IBM Spectrum Scale shared file system that is used for object storage. Therefore, all of the rings are constructed by using 128 virtual devices rather than physical devices. The virtual devices are subdirectories in the IBM Spectrum Scale Object Storage fileset that is created for Swift data. The network addresses that are assigned to the ring devices are from the set of addresses in the CES network address pool, not the main network address of the physical node. The CES addresses are dynamically assigned to the active CES nodes. Therefore, each virtual ring device must always be available as the number of the CES nodes changes. To balance the load among all the CES nodes, all CES network addresses are used in the ring layout.

The Swift subsystem supports creating storage policies that consist of additional object rings that are made of different devices. Object storage leverages this function to enable storage policies that are built upon separate filesets. The storage policy can be used for additional object storage and providing enhanced functions, such as compression, encryption, and file-level object access.

A simple storage policy can be created by running the following command:

```
mmobj policy create gold
```

This command creates a fileset with default values for the storage policy that is named gold. The new ring files are distributed to the protocol nodes and the necessary object services are automatically restarted. Containers can then be assigned to the new policy by passing the policy name in the X-Storage-Policy header when the container is created:

```
X-Storage-Policy: gold
```

This causes the policy identifier to be stored with the container. Objects that are uploaded into the container are stored in the fileset that is associated with the policy.

The storage policies are created and managed by using the `mmobj policy` command, which is described in more detail in IBM Knowledge Center.

For more information about the underlying Swift storage policy function, see this website.

3.3 Swift services

Every IBM Spectrum Scale protocol node can access every virtual device in the shared file system. Because of this, some Swift object services can be optimized to run from a single IBM Spectrum Scale protocol node (see 2.1.1, "Networking" on page 14):

► The swift-object-updater is responsible for updating container listings with objects that were not successfully added to the container when they were initially created, updated, or deleted.

► If object expiration is used, that is, the Swift client uses the X-Delete-At or X-Delete-After headers during an object PUT or POST, the swift-object-expirer ensures a scheduled deletion of objects. The service detects expired objects and marks them for deletion.

► In contrast, some services, that typically run on a single IBM Spectrum Scale protocol node are enhanced to run on every IBM Spectrum Scale protocol node by detecting and using IP address to device mapping in the Ring files for which virtual devices belong to the node hosted IP addresses. It is possible to run the services on each IBM Spectrum Scale protocol node, which spreads the load to all nodes and eases service monitoring. The services work with only the data in the virtual devices that belong to the CES IPs that are hosted on that node.

► Even though objects are not replicated by Swift, the swift-object-replicator runs to periodically clean up tombstone files from deleted objects. It manages the cleanup for all to the hosted IP addresses mapped virtual devices.

► Both openstack-swift-account-auditor and openstack-swift-container-auditor ensure the integrity of the account and container databases by running queries on the database that are found in the account and container data paths to ensure that all entries are valid.

► The openstack-swift-container-updater ensures that account databases are up-to-date with the current existing container databases that are found in the container data paths.

> **Note:** The swift-object-auditor compares the checksum in the object file's extended attributes with the checksum of the object data on disk. If there is a discrepancy, the object file is moved to the quarantine directory of Swift, with the expectation that the object-replicator will eventually replace the quarantined object with a replica object instance.

With IBM Spectrum Scale Object Storage, it is better to use storage controller capabilities, such as IBM Spectrum Scale RAID checksums and disk scrubbing and auditing capabilities. For this reason, the swift-object-auditor service is not run on any node in single region deployments.

If the Swift multi-region function is used, the swift-object-auditor is started to ensure the integrity of the object across clusters.

Table 3-1 lists each of the Swift services and the set of IBM Spectrum Scale protocol nodes on which they must be run.

Table 3-1   Swift services

Object service	IBM Spectrum Scale protocol node
openstack-swift-account	All
openstack-swift-account-auditor	All
openstack-swift-account-reaper	All
openstack-swift-account-replicator	All
openstack-swift-container	All
openstack-swift-container-auditor	All
openstack-swift-container-updater	All
openstack-swift-container-replicator	All
openstack-swift-object	All
openstack-swift-object-auditor	object_singleton_node[a]
openstack-swift-object-expirer	object_singleton_node
openstack-swift-object-replicator	All
openstack-swift-object-updater	object_singleton_node
openstack-swift-proxy	All
memcached	All
openstack-swift-object-sof	All[b]
openstack-keystone	All[c,d]
postgresql-obj	object_database_node[c]

a. If multi-region object deployment is enabled.
b. If unified file and object access is enabled.
c. If local OpenStack Keystone Identity Service is configured.
d. Updated to httpd (Keystone) on all nodes, if using local authentication.

IBM Spectrum Scale Unified File and Object Storage installation

This chapter provides instructions for the IBM Spectrum Scale Unified File and Object Storage installation process. For information about ports that are required by IBM Spectrum Scale services, see IBM Knowledge Center.

For more information about firewall recommendations for protocol access, see IBM Knowledge Center.

4.1 Installation overview

There are two methods to install IBM Spectrum Scale Object Storage in an existing IBM Spectrum Scale environment:

► The installation toolkit:

The spectrumscale installation toolkit automates the steps that are required to install IBM Spectrum Scale, deploy protocols, and install updates and patches.

For information about deploying IBM Spectrum Scale object by using the spectrumscale installation toolkit, see IBM Knowledge Center.

► Manual:

IBM Spectrum Scale Object Storage is typically installed by using the spectrumscale installation toolkit. If you do not want to use the spectrumscale installation toolkit, you can manually install IBM Spectrum Scale for object storage. The IBM Spectrum Scale self-extracting package consists of all the required packages for IBM Spectrum Scale object manual installation.

For information about deploying IBM Spectrum Scale object manually, see IBM Knowledge Center.

Installation flow

The steps in the installation process are listed here to provide an overview. The details of each step are described in 4.4, "Object software installation and configuration" on page 32.

1. Ensure that the prerequisites that are defined in 4.2, "Installation prerequisites" on page 30 are satisfied.

2. Extract the IBM Spectrum Scale self-extracting package.

3. Record the cluster information and the IBM Spectrum Scale object store configuration by using the installer toolkit.

4. Perform the deployment by using the installer toolkit.

Note: The installer can also be used to add or remove object nodes. This process is described in 5.2, "Adding a protocol node" on page 42 and 5.3, "Removing a protocol node" on page 42.

4.2 Installation prerequisites

There are several prerequisites for installing IBM Spectrum Scale Object Storage by using the Installer toolkit.

SSH and network setup

This is a general IBM Spectrum Scale prerequisite, and not specific to protocols other than protocol-specific ports that must be open on the firewall.

The node that is used for initiating the installation of the object protocols by using the installation toolkit must be able to communicate through an internal or external network with all protocol nodes to be installed. All nodes also require SSH keys to be set up so that the installation toolkit can run remote commands without any prompts. Examples of prompts include a prompt asking for a remote node password or a prompt asking a yes-or-no question.

No prompts should exist while using SSH among any cluster nodes to and from each other, and to and from the node designated for installation.

Repository setup

This is a general IBM Spectrum Scale prerequisite, and not specific to protocols.

Important: You must disable all configured EPEL repositories on all nodes where IBM Spectrum Scale Object Storage software will be installed before proceeding with the deployment. The spectrumscale installation toolkit contains all the necessary code for installation. However, there might be base RHEL 7.x RPMs required as prerequisites. To satisfy any prerequisite, it might be necessary for your RHEL 7.x nodes to have access to a DVD repository (or to an external repository accessible by network).

NTP setup

Configure Network Time Protocol (NTP) on all nodes in the system to ensure that the clocks of all of the nodes are synchronized. Clocks that are not synchronized cause debugging issues and authentication problems with the protocols.

Supported OS versions for object deployment

IBM Spectrum Scale Object Storage can be deployed only on RHEL 7.0, RHEL 7.1, RHEL 7.2, and RHEL 7.3.

Detailed prerequisites for IBM Spectrum Scale of object store are documented in IBM Knowledge Center.

4.3 IBM Spectrum Scale installation and configuration

IBM Spectrum Scale must be installed on every object storage node in the system. For more information, see IBM Knowledge Center.

Note: The IBM Spectrum Scale software is not required on the node that hosts the Keystone identity service if it is deployed on a separate node from the object storage nodes.

An IBM Spectrum Scale cluster and file system are required. If this infrastructure does not exist, you must install the IBM Spectrum Scale software and create an IBM Spectrum Scale cluster by using the spectrumscale installation toolkit or IBM Spectrum Scale commands. Then, create an IBM Spectrum Scale file system and mount the IBM Spectrum Scale file system on all protocol nodes. The spectrumscale installation toolkit creates an independent fileset in the file system that you name. This independent fileset is used only for Swift object storage.

4.4 Object software installation and configuration

After you have ensured that all of the prerequisites that are described in 4.2, "Installation prerequisites" on page 30 and 4.3, "IBM Spectrum Scale installation and configuration" on page 31 are met, you can begin the installation. Use the example in the following sections to start the installation.

Use this information to deploy the object protocol in an IBM Spectrum Scale cluster by using the spectrumscale installation toolkit.

Setting up a node that starts the installation

Setup is necessary unless `spectrumscale setup` has been run on this node for a past IBM Spectrum Scale installation or protocol deployment. Select an IP existing on this node that is accessible to and from all nodes by using promptless SSH:

```
./spectrumscale setup -s IP
```

If the `spectrumscale` command is run from a location outside of any of the nodes to be installed, an IBM Spectrum Scale admin node is required:

```
./spectrumscale node add hostname -a
```

Adding protocol nodes

Deployment of protocol services is performed on a subset of the cluster nodes that are designated as protocol nodes. Add protocol nodes by using the following commands:

```
./spectrumscale node add hostname -p
./spectrumscale node add hostname -p
…..
```

Protocol nodes have an additional set of packages that is installed that allow them to run the object protocol services.

Assigning protocol IPs (CES IPs)

Data is served through these protocols from a pool of addresses that is designated as Export IP addresses or CES public IP addresses by using the following command:

```
./spectrumscale config protocols -e IP1, IP2, IP3…
```

Verify that the file system mount points are as expected by running the following command:

```
./spectrumscale filesystem list
```

> **Note:** Skip this step if you have set up file systems and NSDs manually and not through the installation toolkit.

Configuring protocols to point to a shared root file system location

A `ces` directory is automatically created at the root of the specified file system mount point. It is used for protocol administration and configuration, and needs at least 4 GB of memory. Upon completion of object protocol deployment, the IBM Spectrum Scale configuration points to this as `cesSharedRoot`. The `cesSharedRoot` should be a separate file system. The **spectrumscale config protocols** command can be used to define the shared file system (**-f**) and mount point (**-m**):

```
./spectrumscale config protocols -f fs1 -m /ibm/fs1
```

Enabling the object protocol

Enable the object protocol on your IBM Spectrum Scale cluster by running the following command:

```
./spectrumscale enable object
```

Configuring the object protocol

Complete the following steps:

1. Configure the admin user password and a database password to be used for object operations by running the following command:

    ```
    ./spectrumscale config object -au admin -ap -dp
    ```

2. Configure the object endpoint by using a single host name that clients use to access object storage. This can be a round-robin DNS entry mapping to all CES IPs, or a load balancer virtual host name for object storage. Run the following command:

    ```
    ./spectrumscale config object -e hostname
    ```

3. Specify a file system and fileset name where your object data goes by running the following commands:

    ```
    ./spectrumscale config object -f fs1 -m /ibm/fs1
    ./spectrumscale config object -o Object_Fileset
    ```

Setting up authentication

Authentication must be set up before using the object protocol. If you are unsure of the appropriate authentication configuration, you can skip this step and revisit it by rerunning the deployment at a later time or manually by running the **mmuserauth** commands.

Install the Toolkit AD example for object by running the following command:

```
./spectrumscale auth object ad
```

Reviewing your configuration

Review the list of nodes to be configured and then perform the precheck operation by running the following command:

```
./spectrumscale node list
./spectrumscale deploy -precheck
```

Starting the deployment

Run the following command to deploy object protocols:

```
./spectrumscale deploy
```

For more information about object protocol installation and configuration, see IBM Knowledge Center.

4.5 Postinstallation

Installation can be verified after the IBM Spectrum Scale cluster is installed. This section also describes various tuning considerations for the cluster after the installation.

4.5.1 Verifying the installation

This section describes the installation verification steps.

Basic verification

Run the basic Swift commands on a protocol node to verify object installation. If the installation completes successfully, you can list all containers, upload a sample object to a container, and list that container and see the object. For steps about object installation basic verification, see IBM Knowledge Center.

Verifying unified file and object access

Enable the file-access capability and perform verification steps for unified file and object access by performing the following steps:

1. Enable file access by running the following command:

   ```
   mmobj file-access enable
   ```

2. Verify that the unified file and object services are running by running the following commands:

   ```
   mmces service list -v  | grep "openstack-swift-object-sof"
   OBJ:openstack-swift-object-sof is running
   mmces service list -v  -a | grep "ibmobjectizer"
   srnode3:  OBJ:ibmobjectizer is running
   ```

3. Create a storage policy for file access by running the following command:

   ```
   mmobj policy create unifiedpolicy --enable-file-access
   ```

4. Create and associate a container to this storage policy by running the following commands:

   ```
   source ~/openrc
   swift post container1 --header
   "X-Storage-Policy: unifiedpolicy"
   ```

5. Upload an object into this container by running the following command:

   ```
   swift upload container1 imageA.JPG
   ```

6. Verify that the object can be accessed from the file interface by running the following commands:

   ```
   Find the path created for the container by finding it in the newly created
   fileset:
   export FILE_EXPORT_PATH=`find
   /ibm/gpfs0/obj_unifiedpolicy/
   -name "container1"`
   # echo $FILE_EXPORT_PATH
   /ibm/gpfs0/obj_unifiedpolicy/s10041510210z1device1/
   AUTH_09271462d54b472c82adecff17217586/container1
   ```

Objects that are created under container1 must be available in this path as files and accessible by using the POSIX interface. They should also be accessible by using NFS and SMB interfaces. For more information about how to access them by using the NFS and SMB interface, see IBM Knowledge Center.

For more information about creating and using a unified file and object storage policy, see IBM Knowledge Center.

4.5.2 Swift tuning parameters

This section describes the most commonly used tuning parameters for the OpenStack Swift services.

Worker threads

You might have to modify the value of the `Workers` parameter in Swift server configurations. This parameter is set to `auto` for a proxy server. With the `auto` setting, the worker count is automatically set to the number of cores that are detected on the node where the proxy server is running.

In IBM Spectrum Scale, the `Workers` parameter is set by default to `3` in the object server and `2` in the account and container server configuration files. Depending on the memory and number of cores in your protocol nodes, you most likely want to increase these settings.

These services should be set to a percentage of the number of cores on your protocol nodes:

► The object server should be set to 75% of the core count.

► The container server should be set to 50% of the core count.

► The account server should set to 25% of the core count.

To increase these settings, modify the account, container, and object server configuration by running **mmobj config change** (The values that are given here are examples. Adjust these values to match your configuration.):

```
mmobj config change --ccrfile object-server.conf --section DEFAULT --property
workers --value 8
mmobj config change --ccrfile container-server.conf --section DEFAULT --property
workers --value 4
mmobj config change --ccrfile account-server.conf --section DEFAULT --property
workers --value 4
```

For more information, see IBM Knowledge Center.

Node and connection timeouts

The spectrumscale installation toolkit sets two important Swift timeout values in the proxy server configuration file. Both **node_timeout** and **conn_timeout** are set to 120 seconds to minimize timeouts during periods of heavy load. These values can be changed depending upon the workload type by running the following commands:

```
mmobj config change --ccrfile proxy-server.conf --section app:proxy-server
--property node_timeout --value 180
mmobj config change --ccrfile proxy-server.conf --section app:proxy-server
--property conn_timeout --value 120
```

Objectizer service interval

The default interval between the completion of a unified file and object objectizer cycle and the starting of the next cycle is 30 minutes. However, this interval must be properly planned based on the following items:

► The frequency and the number of new file ingestions that you are expecting to be objectized.

► The number of protocol nodes you have deployed.

► How quickly you need the imported files to be objectized.

For more information and instructions about how to set this interval, see IBM Knowledge Center.

Setting quality of service on ibmobjectizer

The periodic scans that are run by the ibmobjectizer service are resource-intensive and might affect the object IO performance. Quality of service (QoS) can be set on the ibmobjectizer service depending upon the I/O workload and the priority at which the ibmobjectizer service must be run.

For more information and instructions about how to set QoS, see IBM Knowledge Center.

4.5.3 IBM Spectrum Scale tuning parameters for object storage

This section describes general IBM Spectrum Scale tuning and also includes commands that are related to tuning. A comprehensive description of IBM Spectrum Scale tuning can be found at this website.

Setting the number of inodes

The IBM Spectrum Scale default maximum number of inodes is acceptable for most configurations, but for large systems that expect hundreds of millions of objects or more, the maximum number of inodes set at fileset creation time might have to be increased. The maximum number of inodes must be larger than the sum of the maximum number of expected objects and the inodes that are required by the Swift directory structure.

The maximum number of inodes must be set to at least 2.5 times the maximum number of expected objects. So, for example, if the expected number of objects is 100 million, the maximum number of inodes must be set to 250 million.

The installer sets the maximum number of inodes to 8 million by default, and all of them are preallocated.

To modify the maximum inode limit for a fileset, run the following command:

```
mmchfileset FileSystem Fileset --inode-limit
MaxNumInodes[:NumInodesToPreallocate]
```

Note: Specifying a relatively large number of pre-allocated inodes improves import performance, but overallocating inodes can waste metadata space unnecessarily.

IBM Spectrum Scale file system cache

The IBM Spectrum Scale pagepool caches file system information in the memory for fast access (similar to the Linux Page Cache). Run `mmchconfig` to set this IBM Spectrum Scale pagepool to be at least 30 - 50 percent of the memory on each object storage node.

maxFilesToCache

This parameter limits the total number of different files that can be cached at one time. It must be set by running `mmchconfig` and be large enough to handle the number of concurrently open files (objects) and allow the caching of recently used files.

maxStatCache

This parameter sets aside additional pageable memory to cache attributes of files that are not currently in the regular file cache. Increasing this value can improve the performance of background Swift services that scan directories.

> **Note:** If you change the `maxFilesToCache` value but not the `maxStatCache` value, the `maxStatCache` value defaults to four times the `maxFilesToCache` value.

seqDiscardThreshold

The default value of this parameter is 1 MB. Therefore, if you have a file that is sequentially read and is greater than 1 MB, IBM Spectrum Scale does not keep the data in the cache after consumption. Because everything in Swift is read sequentially, unless this value is increased, IBM Spectrum Scale does not cache anything. Therefore, this value must be increased to the largest expected object size that must be cached in memory.

tokenMemLimit

This parameter sets the size of the memory that is available for the manager nodes to use for caching tokens. The parameter controls the allocation of memory that is separate from the pagepool. If the number of objects increases, or if high periods of activity are expected, increase the amount of memory to cache tokens to 2 GB or larger.

workerThreads

Controls an integrated group of variables that tune the file system performance. Use this variable to tune file systems in environments that are capable of high sequential or random read/write workloads or small-file activity. To support periods of high activity, increase this value to `1024`. The `workerThreads` parameter can be reduced without having to restart the GPFS daemon. Increasing the value of `workerThreads` requires a restart of the GPFS daemon.

4.5.4 Removing the object protocol

Disabling the object service disables the object protocol and discards the OpenStack Swift configuration and ring files from the CES cluster. If the OpenStack Keystone configuration is configured locally, disabling object storage also discards the keystone configuration and database files from the CES cluster. However, to avoid accidental data loss, the associated filesets that are used for the object data are not automatically removed during disable. The filesets for the object data and any fileset that are created for optional object storage policies must be removed manually. If you enable the object service later, either different fileset names must be specified or the existing filesets must be cleaned up.

Additionally, because Swift configuration data, such as configuration and ring files, is needed to recover from an accidental object service disable, the configuration data is saved in a backup that is in the original object fileset under the `/objbackup` directory. This is also true for the keystone service configuration data and the related keystone database.

To disable the object protocol, complete the following steps:

1. Remove the object authentication by running the following commands:

```
mmuserauth service remove --data-access-method object
mmuserauth service remove --data-access-method object --idmapdelete
```

2. Disable the object protocol by running the following command:

```
mmces service disable obj
```

3. Unlink and remove the base object fileset and filesets that are created for storage policies by using the following commands for each fileset:

```
munlinkfileset fs1 object_fileset
mmdelfileset fs1 object_fileset -f
```

For more information about the cleanup procedure, see IBM Knowledge Center.

System administration considerations

This chapter describes Swift administrative actions that behave differently in an IBM Spectrum Scale Unified File and Object Storage environment, and the operational considerations that we encountered during testing.

One of the benefits of using a clustered file system for Swift storage is simplified maintenance. Nodes can be added or replaced with minimal effort and with no need for the rebalancing or movement of data. This chapter describes the process for adding and removing an IBM Spectrum Scale protocol node in your system.

For performance reasons, it is preferable to place account and container data in a higher performing storage pool. This chapter describes the procedure for this placement by using the ILM placement policies.

Finally, this chapter provides suggestions for several other system administration areas.

5.1 Managing the Swift services

This section describes using commands to manage the Swift services.

The mmces service command

IBM Spectrum Scale uses the `mmces service` command to start, stop, and list (status) Object services on all protocol nodes.

To start the object service on all nodes, run the following command from any IBM Spectrum Scale node:

```
mmces service start OBJ -a
```

You can specify a particular node by using the **-N <Node>** option rather than **-a** to start object services on that particular node.

To stop the object service on all nodes, run the following command from any IBM Spectrum Scale node:

```
mmces service stop OBJ -a
```

You can specify a particular node by using the **-N <Node>** option rather than **-a** to stop object services on that particular node.

To list the status of the object service, run the following command from any IBM Spectrum Scale node:

```
mmces service list -a
```

You can specify a particular node by using the **-N <Node**> option rather than **-a** to list the status of object services on that particular node. Option **-v** lists the object services and their status.

The mmobj command

IBM Spectrum Scale uses the `mmobj` command to manage the configuration of the object protocol service and administers storage policies for object storage, unified file and object access, and multi-region object deployment. Here are some examples:

► To list the storage policies, run the following command:

```
mmobj policy list
```

► To list object configuration settings for proxy-server.conf, DEFAULT section, run the following command:

```
mmobj config list --ccrfile proxy-server.conf --section DEFAULT
The system displays output similar to this:
[DEFAULT]
bind_port = 8080
workers = auto
user = swift
log_level = ERROR
```

► To create a storage policy with compression that is enabled for all objects belonging to this policy, run the following command:

```
mmobj policy create CompressionTest --enable-compression
```

The system displays output similar to the following output:

```
[I] Getting latest configuration from ccr
[I] Creating fileset /dev/gpfs0:obj_CompressionTest
[I] Creating new unique index and build the object rings
[I] Updating the configuration
[I] Uploading the changed configuration
```

For more information about all the options of the `mmobj` command, see IBM Knowledge Center.

The mmuserauth command

IBM Spectrum Scale uses the `mmuserauth` command to manage the authentication configuration of the object protocol service.

The `mmuserauth` command is used to create, list, verify, and remove the object authentication configuration.

The object protocol can be configured with one of the following authentication methods:

▶ local: Postgres running on IBM Spectrum Scale is used for storing user identities.

▶ ldap: Keystone is configured to use an external LDAP server as the user identity back end.

▶ ad: Keystone is configured to use an external Active Directory server as the user identity back end.

▶ userdefined: The IBM Spectrum Scale object protocol is configured with Keystone running outside of the IBM Spectrum Scale environment.

Here are some examples:

▶ To configure local authentication for object access, run the following command:

```
mmuserauth service create --data-access-method object --type local
--ks-admin-user admin --ks-admin-pwd *******
```

The system displays an output similar to this one:

```
Object configuration with local (Database) as identity backend is completed
successfully.
Object Authentication configuration completed successfully.
```

▶ To list the authentication configuration, run the `mmuserauth service list` command as shown in the following example:

```
mmuserauth service list -data-access-method object
```

The system displays output similar to this one:

```
OBJECT access configuration : LOCAL
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_KS_SSL           false
ENABLE_KS_CASIGNING     false
KS_ADMIN_USER           admin
```

For more information about the `mmuserauth` command, see IBM Knowledge Center.

Configuring Keystone for HTTPS

The Keystone can be configured with HTTPS (SSL) for increased security. The communication between the object/Keystone client and Keystone server will be over HTTPS (SSL) in this case.

IBM Spectrum Scale can configure the communication between the Keystone server and the Identity server (Active Directory (AD) and LDAP) over TLS.

Figure 5-1 shows the various ways to configure the object authentication and the various security aspects that are involved.
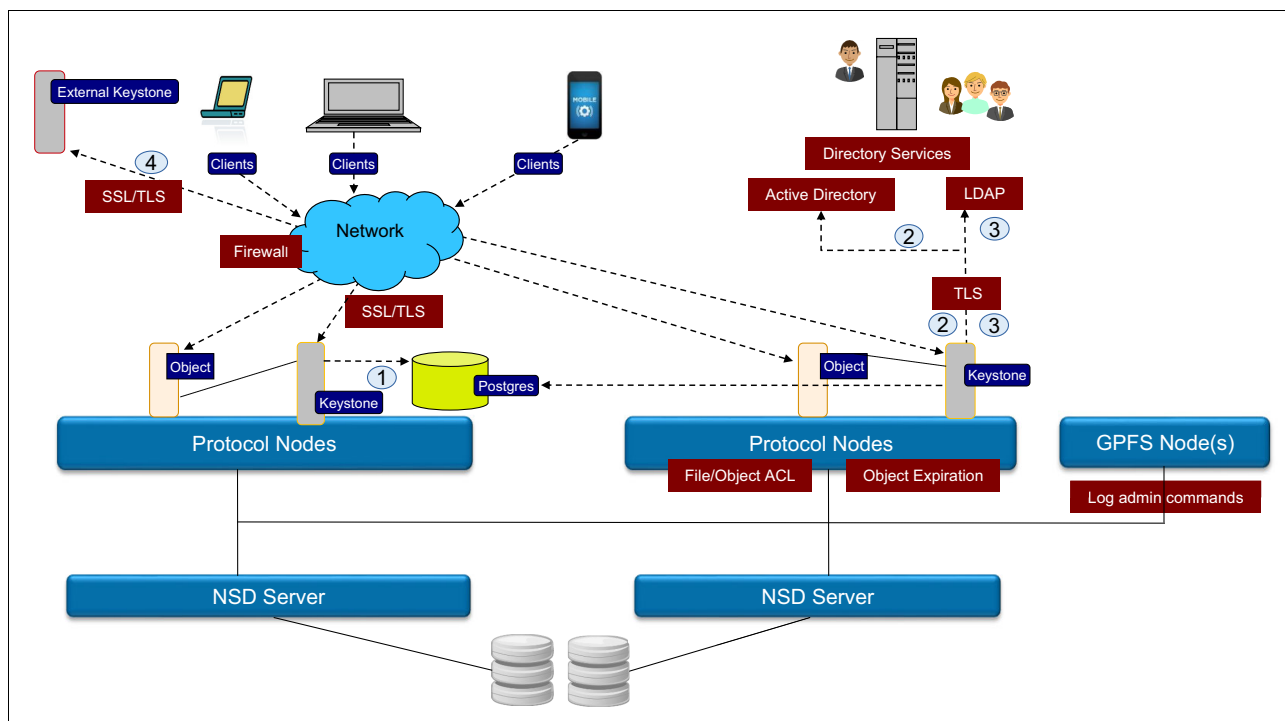
Figure 5-1   Configuring the object authentication

5.2 Adding a protocol node

Adding a protocol node to an existing IBM Spectrum Scale cluster can be achieved manually or by using the spectrumscale installer toolkit. It is preferable to use the latter.

To add a protocol node to an existing IBM Spectrum Scale cluster, see IBM Knowledge Center.

5.3 Removing a protocol node

To delete a protocol node from an existing IBM Spectrum Scale cluster, IBM Knowledge Center.

5.4 Account and container data placement policies

For performance reasons, it might be preferable to place account and container data in a separate storage pool that is backed by faster solid-state device (SSD) storage. This can be easily accomplished by using the IBM Spectrum Scale ILM placement policies.

To implement the placement policies, a dependent fileset must be created to hold all account and container data. A storage pool is then created that is backed by the preferred storage. A file placement policy must then be written and deployed so that all files that are created in the new fileset are automatically placed in the new storage pool. The installer sets the devices parameter in both account server and container server configuration files to use a directory named `ac` under the object fileset. The devices parameter in the object server configuration file is set to use a directory that is named `o` under the object fileset. The devices parameter in the account server and container server configuration files is set to use a directory named `ac` under the object fileset, which allows the account and container fileset to be kept separate and easily linked to the ac directory.

Currently, the installer does not automatically configure the account and container fileset or create the placement policies. This task must be performed by the user. An example is provided below.

In the following example, the IBM Spectrum Scale file system device name is `gpfs`, the file system mount point is `/gpfs0`, and the object independent fileset is `objectfs`.

> **Note:** Delete the `ac` directory before you start this process. If you already have account and container data in the `ac` directory, copy this data to a temporary location and then delete the `ac` directory.

To configure the account and container filesets, complete the following steps:

1. Create a stanza file that is named `nsd_file` to define the SSD devices that are used in the new storage pool. In this example, the pool is called poolAC and there are two devices:

   ```
   %nsd:
   device=/dev/dm100
   usage=dataOnly
   pool=poolAC
   %nsd:
   device=/dev/dm101
   usage=dataOnly
   pool=poolAC
   ```

2. Create the NSDs by running the following command:

   ```
   mmcrnsd -F nsd_file -v no
   ```

3. Add the NSDs to your IBM Spectrum Scale file system by running the following command:

   ```
   mmadddisk gpfs -F nsd_file
   ```

4. Create the `object_acfs` dependent fileset that shares inode space with existing object fileset objectfs for account and container data by running the following command:

   ```
   mmcrfileset gpfs object_acfs --inode-space objectfs
   ```

5. Stop object service on all node by running the following command:

   ```
   mmces service stop OBJ -a
   ```

6. Link the new dependent fileset as `ac` under the `objectfs` independent fileset by running the following command:

```
mmlinkfileset gpfs object_acfs -J /gpfs0/objectfs/ac
Fileset object_acfs linked at /gpfs0/objectfs/ac
```

7. Assign ownership to the `ac` directory by running the following command:

```
chown swift:swift /gpfs0/objectfs/ac
```

8. Create a placement.policy file. Here is an example:

```
/* policy rules to place account and container files into poolAC */
/* PLACEMENT policy 1 - put account and container files in poolAC
poolAC is previously created storage pool and object_acfs is previously created
dependent fileset */
RULE 'db files' SET POOL 'poolAC' FOR FILESET('object_acfs')
/* Default placement policy rule */ RULE 'default' SET POOL 'system'
```

9. To test the policy, run the following command:

```
mmchpolicy gpfs placement.policy -I test
Validated policy 'placement.policy': Parsed 2 policy rules
```

10. If no errors were detected, apply the policy by running the following command:

```
mmchpolicy gpfs placement.policy -I yes
Validated policy 'placement.policy': Parsed 2 policy rules.
Policy 'placement.policy' installed and broadcast to all nodes
```

> **Note:** If you copied any existing account and container data to a temporary location, copy it back to the new `ac` directory. Preserve the original file permission during the copy process. If SELinux is enabled on the cluster, correct the SELinux permission by running following command:
>
> ```
> /usr/sbin/restorecon -R <path of ac directory>
> ```
>
> An example of this command usage is:
>
> ```
>  /usr/sbin/restorecon -R /gpfs0/objectfs/ac
> ```

11. Start the object service by running the following command:

```
mmces service start OBJ -a
```

5.5 Other data tiering use cases

It is possible to use IBM Spectrum Scale ILM policies to better manage your object data. For example, you can define migration policies so that object data is moved between storage tiers that are defined in IBM Spectrum Scale storage pools. This movement of data is done automatically by IBM Spectrum Scale without changing the object's URL or the location of the object file in the underlying file system.

There are two approaches that can be used. The first is based on the heat (frequency of access) of objects or files in a fileset. For an example about how to configure this approach, see IBM Knowledge Center.

The second approach is more experimental, but demonstrates how data can be migrated between tiers based on object metadata. With this approach, a client can modify metadata on one or a group of objects, and the data is moved between storage tiers based on those changes. For an example of this approach, see this website.

5.6 Process monitoring

IBM Spectrum Scale comes with a system health monitoring framework. It monitors IBM Spectrum Scale for object functions. The status of relevant system services and accesses to ports are checked. It also respects the singleton node and database node assignments for services (see 3.3, "Swift services" on page 27).

It periodically checks the service status and compares the status with the expected status. If a mismatch is detected, the framework tries to correct the status by either starting or stopping the service. Additionally it also monitors service functions by trying to access ports in use by the Swift services. It expects a reply within a stipulated time.

If either the service status mismatches or the port requests are not answered after the corrective actions are run, the node is set to an unhealthy state, which causes the CES IP addresses that are hosted on the unhealthy node to be moved to a different protocol node that is in the healthy state. The framework ensures that the services are available to the clients, connecting to a certain CES IP.

For more information about the system health framework, see IBM Knowledge Center.

For more information about CES monitoring and troubleshooting, see IBM Knowledge Center.

For more information about the CES IP address failover, see IBM Knowledge Center.

The IBM Spectrum Scale `mmhealth` command detects issues that can cause a node failover. For more information about this command, see IBM Knowledge Center.

The framework can list events that caused a failover, such as the account service not responding to the port request within the stipulated amount of time.

The system health status can also be viewed by the IBM Spectrum Scale GUI. It is possible to use Simple Network Management Protocol (SNMP) traps and emails to notify when significant events are detected. For more information, see IBM Knowledge Center.

Additionally, the framework ensures that a node that just joined the cluster after a start or restart pulls the latest object configuration (including the Swift rings) from the clustered configuration repository. After the configuration is updated on the node, it ensures that the required services are restarted. The same process happens if any Object (Swift, Keystone) configuration was changed by running the IBM Spectrum Scale `mmobj config` command.

5.7 Security-Enhanced Linux considerations

When the SELinux mode is *Enforcing* and a user on the protocol node must run Swift helper commands, you must run the `runcon` command. Otherwise, the system might generate SELinux failure logs. For example, the `*.recon` files in the `/var/cache/swift` directory are updated by Swift helper services such as replicator and updater.

Complete the following steps:

1. To run the `helper` command, run the following command:

   ```
   runcon -t swift_t -r system_r swift helper command
   ```

2. To check whether the SELinux context is correct, run the `matchpathcon` command:

   ```
   matchpathcon -V /var/cache/swift/*
   ```

The system displays the following output:

```
/var/cache/swift/account.recon verified.
/var/cache/swift/container.recon has context
unconfined_u:object_r:var_t:s0, should be
system_u:object_r:swift_var_cache_t:s0
/var/cache/swift/object.recon verified
```

3. To fix the SELinux context in the file, run the **restorecon** command:

```
restorecon /var/cache/swift/container.recon
```

5.8 Port security considerations

The Swift object, container, and account servers are configured to use ports 6200 and 6203, 6201, and 6202 respectively. The embedded Keystone server is accessed on ports 5000 and 35357. These ports must be reachable by the proxy server on each object storage node. In a production deployment, these ports must be blocked so that they cannot be accessed from non-protocol nodes.

> **Note:** The Swift object, container, and account servers do not authenticate requests that they receive. For this reason, it is critical to ensure that firewall policies are set up to protect these ports from outside access in a production deployment.

For more information about the ports that are used by object storage, see IBM Knowledge Center.

5.9 Overall security considerations

For a comprehensive description of security, including authentication, authorization, secure data at rest, secure data in transit, and secure administration, see *IBM Spectrum Scale Security*, REDP-5426, found here.

5.10 Configuring rsync to limit host access

To limit the hosts that can interact with your rsync environment, add the list of all the network addresses of the object storage nodes to the hosts allow configuration parameter in the `/etc/rsyncd.conf` file. This task must be done on each node and updated any time that a node is added to or deleted from the environment. Here is an example:

```
hosts allow = 192.167.11.10, 192.167.11.11
```

This line allows the two specified hosts to connect to rsync, and all others are rejected. For more information about configuring rsync, see the following website.

Swift feature overview

An advantage of Swift over other object stores is its rich set of features and its extensible design, which allows users to easily add middleware for their applications. For more information, see the following website.

There is a large list of features and middleware, and more features are added regularly. This chapter describes the high priority set of features that are currently tested to help ensure their general stability and correctness. For more information, see the following website.

This chapter describes the following Swift features that have been tested:

► Quotas: Swift quotas allow a specific amount of disk capacity to be allocated to either containers or accounts by using Swift quotas. They also allow a limit on the maximum number of objects to be specified for containers or accounts.

> **Note:** Because of Swift eventual consistency semantics, these limits might be exceeded in certain circumstances.
>
> Although IBM Spectrum Scale quotas do not explicitly interact with Swift quotas, it still might be useful to employ IBM Spectrum Scale quotas to limit the amount of space or the number of inodes that is consumed by the object store. To do this task, define IBM Spectrum Scale quotas on the independent filesets that are used for storing object data by specifying the maximum size or maximum inode usage that the object store or storage policy can consume.
>
> For more information about IBM Spectrum Scale quotas, see IBM Knowledge Center.

► Access control lists (ACLs): Swift ACLs and the Keystone authentication server enable user-based and role-based access that is configured for accounts, containers, and individual objects.

> **Note:** Do not use IBM Spectrum Scale ACLs in the object-independent filesets.

- ► Very large object support: The maximum single object size that has been tested is 5 TB, which is much larger than the typical Swift object size limit (5 GB). This size is possible because of the following conditions:
  - – Load balancing across IBM Spectrum Scale protocol nodes to increase throughput is no longer necessary because each IBM Spectrum Scale protocol node can read and write by using the full storage network bandwidth (instead of needing to share the bandwidth with the creation of one or more object replicas).
  - – Object placement imbalance is not an issue with IBM Spectrum Scale because of its shared storage architecture.
- ► Swift object segmentation can be used to store arbitrarily large objects in the object store. For more information, see the following website.
- ► Versioning: Normally, when an object is updated, the new object replaces the existing object in the object store. Swift versioning can be used to retain multiple versions of an object, which are automatically created upon object update. For more information, see the following website.
- ► Swift3 (S3 compatibility) middleware can be used to allow your object store to interoperate with applications that use the S3 protocol. S3 access can be enabled or disabled running the `mmobj s3` command. For more information, see IBM Knowledge Center.

  For a list of the supported S3 operations, see the following website.

Data protection: Backup, restore, and disaster recovery

Enterprise data requires proven and robust data management tools to ensure that data is not lost due to user errors, software errors, or even hardware failures and errors. Generic OpenStack Swift replication protects against hardware failures, but it is not an end-to-end solution that allows recovery of data that has been incorrectly modified by a user, software application, or in rare cases, hardware corruption. Object versioning can assist with this situation, but it was not designed for this purpose. For example, with traditional object storage, there is no way to instantly create a version or a copy of the entire object store. Older versions cannot be deleted without deleting the latest version and it does not include account or container database information.

You can use IBM Spectrum Scale Unified File and Object Storage to create consistent point-in-time copies of the object store to protect against data loss or corruption. These copies can be either created and stored instantly within IBM Spectrum Scale by using snapshots, or stored in an external mass capacity storage system by using backups.

7.1 IBM Spectrum Scale independent filesets

By using storage policies, different types of object data can be stored in different independent filesets. Additionally, account and container data can be stored in a separate fileset to leverage solid-state drives (SSDs) by using IBM Spectrum Scale information lifecycle management (ILM) placement policies. In the future, you can also spread object data across different file systems for extremely large-scale object storage deployments.

Therefore, it is not always practical to store all of your data in a single independent fileset. Instead, you can have a collection of filesets that operate independently. For certain operations, you can manage all or a subset of these snapshots as a group. In particular, for snapshot and backup operations, you can create snapshot groups that include account and container data, and selected object data.

7.2 Snapshots

IBM Spectrum Scale fileset snapshots provide a method of creating instant point-in-time copies of a fileset. You can have several storage policies for different types of object data, each with a different backup strategy. For example, business critical data might be stored in containers that belong to one storage policy (and fileset), and are backed up nightly. Scratch data might be stored in containers that belong to a second storage policy (and fileset) and are backed up weekly. These containers all can belong to the same file system, and the entire file system is backed up monthly.

To make a consistent snapshot, you must back up the fileset that contains account and container data at the same time as the object data. This task can be accomplished in different ways:

► Create a file system snapshot that has a consistent view of all of the filesets, including account, container, and object data. Backup and restore operations can be performed from the data that is held in that snapshot.

► Create a group of snapshots that includes the independent fileset holding account and container data and the filesets containing the object data of interest. This task should be done at a period of low activity or with the object interface stopped to ensure a consistent state. In this case, a snapshot naming convention can be used to help identify the snapshots belonging to the same group.

For more information about snapshots, see the `mmcrsnapshot` command in *IBM Spectrum Scale Version 4 Release 2.2 Command and Programming Reference*, SA23-1456-04.

7.3 Disaster recovery options

There are two options for disaster recovery in IBM Spectrum Scale Object Storage. The correct choice for your system depends on the requirements of your use case and the capabilities of your wide area network.

7.3.1 Active File Management-based Asynchronous Disaster Recover

IBM Spectrum Scale protocols disaster recovery (DR) uses Active File Management (AFM) based Asynchronous Disaster Recovery (AFM DR) to provide a solution that allows an IBM Spectrum Scale cluster to fail over and fail back to another cluster and back up and restore the protocol configuration information when a secondary cluster is not available. This supports up to two sites, where only one site is active at any point. This can be used for all storage policies, including those with file access enabled. The AFM DR limitations are documented in IBM Knowledge Center. For more information about this DR approach, see IBM Knowledge Center.

7.3.2 Swift multi-region active-active replication

With the OpenStack Swift multi-region capability, you can configure an active-active configuration with more than two sites. The tested limit with IBM Spectrum Scale V4.2.2 is three sites. With storage policies, you can specify the number of replicas of a group of objects that should be maintained. You can, for example, create storage policies that replicate object data to every site, policies that replicate object data only to two sites, and policies where object data is stored only at a single site. Account and container data is always replicated to every site. Swift multi-region does not support storage policies with file access enabled.

7.4 Backing up and restoring the object store

Snapshots are a good way to protect data from various errors and failures. Moving them to a separate backup storage system can provide better protection against catastrophic failures of the entire storage system and might even allow the data to be stored at a lower cost.

The IBM Spectrum Scale `mmbackup` command can be used to back up a file system or an independent fileset to IBM Spectrum Protect.

To back up the entire object store consistently, create a snapshot of the file system. The file system snapshot can then be moved to backup media by running the `mmbackup` command.

It might be more efficient to back up only filesets that are associated with the selected storage policies. In such a case, create a snapshot of the account and container fileset and of the fileset of the storage policy and back up the pair together by running the `mmbackup` command.

You can run the `mmbackup` command to create incremental backups so that only objects that changed since the last backup are moved to backup media.

For more information about the `mmbackup` command, see IBM Knowledge Center.

Objects can be restored by using the IBM Spectrum Protect command-line interface (CLI) to restore either individual objects or all of the objects in a fileset.

While restoring an entire storage policy, all the container databases that are associated with that policy must also be restored.

The AFM DR capability can also be used to back up the entire contents of the protocol-enabled cluster. For more information, see IBM Knowledge Center.

8

Summary

This chapter summarizes further areas of investigation and concludes with how
IBM Spectrum Scale Unified File and Object Storage solves business problems by providing a
new type of cloud storage.

53

8.1 Future investigation

This paper focuses on the general architecture and setup of IBM Spectrum Scale Object Storage. To realize the vision that was described earlier in this paper, IBM continues to extend the object storage features.

Here are some of the high-priority items that are being considered:

► Integration of external tape pools to further decrease the cost per gigabyte. See *Active Archive Implementation Guide with IBM Spectrum Scale Object and IBM Spectrum Archive*, REDP-5237 here.

Also, see the SwiftHLM project for a description of how the Swift API can be extended to better support high latency media, such as tape or optical disk, at this website.

► The ability to search for objects in a container or account that have specific metadata values. See IBM Spectrum Scale Object Metadata Search Open Beta for a description of this integration at this website.

► For improved scalability, add support for multiple file systems to provide storage to a single object storage cluster.

► The ability to designate objects as immutable for a duration so that they cannot be updated or deleted during that time.

Note: Any references to future investigation are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

8.2 Conclusion

IBM Spectrum Scale is a proven, enterprise-class file system, and OpenStack Swift is a best-of-class object-based storage system. IBM Spectrum Scale Object Storage combines these technologies to provide a new type of cloud storage that includes efficient data protection and recovery, proven scalability and performance, snapshot, and backup and recovery support, and information lifecycle management (ILM). Through these features, IBM Spectrum Scale Object Storage can help simplify data management and allow enterprises to realize the full value of their data and enable new, enterprise-ready cloud workloads.

Related publications

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

IBM Redbooks

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

► *Active Archive Implementation Guide with IBM Spectrum Scale Object and IBM Spectrum Archive*, REDP-5237

► *Enabling Hybrid Cloud Storage for IBM Spectrum Scale Using Transparent Cloud Tiering*, REDP-5411

► *IBM Private, Public, and Hybrid Cloud Storage Solutions*, REDP-4873

► *IBM Software-Defined Storage Guide*, REDP-5121

► *IBM Spectrum Scale in an OpenStack Environment*, REDP-5331

► *IBM Spectrum Scale Security*, REDP-5426

You can search for, view, download, or order these documents and other Redbooks, Redpapers, web docs, drafts, and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

► *IBM Spectrum Scale Version 4 Release 2.2 Administration Guide*, SA23-1455

► *IBM Spectrum Scale Version 4 Release 2.2 Concepts, Planning, and Installation Guide*, GA76-0441

► *IBM Spectrum Scale Version 4 Release 2.2 Command and Programming Reference*, SA23-1456

► *IBM Spectrum Scale Version 4 Release 2.2 Problem Determination Guide,* GA76-0443

► *"*GPFS-based implementation of a hyper-converged system for software defined infrastructure" by Azagury, et al, in IBM Journal of Research and Development 58 (2), 1-12, 2014

Online resources

This website is also relevant as a further information source:

► IBM Spectrum Scale

  http://www.ibm.com/systems/storage/spectrum/scale/index.html

IBM Knowledge Center topics

► Elastic Storage Server planning and service information

  http://www.ibm.com/support/knowledgecenter/P8ESS/p8ehc/p8ehc_storage_landing.htm

  http://www.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=DCD12377USEN

► IBM Spectrum Scale Administering

  http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.2/com.ibm.spectrum.scale.v4r22.doc/bl1adm_administering_kclanding.htm

► IBM Spectrum Scale Command Reference

  http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.2/com.ibm.spectrum.scale.v4r22.doc/bl1adm_command.htm

► IBM Spectrum Scale Configuring

  http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.2/com.ibm.spectrum.scale.v4r22.doc/bl1adm_configuring_kclanding.htm

► IBM Spectrum Scale Frequently Asked Questions (FAQ)

  http://www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html?cp=STXKQY_4.2.2

► IBM Spectrum Scale Installing and upgrading

  http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.2/com.ibm.spectrum.scale.v4r22.doc/bl1ins_installingandupgrading_kclanding.htm

► IBM Spectrum Scale Monitoring

  http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.2/com.ibm.spectrum.scale.v4r22.doc/bl1adv_monitoring_kclanding.htm

► IBM Spectrum Scale Planning

  http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.2/com.ibm.spectrum.scale.v4r22.doc/bl1in_PlanningForIBMSpectrumScale.htm

► IBM Spectrum Scale Product Overview

  http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.2/com.ibm.spectrum.scale.v4r22.doc/bl1in_IntroducingIBMSpectrumScale.htm

► IBM Spectrum Scale Programming Reference

  http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.2/com.ibm.spectrum.scale.v4r22.doc/bl1adm_programmingref_kclanding.htm

- ► IBM Spectrum Scale RAID information

  http://www.ibm.com/support/knowledgecenter/en/SSYSP8/gnrfaq.html
  http://www.ibm.com/support/knowledgecenter/en/SSYSP8_4.5.0/c2766583.pdf?view=kc

- ► IBM Spectrum Scale Troubleshooting

  http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.2/com.ibm.spectrum.scale.
  v4r22.doc/bl1pdg_troubleshooting_kclanding.htm

IBM developerWorks topics

- ► File System Planning

  https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General+
  Parallel+File+System+%28GPFS%29/page/File+System+Planning

- ► Tuning Parameters

  https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General%
  20Parallel%20File%20System%20%28GPFS%29/page/Tuning%20Parameters

OpenStack topics

- ► Authentication

  http://docs.openstack.org/developer/swift/api/authentication.html

- ► The Auth System

  http://docs.openstack.org/developer/swift/overview_auth.html#keystone-auth

- ► Cluster Telemetry and Monitoring

  http://docs.openstack.org/developer/swift/admin_guide.html#cluster-telemetry-an
  d-monitoring

- ► General System Tuning

  http://swift.openstack.org/deployment_guide.html#general-system-tuning

- ► Middleware

  http://docs.openstack.org/developer/swift/middleware.html

- ► Middleware and Metadata

  http://docs.openstack.org/developer/swift/development_middleware.html

- ► Object Storage Command line

  http://docs.openstack.org/cli-reference/swift.html

- ► Object Versioning

  http://docs.openstack.org/developer/swift/overview_object_versioning.html

Other websites

- ► Amazon Simple Storage Services (S3)

  http://aws.amazon.com/documentation/s3

- ► Cyberduck

  http://cyberduck.io

- ► rsyncd.conf

  http://rsync.samba.org/ftp/rsync/rsyncd.conf.html

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Get connected

ibm.com/redbooks