

Unlocking Data Insights and AI: IBM Storage Ceph as a Data Lakehouse Platform for IBM watsonx.data and Beyond

Daniel Parkes
Daniel Dominguez Cuadrado
Franck Malterre
Jean-Charles (JC) Lopez
Jun Liu
Kyle Bader
Poyraz Sagtekin
Tushar Agrawal
Vasfi Gucer



Analytics

Storage





IBM Redbooks

**IBM Storage Ceph as a Data Lakehouse Platform for
IBM watsonx.data and Beyond**

July 2024

Note: Before using this information and the product it supports, read the information in “Notices” on page xv.

First Edition (July 2024)

This edition applies to IBM Storage IBM Storage Ceph 7.x and IBM watsonx.data 1.x.

© Copyright International Business Machines Corporation 2024. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	vii
Tables	ix
Examples	xi
Notices	xv
Trademarks	xvi
Preface	xvii
Authors	xvii
Now you can become a published author, too!	xx
Comments welcome	xx
Stay connected to IBM Redbooks	xx
Chapter 1. Introduction	1
1.1 History	2
1.2 Ceph versions	2
1.3 Ceph and storage challenges	4
1.3.1 Data keeps growing	4
1.3.2 Technology changes	4
1.3.3 Data organization, access, and costs	4
1.3.4 Unlocking data value in a distributed landscape	5
1.3.5 Ceph approach	5
1.3.6 Ceph storage types	5
1.4 What is new in IBM Storage Ceph 7.0	6
1.4.1 New features	6
1.4.2 Enhancements	7
1.4.3 Hardware	8
Chapter 2. The modern data lake architecture	9
2.1 Introduction	10
2.2 Data zones	11
2.3 Organizing the data lakehouse	12
2.3.1 Buckets	12
2.3.2 Catalogs, schemas, and tables	13
Chapter 3. Building a scalable data lake with IBM Storage Ceph Object Storage ...	15
3.1 IBM Storage Ceph Object Storage: An ideal platform for data lakes	16
3.2 Security	17
3.2.1 Access control	18
3.2.2 Encryption	20
3.2.3 Object lock	21
3.2.4 Versioning	21
3.3 Scalability	22
3.4 Efficiency	25
3.4.1 Erasure coding	25
3.4.2 Compression	26
3.4.3 Lifecycle management	28
3.5 Data management	29

3.5.1 Ceph Object Gateway placement targets	29
3.5.2 Ceph Object Gateway Storage classes	30
3.5.3 Ceph Object Gateway bucket notifications	33
3.6 Regional data resiliency in Ceph.	36
3.6.1 Archive zone	42
Chapter 4. Replacing Hadoop Distributed File System with IBM Storage Ceph Object Storage	47
4.1 Hadoop introduction	48
4.1.1 The birth of Hadoop: A response to big data challenges	48
4.1.2 Hadoop's rise: Powering the big data revolution.	48
4.2 HDFS introduction.	50
4.3 The rise of object storage	50
4.4 Accessing S3 Object Storage from Hadoop (Amazon S3A File System)	51
4.5 IBM Storage Ceph Object Storage integration with Hadoop.	53
4.6 IBM Storage Ceph Object Storage and S3A.	54
4.7 Example of connecting Hadoop with IBM Storage Ceph by using S3A	54
4.8 Migrating from HDFS to S3A.	58
4.8.1 Storage abstraction.	58
4.8.2 The DistCp tool	59
4.8.3 Application-layer tools for data movement	62
4.9 A real-world HDFS to S3A migration example	62
4.9.1 The environment.	62
4.9.2 Current solution.	63
4.9.3 S3A Ceph-based solution	63
Chapter 5. IBM Storage Ceph with IBM watsonx.data	65
5.1 Introducing IBM watsonx.data.	66
5.2 IBM watsonx.data infrastructure components.	67
5.3 Typical use cases for IBM watsonx.data.	68
5.4 IBM watsonx.data UI.	69
5.4.1 Infrastructure manager	70
5.4.2 Data manager	71
5.4.3 Query workspace	72
5.4.4 Query history.	73
5.4.5 Access control.	74
Chapter 6. Retail use case	77
6.1 Use case introduction	78
6.1.1 Objectives	78
Chapter 7. Ingest: Landing and raw zones	81
7.1 Ingestion from stores or branches to the landing zone	82
7.1.1 Ingest: Lifecycle policy expiration	83
7.1.2 Ingest: Data at rest encryption server-side encryption with Amazon Key Management Service	86
7.1.3 Ingest: Object storage identity provider authentication with single sign-on.	86
7.1.4 Ingest: IAM role-based access control-based authorization	88
7.2 Raw zone: Branch stores classification data pipeline	90
7.2.1 Raw zone: Red Hat OpenShift and Kafka setup.	92
7.2.2 Raw zone: S3 bucket notification configuration	93
7.2.3 Raw zone: Setting up the raw zone buckets.	95
7.2.4 Raw zone: S3 object lock and object versioning configuration	95
7.2.5 Raw zone: S3 SSE-KMS encryption at rest	97

7.2.6 Raw zone: S3 Multi-Factor Authentication delete	97
7.2.7 Raw zone: Data tiering configuration - creating a cold storage class	100
7.2.8 Raw zone: Authentication - IAM role configuration for the raw zone	103
7.2.9 Raw zone: Serverless data pipeline workflow	106
7.2.10 Raw zone: Knative and Kafka service deployment	108
7.2.11 Raw zone: Ingest data pipeline verification	112
7.2.12 Raw zone: Hadoop data ingest into the raw zone	114
7.2.13 Raw zone: E-commerce platform data ingest	114
7.2.14 Early raw zone data filtering with S3 Select	115
Chapter 8. Transform: Staging and curated zones	119
8.1 Points of sale (physical stores)	120
8.1.1 Assigning bucket policies and object tags for effective S3 access control	120
8.2 E-commerce	132
8.2.1 E-commerce: Ingest	133
8.2.2 E-commerce: Browser log workflow	133
8.2.3 E-commerce: Transactions workflow	134
8.2.4 E-commerce: Initial data cleansing	134
Chapter 9. Consume	155
9.1 Introduction	156
9.2 Visualization: E-commerce example	156
9.2.1 Deploying and configuring Apache Superset	157
9.2.2 Connecting Apache Superset to PrestoDB (watsonx.data)	159
9.2.3 Creating a graph or data set from an SQL query	163
9.3 Curated point-of-sale SQL queries	170
9.3.1 IBM watsonx.data Single Sign-On authentication setup	170
9.3.2 Consuming the curated data from watsonx.data	176
9.3.3 Visual Explain	178
Appendix A. Configuring RADOS Gateway by using the IBM Storage Ceph GUI.	181
Configuring RADOS Gateway by using the IBM Storage Ceph GUI	182
RADOS Gateway user creation	182
Bucket creation	184
Appendix B. Configuring the command-line tools for IBM watsonx.data	185
Configuring command-line tools for watsonx.data	186
Installing the ibm-lh-client utility	186
Setting up the ibm-lh utility	188
Creating a schema	188
Creating a table and ingesting data	189
Appendix C. Additional material	191
Locating the GitHub material	191
Cloning the GitHub material	191
Abbreviations and acronyms	193
Related publications	195
IBM Redbooks	195
Online resources	195
Help from IBM	195

Figures

3-1 IBM Storage IBM Storage Ceph features	17
3-2 Attribute-based access control example	19
3-3 Multisite one-realm	38
3-4 Multisite two-realms	39
3-5 Multisite bucket-granularity	41
3-6 Visual representation of how archive zones manage S3 object versions	43
5-1 IBM watsonx platform	66
5-2 IBM watsonx platform together with IBM Data and Data Fabric services	67
5-3 IBM watsonx.data lakehouse architecture and its value proposition.	68
5-4 IBM watsonx.data console home	69
5-5 Infrastructure manager	70
5-6 Data manager window	71
5-7 Query workspace	73
5-8 Query history window	74
5-9 Infrastructure Access control window	75
5-10 Access policies	76
6-1 Point of sales example use case for the modern data lake	79
6-2 Point of sales example use case for the modern data lake	80
7-1 Landing zone diagram	82
7-2 Red Hat Single Sign-On	87
7-3 User lookup	87
7-4 Data pipeline	91
7-5 User map	103
7-6 High-level description of the ingest data workflow	107
7-7 Data corresponding bucket flow	108
7-8 Visual representation of the workflow	111
7-9 Data pipeline architecture for e-commerce	115
8-1 Add bucket option	122
8-2 Adding a bucket	123
8-3 Manage associations	124
8-4 Presto engine restarts	125
8-5 Create schema	126
8-6 Create table from	127
8-7 Uploading a sample file from your computer to watsonx.data	128
8-8 Summary window	128
8-9 Ingest data	129
8-10 Selecting Iceberg copy loader.	129
8-11 Selecting the files within the confidential bucket.	130
8-12 Specifying your target for the ingestion.	130
8-13 Summary window	131
8-14 Query workspace	131
8-15 Data pipeline architecture for e-commerce	132
8-16 E-commerce ingest high-level workflow	133
8-17 Infrastructure manager: Catalogs	142
8-18 Infrastructure manager: Buckets.	142
8-19 Data manager objects.	143
8-20 Catalogs view	151
9-1 Sign In window	158

9-2	Configuring Superset to access the PrestoDB SQL engine	159
9-3	Selecting Settings > Database Connections.	159
9-4	Adding a database connection	159
9-5	Selecting Presto from the list	160
9-6	BASIC Configuration tab.	161
9-7	Adding the entry "verify:false" in the Advanced security settings	162
9-8	Superset SQL Lab	163
9-9	The table schema: Displaying a detailed overview of the data structure	164
9-10	Superset displays the results	165
9-11	Visualizing the data.	166
9-12	Top-selling e-commerce products.	167
9-13	Save chart.	168
9-14	Building more charts	169
9-15	Moving the chart from draft to published.	169
9-16	Identity providers option	171
9-17	New connection	171
9-18	OIDC option	172
9-19	Entering the details	172
9-20	Add users	173
9-21	Platform access	173
9-22	Assigning roles	174
9-23	Summary.	175
9-24	Query workspace	176
9-25	Generate path option	177
9-26	Worksheet results	178
9-27	Visual Explain	178
A-1	Users option on the Object Gateway interface	182
A-2	Creating an RGW user	183
A-3	Create Bucket.	184

Tables

- 1-1 Upstream and downstream IBM Storage Ceph versions 3
- 1-2 Downstream product lifecycle 3
- 3-1 Event types that can be used to trigger an S3 bucket notification 34
- 3-2 Synchronization policy group states 40
- B-1 The ibm-lh-client utility 186

Examples

3-1	Updating an existing placement target to enable compression	27
3-2	S3 Bucket Lifecycle policy	32
4-1	Creating a user specifically for Hadoop access	54
4-2	Providing encryption at rest for the data sets	55
4-3	Validating the S3 endpoint and ensuring that it is operational as intended	55
4-4	Uploading an object and using SSE-KMS to encrypt the object at rest	56
4-5	Configuring an S3 bucket encryption policy	56
4-6	Enabling the hadoop-aws module.	56
4-7	Configuring IBM Storage Ceph Object Storage SSE-KMS.	57
4-8	Checking whether you can access the hadoop bucket by using S3A from Hadoop.	58
4-9	Accessing the data on the HDFS file system	58
4-10	Copying the hosts file from HDFS to S3	59
4-11	Switching between storages	59
4-12	Creating a Hive external table for reference	60
4-13	Copying the data set from HDFS to S3.	61
4-14	Running distcp to copy the data from the HDFS source to the S3 destination	61
4-15	Modifying the location of the table to start using the new s3a location	61
4-16	Checking that the queries work against the data set	61
7-1	Local Object Gateway user.	83
7-2	Configuring the AWS CLI for the ingest user	83
7-3	Landing zone bucket ingest	83
7-4	Deleting the contents of buckets that are older than 24 hours	83
7-5	Checking the lifecycle configuration for bucket “ingest”	84
7-6	Changing the interval to 60 in the IBM Storage Ceph RGW configuration database and an object gateway service restart.	84
7-7	Uploading a file that is called “testlc” without the S3 tag set to processed=true	85
7-8	Uploading an object that is called “shop4_02_04_2024.csv” with the tag set to processed=true.	85
7-9	Confirming that the lifecycle job successfully finished	85
7-10	Testing the lifecycle configuration.	85
7-11	IBM Storage Ceph configuration options that we are using to integrate IBM Guardium Key Lifecycle Manager with IBM Storage Ceph.	86
7-12	Configuring an S3 put-bucket-encryption policy	86
7-13	Group membership of the users is part of the JWT token.	87
7-14	Granting the shoppingest user full access to managing roles	88
7-15	Creating the IAM role with its policy doc.	88
7-16	Configuring the IAM policy	89
7-17	Testing the entire workflow	90
7-18	Checking that we can upload objects but not delete them	90
7-19	Kafka deployment running on Red Hat OpenShift Container Platform	92
7-20	Creating shop-ingest-pipe.	92
7-21	Configuring the ingest bucket and adding an SNS topic.	93
7-22	Configuring the notification event on the ingest bucket	93
7-23	Using a PUT request on a test object to the `ingest` bucket.	94
7-24	Checking for the notification event in Kafka	94
7-25	Creating the two buckets for storing classified data	95
7-26	Verifying that uploads of objects to this bucket do not have an object lock retention policy	96

7-27	Creating the bucket with the parameter object-lock-enabled-for-bucket.	96
7-28	Enabling object versioning	96
7-29	Configuring encryption at rest	97
7-30	The dnf install oathtool command	98
7-31	Creating a seed by using the system's "urandom" module	98
7-32	Configuring a TOTP entry for the account	98
7-33	The radosgw-admin mfa check command	98
7-34	Creating a TOTP PIN with the oathtool	99
7-35	Enabling the MFADelete option on the confidential bucket	99
7-36	Verifying that MFA is in place	99
7-37	Trying to do a normal s3api delete-object	99
7-38	Adding the --mfa option	99
7-39	The IBM Storage Ceph osd tree command	100
7-40	Configuring the failure domain to the host and setting the device class to an SSD .	100
7-41	Creating the rule that uses the HDD device class	101
7-42	Creating a pool that uses the cold CRUSH rule	101
7-43	STANDARD storage class	101
7-44	Adding the storage class to the zonegroup	102
7-45	Configuring the datapool for the COLD storage class	102
7-46	Lifecycle policy rule	103
7-47	IAM role doc policy	104
7-48	Configuring a policy that is specific to accessing the ingest bucket	104
7-49	Adding a second policy	105
7-50	Testing the raw-zone-admin	105
7-51	Testing the ingest bucket	106
7-52	Testing the anonymized and confidential buckets	106
7-53	Knative operators that are deployed and running on our Red Hat OpenShift Container Platform cluster.	108
7-54	Deploying the Eventing, Serving, and Knative Kafka operators	109
7-55	Creating a Knative serving service	109
7-56	The oc get deployment command	110
7-57	Python classification script	111
7-58	Sample data set generator	112
7-59	Object created event arrived	112
7-60	Application pod logs	113
7-61	Checking whether the .csv file landed at its final destination	113
7-62	Checking whether the object has the S3 tag for the data security classification. . .	113
7-63	Final check	114
7-64	Code snippet	116
7-65	E-commerce example app log	117
8-1	Creating the buckets	120
8-2	Assigning a bucket policy to the confidential bucket	120
8-3	Assigning a bucket policy to the anonymized bucket	121
8-4	Sample data set	134
8-5	Granting the user the roles capability	135
8-6	Confirming the existence of the ecommraw bucket within the raw zone	135
8-7	Creating a local RGW user	135
8-8	JSON IAM role doc policy	136
8-9	Listing the recently created role by using the S3 API	136
8-10	First role policy	137
8-11	Second role policy	137
8-12	Assuming the "spark-ecomm" role	138
8-13	Snippet of the Spark job S3A configuration	138

8-14	Leveraging a combination of IAM roles and temporary credentials	139
8-15	Running the Spark job	140
8-16	Data verification checks	140
8-17	Data verification checks	141
8-18	Transaction data set	141
8-19	Connecting to a CP4D watsonx.data Presto instance	143
8-20	Selecting a default catalog and schema	144
8-21	Confirming that our data is accessible for querying through Presto	144
8-22	Creating the browser log table	145
8-23	Issuing a SELECT statement to fetch data from the newly created table	146
8-24	Running a CALL command to synchronize or refresh the table partition metadata	146
8-25	JOIN query between the transactions and browser log tables	147
8-26	SQL JOIN query	148
8-27	Creating a view	149
8-28	Creating and configuring the bucket in watsonx.data	150
8-29	Creating a table from the JOIN statement	151
9-1	The podman run command	157
9-2	Setting up an administration account	158
9-3	Superset UI is available through a web browser on port 8088	158
9-4	URI that is used for this example	161
9-5	Query to see the top 10 selling items	164
9-6	SQL query	177
9-7	Connecting to your Presto engine	179
9-8	SQL query to determine the most popular payment methods	179
B-1	Creating an installation directory and browsing to it	186
B-2	Setting up the environment variables	186
B-3	Release tags with Cloud Pak for Data versions	187
B-4	The watsonx.data client package	187
B-5	The watsonx.data client pkg.tar file	187
B-6	Authenticating Docker with the IBM Container Registry	187
B-7	Adding the username and password for authentication	188
B-8	Logging in by using presto-cli	188
B-9	Creating your schema by using the presto CLI	189
B-10	Showing the schemas	189
B-11	Creating a table	189
B-12	Defining your S3 credentials as environment variables	189

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

Cognos®

Db2®

Guardium®

IBM®


IBM Cloud®

IBM Cloud Pak®

IBM Security®

Netezza®

Redbooks®

Redbooks (logo) ®

The following terms are trademarks of other companies:

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

ITIL is a Registered Trade Mark of AXELOS Limited.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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

Preface

IBM watsonx.data empowers enterprises to scale their analytics and AI capabilities with a purpose-built data store, leveraging an open lakehouse architecture. Through its robust querying, governance, and open data formats, IBM watsonx.data facilitates seamless data access and sharing. With IBM watsonx.data, you can swiftly connect to data, extract actionable insights, and optimize data warehouse expenses.

IBM Storage IBM Storage Ceph offers high-performance, scalable object storage that can handle large data sets efficiently. IBM watsonx.data and IBM Storage IBM Storage Ceph can be a powerful combination for building a scalable and cost-effective data lakehouse solution.

This IBM Redbooks® publication dives into how the IBM Storage IBM Storage Ceph robust storage capabilities and the IBM watsonx.data advanced analytics features come together to form a powerful data and AI platform. This platform empowers you to unlock valuable insights from your data and make data-driven decisions.

Using a real-world scenario involving an Amazon Simple Storage Service (S3) data lake, you can explore each stage of the data pipeline (ingestion, transformation, and consumption) with step-by-step instructions and hands-on examples. All code samples that are created for the book's scenarios are available at the [Redbooks GitHub](#) repository.

The target audience for this book is data architects, data analysts, data engineers, and data scientists working on creating AI and data lakehouse solutions.

Note: For more information about the IBM Storage IBM Storage Ceph architecture and technology that is behind the product, see *IBM Storage Ceph Concepts and Architecture Guide*, REDP-5721.

Authors

This book was produced by a team of specialists from around the world.



Daniel Parkes works in the IBM Storage IBM Storage Ceph Product Management team, where he focuses on the IBM Storage IBM Storage Ceph Object Storage offering. An active open-source enthusiast participant in IBM Storage Ceph technical discussions, Daniel demonstrates a strong commitment to customer success, actively supporting their adoption of IBM Storage Ceph solutions on their path towards digital excellence.



Daniel Dominguez Cuadrado is a Data Services Senior Specialist Solution Architect at Red Hat. He is an experienced IBM Storage IBM Storage Ceph Expert, with nine years of experience working with IBM Storage IBM Storage Ceph in the field with professional services, and then as pre-sales on the Red Hat EMEA storage team.



Franck Malterre is an IT professional with a background of over 25 years in designing, implementing, and maintaining large x86 physical and virtualized environments. For the last 10 years, he has specialized in the IBM File and Object Storage Solution by developing IBM Cloud Object Storage, IBM Storage Scale, and IBM Storage IBM Storage Ceph live demonstrations and proofs of concept for IBM® personnel and IBM Business Partners.



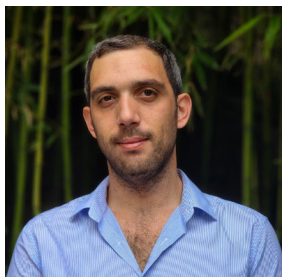
Jean-Charles (JC) Lopez has been in storage for 80% of his professional career, and has been working in software-defined storage since 2013. JC helps people understand how they can move to a containerized environment when it comes to data persistence and protection. His go-to solutions are Fusion, IBM Storage Ceph, and Red Hat OpenShift. He takes part in the Upstream Community and downstream versions of IBM Storage Ceph and Red Hat OpenShift.



Jun Liu serves as the architect for IBM watsonx.data, primarily focusing on the authentication and authorization architecture across various components of the product. His current emphasis is on enhancing data security and governance within the platform. Before working on the IBM watsonx project, Jun Liu was the Technical Architect of Data Virtualization Console. He leads the Data Virtualization Console development and new feature integration team. Before working on the Data Virtualization project, he worked for various projects on IBM Db2® query monitoring and optimization. He has been with IBM since 2005.



Kyle Bader is a Senior Technical Staff Member and Principal Portfolio Architect for IBM Storage Ceph based offerings at IBM. He has spent a decade designing IBM Storage Ceph based storage clusters and solutions. His current interests lie at the intersection of distributed storage, containers, data-intensive applications, and machine learning (ML).



Poyraz Sagtekin joined IBM in 2016 after completing his Computer Engineering degree at Middle East Technical University. Since then, he has worked in diverse areas, including IoT systems, IBM Cloud®, Kubernetes, Red Hat Enterprise Linux (RHEL), and IBM Power servers. Currently, he works as a Hybrid Cloud Services Consultant at IBM Systems Expert Labs in Türkiye. He is an IBM recognized speaker and educator.



Tushar Agrawal leads a team of Customer Success Architects in the US National Market to drive adoption of the hybrid cloud and artificial intelligence (AI) platforms that are provided by IBM. Tushar focuses on ethical AI principles and collaborates with his customers to develop innovative solutions to address business transformation challenges. Tushar is an experienced enterprise architect for large-scale data processing. Tushar successfully launched 50+ complex solutions for customers across industry verticals and segments. Tushar has filed 180+ patent applications with the US Patent Office, and continuously works on innovation with emerging technologies to co-create intellectual property for IBM. Tushar holds a BS in Computer Science & Engineering degree from MNNIT Allahabad. He also holds an MBA degree. He is pursuing a PhD in Software Engineering from North Dakota State University, Fargo, ND.



Vasfi Gucer works as the Storage Team Leader on the IBM Redbooks Team. He has more than 30 years of experience in the areas of systems management, networking hardware, and software. He writes extensively and teaches IBM classes worldwide about IBM products. For the past decade, his primary focus has been on storage, cloud computing, and cloud storage technologies. Vasfi is also an IBM Certified Senior IT Specialist, Project Management Professional (PMP), IT Infrastructure Library (ITIL) V2 Manager, and ITIL V3 Expert.

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

Elias Luna, Henry Vo, Greg Deffenbaugh, Matt Benjamin, Wade Wallace
IBM US

Kelly Schlamb
IBM Canada

The team extends its gratitude to the Upstream Community, IBM, and Red Hat IBM Storage Ceph Documentation teams for their contributions to continuously improve IBM Storage Ceph documentation.

Now you can become a published author, too!

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book 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, IBM Redbooks
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on LinkedIn:

<https://www.linkedin.com/groups/2130806>

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

<https://www.redbooks.ibm.com/subscribe>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<https://www.redbooks.ibm.com/rss.html>



Introduction

This chapter describes the origin story of IBM Storage Ceph, a software-defined storage solution, and the fundamental architectural concepts that underpin its functions.

This chapter has the following sections:

- ▶ History
- ▶ Ceph versions
- ▶ Ceph and storage challenges
- ▶ What is new in IBM Storage Ceph 7.0

Important: IBM Storage Ceph is the official name for IBM's Ceph-based storage product, while the community version is simply called Ceph. Red Hat continues to offer Red Hat Ceph Storage, primarily for Red Hat OpenStack Platform. Since early 2023, IBM has managed Ceph for standalone storage and broader cloud-native architectures, following the transition of the Red Hat Ceph storage team and technologies to IBM Storage. These naming conventions are used throughout this IBM Redbooks publication.

1.1 History

The Ceph project was born after recognizing that the [Lustre](#) architecture had its flaws in terms of metadata lookup. In Lustre, the lookup function that identifies where a particular file is located relies on a specific software component that is called the Metadata Server (MDS) that must be queried before accessing a file or a directory. Under heavy loads, this MDS can become a bottleneck.

To solve this inherent Lustre architecture problem, Sage Weil imagined a new mechanism to distribute and locate the data in a distributed and heterogeneous structured storage cluster. This new concept is metadata-less and relies on a pseudo-random placement algorithm to do so.

The novel algorithm, which is named Controlled Replication Under Scalable Hashing ([CRUSH](#)), leverages a sophisticated calculation to optimally place and redistribute data across the cluster, minimizing data movement when the cluster's state changes, all without any extra metadata.

CRUSH distributes the data across all devices in the cluster to avoid the classic bias of favoring empty devices to write new data. This approach eliminates the potential bottleneck that is caused in other systems where empty devices are overloaded with new writes due to capacity balancing.

CRUSH manages data distribution throughout the Ceph storage cluster's lifecycle, including expansion, reduction, and device failures. This approach ensures a balanced distribution of both old and new data across all physical disks. As a result, I/O operations (reads/writes) are evenly spread across the storage system, leading to optimal performance and scalability.

To enhance data distribution, the solution enables the breakdown of large elements (for example, a 100 GiB file) into smaller elements, each assigned a specific placement through the CRUSH algorithm. Therefore, reading a large file leverages multiple physical disk drives rather than a single disk drive.

Sage Weil prototyped the new algorithm and created a new distributed storage software solution, Ceph (short for cephalopod). The name Ceph was chosen as a reference to the ocean and the life that it harbors because Santa Cruz, CA is a Pacific Ocean coastal town, where Sage lives.

On January 2023, all Ceph developers and product managers were moved from Red Hat to IBM to provide greater resources for the future of the project. The Ceph project at IBM remains an open-source project and code changes still follow the upstream first rule.

1.2 Ceph versions

All Ceph community versions were assigned the name of a member of the Cephalopoda natural sciences family. The first letter of the name helps identify the version.

Table 1-1 on page 3 represents all Ceph community version names with the matching Inktank, Red Hat Ceph Storage, or IBM Storage Ceph version leveraging it.

Table 1-1 Upstream and downstream Ceph versions

Name	Upstream version	Release	End of life (EOL)	Downstream product
Argonaut	0.48	2012-07-03	N/A	
Bobtail	0.56	2013-01-01	N/A	Inktank Ceph Enterprise 1.0
Cuttlefish	0.61	2013-05-07	N/A	
Dumpling	0.67	2013-08-01	2015-05-01	Inktank Ceph Enterprise 1.1
Emperor*	0.72	2013-11-01	2014-05-01	
Firefly	0.80	2014-05-01	2016-04-01	Red Hat Ceph Storage 1.2 Inktank Ceph Enterprise 1.2
Giant*	0.87	2014-10-01	2015-04-01	
Hammer	0.94	2015-04-01	2017-08-01	Red Hat Ceph Storage 1.3
Infernos*	9.2.1	2015-11-01	2016-04-01	
Jewel	10.2	2016-04-01	2018-07-01	Red Hat Ceph Storage 2
Kurgan*	11.2	2017-01-01	2017-08-01	
Luminous	12.2	2017-08-01	2020-03-01	Red Hat Ceph Storage 3
Mimic*	13.2	2018-06-01	2020-07-22	
Nautilus	14.2	2019-03-19	2021-06-30	Red Hat Ceph Storage 4
Octopus*	15.2	2020-03-23	2022-08-09	
Pacific	16.2	2021-03-31	2023-10-01	IBM Storage Ceph 5 (start with 5.3) Red Hat Ceph Storage 5
Quincy	17.2	2022-04-19	2024-06-01	IBM Storage Ceph 6 (start with 6.1) Red Hat Ceph Storage 6
Reef	18.2	2023-08-07	2025-08-01	IBM Storage Ceph 7 (start with 7.0) Red Hat Ceph Storage 7
Squid

The versions that are identified by an * with no corresponding downstream product are considered Ceph development versions with a short lifespan and not used by Inktank, Red Hat, or IBM to create a durable enough product. Table 1-2 shows the downstream product lifecycle.

Table 1-2 Downstream product lifecycle

Product	General availability (GA)	EOL	Extended support is available until
Inktank Ceph Enterprise 1.1		2015-07-31	N/A
Inktank Ceph Enterprise 1.2 Red Hat Ceph Storage 1.2		2016-05-31	N/A

Product	General availability (GA)	EOL	Extended support is available until
Red Hat Ceph Storage 1.3		2018-06-30	N/A
Red Hat Ceph Storage 2		2019-12-16	2021-12-17
Red Hat Ceph Storage 3		2021-02-28	2023-06-27
Red Hat Ceph Storage 4		2023-03-31	2025-04-30
Red Hat Ceph Storage 5.3 IBM Storage Ceph 5.3	2023-03-10	2024-08-31	2027-07-31
Red Hat Ceph Storage 6.1 IBM Storage Ceph 6.1	2023-08-18	2026-03-20	2028-03-20
Red Hat Ceph Storage 7.0 IBM Storage Ceph 7.0	2023-12-08		

1.3 Ceph and storage challenges

Persistent storage has always faced the same basic challenges.

1.3.1 Data keeps growing

As information technology evolved, it moved from character-based data (text) to multimedia data (images, videos, and audio). This change accelerated data growth and democratized data creation, so data could be generated from various devices and applications. Therefore, the traditional concept of structured data is increasingly complex.

1.3.2 Technology changes

As the information technology lifecycle accelerates, the steady centralized data center morphs into a more decentralized organization relying on application and server communication like never before, when all CPUs were in the same room with passive terminals requiring access to its processing capabilities.

1.3.3 Data organization, access, and costs

Historically, data was in centralized repositories that used various media like magnetic disks, tapes, and punch cards. However, access to this data was tightly controlled and often required physical interaction (like swapping tapes or sorting cards). Preserving the availability and integrity of this diverse data landscape was a paramount responsibility for IT departments.

1.3.4 Unlocking data value in a distributed landscape

As data fragments across infrastructure and is used by diverse users and devices, the true value of Ceph is analyzing vast, multifaceted data sets. This process requires processing information from multiple sources and formats, often through advanced techniques like artificial intelligence (AI) and machine learning (ML). The key to unlocking this value hinges on real-time data accessibility, regardless of location, without human intervention for speed and efficiency.

Ceph addresses the challenges of modern data storage head-on. Designed for high availability, it eliminates single points of failure. Ceph boasts exceptional scalability so that you can seamlessly add storage capacity as your needs grow. Day-to-day management is streamlined, requiring minimal intervention beyond replacing failed physical resources like nodes or drives. Notably, Ceph prioritizes data integrity while offering a cost-effective alternative to expensive proprietary storage solutions.

1.3.5 Ceph approach

To address all these challenges, Ceph relies on a layered architecture with an object store at its core. This core, which is known as the *Reliable Autonomic Distributed Object Store (RADOS)*, provides the following features:

- ▶ Stores the data with data protection and consistency as goal number one.
- ▶ Follows a scale-out model to cope with growth and changes.
- ▶ Present no single point of failure.
- ▶ Provides CRUSH as a customizable metadata-less placement algorithm.
- ▶ Runs as a software-defined storage solution on commodity hardware.
- ▶ Open source to avoid vendor lock-in.

1.3.6 Ceph storage types

Ceph is known as a unified storage solution and supports the following types of storage:

- ▶ Block storage by using a virtual block device solution that is known as *RADOS Block Device (RBD)*.
- ▶ File storage by using a Portable Operating System Interface (POSIX) compliant shared file system solution that is known as *Ceph Filesystem (CephFS)*.
- ▶ Object storage by using an OpenStack Swift and Amazon Simple Storage Service (S3) compatible gateway that is known as the *RADOS Gateway (RGW)*.

The different types of storage are segregated and do not share data between them while sharing a physical storage pool of storage devices. A custom CRUSH configuration can create segmented pools of storage at the node level.

The different types of storage are identified and implemented as Ceph access methods on the native RADOS API. This API is known as `librados`.

Ceph is written entirely in C and C++, except for some API wrappers that are language-specific (For example, the Python wrapper for `librados` or `librbd`).

Note: For more information about the IBM Storage Ceph architecture and technology that is behind the product, see *IBM Storage Ceph Concepts and Architecture Guide*, REDP-5721.

1.4 What is new in IBM Storage Ceph 7.0

This major release marks a significant upgrade by migrating the upstream codebase from Quincy (used in IBM Storage Ceph 6) to Reef. This shift enables enhanced performance and stability for your storage system. Also, IBM Storage Ceph 7.0 introduces a range of new features and improvements.

1.4.1 New features

This section delves into some of the new features that were introduced with IBM Storage Ceph 7.0. For more information, see [IBM Storage Ceph Version 7.0 Release Notes](#).

Note: This book is based on IBM Storage Ceph 7.0.

Information gathering and Call Home

The new *Call Home* integrated feature automatically gathers information about your IBM Storage Ceph cluster, simplifying troubleshooting and improving your support experience.

Key alert data is securely shared with IBM Storage Insights to improve your support experience. This two-way communication enables IBM Product Managers to make data-driven decisions that benefit customers while you gain a centralized view of all your deployed clusters through a simple interface.

Network File System over CephFS

With this new release, you can share data between CephFS native and Network File System (NFS) clients. Non CephFS capable servers can access the data that is stored in the CephFS through new NFS export capabilities that are carried by the NFS Ganesha Server software.

This new feature provides high availability through multiple NFS gateways and support for a Virtual IP address, and it can be configured by using the integrated IBM Storage Ceph Dashboard UI.

Write Once Read Many certification

Cohasset Associates, Inc. evaluated IBM Storage Ceph Object Lock against the electronic records requirements that are mandated by various regulatory bodies. They concluded that IBM Storage Ceph, when configured and used with Object Lock, adheres to the electronic record keeping system stipulations of the US Securities and Exchange Commission (SEC) and Financial Industry Regulatory Authority (FINRA) rules:

- ▶ SEC 17a-4(f)
- ▶ SEC 18a-6(e)
- ▶ FINRA 4511(c)
- ▶ Commodity Futures Trading Commission (CFTC) 1.31(c)-(d)

Note: The SEC stipulates record keeping requirements, including retention periods. FINRA rules regulate member brokerage firms and exchange member markets.

For more information, see the [Cohasset certification](#).

Object Storage Gateway

IBM Storage Ceph 7.0 brings the following enhancements to the S3 object store:

- ▶ Configure RGW multisite through the Dashboard UI.
- ▶ Multisite asynchronous replication status and monitoring.
- ▶ Object-level interaction through the Dashboard UI (For example, add or remove object).

CephFS

This feature provides improved health and performance monitoring of the CephFS together with better capacity utilization tracking.

CephFS volume management is now integrated with the IBM Storage Ceph Dashboard UI to provide the following functions:

- ▶ Create a snapshot.
- ▶ Delete a snapshot.
- ▶ Access snapshots and encryption settings.
- ▶ Manage snapshots.

Non-Volatile Memory Express over Fabric technology preview

Administrators can deploy Non-Volatile Memory Express over Fabric (NVMe-oF) gateways through the standard **cephadm** utility. NVMe-oF allows Non-Volatile Memory Express (NVMe) initiators, typically on bare-metal servers, to access IBM Storage Ceph virtual block devices by using the NVMe-oF protocol without any Ceph code through Linux NVMe kernel modules.

Archive zone technology preview

The S3 object store that is deployed by using the IBM Storage Ceph RGW feature can be configured to push every version of every object to an archive zone. The archive zone acts as an object catalog where objects are not deleted even if the original object is deleted. As such, the archive zone can be used to restore any version of any object that was ever archived. This function can be useful to enact compliance rules within the IBM Storage Ceph cluster.

The archive mechanism can be configured at the S3 bucket level to selectively archive data. For more information about the archive zone feature, see 3.6.1, “Archive zone” on page 43.

NFS over S3 technology preview

You can access an S3 bucket by using an NFS client to facilitate the migration of existing file system-based data or the sharing of S3 data with legacy applications.

1.4.2 Enhancements

This section delves into some of the enhancements that are included in IBM Storage Ceph 7.0.

Data protection

This new version offers support for Erasure Coding 2+2 for a better efficiency within small-scale IBM Storage Ceph clusters.

As a best practice, size your cluster with one extra node than the protection scheme to allow normal cluster healing and data recovery (for example, 2+2 Erasure Coding data protection with five IBM Storage Ceph nodes).

Object Storage Gateway

This new version offers support for S3 Select for the following formats:

- ▶ CSV
- ▶ JavaScript Object Notation (JSON)
- ▶ Parquet

The new feature *Datacenter-Data-Delivery Network (D3N)* uses high-speed storage such as NVMe flash or dynamic RAM (DRAM) to cache data sets on the access side. D3N improves the performance of big-data jobs running in analysis clusters by speeding up recurring reads from the data lake.

With this version, you can use *bucket-level granularity* when configuring the RGW multisite asynchronous replication to reduce inter-site replication traffic and potentially reducing your recovery point objective (RPO) by limiting the amount of data that is replicated so that you focus on the critical data.

This version introduces *policy-based archive and migration strategies* into Amazon Web Services (AWS) S3 and AWS compatible S3 endpoints such as IBM Storage Ceph and IBM Cloud Object Storage. These policies can be managed by the user by using standard S3 bucket policies.

Better *parallelism* is introduced with this new version to provide faster asynchronous replication in Object Storage Gateway multisite setups.

Security

Transport Layer Security (TLS) is enabled across all components of the Ceph monitoring stack for heightened security. The new security model is aligned with security best practices and covers Prometheus, Alertmanager, and the Node Exporter.

Flexibility

Placing a host in maintenance mode is now protected by using a flag to prevent placing a host in maintenance mode by mistake, which stops all the Ceph daemons on this node.

Nodes can now be drained without deleting configuration information, but can also be placed in a managed or unmanaged state at will.

Note: For more information, see the [IBM Storage Ceph IBM Documentation page](#).

1.4.3 Hardware

IBM now offers IBM Storage Ceph Ready nodes for customers that need pre-validated hardware platforms to run this Ceph software offering. These ready nodes come in two modes:

- ▶ Capacity-optimized Ready Nodes (hard disk drive (HDD)-based).
- ▶ Performance-optimized Ready Nodes (flash-based).

Note: For more information, see the [Ceph Ready Nodes Data Sheet](#).

The IBM Storage Ceph RGW supports *Intel QuickAssist Technology (QAT)* encryption and compression for enhanced efficiency on both types of operations in your lakehouse environment. The Performance-Optimized nodes are fitted with the appropriate Intel components, and the Capacity-Optimized nodes will be fitted with it in their next update cycle.



The modern data lake architecture

Imagine a central location where you can store all your data regardless of format or structure. This location is the core idea behind a modern data lake architecture. It acts as a central repository for housing massive volumes of data, from raw sensor data to processed business transactions. A modern data lake architecture provides a flexible and scalable foundation for organizations to leverage the power of all their data.

This chapter introduces the modern data lake architecture.

This chapter has the following sections:

- ▶ Introduction
- ▶ Data zones
- ▶ Organizing the data lakehouse

2.1 Introduction

Data lakes revolutionized data storage by introducing schema-on-read, polyglot data handling, and scalable storage. This approach contrasted with traditional data warehouses, which relied on schema-on-write, proprietary data formats, and expensive and often limited storage.

Although Apache Hive paved the way for data lakes, it predated the widespread adoption of cloud services. The Google cloud offering would not arrive for another 3 years. As a result, early data lake adopters turned to Hive paired with Hadoop Distributed File System (HDFS), a scalable, on-premises file system running on affordable hardware. This approach offered a cost-effective alternative to data warehouses initially, but later, cloud adoption exposed limitations in HDFS.

The cloud-native paradigm prioritizes decoupling storage and compute for independent scaling, something HDFS inherently struggled with. Cloud object storage (Amazon Simple Storage Service (S3)) services offered unmatched ubiquity, scalability, and cost-effectiveness compared to HDFS. Unfortunately, Hive table organization became entrenched in file system semantics.

However, new data formats like Parquet and Avro emerged, alongside engines like Spark, Impala, and Presto that could readily process them. This development enabled polyglot data retrieval, processing, and analytics, reducing vendor lock-in and breaking down data silos.

Data warehouses are still in enterprise data centers. Although data lakes afforded great flexibility and economics, Hive tables do not map well to object storage, and performance optimization is difficult. If Hive is the precursor to data lakes, Apache Iceberg is the precursor to the data lakehouse. Iceberg is designed for object storage, so it addresses many performance and correctness deficiencies of Hive tables. Iceberg is the current standard for curated tabular data products.

What is a data lakehouse?: Data lakehouses combine the strengths of data warehouses and data lakes into a single, powerful platform.

- ▶ Data warehouses: Known for their performance and structured data organization, data warehouses are ideal for business intelligence (BI) tasks and fast querying. However, they can be expensive to maintain and struggle to handle diverse data types.
- ▶ Data lakes: Offering unparalleled scalability and flexibility, data lakes can store any type of data, but their unstructured nature can hinder efficient analysis.

Data lakehouses bridge the gap between these two approaches. They leverage cloud object storage to handle a wide range of data formats (structured, semi-structured, and unstructured) while maintaining the organization and manageability of a data warehouse. This approach translates to several key benefits:

- ▶ Improved data accessibility: Users can access and process data by using various tools (SQL and machine learning (ML)) across all data types.
- ▶ Simplified data exploration: Having all your data in one place enables broader and faster data analysis and discovery.
- ▶ Enhanced scalability and cost-efficiency: Cloud storage provides a scalable and potentially more cost-effective solution compared to a traditional data warehouse infrastructure.
- ▶ Stronger data governance: Centralized data management facilitates access controls and regulatory compliance.

For more information about data lakehouses, see this [IBM website](#).

2.2 Data zones

Data strategies must acknowledge *polyglot persistence*, which is recognition that source data might be unstructured, or serialized in a structured, semi-structured, or multi-modal data formats. Therefore, an approach to organizing data that logically partitions data into various zones is needed. Zoning data provides isolation, and informs the intended consumers about the quality of the data. The exact name and number of zones might vary depending on the organization, but generally there is one zone that is synonymous with *raw data*, another zone that is synonymous with *staging data*, and another zone that is synonymous with *curated data*.

When data arrives, it might be unrefined or not stored in optimized formats, and it often lacks normalization. Data in this stage is in the raw zone (see Chapter 7, “Ingest: Landing and raw zones” on page 83). The data might be machine logs, sensor data, unstructured text, or semi-structured events from applications. Raw data is not ready to be consumed by BI tools, train models, or be used for *Retrieval-Augmented Generation (RAG)*. The raw data must be processed by engines that label, enrich, normalize, and encode it in more optimized formats.

What is RAG?: [RAG](#) is a technique for improving the accuracy and reliability of AI models by referencing trusted external sources before generating text.

This processing does not happen in a single step. In between raw and curated zones is the *staging zone*, which is a place to store data between processing steps (see Chapter 8, “Transform: Staging and curated zones” on page 121). The final zone is the *curated zone*, which is the high-quality data that constitutes a data product (see Chapter 9, “Consume” on page 157).

As stated in 2.1, “Introduction” on page 10, if the data is tabular in nature, it should be represented as Iceberg tables after it is in the curated zone. Other examples of curated data sets are flattened features that are serialized in multimodal data formats like TFRecord or Petastorm, or embeddings for RAG that are indexed by vector databases like Milvus. Regardless of the zone or type, all data must be persisted in cost-effective, scalable object storage.

IBM Storage Ceph is open-source and software-defined. It runs on industry-standard hardware, and has best-in-class coverage of the S3 API. IBM Storage Ceph was designed as an object store, in contrast with approaches that bolt-on S3 API servers to a distributed file system. With Ceph, data placement is by algorithm instead of by lookup so that Ceph scales well into [billions](#) of objects, even on modestly sized clusters. Data that is stored in Ceph is protected with efficient erasure coding, through in-flight and at-rest checksums and encryption, and through robust access control that integrates with enterprise identity systems.

Note: For more information, see *IBM Storage Ceph Concepts and Architecture Guide*, REDP-5721.

2.3 Organizing the data lakehouse

This section explains some strategies for organizing your data lakehouse.

2.3.1 Buckets

A *bucket* is a logical container to organize collections of objects. At the infrastructure level, buckets are a resource that is administered and owned by an account. There is a multitude of properties that are configured at the bucket level, including retention, encryption, versioning, notification, access policies, and more. Some of these properties apply to all objects in a bucket, and others can be scoped to objects that match specified prefixes or tags.

Ceph buckets excel at handling massive object volumes by supporting hundreds of millions of objects. This scalability stems from Ceph internal bucket index sharding, which eliminates the need for prefix rate limiting and optimizes performance.

To ensure data isolation, data belonging to different zones should be stored in separate buckets. The same situation is true for data that is represented in disparate formats. Use separate buckets for tabular data with different structures, data with specific offline functions (data that has unique requirements for access or management when the Ceph cluster itself is offline), and objects that are related to vector databases.

Note: A *vector database* is a type of database that is designed to store and manage data as multidimensional vectors. These vectors are mathematical representations of information, where each dimension corresponds to a specific feature or attribute.

2.3.2 Catalogs, schemas, and tables

Hive and Iceberg catalogs are specialized technical catalogs that manage tabular data. They act as registries and store information about schemas. Each schema defines the structure and organization of tables, including their offline features. These catalogs play a crucial role in commit protocols that are used during table updates, which ensure data consistency and integrity.

Although both catalogs offer these functions, Iceberg stands out for its superior performance and robust safety features. In practice, it is common to use a separate S3 bucket for each catalog. This approach stems from the fact that many properties, like access control or lifecycle management, are configured at the bucket level. Different catalogs might require distinct settings for optimal functions.

This chapter explored the fundamentals of modern data lake architecture. The following chapters delve into why IBM Storage Ceph is a perfect fit for data lakehouses, and showcase a real-world scenario from the retail industry.



Building a scalable data lake with IBM Storage Ceph Object Storage

Built on the highly scalable foundation of the Reliable Autonomic Distributed Object Store (RADOS), IBM Storage Ceph extends its capabilities with more access methods that are integrated through the RADOS Gateway (RGW). This combination delivers a powerful Amazon Simple Storage Service (S3) object store. As a result, users can leverage the familiar S3 interface while benefiting from the inherent scalability and flexibility of IBM Storage Ceph.

This chapter explores the S3 compatible object storage features that are offered by IBM Storage Ceph. This chapter describes why these features make IBM Storage Ceph an ideal choice for data lake architectures. Also, this chapter examines the growing popularity of the S3 protocol and its features among data engineering, AI, and analytics teams worldwide.

This chapter has the following sections:

- ▶ IBM Storage Ceph Object Storage: An ideal platform for data lakes
- ▶ Security
- ▶ Scalability
- ▶ Efficiency
- ▶ Data management
- ▶ Regional data resiliency in IBM Storage Ceph

3.1 IBM Storage Ceph Object Storage: An ideal platform for data lakes

IBM Storage Ceph, based on the open-source [Ceph Distributed Storage System](#), is a powerful and cost-effective solution that is available to organizations to harness the full potential of their data assets. IBM Storage Ceph should be used for creating data lakes or lakehouses because of the following key advantages:

- ▶ **Cost-effectiveness:** IBM Storage Ceph uses commodity hardware and open-source software to reduce the upfront infrastructure costs.
- ▶ **High scalability:** IBM Storage Ceph enables horizontal scaling to accommodate large volumes of growing data in a data lake or lakehouse.
- ▶ **High flexibility:** IBM Storage Ceph can handle various data types (structured, semi-structured, and unstructured data, for example, text, images, videos and sensor data, and others), making it versatile and appropriate for data lakes.
- ▶ **High availability:** IBM Storage Ceph is designed to provide high durability and reliability for the data that is stored in a data lake or lakehouse. Data is always accessible, despite hardware failures or disruptions in the network, by using data replication across multiple geographic locations, which provides redundancy and fault-tolerance to prevent data loss.
- ▶ **High performance:** IBM Storage Ceph enables parallel data access and processing through integration with frameworks like Apache Hadoop and Apache Spark to enable high throughput and low latency for data ingestion, processing, and analysis within a data lake or lakehouse. It also provides a cache data accelerator (Datacenter-Data-Delivery Network (D3N)) and Query pushdown (S3 Select) to improve the performance of analytical workloads. For more information about implementing S3 Select in IBM Storage Ceph, see 7.2.14, “Early raw zone data filtering with S3 Select” on page 117.
- ▶ **Data governance:** IBM Storage Ceph provides efficient management of metadata to enforce data governance policies, track data lineage, monitor data usage, and provide valuable information about the data that is stored in the data lake, such as format, data source, and others.

Figure 3-1 on page 17 shows the IBM Storage Ceph features.

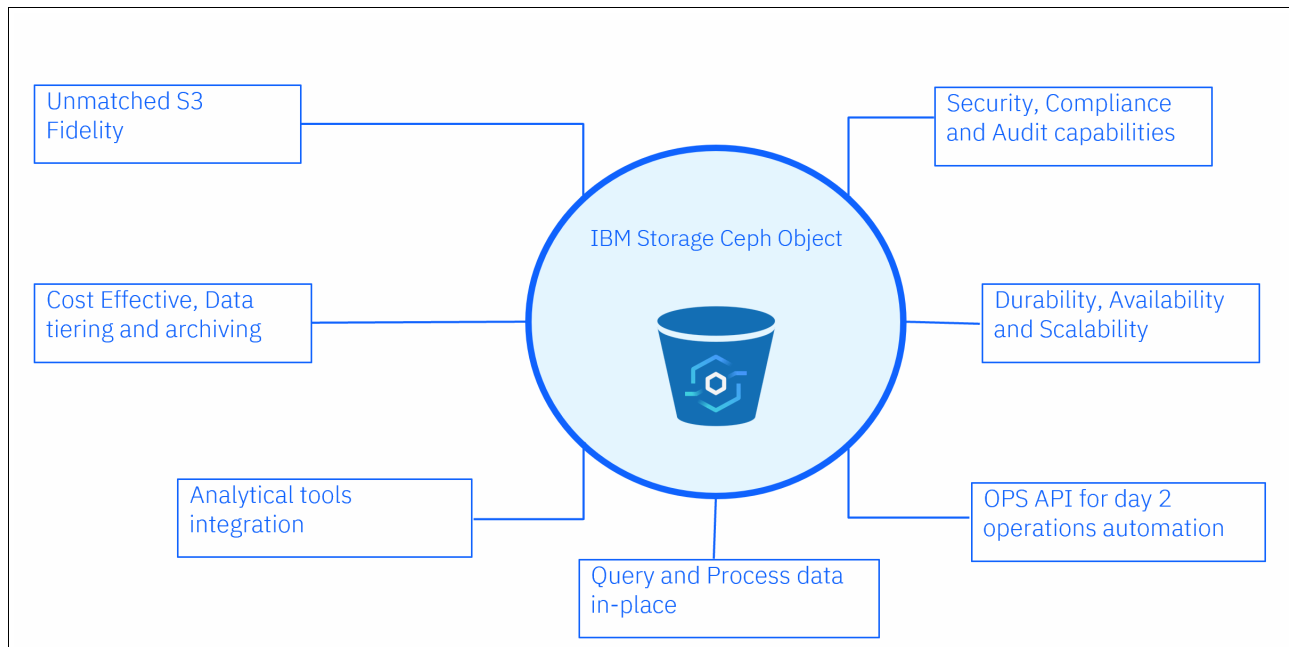


Figure 3-1 IBM Storage Ceph features

Some key features of IBM Storage Ceph are described in the following sections.

3.2 Security

Security is crucial for a data lake or data lakehouse implementations. IBM Storage Ceph consists of multiple security features that make it a robust and ideal choice for building data lakes or lakehouses. Some of these key security features are encryption, access controls, and audit logging. These features protect sensitive data that is stored in the data lake and helps organizations stay in compliance with regulatory obligations like General Data Protection Regulation (GDPR), Health Insurance Portability and Accountability Act (HIPAA), Payment Card Industry Data Security Standard (PCI-DSS), and others. The security features enable consumers to maintain trust in their data lake or lakehouse environments.

IBM Storage Ceph prioritizes data security by enabling you to implement best practices:

- ▶ **Data-level access management:** Control access at the data object level. This approach minimizes the attack surface and focuses on protecting your valuable information.
- ▶ **Seamless Identity and Access Management (IAM) or Single Sign-On (SSO) integration:** Integrate IBM Storage Ceph with your existing IAM or SSO systems. Simplifies user management and ensures consistent access control policies.
- ▶ **Granular control with role-based access control (RBAC) or attribute-based access control (ABAC):** Implement RBAC or ABAC to grant users access based on their roles or specific attributes. Ensures that only authorized users can access specific data within your data lakehouse.

For more information about best practices for data lake security, see the following resource:

- ▶ [Building a Secure Enterprise Data Lakehouse with IBM watsonx.data](#)
- ▶ [Enabling Secure Generative AI with IBM watsonx.data — Navigating the Landscape of Data Privacy](#)

3.2.1 Access control

IBM Storage Ceph empowers organizations to safeguard their data assets with comprehensive access control mechanisms. These policies are built on user identities, roles, and assigned permissions. Through robust *authentication and authorization processes*, only authorized users and applications can access or modify data within the storage cluster's data lake.

The RADOS layer in IBM Storage Ceph offers native support for secure communication by using the cephx protocol. This protocol validates the identity of users and applications attempting to access the storage cluster by using cryptographic authentication.

Note: Cephx is a lightweight authentication system similar to Kerberos, but designed for the Ceph ecosystem. For more information about the cephx protocol, see *IBM Storage Ceph Concepts and Architecture Guide*, REDP-5721.

Also, IBM Storage Ceph seamlessly integrates with various external identity providers (IDPs) and IAM systems, which include industry-standard protocols like Lightweight Directory Access Protocol (LDAP) and Active Directory (AD), along with compatible OpenID Connect (OIDC) providers such as IBM Secure Verify or Red Hat SSO. This flexibility leverages existing authentication infrastructure for a unified access control experience.

Note: For more information about how to configure authentication with SSO (OIDC) by using IBM Security® Verify, see [Authentication and Authorization of Ceph Object Storage by using Open ID Connect \(OIDC\)](#).

IBM Storage Ceph empowers organizations to implement robust access management for their data lake, especially in large deployments. You can achieve this goal by using the following features:

- ▶ **External identity integration:** Streamline access control by leveraging existing user identities that are stored in external directories. This integration can enforce mechanisms like multi-factor authentication (MFA) to add an extra layer of security, particularly for sensitive actions like data deletion.

MFA delete feature: The MFA delete feature strengthens your data security by requiring an extra verification step before objects can be deleted. This safeguard helps prevent accidental or unauthorized deletion of critical data within your IBM Storage Ceph storage system. For more information about how to implement MFA delete in IBM Storage Ceph, see 7.2.6, “Raw zone: S3 Multi-Factor Authentication delete” on page 99.

- ▶ **Granular authorization policies:** Define which users or groups can perform specific actions (read, write, or delete) on objects or buckets within the data lake. The Ceph Object Gateway (RGW) enforces these policies, ensuring that only authorized users can access data through APIs. For more information about step by step instructions for implementing granular authorization policies in IBM Storage Ceph, see 8.1.1, “Assigning bucket policies and object tags for effective S3 access control” on page 122.
- ▶ **Simplified RBAC:** Create pre-defined roles with specific permission sets. Assign these roles to users and applications for simplified management compared to managing individual permissions for each user.

- **ABAC for scalability:** Manage access rights based on tags that are assigned to resources. Simplifies access control, especially when adding resources or modifying privileges for many users at once. ABAC allows for fewer, more concise permission policies.

Note: For more information about ABAC, see [IBM Storage Ceph Object Storage Authorization with Attribute-based Access Control \(ABAC\)](#).

In practice, you can leverage a combination of RBAC and ABAC to create the most effective access control policies for your data lake. This approach offers a balance between management and granular control over data access (Figure 3-2).

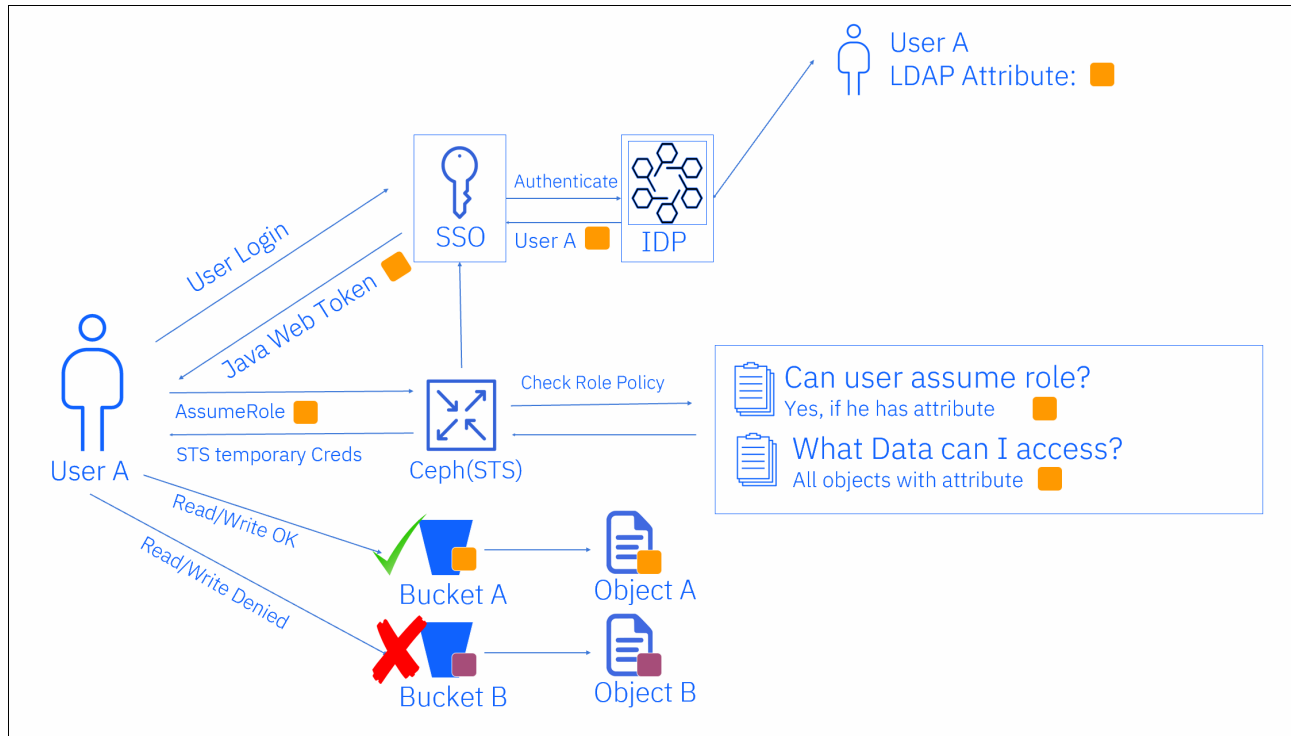


Figure 3-2 Attribute-based access control example

What is the Security Token Service (STS)? RGW implements a subset of functions from Amazon STS. This subset is known as Ceph STS. It enables users and administrators to manage authentication and access control for object storage within a IBM Storage Ceph cluster. Users first authenticate against STS, and on success, receive short-lived S3 access and a secret key that can be used in subsequent requests.

IBM Storage Ceph further provides audit logging that records data lake access actions and attempts that are performed by users or applications within the storage cluster. Information like the timestamp, source IP address, and user identity, are captured in the audit logs to help with compliance, reporting, security monitoring, and investigation.

How to implement access control in IBM Storage Ceph: For more information about how to implement access control in IBM Storage Ceph, see 7.1.3, “Ingest: Object storage identity provider authentication with single sign-on” on page 88 and 7.1.4, “Ingest: IAM role-based access control-based authorization” on page 90.

Note: For more information about audit logging, see [rgw_enable_ops_log](#). Also, IBM Storage Ceph provides centralized logging. For more information about centralized logging, see [Centralized Logging on the Ceph Dashboard](#).

3.2.2 Encryption

IBM Storage Ceph helps organizations secure their data lakes and lakehouses by protecting data integrity and confidentiality. The encryption feature mitigates the risk of data breaches and enables organizations to ensure that their data lakes or data lakehouses comply with industry regulations and data protection laws to safeguard sensitive data.

Data-at-rest encryption

IBM Storage Ceph safeguards your data by encrypting it at rest. This encryption is crucial to protect sensitive information in case of a physical disk theft or unauthorized access.

Hardware Security Modules (HSM) or Key Management Services (KMS) are typically responsible for generating, storing, and distributing encryption keys. IBM Storage Ceph manages encryption keys by integrating them with a Key Management Interoperability Protocol (KMIP) compliant KMS, such as IBM Guardium® Key Lifecycle Manager (GKLM).

Note: Ceph RGW also has built-in support for integrating with Hashicorp Vault. This integration uses Vault native APIs and functions for authentication and key access.

IBM Storage Ceph provides *built-in support* for various server-side encryption (SSE) options like Server-Side Encryption with Amazon Key Management Service (SSE-KMS), Server-Side Encryption with Amazon S3 Managed Keys (SSE-S3) and Server-Side Encryption with Customer-Provided Keys (SSE-C).

- ▶ *SSE-KMS*, where the client-provided keys are stored in a KMS and the storage system retrieves keys from an external key manager on behalf of a calling principal.
- ▶ *SSE-C*, where the client provides its own keys within each bucket interaction.
- ▶ *SSE-S3*, where all object data is encrypted or decrypted transparently by the storage system by using per bucket keys. Makes key management invisible to the calling principal.

For any object, any one of the SSE options can be used to protect data-at-rest. The SSE option can be selected based on the security requirements, compliance mandates, and performance.

How to implement encryption in IBM Storage Ceph: For more information about how to implement encryption in IBM Storage Ceph, see 7.2.5, “Raw zone: S3 SSE-KMS encryption at rest” on page 92.

Data-in-motion encryption

IBM Storage Ceph uses Transport Layer Security (TLS) to encrypt data that is transmitted over the network to ensure a secure communication channel. **cephadm** simplifies the setup by using automated TLS configuration for the ingress service.

Note: For more information about TLS config automation with `cephadm`, see [Setting up HTTPS](#).

A TLS handshake can be done by using symmetric encryption algorithms or mutual authentication between clients and servers to ensure that identity verification is done for a secure data transfer. IBM Storage Ceph further supports an on-wire encryption by using the Messenger v2 protocol to provide encapsulation of the authentication payload and enable authentication with Ceph Authentication Protocol (cephx).

Note: For more information about the Messenger v2 protocol, see [Messenger v2](#).

3.2.3 Object lock

A key requirement for data lake and data lakehouse is to ensure that the critical data is not altered by accident or by unauthorized or malicious activity. IBM Storage Ceph provides an S3 object lock API that enables an *object lock* that prevents objects from being altered or deleted for a retention period. Here are the features of an object lock:

- ▶ **Retention period:** When creating an object, define a retention period during which it is locked. This time frame ensures data integrity and compliance with regulations.
- ▶ **Legal holds:** Even after the retention period expires, legal holds can be applied to extend the object lock indefinitely, which provides more control for sensitive data.
- ▶ **Granular access control:** IBM Storage Ceph restricts setting, modifying, or removing object lock settings to only authorized administrators. Enforces data governance by leveraging familiar S3 APIs.

An S3 object lock compliance mode is a *Write Once Read Many (WORM)* approach that is enforced on locked objects during the retention period. Object lock facilitates data immutability and prevents modification or deletion of the stored data. Using this capability in a data lake or lakehouse enables organizations to adhere with regulations such as GDPR, HIPAA, and other data retention mandates.

Key takeaway: S3 object lock with IBM Storage Ceph offers a robust solution for securing critical data in data lakes and data lakehouses, ensuring compliance and preventing unauthorized modifications.

How to implement object lock in IBM Storage Ceph: For more information about how to implement an object lock in IBM Storage Ceph, see 7.2.4, “Raw zone: S3 object lock and object versioning configuration” on page 97.

Note: Consulting company Cohasset concluded that IBM Storage Ceph, when properly configured and satisfying additional considerations, meets the electronic record-keeping system requirements of SEC Rules 17a-4(f)(2), 18a-6(e)(2), and Financial Industry Regulatory Authority (FINRA) Rule 4511(c), and supports the regulated entity in its compliance with the audit system requirements in SEC Rules 17a-4(f)(3)(iii) and 18a-6(e)(3)(iii). In addition, the assessed capabilities meet the principles-based electronic records requirements of Commodity Futures Trading Commission (CFTC) Rule 1.31(c)-(d). For more information, see the [Cohasset certification](#).

3.2.4 Versioning

IBM Storage Ceph supports the S3 Object versioning API. IBM Storage Ceph creates a version of the object on every write action (for example, **COPY**, **PUT**, or **POST**) when the versioning feature is enabled at the bucket level. The older versions are retained in the storage with a unique identifier, for example, timestamp, along with the current version, enabling users to access older versions as needed for historical bookkeeping through S3 versioning APIs. This feature safeguards your data by enabling you to recover previous versions of the object in, for example, the following situations:

- ▶ Accidental deletion
- ▶ Data corruption
- ▶ Hardware failures
- ▶ Cyberattacks
- ▶ Overwrites due to errors

Note: For more information about the S3 Object versioning API, see [How S3 Versioning Works](#).

If needed, an older version can be removed by specifying the name and the object's version ID. Previous object versions are retained until manually deleted or automatically removed based on lifecycle rules.

Object versioning can only be paused once enabled. However, bucket versioning can be combined with an object lifecycle to set an expiration (as number of days). Each version of an object counts like another object in the bucket. Therefore, administrators can define lifecycle policies at bucket level to move retained versions to a different storage class or delete older versions based on criteria, such as age, last access date, and others to optimize storage usage and reduce associated costs.

Key takeaways:

- ▶ Bucket versioning can be used with object lifecycle management to optimize storage usage and costs.
- ▶ Versioning improves data resilience, data protection, and data governance effectively within the data lake storage.

How to implement versioning in IBM Storage Ceph: For more information about how to implement versioning in IBM Storage Ceph, see 7.2.4, “Raw zone: S3 object lock and object versioning configuration” on page 97.

3.3 Scalability

For scalability, IBM Storage Ceph provides some key features that are described in this section. Some of these features are specific to the RGW, and others are provided by RADOS.

- ▶ RADOS scalability to thousands of object storage daemons (OSDs).
- ▶ Controlled Replication Under Scalable Hashing (CRUSH) to distribute objects across OSDs.
- ▶ Highly available and scalable through multiple RGWs.
- ▶ Highly scalable bucket indexing with dynamic resharding.

- Compatible with many HTTP load balancers, proprietary or open source, hardware or software.
- Indexless buckets.

In large object stores, a key design consideration is *bucket indexing*. Bucket indexes enable users and applications to efficiently *remember* the contents of an object bucket. Imagine a bucket as a directory in a file system. A bucket index acts like a list of all the files within that directory.

This analogy can be further extended to a familiar command. If you are comfortable with Linux environments, using a bucket index is similar to using the `/bin/ls` command to see a directory's contents.

In large environments, the time that it takes to list all the objects that are stored in a bucket grows longer. A naive approach to indexing might require locking the entire index for a single application or user uploading a tiny (1 KiB) or large (thousands of MiB) object.

To address this scalability challenge, IBM Storage Ceph uses a technique that is called *bucket sharding*. This approach splits each bucket index into smaller, manageable slices (shards) with a configurable upper threshold (the default is 100,000 entries). Each shard is stored as key-value data on a distinct RADOS object (for more information, see *IBM Storage Ceph Concepts and Architecture Guide*, REDP-5721). The existence of multiple slices permits the parallelization of tasks when doing a complete listing operation on a single bucket.

In IBM Storage Ceph 7.0, each bucket starts with 10 index shards that can accommodate 1,000,000 objects in a single bucket. *Dynamic Bucket Index Resharding* is enabled by default to preserve performance during listing or update operations going through the Ceph RGW.

IBM Storage Ceph can set the following behaviors for dynamic resharding:

- Enable or disable dynamic resharding (`rgw_dynamic_resharding`, with a default of true).
- Number of objects per shard (`rgw_max_objs_per_shard`, with a default of 100000).
- Maximum number of shards through dynamic resharding (`rgw_max_dynamic_shards`, with a default of 1999).
- Lock during dynamic resharding (`rgw_reshard_bucket_lock_duration`, with a default of 360).

If required, the administrator can manually reshard a bucket index by using the following command:

```
radosgw-admin reshard add --bucket {bucket_name} --num-shards
```

Important: Before resharding a bucket manually, always make a backup of the current bucket index content by running the following command:

```
radosgw-admin bi list --bucket={bucket_name} >bucket.backup
```

Add quotation marks around the {bucket_name} in case your bucket name contains special characters or spaces. For example:

```
radosgw-admin bi list --bucket="{bucket_name}" >bucket.backup
```

To view the outstanding resharding requests, run the following command:

```
radosgw-admin reshard list
```

To view the status of a reshard operation, run the following command:

```
radosgw-admin reshard status --bucket {bucket_name}
```

IBM and Red Hat conducted a series of scalability tests for IBM Storage Ceph storage. Their latest test focused on evaluating the ability of IBM Storage Ceph to handle massive data sets. They successfully ingested over 10 billion objects into an IBM Storage Ceph cluster, demonstrating its exceptional scalability. For more information about this test, see the video at [Red Hat](#).

The scalability of IBM Storage Ceph is made possible by the highly scalable and highly available nature of RADOS and the CRUSH algorithm that efficiently distributes objects across all the OSDs that are available in your IBM Storage Ceph cluster.

Scalability and performance with 1 TBps throughput: For more information about the scalability and performance of Ceph with 1 TBps throughput, see [Ceph: A Journey to 1 TiB/s](#).

IBM Storage Ceph can be paired with scalability only in terms of bandwidth and number of operations per second. This goal can be achieved with a combination of the RESTful nature of the S3 protocol and deploying multiple RGWs, where you place all or a subset of them behind an HTTP load balancer. The supported deployment tool that is used by IBM Storage Ceph, **cephadm**, supports the deployment of an ingress service based on HAProxy for teams that do not have load balancers that are available. The built-in ingress service can also be paired with existing onsite traffic distribution mechanisms.

Key takeaways:

- ▶ Horizontal scaling: IBM Storage Ceph scales by adding more commodity hardware nodes. You can add storage capacity, performance, or both by adding more servers. There is no single point of failure, so you can expand without worrying about bottlenecks.
- ▶ Distributed architecture: IBM Storage Ceph uses a distributed architecture with no central controllers. There is no single point of failure and all nodes cooperate to manage the storage pool.
- ▶ Automatic data placement and replication: IBM Storage Ceph automatically places and replicates data across the cluster to ensure redundancy and availability. Eliminates the need for manual data balancing.

IBM Storage Ceph also offers *index-less buckets*, a feature that is suited for custom applications that can trade the ability to list buckets for higher write operations per second (index-less buckets do not help read performance). Here is how it works:

- ▶ Application-managed indexing: With index-less buckets, the responsibility of maintaining the object list shifts from the Ceph RGW to the application layer.
- ▶ External database integration: Applications typically leverage a separate database (often with Flash Storage) to manage the object index. This approach ensures fast access and retrieval of object information by using database-specific APIs or query languages.
- ▶ Zonal separation: Although index-less and indexed buckets can coexist within an IBM Storage Ceph cluster, they must be placed in separate RGW zonegroups.

Zones and zonegroups:

- ▶ A *zone* represents a specific location where objects are stored within the Ceph cluster. Each zone is backed by its own storage pool.
- ▶ A *zonegroup* is a collection of zones that are logically grouped. By default, data and metadata are automatically synchronized across all zones within a zonegroup. Ensures the redundancy and availability of your objects.

Analytical workloads in IBM Storage Ceph:

IBM Storage Ceph offers two features that can benefit analytical workloads:

- ▶ *D3N* accelerates data access for large data sets by using a local caching layer (RAM or Non-Volatile Memory Express (NVMe)) at the edge. Positions frequently accessed data closer to RGWs, improving performance. Also, the overall infrastructure maintains a good total cost of ownership (TCO) by leveraging cheaper, high-capacity hard disk drive (HDD) storage for the physical back end. For more information, see the [IBM Storage Ceph Documentation](#).
- ▶ *S3 Select* helps you build a more efficient pipeline between your S3 clients and your S3 endpoint by using an SQL-like syntax to select subsets of the data that is stored in the S3 object store. Results in better data targeting and improves latency and throughput while reducing network usage for better TCO in the public cloud and on-premises. For more information, see the [IBM Storage Ceph Documentation](#).

By leveraging these features and carefully considering your specific needs, IBM Storage Ceph can be a powerful platform for storing and analyzing large data sets.

3.4 Efficiency

Data lakes rely on the ability to access massive data sets for comprehensive and nuanced insights. However, traditional hierarchical file systems, often reliant on directory lookups, struggle with scalability. This limitation becomes evident as data volumes reach the exabyte range. However, object storage excels in scaling efficiently to accommodate such vast quantities of data. Unlike file systems that become cumbersome with exponentially growing directories, object stores manage data as individual objects, enabling seamless scalability into the exabyte realm.

As the environment becomes larger, you must provide the best data durability possible. To address the data volume and the durability challenges, object stores rely on features such as:

- ▶ Erasure coding for better raw to usable storage ratio (TCO and durability at scale).
- ▶ Compression to enhance the raw to usable ratio (TCO at scale).
- ▶ Lifecycle policies to automate object expiration or transition (TCO at scale).

3.4.1 Erasure coding

IBM Storage Ceph supports multiple data protection schemes to address performance and cost requirements. The data protection methods follow two different techniques:

- ▶ Replicated pools:

- All objects that are stored in a pool are fully replicated according to the **size** parameter that is assigned to the pool.
- Provides better performance and faster recovery of the pool.
- Provides a better option for small objects and small writes.
- Provides a higher raw to usable storage ratio.

- ▶ Erasure coded (EC) pools:
 - All objects that are stored in the pool benefit from *erasure coding*, a data protection technique that is similar to RAID 5 or RAID 6. Unlike traditional hardware RAID, erasure coding is implemented entirely in software, offering greater flexibility in terms of data distribution and fault tolerance.
 - Provides a longer recovery time for the pool.
 - Requires parity calculation on every write.
 - Provides better durability at a lower cost than replicated pools.
 - Provides a better raw to usable storage ratio.

Replicated pools create full copies of the data and offer a fixed raw to usable storage ratio that is aligned with the pool **size** parameter:

- ▶ 2:1 if size=2 (2 raw bytes are used for every usable byte that is stored.)
- ▶ 3:1 if size=3 (3 raw bytes are used for every usable byte that is stored.)

Note: A replication factor of 2 is supported only when using flash-based OSDs for the pool.

EC pools support different EC profiles to better align with the nature of the data being stored, the durability requirements, and the cost objectives:

- ▶ 2+2: Two data chunks and two parity chunks (2 raw bytes are used for every usable byte that is stored.)
- ▶ 4+2: Four data chunks and two parity chunks (1.5 raw bytes are used for every usable byte that is stored.)
- ▶ 8+3: Eight data chunks and three parity chunks (1.375 raw bytes are used for every usable byte that is stored.)
- ▶ 8+4: Eight data chunks and four parity chunks (1.5 raw bytes are used for every usable byte that is stored.)

With erasure coding, you get a better raw to usable storage ratio and enhance the durability of your data so that it can survive up to four component failures without any data loss.

Tip: Erasure Coding 2+2 and Replicated Factor 2 provide the same raw to usable overhead. All other EC profiles provide a better effective capacity.

Key takeaway: IBM Storage Ceph erasure coding offers a storage-efficient way to ensure data durability and fault tolerance compared to replication. It is ideal for data that is less performance-critical but still requires durability and fault tolerance, such as backups, archives, or historical data.

3.4.2 Compression

For data lake and data lakehouse deployments, IBM Storage Ceph offers inline compression at the *RADOS Gateway Placement Target* level. The compression policy is configured at the placement level within the RGW and helps you optimize storage utilization within your data lake by automatically compressing data as it enters the system. It reduces storage requirements and can improve overall data processing efficiency.

You can configure placement targets to apply compression policies to specific data categories or workloads within your data lake.

Benefits of inline compression:

- ▶ Reduces storage requirements by compressing data dynamically.
- ▶ Potentially improves data processing efficiency due to smaller data sizes.

The supported compression plug-ins include the following ones:

- ▶ **Snappy:** A fast and lightweight algorithm, it is a good choice for general-purpose compression where speed is a priority. It offers a decent balance between compression ratio and decompression speed.
- ▶ **zlib (DEFLATE):** A widely used and well-established algorithm that is known for its good balance between compression ratio and decompression speed. It offers a denser compression compared to Snappy, but decompression might require slightly more processing power.
- ▶ **zstd (Zstandard):** A modern algorithm that is gaining popularity due to its ability to achieve high compression ratios with moderate decompression speed. It is a good option if storage efficiency is a top priority, and you can tolerate slightly slower decompression times compared to Snappy.

Choosing the right algorithm: The optimal compression algorithm for your IBM Storage Ceph storage depends on your specific needs and data types. Here are some factors to consider when making your choice:

- ▶ **Data type:** Different data types compress better with certain algorithms. Text files might compress well with zlib, but media files like images or audio might benefit more from zstd high compression capabilities.
- ▶ **Performance versus compression ratio:** Faster algorithms like Snappy offer a tradeoff by achieving lower compression compared to denser algorithms like zstd. If storage space is at a premium, zstd might be a good choice, but if faster data processing is crucial, Snappy might be preferable.
- ▶ **Processing power:** Decompressing data requires some processing overhead. Consider the resources that are available for decompression during data retrieval and choose an algorithm that balances compression efficiency with decompression speed based on your workload requirements.

Compression can be set at the creation time of the RGW placement or enabled afterward. Example 3-1 illustrates how you can update an existing placement target to enable compression through zlib.

Example 3-1 Updating an existing placement target to enable compression

```
$ radosgw-admin zone placement modify \  
    --rgw-zone {zone_name} \  
    --placement-id {placement_target_name} \  
    --storage-class {storageclass_name} \  
    --compression zlib
```

To disable compression, use an empty string such as "" or '' as the compression parameter.

Tips:

- ▶ Enabling compression does not compress the data that is already written unless it is updated or rewritten.
- ▶ Disabling compression does not decompress the data that is already written compressed unless it is updated or rewritten.

Note: All-flash [IBM Storage Ready Nodes](#) include hardware that is capable of Intel QuickAssist Technology (QAT) acceleration. IBM Storage Ceph is one of the software options for IBM Storage Ready Nodes.

Intel QAT is a hardware acceleration technology that offloads tasks from the CPU to dedicated hardware components on Intel processors. These components can perform certain tasks, like compression and decompression, faster than the CPU itself.

For more information about IBM Storage Ready Nodes, see [IBM Documentation](#).

3.4.3 Lifecycle management

Ceph RGW supports many bucket lifecycle management settings to make the best usage of your S3 compatible object store:

- ▶ Object transition
- ▶ Object expiration
- ▶ Versioning
- ▶ Object lock

Object transition migrates data between storage classes after a specific amount of time. The data transition feature empowers you to optimize storage efficiency and performance. It automatically migrates *hot data* (frequently accessed objects) to a high-performance RGW placement target. This target might use NVMe drives for exceptional speed.

Less frequently accessed data is automatically moved to a lower-cost yet still reliable RGW placement target. This target might leverage near-line SAS (NL-SAS) drives for cost-effective storage. It helps enhance your TCO by limiting the usage of expensive NVMe physical drives for older data.

Object expiration configures a bucket so that older data is automatically deleted after a specific amount of time or on a specific date without any action from any S3 client application. It lowers the overall space usage within your IBM Storage Ceph cluster and improves your TCO over time.

Object versioning keeps older versions of objects that are accessible within a bucket where this feature is enabled. The application or an administrator does not need to require data restoration from backup and restore software, and requires less time to get the old data back for reprocessing. It helps reduce the staff that is needed to get access to older version of your data without any physical media handling.

Object locking ensures that the data that is stored in a bucket cannot be deleted or overwritten. You do not need to deploy a specific third-party software or hardware to ensure that your data, once stored, is immutable and cannot be deleted.

To use any of the lifecycle management features, assign the appropriate policy to the bucket. The appropriate policy can even be assigned directly by the user based on their level of permissions on the bucket (for example, the owner of the bucket can assign any bucket policy).

For more information about lifecycle policies for IBM Storage Ceph, see [Bucket lifecycle](#).

Key takeaway: The bucket lifecycle configuration can automatically migrate objects to more cost-effective storage classes, archive inactive data, or even delete data based on your specific needs. It helps you optimize storage costs and ensure that your valuable data remains readily accessible when needed.

How to implement lifecycle policies in IBM Storage Ceph: For more information about how to implement lifecycle policies in IBM Storage Ceph, see 7.1.1, “Ingest: Lifecycle policy expiration” on page 85.

3.5 Data management

The Ceph Object Gateway excels in its ability to strategically place and manage data. System administrators can leverage placement targets and storage classes to fine-tune the gateway for diverse application needs. This granular control enables precise alignment with performance, availability, and cost objectives across various workloads. By optimizing storage utilization across the entire IBM Storage Ceph cluster, this adaptability enhances overall functions and resource efficiency.

Complementing the scalability features, Ceph Object Gateway boasts advanced data management functions. These features, like bucket notifications and the bucket inventory, facilitate data-driven strategies within the enterprise. Crucially, these features preserve the isolation between the client that is producing the data and the core architecture of the business, ensuring security and streamlined workflows.

IBM Storage Ceph offers two key features that improve data operations within the Ceph Object Gateway:

- ▶ **Bucket notifications:** Trigger automated actions in response to predefined events happening within a bucket. Enables real-time data management workflows.
- ▶ **Bucket inventory:** Facilitates efficient data organization, auditing, reporting, and compliance tasks. Bucket inventory streamlines essential data management activities.

3.5.1 Ceph Object Gateway placement targets

The operation of the Ceph Object Gateway involves the allocation of client bucket and object data to precise placement targets, which manage the storage of this data within corresponding pools. In the absence of configured placement targets by cluster administrators, the Ceph Object Gateway resorts to default targets and pools.

Furthermore, storage policies serve as a mechanism for Ceph Object Gateway clients to tailor their storage strategy to their specific requirements. These policies empower clients to designate particular types of storage media, such as NVMe drives, solid-state drives (SSDs), or SAS or SATA drives, each distinguished by unique attributes relating to durability, replication, erasure coding, and more.

Placement targets dictate the association of pools with individual buckets. When a bucket is created, its placement target is fixed and cannot be altered.

By using various placement targets within the Ceph Object Gateway, cluster administrators may precisely dictate the location of their data. For example:

- ▶ Strategically allocate bucket data to different performance tiers by statically assigning placement targets:
 - Bronze tier: Designate the placement target with a pool that is supported by rotational devices.
 - Silver tier: Designate the placement target with a pool that is supported by SSDs.
 - Gold tier: Designate the placement target with a pool that is supported by NVMe drives.
- ▶ Implement different data protection mechanisms for buckets based on their requirements:
 - Use replication for critical data, ensuring high redundancy.
 - Employ EC for other buckets to achieve superior storage efficiency, with slightly reduced redundancy.
- ▶ Tailor placement targets based on the nature of the data to be stored:
 - Archival data: Assign to a placement target by using rotational devices for cost-effective, long-term storage.
 - I/O sensitive workloads: Assign to a placement target by using all-flash devices for optimal performance.
- ▶ Optimize storage efficiency by configuring compression on select placement targets:
 - Allocate one placement target to compress data, enhancing storage efficiency.
 - Reserve another placement target for uncompressed data storage, ensuring data integrity and accessibility.

Ceph Object Gateway empowers administrators with seamless control over data storage during bucket creation. Through placement targets, they can define various storage configurations that are tailored to specific data management needs. These targets might prioritize performance, optimize cost, or ensure high availability.

Users benefit from this flexibility when creating buckets. They can select the most suitable placement target based on their data's characteristics. It simplifies storage allocation and ensures that data is placed in the most appropriate location within the IBM Storage Ceph cluster.

However, placement targets are *fixed* after a bucket is created. Choosing the right target during bucket creation is crucial for optimal data segmentation and management.

For more information, see “Bucket creation” on page 186.

3.5.2 Ceph Object Gateway Storage classes

Placement targets in Ceph Object Gateway serve as a valuable feature for statically assigning the physical storage location of your data on bucket creation. However, there are instances where it becomes advantageous to provide users with the capability to dynamically migrate their data between various logical partitions based on predefined criteria, such as time. This function is facilitated by Storage classes in Ceph Object Gateway, which are governed by S3 Bucket Lifecycle rules, enabling the automation of object transitions between Storage classes.

With Storage classes, users may categorize their data and define lifecycle policies that automatically move objects between different storage tiers over time. This dynamic data management approach enables optimized resource utilization and cost efficiency because data can be stored on appropriate storage tiers based on its access patterns, importance, or other criteria.

By leveraging S3 Bucket Lifecycle rules, administrators can configure policies to automatically migrate data between Storage classes, ensuring that the data remains aligned with changing business requirements and data lifecycle stages. This comprehensive data management framework enhances the agility and scalability of Ceph Object Gateway deployments, empowering users to effectively manage their data storage and access needs over time.

When you use Storage classes in Ceph Object Gateway, users may strategically determine the placement of their data based on time, enabling various optimization strategies:

- ▶ Implementing diverse data protection mechanisms for objects according to their requirements and access patterns:
 - Employ replication for critical data with high access frequency, ensuring robust redundancy.
 - Use EC for less frequently accessed objects, prioritizing storage efficiency while maintaining adequate redundancy.
- ▶ Tailoring Storage classes to suit the characteristics of the data being stored:
 - For archival data, assign a Storage class by using rotational devices for cost-effective long-term storage.
 - For I/O sensitive workloads, designate a Storage class by using all-flash devices to maximize performance for frequently accessed objects.
- ▶ Optimizing storage efficiency through compression within select Storage classes:
 - Allocate one Storage class for compressing data, enhancing storage efficiency for objects accessed less frequently.
 - Reserve another Storage class for uncompressed data storage, ensuring data integrity and accessibility for objects requiring immediate access without compression overhead.

By leveraging these capabilities within Storage classes, users can effectively manage their data storage, balancing performance, cost, and efficiency based on the specific requirements of their workloads.

In addition to using S3 Bucket Lifecycle policies to seamlessly migrate objects between different Storage classes based on predefined transition rules, users also may override the default Storage class when creating an object. This goal can be achieved by including the HTTP header `X-Amz-Storage-Class` along with the request.

This feature empowers users to exert granular control over the storage characteristics of individual objects so that they can specify the Storage class at the time of object creation. By providing this HTTP header, users can tailor the storage attributes of each object to align with specific performance, durability, and cost requirements, regardless of the default Storage class settings.

This level of customization ensures that users can optimize their data storage strategy on a per-object basis, enabling efficient resource utilization and cost management within their Ceph Object Gateway environment.

A standout feature of this capability is the ability to use source filters (prefixes), enabling the migration or expiration of only wanted objects.

When employing Amazon Web Services (AWS) S3 software development kits (SDKs) such as boto3, ensure that non-default storage class names are matched with the ones that are defined by AWS S3. Otherwise, the SDK rejects the request and generates an exception. This situation is why you should adhere to AWS S3 storage class naming conventions to ensure seamless interaction with the SDK and prevent errors during data storage operations. For more information about the naming convention that AWS uses to define the non-default Storage classes, see [AWS Documentation](#).

This example demonstrates how to effectively use Storage classes in Ceph Object Gateway along with S3 Bucket Lifecycle policies to manage object migrations and expirations based on the duration of storage within the IBM Storage Ceph cluster.

Initially, we configured our default Storage class to store objects in a pool by using all-flash devices, ensuring optimal performance for frequently accessed objects. The class is denoted as STANDARD. However, we aim to schedule migrations for these objects to a Storage class by using a pool with more cost-effective and less performant devices after a period of 30 days. This class is designated as STANDARD_IA.

Also, after objects are stored for a year, it becomes unnecessary to retain them further. Therefore, our S3 Bucket Lifecycle policy should be configured to encompass the following actions:

- ▶ Migrate objects from the STANDARD Storage class to the STANDARD_IA Storage class after 30 days of storage.
- ▶ Expire objects after 1 year of storage to ensure efficient resource utilization and storage management.

By implementing the comprehensive S3 Bucket Lifecycle policy that is shown in Example 3-2, we can effectively optimize storage costs, performance, and resource utilization within our Ceph Object Gateway environment while ensuring compliance with data retention requirements.

Example 3-2 S3 Bucket Lifecycle policy

```
<LifecycleConfiguration>
  <Rule>
    <ID>Archive all objects older than 30 days and delete the objects after a
year</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Key takeaway: With Ceph Object Gateway Storage classes, you can tailor your storage infrastructure to your specific data access patterns and budget constraints, leading to improved efficiency, cost optimization, and better data management.

How to implement storage classes in IBM Storage Ceph: For more information about how to implement storage classes in IBM Storage Ceph, see Refer to 7.2.7, “Raw zone: Data tiering configuration - creating a cold storage class” on page 102.

3.5.3 Ceph Object Gateway bucket notifications

One of the most notable features of object storage, especially within the context of Ceph Object Gateway, is its support for bucket notifications. This feature enables users to receive notifications about various events occurring within their object storage environment, empowering them to take automated actions in response.

With bucket notifications, users can configure different events to trigger notifications to designated services, allowing for seamless integration with downstream systems and workflows. For example, events such as object creation, deletion, or modification can serve as triggers for notifications, enabling users to react dynamically to changes in their storage environment.

This capability opens up a wide range of possibilities for automating tasks and implementing event-driven architectures. For example, notifications can be used to trigger data processing pipelines, initiate backup procedures, synchronize data across multiple systems, or update external databases or applications in real time.

By leveraging bucket notifications, users can enhance the agility, efficiency, and reliability of their object storage deployments. They can build robust, automated workflows that respond swiftly to changes and events within the storage environment, improving productivity, reducing operational overhead, and enabling more seamless integration with other systems and services.

Here is a detailed list of potential benefits:

- ▶ **Flexible integration:** Ceph Object Gateway S3 bucket notifications offer seamless integration with existing infrastructure and applications, enabling users to leverage familiar tools and workflows.
- ▶ **Event-driven architecture:** By responding to S3 bucket events, users can build event-driven architectures and automate workflows. For more information about implementing an event-driven architecture with IBM Storage Ceph, see 7.2.9, “Raw zone: Serverless data pipeline workflow” on page 108.
- ▶ **Scalability and performance:** Ceph Object Gateway is designed for scalability and high-performance object storage. S3 bucket notifications scale with the size and complexity of the IBM Storage Ceph cluster, ensuring reliable event delivery and efficient processing.
- ▶ **Customizable actions:** Users can define custom actions for running in response to S3 bucket events, such as invoking external APIs, running scripts, or triggering business logic.
- ▶ **Real-time monitoring and alerting:** S3 bucket notifications provide real-time visibility into object-related activities within IBM Storage Ceph clusters, enabling users to monitor, track, and analyze changes as they occur. It facilitates proactive monitoring, troubleshooting, and alerting.

- **Enhanced data management:** Notifications can be used to enforce data management policies, implement access controls, and enforce compliance requirements, contributing to a secure and well-managed storage environment.

Overall, Ceph Object Gateway S3 bucket notifications offer a robust mechanism for event-driven architecture, enabling users to build scalable, responsive, and automated applications while enhancing visibility, efficiency, and security in their IBM Storage Ceph storage workflows.

In the upcoming IBM Storage Ceph 8 release, the enhanced notification_v2 format for bucket notifications and topics will introduce robust support for multisite replication and demonstrate remarkable scalability across numerous topics. This advancement represents a significant leap forward in the capabilities of IBM Storage Ceph storage, particularly in distributed environments requiring seamless data replication and extensive topic management.

Ceph Object Gateway offers the capability to send notifications to various destinations, including HTTP endpoints, AMQP 0.9.1 endpoints, and Kafka endpoints. For more information and detailed instructions about configuring the different providers, see [Bucket notifications](#).

Event types

Table 3-1 provides a comprehensive list of the event types that can be used to trigger an S3 bucket notification in Ceph Object Gateway.

Table 3-1 Event types that can be used to trigger an S3 bucket notification

Event	Subevent
s3:ObjectCreated:	* (all subevents)
	Put
	Post
	Copy
	CompleteMultipartUpload
s3:ObjectRemoved:	* (all subevents)
	Delete
	DeleteMarkerCreated
s3:ObjectLifecycle:Expiration:	Current
	NonCurrent
	DeleteMarker
	AbortMultipartUpload
s3:ObjectLifecycle:Transition:	Current
	NonCurrent
s3:ObjectSynced:	* (all subevents)
	Create
	Delete
	DeletionMarkerCreated

Event	Subevent
s3:Replication:	* (all subevents)
	Create
	Delete
	DeletionMarkerCreated

Notification filtering

In Ceph Object Gateway S3 bucket notifications, there are several options that are available for filtering when a notification is triggered:

- ▶ **Prefix/Suffix filtering:** Filters notifications based on the prefix (the beginning) or suffix (the end) of object keys. It is a simple and effective way to filter objects based on their names.
- ▶ **Regular expression matching:** With regular expression filtering, you can define complex patterns by using regular expressions to match object keys. This approach provides flexibility in specifying criteria for filtering notifications beyond simple prefixes or suffixes.
- ▶ **Metadata attribute filtering:** Objects in Ceph Object Gateway can have associated metadata attributes, which are key-value pairs that provide more information about the object. With metadata attribute filtering, you can filter notifications based on the values of these attributes, offering more granular control.
- ▶ **Object tag filtering:** Object tagging enables you to assign custom tags to objects, enabling categorization and organization. Object tag filtering lets you specify certain tags or combinations of tags to filter notifications, triggering notifications based on the presence or absence of specific tags that are associated with objects.

These filtering options offer a comprehensive range of methods to customize the notification system based on object keys, metadata attributes, and tags to cater to diverse use cases and requirements.

Notifications reliability

Notifications can be dispatched either synchronously or asynchronously, each exhibiting distinct characteristics in terms of latency and reliability. This section delves into the expected performance and dependability for both synchronous and asynchronous notification methods.

Synchronous notifications occur immediately as part of the original operation. In this mode, the operation is not considered complete (acknowledged) until the notification successfully reaches its destination (the topic's endpoint). The overall time that it takes for the notification to be sent and confirmed adds to the total wait time (latency) for the original operation.

Unlike synchronous notifications, which are delivered immediately, *asynchronous notifications* are first saved permanently (persistently stored) and then sent to the configured destination (the topic's endpoint) later. The initial operation experiences a delay only when the notification is saved.

To minimize this delay, consider placing the Ceph Object Gateway "log" pool on high-performance storage. This separation (decoupling) between notification delivery and operation execution improves responsiveness by ensuring the operation itself is not slowed down.

Moreover, the reliability of synchronous notifications is bolstered by the immediate acknowledgment on dispatch, ensuring prompt confirmation of notification delivery. Conversely, asynchronous notifications, while offering reduced latency for the original operation, might exhibit a slight delay in notification dispatch due to the asynchronous delivery mechanism. However, their reliance on persistent storage enhances resilience, ensuring that notifications are preserved and eventually dispatched even in system failures or disruptions.

How to implement bucket notifications in IBM Storage Ceph: For more information about how to implement bucket notifications in IBM Storage Ceph, see 7.2.2, “Raw zone: S3 bucket notification configuration” on page 95.

3.6 Regional data resiliency in IBM Storage Ceph

You can ensure regional data resiliency in IBM Storage Ceph by using the Ceph Object Gateway Multisite feature, which enables the deployment of multiple instances of Ceph Object Gateway across different geographical locations while maintaining data consistency and accessibility.

Ceph Object Gateway Multisite has the following features:

- ▶ **Geographical distribution:** By using Ceph Object Gateway Multisite, you can deploy multiple Ceph Object Gateway instances in various geographical locations or data centers. This distribution ensures proximity to users and enables data redundancy and disaster recovery (DR) capabilities.
- ▶ **Bucket replication:** One of the key functions of Ceph Object Gateway Multisite is bucket replication. You can configure either all or specific buckets to be replicated across multiple Ceph Object Gateway instances in different locations. This replication process ensures that data is synchronized between the source and destination buckets in near real-time.
- ▶ **Data consistency:** Ceph Object Gateway Multisite employs the technique of eventual consistency to maintain data consistency across replicated buckets. Replication is asynchronous, meaning that changes that are made to data are propagated across instances with some delay, and it might take time for all instances to synchronize fully.
- ▶ **Active-Active configuration:** Ceph Object Gateway Multisite supports active-active configurations, allowing all instances of Ceph Object Gateway to actively serve client requests, providing load-balancing and scalability benefits. Only the master zone within the zonegroup accepts metadata updates, such as creating users or buckets.
- ▶ **DR:** By replicating data across multiple geographical locations, Ceph Object Gateway Multisite enhances DR capabilities. In a failure or disaster in one location, data remains accessible from other locations, ensuring business continuity and minimal downtime.

Overall, Ceph Object Gateway Multisite provides a robust solution for deploying scalable, highly available, and geographically distributed object storage services that are suitable for a wide range of use cases, including content delivery, backup and archiving, and enterprise storage.

Multisite replication

Since the release of Ceph Jewel (released on 1 April, 2016, it has been possible to use multisite replication in Ceph Object Gateway. When using traditional multisite replication, all objects in all buckets in the multi-zone configuration are replicated, without the possibility to

select which buckets should be replicated and which should not. In newer versions, a new feature called *multisite bucket-granularity sync policies* can define the replication per bucket.

Using multisite replication is highly effective for replicating data across different locations worldwide. Performing this replication at the storage layer is the most efficient approach because it eliminates the need for the application to manage data consistency across multiple locations.

There are different varieties or configurations when setting up multisite replication:

- ▶ Multi-zone
- ▶ Multi-zone groups
- ▶ Multiple realms

Multi-zone

This setup adopts a sophisticated structure. It is composed of a single zonegroup and multiple zones, with each zone containing one or more Ceph Object Gateway instances. Moreover, each zone is supported by its own dedicated IBM Storage Ceph Cluster.

The inclusion of multiple zones within a specific zonegroup serves as a safeguard against disasters affecting individual zones. In a substantial failure in one zone, the presence of other zones ensures continuity. Furthermore, each zone remains active and capable of receiving write operations. Such a multi-zone setup, with several active zones, strengthens DR measures and serves as a solid basis for establishing content-delivery networks.

In the latest versions of IBM Storage Ceph, Dynamic Bucket Index Resharding is supported in multi-zone environments. Each shard within the bucket index efficiently manages its entries until it reaches a specific threshold. When this threshold is surpassed, performance issues might arise. The dynamic resharding feature identifies this condition and automatically increases the number of shards that are used by a bucket's index, which reduces the number of entries in each shard.

Replication operates in an active/active mode for data access, so users may read/write simultaneously from their nearest S3 endpoint location. It enables faster data access and minimizes downtime.

However, only the specified master zone within the zonegroup accepts metadata updates. For example, when creating users and buckets, any metadata changes in non-master zones are relayed to the designated master. In a master failure, a manual failover of the master zone must be initiated.

Multi-zone groups

The Ceph Object Gateway facilitates the usage of multiple zonegroups, with each zonegroup composed of one or more zones. If two zones belong to the same zonegroup, and that zonegroup shares the realm as another zonegroup, the objects that are stored within these zones share a unified object namespace. This approach ensures that object IDs remain unique across both zonegroups and zones.

Ownership of each bucket lies with the zonegroup where it was initially created, and the replication of its object data extends solely to other zones within that zonegroup. Any data requests for buckets originating from other zonegroups are redirected to the zonegroup where the bucket is.

The creation of multiple zonegroups can prove beneficial when you want to establish a shared namespace for users and buckets across numerous zones while segregating object data to specific subsets of those zones.

Alternatively, in scenarios needing data isolation across distinct realms, each realm may have its own single zonegroup. Zonegroups offer flexibility by enabling separate control over data and metadata isolation.

Multiple realms

Realms enable the establishment of policies that are applicable across multiple zonegroups. Each realm possesses a globally unique namespace and can encompass either a single zonegroup or multiple zonegroups. Opting for multiple realms enables the definition of distinct namespaces and configurations, granting flexibility wherein each realm can possess configurations that are independent of ones in other realms. For example, it is not possible to have duplicate bucket or usernames within a single realm.

Figure 3-3 shows the multisite one-realm configuration.

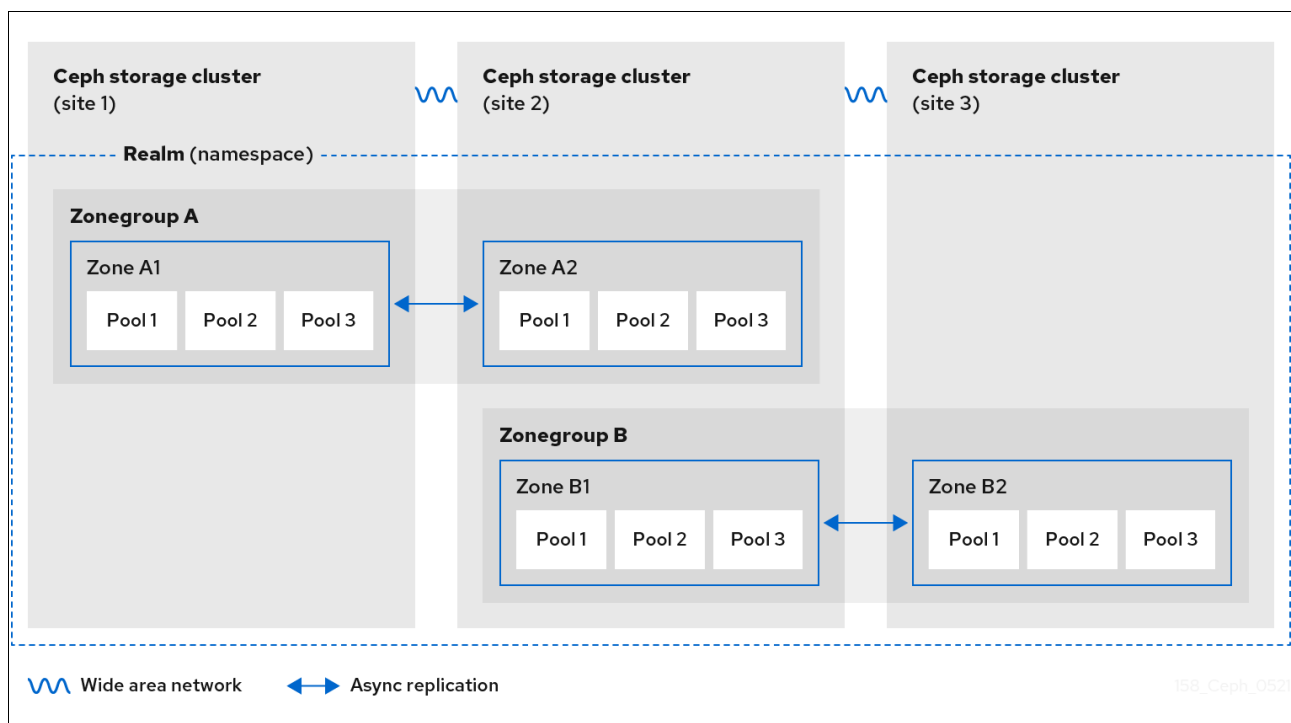


Figure 3-3 Multisite one-realm

A realm serves as a globally unique namespace encompassing one or more zonegroups. Zonegroups are one or more zones, with zones containing buckets, and buckets containing objects. Realms enable the Ceph Object Gateway to accommodate multiple namespaces and their configurations concurrently on a single hardware platform.

Each realm is linked to a period, representing the status of zonegroup and zone configurations at a given time. Whenever modifications are made to a zonegroup or zone, it is essential to update and commit the associated period.

Figure 3-4 on page 40 shows the multisite two-realms configuration.

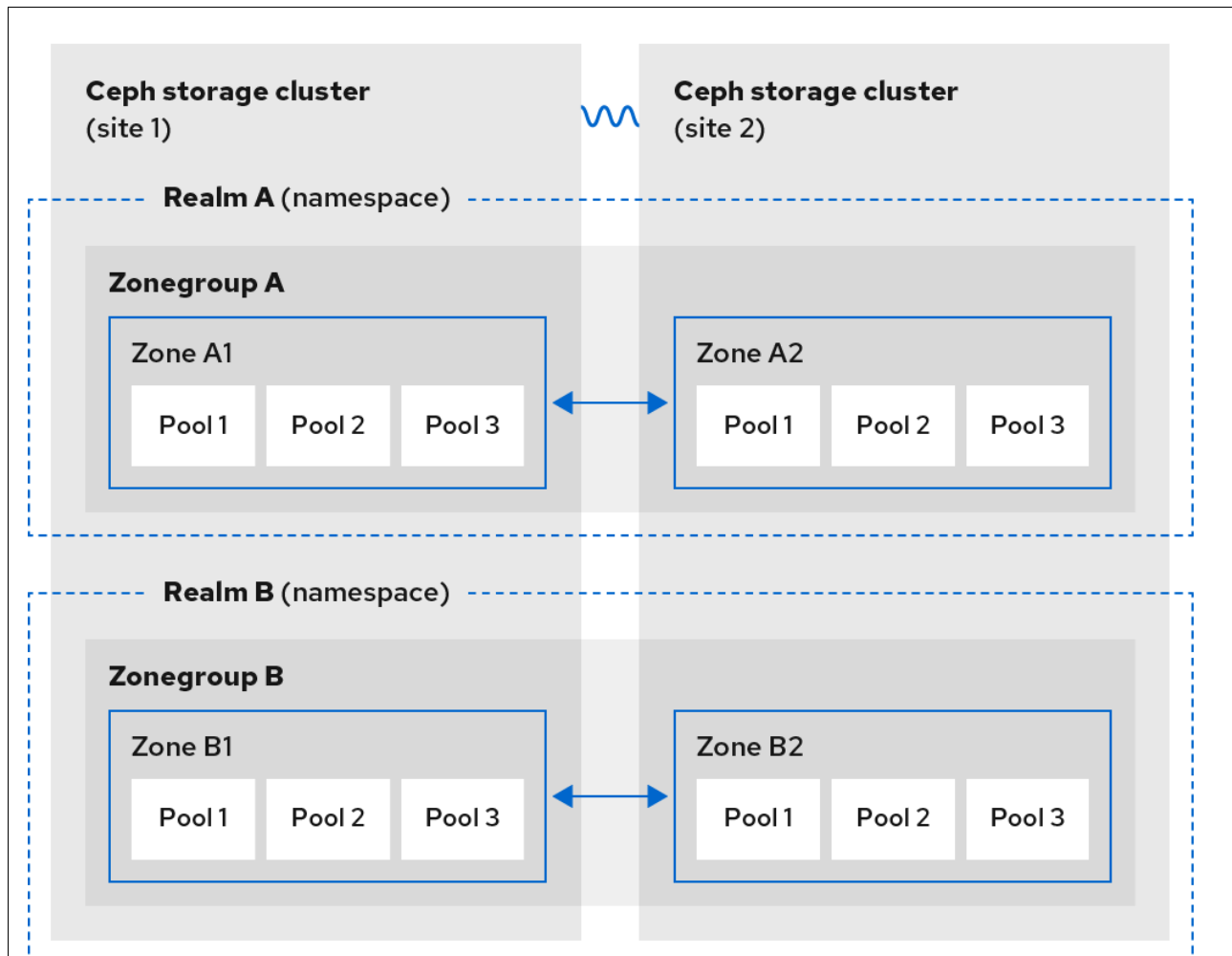


Figure 3-4 Multisite two-realms

With the flexibility that is offered by realms, zonegroups, and zones, the Ceph Object Gateway enables you to tailor your data lakes for data replication across diverse global locations. This approach ensures data consistency and provides robust monitoring metrics to verify that your data is effectively replicated in each zone.

Dedicated Ceph Object Gateway instances for replication

The default configuration of Ceph Object Gateway services includes managing both public-facing S3 requests and replication requests between sites, with shared resources and processing time between these tasks. To enhance this setup, it is a best practice to allocate specific groups of Ceph Object Gateway instances to handle public S3 requests and multisite replication requests between two IBM Storage Ceph clusters.

Although not mandatory, this approach offers several benefits:

- ▶ **Scalability:** With dedicated resource sets for public and replication tasks, you can independently scale Object Gateway instances based on performance requirements, such as increased throughput.
- ▶ **Avoidance of blocking:** Segregated Object Gateway instances prevent sync replication blocking due to the instances being occupied with client-facing tasks or vice versa.

- Improved troubleshooting: Dedicated Object Gateway sets streamline troubleshooting by targeting specific Object Gateway instances based on the issue. This separation also ensures that replication messages do not interfere with client logs or vice versa.
- Enhanced security: Using different Object Gateway sets enables the use of distinct networks with varying security levels, firewall rules, and operating system (OS) security measures. For example, public-facing Object Gateway instances could operate on Network A, and replication Object Gateway instances could use Network B.
- Dedicated network for replication traffic: Dedicated network infrastructure for replication and client traffic ensures no competition for resources, potentially enhancing replication performance.

By default, all Object Gateway instances participate in multisite replication, but two steps are required to remove an Object Gateway instance from this process:

1. Set the `rgw_run_sync_thread` parameter to `false` in the Object Gateway instance. This setting prevents the Object Gateway from transmitting multisite replication data.
2. To avoid receiving replication data, remove Object Gateway instances from the zonegroup and zone replication endpoints.

How to implement dedicated Ceph Object Gateway instances in IBM Storage Ceph:

For more information about how to implement dedicated Ceph Object Gateway instances in IBM Storage Ceph, see this [IBM Storage Ceph Object Storage Multisite Replication Serie Part Three](#).

Multisite bucket-granularity sync policies

The multisite bucket-granularity sync policy provides precise control over how data is transferred between buckets across different zones, expanding on the capabilities of zone sync. Previously, buckets were treated uniformly, with each data zone containing an identical copy of the bucket, ensuring consistency across all zones. However, with the bucket-granularity sync policy, buckets can now vary, enabling one bucket to retrieve data from others in different zones. As a result, the synchronization process, which previously assumed that the source and destination buckets were always the same, can now accommodate this variation.

This synchronization policy enables the creation of multiple groups, each capable of managing lists of data-flow configurations and pipe configurations:

- Data-flow configurations determine how data moves among different zones, supporting both symmetrical data flow, where multiple zones exchange data bidirectionally, and directional data flow, where data moves unilaterally between zones.
- A pipe defines which buckets use specific data flows and includes their associated properties. Meanwhile, a synchronization policy group can be found in three different states, as shown in Table 3-2.

Table 3-2 Synchronization policy group states

Value	Description
enabled	Synchronization is allowed and enabled.
allowed	Synchronization is allowed.
forbidden	Synchronization is not allowed and can override other groups.

The synchronization policy enables the creation of multiple groups, each capable of managing lists of configurations for data flow and pipes. These configurations determine how data moves between different zones, supporting both symmetrical data flow, where multiple zones exchange data bidirectionally, and directional data flow, where data moves unilaterally between zones.

Policies can be defined at the bucket level, inheriting the data flow that is specified in the zonegroup policy but with limitations on what can be defined compared to the zonegroup policy.

When using wildcard zones or wildcard bucket parameters in the policy, all relevant zones or buckets are included. In the context of a bucket policy, this situation refers to the current bucket instance. For DR, entire zones can be mirrored without bucket-level configuration. For more granular control, configure synchronization pipes at the zone group level, and then enable specific bucket synchronization at the bucket level. If necessary, the bucket-level policy can restrict data movement to specific relevant zones.

Figure 3-5 shows multisite bucket-granularity.

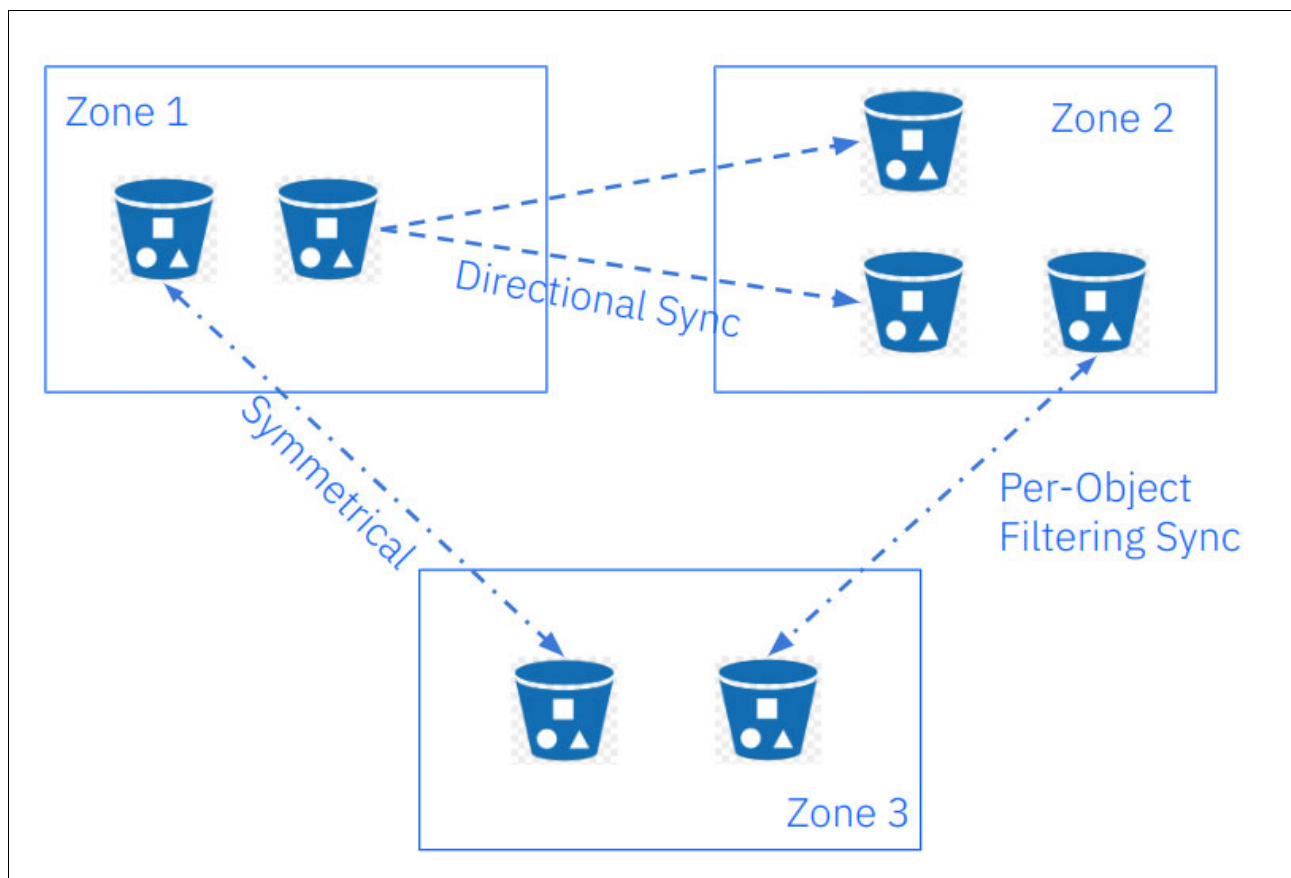


Figure 3-5 Multisite bucket-granularity

Multisite bucket-granularity sync policies offer users increased flexibility and cost savings, unlocking a range of valuable replication features:

- ▶ Replicating entire zones while selectively excluding certain buckets from replication.
- ▶ Replicating specific buckets while selectively excluding entire zones from replication.
- ▶ Implementing bidirectional and unidirectional data flow configurations for each bucket.
- ▶ Replicating one source bucket across multiple destination buckets in different zones.
- ▶ Users may enable or disable synchronization for each individual bucket, allowing for precise control over replication processes.

At the time of writing, buckets must be in different zones. Users cannot enable replication between buckets in the same zone.

A standout feature of this capability is its ability to use source filters, enabling the replication of only wanted objects. This task can involve replicating objects with specific prefixes or ones that are tagged appropriately.

By using multisite bucket-granularity sync policies in your data lakes, you can selectively replicate specific buckets and objects to other zones. For example, if you have a bucket that is dedicated to ingesting data from your edge locations, you might not want to replicate this data until the objects are validated, organized, and cleaned. When these tasks are complete, then you can move the objects to another bucket that is designated for replication so that your data scientists can extract and use the data to enhance your business. This solution is built in to IBM Storage Ceph, so you do not need any external tools to replicate your data to other locations.

How to implement multisite bucket-granularity sync policies in IBM Storage Ceph:

For more information about how to implement multisite bucket-granularity sync policies in IBM Storage Ceph, see [IBM Storage Ceph Object Storage Multisite Replication Series Part Five](#).

How to implement multisite replication in IBM Storage Ceph: For more information about how to implement multisite replication in IBM Storage Ceph, see [IBM Storage Ceph Object Storage Multisite Replication Series Part One](#).

3.6.1 Archive zone

Although many third-party backup solutions can store backups in AWS S3 compatible buckets by using the S3 protocol, finding solutions that can back up the data within those AWS S3 compatible buckets is less common.

Backing up object storage solutions containing billions of objects and petabytes of data presents unique challenges. These challenges include efficiency (completing the backup in a reasonable time frame), cost (storing a massive backup can be expensive), and choosing the appropriate destination for the backup data.

One of the main reasons that is cited by traditional object storage providers for avoiding backing up the data is the outstanding reliability of this type of solution. Replication and erasure coding mechanisms offer excellent protection against hardware or zone failures. However, *these techniques are not backups*. For example, any accidental deletion or a ransomware attack is automatically replicated (either synchronously or asynchronously), potentially causing irreparable damage to the data and rendering it unusable.

Many object storage users protect their data by enabling versioning and the MFA delete in their buckets. This approach is valid when dealing with a single zone. However, when

replicating data to multiple locations, this approach becomes impractical. Enabling versioning features in each zone consumes more capacity and performance than necessary.

The *archive zone feature* provides a solution for effectively managing S3 object versions.

Archive zone feature: A few years ago, Red Hat collaborated with a customer to address their data replication needs. They explored various capabilities within IBM Storage Ceph (formerly Red Hat Ceph Storage) to find the optimal solution for replicating the customer's data across multiple locations.

Through this collaboration, they identified the *archive zone* feature as the most suitable approach. Here is why the archive zone proved to be the ideal solution:

- ▶ The archive zone feature implements a specific archiving behavior that is associated with a single zone without affecting other zones in the zonegroup.
- ▶ An archive zone enables the retention of version history for S3 objects, which can be deleted only through the gateways that are associated with the archive zone.
- ▶ This function proves beneficial in configurations where several, non-versioned zones replicate their data and metadata through their zone gateways, ensuring high availability for users while the archive zone captures all data updates.

Figure 3-6 shows a visual representation of how archive zones manage S3 object versions.

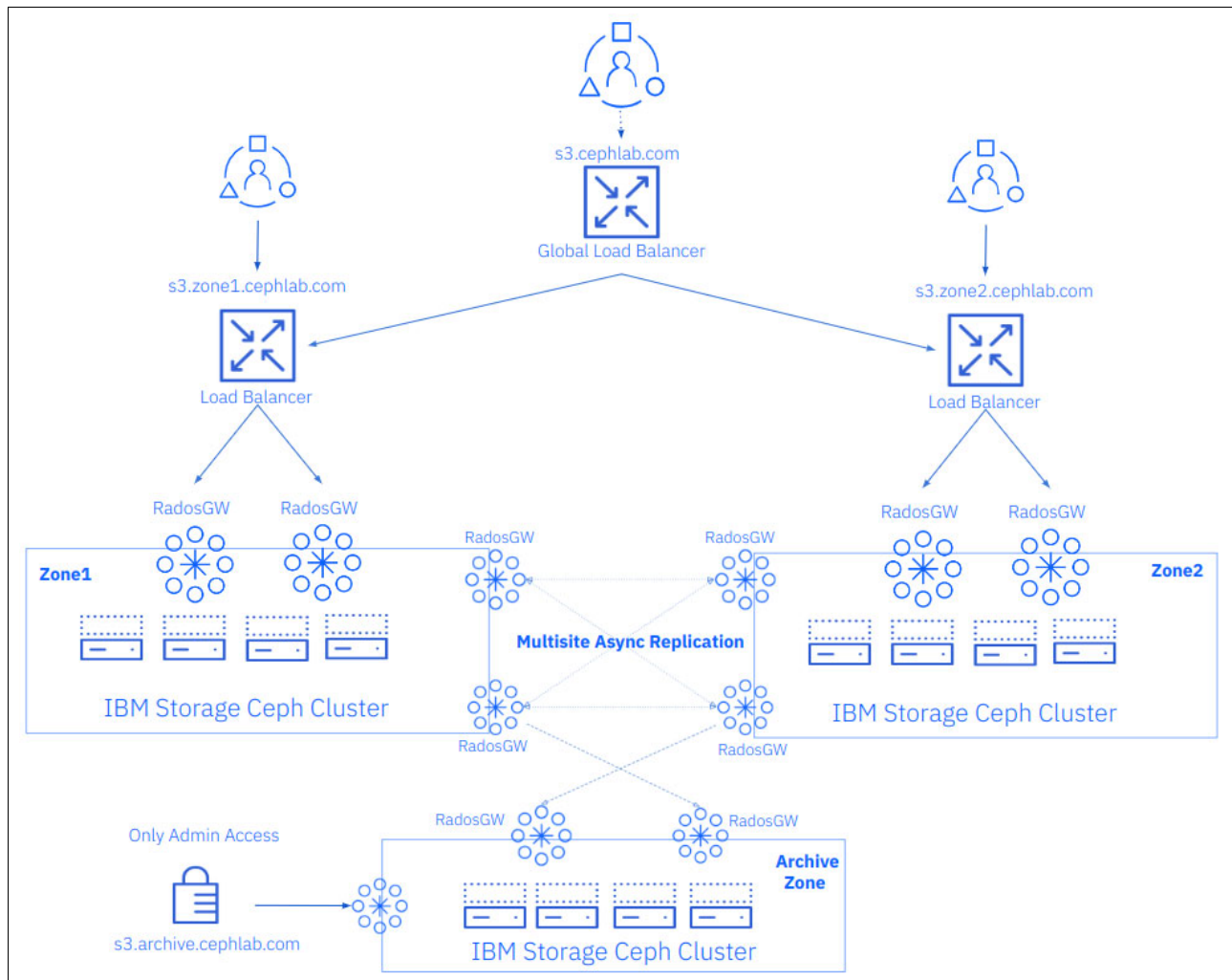


Figure 3-6 Visual representation of how archive zones manage S3 object versions

By leveraging the archive zone feature, you can achieve the following objectives:

- ▶ Reduce the number of copies (versions) of S3 objects across all zones within the zonegroup by enabling versioning exclusively in the archive zone, rather than in other zones.
- ▶ Safeguard and preserve a comprehensive history of S3 objects beyond intentional or unintentional updates that are made by legitimate users in non-archive zones.

Traditional backup methods, with their scheduled intervals, can leave your data vulnerable during gaps between backups. An archive zone in IBM Storage Ceph takes a different approach, offering continuous data protection for superior robustness and effectiveness.

Here is how an archive zone safeguards your data:

- ▶ Automatic versioning: Every time that a user object is updated, the archive zone automatically creates a new version. This approach eliminates the need for scheduled backups and ensures that all changes are captured promptly.
- ▶ Near real-time resilience: By continuously capturing data updates, the archive zone provides near real-time data resilience. You can be confident that any modifications are immediately protected within the archive.

The IBM Storage Ceph archive zone feature boasts the following key attributes:

- ▶ Versioning is automatically enabled for all buckets within the Ceph Object Gateway archive zone, ensuring a comprehensive history of object modifications.
- ▶ With each new object that is uploaded by a user, the archive zone seamlessly replicates the data asynchronously, providing immediate protection against data loss.
- ▶ Every modification that is made to objects within the production zone triggers the generation of a new object version within the archive zone, including delete operations. This approach ensures that a complete record of object changes is maintained for audit and recovery purposes.
- ▶ Immutability is ensured within the archive zone, safeguarding deleted objects from permanent loss. Even if an object is deleted in the production zone, it remains intact within the archive zone, preserving data integrity and compliance requirements.

However, although the archive zone ensures immutability, it does not impose object locks on ingested objects. Objects within the archive zone remain deletable if users possess the necessary permissions through the S3 endpoint.

- ▶ Even if one of our users accidentally delete the entire bucket, you ensure data preservation by atomically renaming the bucket in the archive zone. The renaming follows a specific format:

`'bucket-name-deleted-hash'`

The hash calculation incorporates unique values from the bucket metadata, including the bucket creation date. This approach effectively mitigates potential conflicts and collisions in the naming of buckets within the archive zone.

You may use both the archive zone feature and multisite bucket-granularity sync policies concurrently, which grants precise control over which buckets or individual objects within your storage environment are sent or replicated to the archive zone. This flexibility empowers you to selectively choose which data deserves the added protection of the archive zone.

For example, in scenarios where certain buckets contain disposable or non-critical data, you can opt to exclude their objects from replication to the archive zone. By doing so, you effectively conserve storage capacity within the IBM Storage Ceph cluster and mitigate the transmission of unnecessary data across your network infrastructure. This targeted approach ensures that only essential data is archived, optimizing resource utilization and streamlining data management processes.

One of the primary challenges that is encountered when overseeing the IBM Storage Ceph cluster and activating the archive zone feature is the accumulation of objects (or versions of objects) over time. Because these objects are safeguarded from automatic deletion by users, the volume of stored data can escalate rapidly, posing a potential management issue. To address this concern effectively, implementing lifecycle policies becomes crucial.

Lifecycle policies offer a strategic solution by enabling administrators to specify the number of versions to retain for each object within the IBM Storage Ceph cluster that is designated as the archive zone. By configuring these policies, organizations can proactively manage data retention, ensuring optimal resource utilization and mitigating the risk of storage overload.

How to implement an archive zone in IBM Storage Ceph: For more information about how to implement archive zone in IBM Storage Ceph, see [IBM Storage Ceph Object Storage Multisite Replication Series Part Seven](#).



Replacing Hadoop Distributed File System with IBM Storage Ceph Object Storage

IBM Storage Ceph Object Storage can be a compelling alternative to Hadoop Distributed File System (HDFS), especially for organizations seeking a more scalable, flexible, and potentially cost-effective storage solution. This chapter explores migrating from HDFS to IBM Storage Ceph Object Storage.

This chapter has the following sections:

- ▶ Hadoop introduction
- ▶ HDFS introduction
- ▶ The rise of object storage
- ▶ Accessing S3 Object Storage from Hadoop (Amazon S3A File System)
- ▶ IBM Storage Ceph Object Storage integration with Hadoop
- ▶ IBM Storage Ceph Object Storage and S3A
- ▶ Example of connecting Hadoop with IBM Storage Ceph by using S3A
- ▶ Migrating from HDFS to S3A
- ▶ A real-world HDFS to S3A migration example

4.1 Hadoop introduction

Hadoop is an open-source framework for the efficient storage and processing of large data sets across clusters of computers.

“The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing. The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers by using simple programming models. It is designed to scale from single servers to thousands of machines, each offering local computation and storage. Rather than relying on hardware to deliver high availability, the library itself is designed to detect and handle failure at the application layer, delivering a highly available service on top of a cluster of computers, each of which may be prone to failures.”¹

4.1.1 The birth of Hadoop: A response to big data challenges

In the early 2000s, as the web exploded with data, engineers Doug Cutting and Mike Cafarella realized existing search engine technology could not keep up. To tackle this challenge, they created *Nutch*, a system that was designed for distributed data processing across multiple computers, which could analyze massive data sets efficiently.

As Nutch, a pioneering web crawler, gained traction, its creators recognized the limitations of its initial storage and processing infrastructure. This realization sparked a collaboration with the Apache Software Foundation (ASF), an incubator for open-source projects. In 2005, a pivotal step was taken: the donation of Nutch's core functions (the MapReduce framework for distributed data processing and its file system) to the ASF. This action marked the genesis of a new project that was named *Hadoop* after Doug Cutting's son's stuffed elephant.

Hadoop's open-source nature fostered a global developer community. With continuous contributions adding new features and functions, Hadoop's popularity skyrocketed in the tech world. Organizations embraced its potential for distributed computing, paving the way for big data analytics as we know it today.

4.1.2 Hadoop's rise: Powering the big data revolution

Hadoop's potential was not limited to search engines. Yahoo! announced in 2008 that it was using Hadoop to power its search engine.² Other major companies like Facebook, LinkedIn, and Twitter followed suit, recognizing the power of distributed computing for handling their massive data sets.

Even today, Hadoop remains a crucial tool for organizations seeking to unlock valuable insights from their data. It played a pivotal role in the big data revolution, empowering users to process and analyze information at an unprecedented scale.

Hadoop's story exemplifies the transformative power of open-source collaboration. By leveraging the collective ingenuity of a global developer community, it transformed how we approach data processing and analysis.

¹ <https://hadoop.apache.org/>

² <https://www.wired.com/2011/10/how-yahoo-spawned-hadoop/>

Hadoop has been successful due to several features.

- ▶ Hadoop is an open-source project, which has resulted in a vast community of developers who contribute to the project, add new features, and support users.
- ▶ Hadoop was a highly scalable solution compared to the other technologies that were available at the time, making it an excellent solution for managing large volumes of data with a wide range of data types. Its distributed architecture enabled data to be processed in parallel across a cluster of computers, making it capable of handling massive data sets.
- ▶ At the time, Hadoop was a cost-effective solution for managing large volumes of data compared to its high-end data warehouse competitors. It can operate on commodity hardware, making it a more affordable alternative to traditional data processing systems.
- ▶ Hadoop has a rich ecosystem of tools and technologies that can be used to process, analyze, and visualize data, such as Hive, Pig, Spark, and HBase, which provide more functions and capabilities to Hadoop users.

Over the years, Hadoop and HDFS faced challenges, some of which were resolved while others remain intrinsic to the architecture. Here are some examples:

- ▶ In HDFS, a historical issue is related to poor performance with small files. HDFS is designed to work efficiently with few large files for storing large data sets. If too many small files exist, performance tends to drop. For more information, see [The Small Files Problem - Cloudera Blog](#).
- ▶ While Hadoop and its MapReduce framework were instrumental in big data processing, they can struggle with speed and latency compared to newer tools like Spark. Hadoop breaks down jobs into rigid map and reduce stages, forcing intermediate results to disk storage (HDFS) after each stage.

Spark offers a more flexible approach. It uses Directed Acyclic Graphs (DAGs) to represent jobs so that intermediate results stay in memory while possible. This in-memory processing boosts performance, especially for large data sets. As a result, Spark became the preferred choice for many modern big data workloads that demand faster processing and lower latency.

- ▶ Unlike real-time processing systems, Hadoop's batch-oriented approach limits its usefulness for businesses needing immediate data insights.
- ▶ Setting up and maintaining Hadoop clusters can be daunting for smaller organizations lacking dedicated IT staff. The specialized skills that are required for implementation and management can add complexity for resource-constrained teams.
- ▶ The Hadoop MapReduce framework requires deep expertise, hindering developer productivity. While Hive and Pig provide higher-level abstractions for data access, they introduce more complexity for managing workflows.

Key takeaways: Although Hadoop and its ecosystem (MapReduce, Pig, and Hive) play a pivotal role in pioneering big data analytics, their dominance is shifting. Hadoop deployments are still prevalent, but newer, more general-purpose technologies like Kubernetes and object storage are emerging as strong alternatives. These newer solutions offer greater flexibility and can integrate seamlessly with various data sources, making them well suited for the evolving big data landscape.

4.2 HDFS introduction

The HDFS revolutionized big data storage when it arrived. This distributed file system, a core component of Apache Hadoop, excels at handling massive data sets across clusters. However, as data storage needs evolved, object storage emerged as a compelling alternative due to its inherent architectural advantages.

HDFS is highly available and fault-tolerant, with automatic replication and failover capabilities. If a node in the cluster fails, HDFS can automatically replicate the data to another node, ensuring that data remains available and accessible always.

Data locality was a crucial concept in Hadoop HDFS. Its purpose was to decrease the amount of data that needed to be transmitted over the network. Data locality can enhance data processing performance and minimize network traffic. Modern architectures often prioritize minimizing data movement across the network (reducing network bisection) and have less emphasis on traditional locality optimizations. For more information, see [What about locality?](#)

For more information about the HDFS architecture, see the [HDFS Architecture Guide](#).

4.3 The rise of object storage

Traditionally, data was in file systems that were built for structured information and limited storage needs. However, with the explosion of data, these systems became overwhelmed. This situation paved the way for object storage, a scalable and cost-effective solution that is designed for massive volumes of unstructured data, like sensor readings, logs, and multimedia files.

Initially used for backups and archives, object storage's potential for analytics emerged as data volumes grew. Enter the *Object Data Lake architecture*, which leverages object storage's strengths for data analysis.

Here is why object storage works well for analytics:

- ▶ **Massive unstructured data handling:** It effortlessly stores and processes vast amounts of unstructured data, a hallmark of modern applications.
- ▶ **Scalability for growth:** Object storage scales seamlessly to accommodate ever-increasing data volumes.
- ▶ **Powerful metadata management:** Robust tagging, categorization, and search capabilities make finding specific data a breeze.
- ▶ **Unmatched security:** Encryption, access control, object locking, and versioning ensure that your data remains secure.

Object storage's advantages over traditional file systems became undeniable as analytics use cases multiplied.

If you take a detailed look at the rise of object storage in the Hadoop analytical landscape, here are some of the main contributing factors:

- ▶ HDFS in the cloud often required overprovisioning of instances, particularly when compute needs grew slower than storage. Also, the constant need to store data limited dynamic scaling for cost optimization, as clusters typically ran continuously.
- ▶ HDFS replication on an instance store or Elastic Block Store (EBS) is more expensive than Amazon Simple Storage Service (S3).
- ▶ Name node scalability issues (for example, small files problem).

The cloud model's success influenced on-premises storage, leading to the separation of compute resources and highly scalable object storage by using an erasure-coded (EC) S3. Network advancements made this approach feasible, eliminating the need for colocated compute and storage for locality optimization.

Data lakes can be used to collect data from multiple sources, resulting in an accumulation of billions of files and petabytes of data. This volume of data surpasses the capability of conventional database technologies that run on traditional file systems, like relational database management systems (RDBMS), which were initially created to manage structured data.

Hadoop took market share from conventional databases and data warehousing technologies. The cloud forced Hadoop to evolve to embrace decoupled compute and object storage.

As more organizations and analytical tools adopt object storage as a key component of their data lake strategies, object storage becomes a critical part of the modern data analytics landscape.

4.4 Accessing S3 Object Storage from Hadoop (Amazon S3A File System)

The rise of cloud storage, specifically Amazon Simple Storage Service (S3), opened doors for integrating storage with big data frameworks like Apache Hadoop.

Here is a timeline of key developments:

- ▶ Early efforts: After S3's launch, the initial attempt to bridge the gap was the S3 block file system, which offered limited functions.
- ▶ Amazon S3 Native Filesystem (S3N): Marked a significant step forward by enabling Hadoop to directly work with S3 data. It eventually merged with the Hadoop project.
- ▶ Amazon Elastic MapReduce (EMR) and S3 Client: Amazon EMR included its own S3 client, highlighting the growing need for seamless integration.
- ▶ Amazon S3A File System (S3A): An open-source S3 object storage connector that emerged as a response to the limitations of proprietary solutions.

This ongoing development allows Hadoop applications to seamlessly interact with S3 data, fostering greater flexibility and cost-efficiency.

Key takeaway: The continuous evolution of S3 file system clients for Hadoop reflects the increasing importance of object storage for big data processing.

Hadoop applications can easily leverage Amazon S3 storage by using the S3A connector. Specify "s3a://" followed by the S3 bucket path when accessing data. The S3A connector acts as a bridge, transparently converting Hadoop file system operations into S3 object interactions. This seamless integration offers a couple of advantages:

- ▶ One-for-one mapping: Files are represented as individual objects in S3, maintaining a clear and consistent relationship (clarifies the "one to one relationship" concept).
- ▶ Broader accessibility: Because the S3A connector handles the conversion, other applications can now directly read/write data in the S3 lake. Eliminates the need for a dedicated Hadoop file system client, which was previously limited primarily to Java applications.

Hadoop-AWS module: Integration with Amazon Web Services provides the following S3A high-level feature description:

- ▶ Supports per-bucket granular configuration.
- ▶ Directly reads/writes S3 objects.
- ▶ Compatible with standard S3 clients.
- ▶ Compatible with files that are created by the older s3n:// client and Amazon EMR s3:// client.
- ▶ Supports multipart uploads for multi-GB objects.
- ▶ Offers a high-performance random I/O mode for working with columnar data, such as Apache Optimized Row Columnar (ORC) and Apache Parquet files.
- ▶ Uses the Amazon Java V2 software development kit (SDK) with support for the latest S3 features and authentication schemes.
- ▶ Supports authentication by using environment variables, Hadoop configuration properties, the Hadoop key management store Identity and Access Management (IAM) roles, and even custom credentials providers.
- ▶ Supports S3 Server-side Encryption for reading/writing server-side encryption (SSE) with Amazon S3 Managed Keys (SSE-S3), Server-Side Encryption with Amazon Key Management Service (SSE-KMS), and Server-Side Encryption with Customer-Provided Keys (SSE-C).
- ▶ Supports S3-CSE client-side encryption.
- ▶ Instrumented with Hadoop metrics.
- ▶ Actively maintained by the open-source community.

While S3A attempts to map the HDFS API closely to the S3 API, there are some semantic differences that are inherent to object storage:³

- ▶ Data consistency: Eventual consistency ensures that all replicas of your data eventually reflect the latest changes. However, there might be a temporary delay before updates are propagated across all locations. Creations, updates, and deletions might not be visible immediately across all systems.
- ▶ Non-atomic operations: Renaming and deleting operations are not strictly atomic (occurring instantaneously). IBM Storage Ceph uses a *magic committee* to minimize inconsistencies, but these operations can take some time to complete, especially for large directories. This temporary inconsistency might be visible to other processes accessing the data.

³ https://es.wikipedia.org/wiki/Apache_Hadoop

For more information, see [The anatomy of the S3A file system](#). For more information about the S3A connector/client on the upstream, see [Hadoop-AWS module: Integration with Amazon Web Services](#).

4.5 IBM Storage Ceph Object Storage integration with Hadoop

Hadoop and Spark use the S3A interface to access data in an S3 object store like IBM Storage Ceph, instantly unlocking all benefits that Ceph S3 API object storage brings to the table:

- ▶ Best-in-class implementation of the S3 and S3 adjacent APIs, including IAM and Security Token Service (STS).
- ▶ Multi-cluster federated bucket namespaces.
- ▶ Scalability to billions of objects.
- ▶ Unmatched security:
 - Federal Information Processing Standards (FIPS) 140-2 validated cryptography for in-transit and SSE⁴
 - IAM and bucket policy
 - STS with OpenID Connect (OIDC) integration
 - Bucket versioning
 - Object lock
 - Multi-factor authentication (MFA) delete

In addition to the S3 features that are available in IBM Storage Ceph, S3A integration offers several benefits for Hadoop users:

- ▶ With HDFS, connecting different versions of Hadoop and Spark to query the same data set is impossible. With S3A and object storage, you can connect different tool versions and query the same data set, which provides flexibility to data scientists.
- ▶ Decoupling Storage and Compute. With Hadoop and HDFS, a tight coupling between storage and compute does not allow scaling the components individually. With the S3A integration and IBM Storage Ceph, each component, compute, and storage feature is decoupled and can be scaled independently of the others as needed.
- ▶ Notably, IBM Storage Ceph provides EC pools that offer higher efficiency than the initial HDFS replication scheme. Leads to a significant improvement in storage optimization and cost-effectiveness.
- ▶ IBM Storage Ceph offers multisite replication. Hadoop and Spark users can use this feature to provide disaster recovery (DR) capabilities to their data lakes without needing third-party tools, such as WANdisco, and reduce the maintenance costs for DR capabilities.
- ▶ IBM Storage Ceph Object Storage provides high-end security, encryption at rest and in transit, advanced authentication and authorization schemes through STS and IAM roles, SSE (SSE-KMS, SSE-C, and SSE-S3), MFA delete, audit logs, and object lock providing immutability.

⁴ One benefit of transparently representing files as objects 1:1 (S3N or S3A) was that other applications could read/write to the data lake without a Hadoop file system client, which was largely only available to Java applications.

4.6 IBM Storage Ceph Object Storage and S3A

IBM Storage Ceph provides tight integration with the S3A implementation. Several IBM customers successfully run their analytical frameworks based on Hadoop and Spark by using IBM Storage Ceph Object Storage capabilities with S3A.

Joint efforts between IBM Storage Ceph Engineering and IBM customers developed and extended the S3A functions in different areas, for example, a custom identity provider (IDP) that enhances the integration of S3A with the STS S3 feature. This enhancement enabled users to remove and transform their role-based access control (RBAC) authorization framework into a new full-fledged attribute-based access control (ABAC) that is provided by the STS/IAM features in IBM Storage Ceph Object.

The combination of Secure Token Service Authentication with the flexibility of ABAC dramatically simplifies the application authentication policy.

For more information about IBM Storage Ceph authentication and authorization schemes with STS and IAM roles, see *IBM Storage Ceph Solutions Guide*, REDP-5715. For a hands-on example of implementing an ABAC policy, see [IBM Storage Ceph Object Storage Authorization with Attribute-based Access Control \(ABAC\)](#).

4.7 Example of connecting Hadoop with IBM Storage Ceph by using S3A

This section provides a basic example of configuring Hadoop to connect to IBM Storage Ceph Object Storage by using the S3A client/connector.

In the example environment, we set up a simple Hadoop environment on a Red Hat Enterprise Linux (RHEL) node that is called “linux1.” We also have an IBM Storage Ceph Cluster running with our S3 endpoint set, and the configured S3 restful API endpoint is <https://s3.cephlabs.com>.

To configure Hadoop to connect to IBM Storage Ceph Object Storage by using the S3A client/connector, complete the following steps:

1. Set up Ceph for object storage. Create a user specifically for Hadoop access. This user has a UID and display name of “Hadoop” (Example 4-1).

Example 4-1 Creating a user specifically for Hadoop access

```
# radosgw-admin user create --uid hadoop --display-name hadoop
  "user_id": "hadoop",
  "display_name": "hadoop",
  "email": "",
  "suspended": 0
  "max_buckets": 1000,
  "subusers": [],
  "keys": [
    {
      "user": "hadoop",
      "access_key": "QN71C16RLCRTCQT6WEUC",
      "secret_key": "sxAILYNUgsXuUhIIDsVTCy1AAJbaUf6EHNTAN5A1"
    }
  ],
```

```

"swift_keys": [],
"caps": [],
"op_mask": "read, write, delete",
"default_placement": "",
"default_storage_class": "",
"placement_tags": [],
"bucket_quota": {
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
},
"user_quota": {
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
},
"temp_url_keys": [],
"type": "rgw",
"mfa_ids": []

```

2. For this example, we already configured and integrated a Key Management Interoperability Protocol (KMIP)-compliant key manager ([IBM Guardium Key Lifecycle Manager \(GKLM\)](#)) with our IBM Storage Ceph deployment so that we can provide encryption at rest for our data sets (Example 4-2).

Example 4-2 Providing encryption at rest for the data sets

```

# ceph config dump | grep rgw_crypt
client.rgw                                advanced rgw_crypt_kmip_addr
10.251.0.35:5696
client.rgw                                advanced
rgw_crypt_kmip_client_cert                /var/run/ceph/rgw.s3.cephlabs.com.cer
client.rgw                                advanced rgw_crypt_kmip_client_key
/var/run/ceph/rgw.s3.cephlabs.com.key
client.rgw                                advanced rgw_crypt_require_ssl
false
client.rgw                                advanced rgw_crypt_s3_kms_backend
kmip

```

3. Leverage the Amazon Web Services (AWS) CLI to validate the S3 endpoint and ensure that it is operational as intended. Use the access key and secret key that is provided when creating the Hadoop user for authentication (Example 4-3).

Example 4-3 Validating the S3 endpoint and ensuring that it is operational as intended

```

$ grep -A 2 hadoop .aws/credentials
[hadoop]
aws_access_key_id = QN71C16RLCRTCQT6WEUC
aws_secret_access_key = sxAILYNUgsXuUhIIDsVTCy1AAJbaUf6EHNTAN5A1
$ aws --endpoint https://s3.cephlabs.com --profile hadoop s3 mb s3://hadoop/
--region default
make_bucket: hadoop

```

4. The credentials (access key and secret key) should successfully authenticate at the S3 endpoint. Create a bucket that is called `hadoop` to store the objects. As a final check, upload an object and encrypt it at rest by using SSE-KMS with the key ID `"rgw00dc42a9b000000000"` (Example 4-4).

Example 4-4 Uploading an object and using SSE-KMS to encrypt the object at rest

```
aws --endpoint https://s3.cephlabs.com --profile hadoop s3 cp /etc/hosts
s3://hadoop --sse=aws:kms --sse-kms-key-id rgw00dc42a9b000000000
upload: ../../etc/hosts to s3://hadoop/hosts
$ aws --endpoint https://s3.cephlabs.com --profile hadoop s3api head-object
--bucket hadoop --key hosts | grep Server
"ServerSideEncryption": "aws:kms",
```

5. Beyond individual object encryption, you can configure an S3 bucket encryption policy. This policy automates the process, transparently encrypting every object that is uploaded to the bucket. This approach eliminates the need for manual encryption on each object upload, streamlining the workflow for users (Example 4-5).

Example 4-5 Configuring an S3 bucket encryption policy

```
$ cat kms-config.json
"Rules": [
  {
    "ApplyServerSideEncryptionByDefault": {
      "KMSEncryption": "aws:kms",
      "SSEAlgorithm": "aws:kms"
    }
  }
]
$ aws --profile hadoop --endpoint-url=http://s3.cephlabs.com s3api
put-bucket-encryption --bucket hadoop --server-side-encryption-configuration
file://kms-config.json
```

6. You can configure SSE-KMS encryption by using the S3A integration (`fs.s3a.encryption.algorithm`, `fs.s3a.encryption.key`). In this example, we set the encryption at the level of the IBM Storage Ceph bucket. Any object that is uploaded to this bucket is automatically encrypted.

We configure the Hadoop side first. The `hadoop-aws` module must be enabled in the `hadoop-env.sh` file (Example 4-7 on page 59).

Example 4-6 Enabling the hadoop-aws module

```
$ cat /usr/local/hadoop/etc/hadoop/hadoop-env.sh | grep TOOL
export HADOOP_OPTIONAL_TOOLS="hadoop-aws"
```

7. Add S3A properties to the Hadoop configuration file `core-site.xml`. This example is simplified to illustrate the basic setup. There are many other S3A properties that you can configure for specific performance or security needs. In this example, we use `SimpleAWSCredentialsProvider`. This provider uses the access and secret keys that we provided for the user in step 3 on page 57.

Note: There are several ways to authenticate with S3A.

- ▶ Authentication providers: These [authentication providers](#) offer pre-configured options that streamline the setup.
- ▶ Environment variables: Set the environment variables to provide your credentials directly.
- ▶ AWS credential files: Use the standard AWS credential file format for authentication.

For enhanced security, consider using the Hadoop [credentials provider](#). It creates an encrypted credentials provider to safeguard your access keys.

We also configure Ceph Object Storage SSE-KMS (the built-in Ceph encryption), so all the data that we upload to Hadoop by using S3A is encrypted at rest (Example 4-7).

Example 4-7 Configuring Ceph Object Storage SSE-KMS

```
<property>
  <name>fs.s3a.impl</name>
  <value>org.apache.hadoop.fs.s3a.S3AFileSystem</value>
</property>
<property>
  <name>fs.s3a.access.key</name>
  <value>6N85CBHVHKEZ6EM0PYJH</value>
</property>
<property>
  <name>fs.s3a.secret.key</name>
  <value>svP1GMkpG01gRkmdhask3fuVTvPu1ZZUk51c5yTF</value>
</property>
<property>
  <name>fs.s3a.aws.credentials.provider</name>
  <value>
    org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider,
  </value>
</property>
<property>
  <name>fs.s3a.connection.ssl.enabled</name>
  <value>true</value>
</property>
<property>
  <name>fs.s3a.endpoint</name>
  <value>https://s3.cephlabs.com</value>
</property>
<property>
  <name>fs.s3a.endpoint.region</name>
  <value>default</value>
</property>
<property>
  <name>fs.s3a.path.style.access</name>
  <value>>false</value>
</property>
```

8. After configuring the properties, restart the Hadoop services. When they are running, verify whether you can access the hadoop bucket by using S3A from the Hadoop application (Example 4-8).

Example 4-8 Checking whether you can access the hadoop bucket by using S3A from Hadoop

```
$ hadoop fs -ls s3a://hadoop/
Found 1 items
rw-rw-rw-  1 hadoopuser hadoopuser      821 2024-04-02 06:34 s3a://hadoop/hosts
$ hadoop fs -cp file:///etc/nsswitch.conf s3a://hadoop/
$ hadoop fs -ls s3a://hadoop/
Found 2 items
rw-rw-rw-  1 hadoopuser hadoopuser      821 2024-04-02 06:34 s3a://hadoop/hosts
rw-rw-rw-  1 hadoopuser hadoopuser     2124 2024-04-02 07:02
s3a://hadoop/nsswitch.conf
```

We now have our S3A connector that is configured to access IBM Storage Ceph Object S3 buckets from Hadoop.

4.8 Migrating from HDFS to S3A

There are different ways to approach migration from HDFS to S3. It can be *partial migration*, where you keep the old data in HDFS and migrate only the new incoming data sets into object storage. You can do a *complete migration* of all the data from HDFS to S3, and deprecate the HDFS storage nodes afterward.

This section describes three different techniques for HDFS to S3 migration.

4.8.1 Storage abstraction

One key advantage of migrating HDFS data to S3 is Hadoop's ability to switch between HDFS and S3 storage back ends seamlessly. This task is accomplished by specifying the appropriate protocol. In S3, the protocol scheme is “s3a://”, and for HDFS, it is “hdfs://”.

By specifying the correct protocol at run time, applications can easily migrate between HDFS and S3 storage back ends without complex modifications to the application's underlying code. This approach simplifies migration, enabling developers to update their applications and start using S3 easily.

To do an HDFS to S3A migration in the example environment, complete the following steps:

1. Access the data on the HDFS file system with the “hdfs://” prefix (Example 4-9).

Example 4-9 Accessing the data on the HDFS file system

```
$ hadoop fs -mkdir hdfs://linux1:9000/example1
$ hadoop fs -copyFromLocal /etc/hosts hdfs://linux1:9000/example1/
$ hadoop fs -cat hdfs://linux1:9000/example1/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
```

2. Copy the hosts file from HDFS to S3. For S3, reference the “s3a://” prefix with the bucket name, as shown in Example 4-10 on page 61.

Example 4-10 Copying the hosts file from HDFS to S3

```
$ hadoop fs -cp hdfs://linux1:9000/example1/hosts s3a://hadoop/hostsnew
$ hadoop fs -cat s3a://hadoop/hostsnew
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1        localhost localhost.localdomain localhost6 localhost6.localdomain6
```

3. Switching storage configurations is simple, as shown in Example 4-11. Specify a prefix to direct data to your preferred storage location. If you do not provide a prefix at run time, the default storage location that is set in your `core-site.xml` file is used. With this approach, complex code modifications are not required, which aligns with the concept of a migration strategy that minimizes code changes.

Example 4-11 Switching between storages

```
$ cat core-site.xml | grep -A1 fs.defaultFS
    <name>fs.defaultFS</name>
    <value>hdfs://linux1:9000</value>$ cat core-site.xml | grep -A1
fs.defaultFS
    <name>fs.defaultFS</name>
    <value>s3a://hadoop</value>
$ hadoop fs -ls /
Found 4 items
rw-rw-rw-  1 hadoopuser hadoopuser      821 2024-04-02 06:34 /hosts
rw-rw-rw-  1 hadoopuser hadoopuser      158 2024-04-02 08:27 /hostsnew
rw-rw-rw-  1 hadoopuser hadoopuser    2124 2024-04-02 07:02 /nsswitch.conf
rw-rw-rw-  1 hadoopuser hadoopuser      821 2024-04-02 07:50 /test2
```

Key takeaways:

- ▶ Although you can change the default storage to S3A as shown in Example 4-11 on page 61, this approach requires specifying a bucket name for each operation, which breaks compatibility with existing code that relies on HDFS access behavior. Therefore, it might not be the most convenient solution for all use cases.
- ▶ Gradual migration is a viable option to address this issue. You can start by migrating specific data sets to S3A while keeping HDFS for existing code to leverage S3A's advantages incrementally while minimizing code changes.
- ▶ Two further migration strategies are presented in 4.8.2, “The DistCp tool” on page 61 and 4.8.3, “Application-layer tools for data movement” on page 64.

4.8.2 The DistCp tool

To migrate or copy data between different sources, Hadoop provides the *DistCp* tool. DistCp can be used to move vast amounts of data between different storage back ends, so it is a perfect match for migrating data from HDFS to S3. As defined by the DistCp Hadoop documentation:

“DistCp (distributed copy) is used for large inter/intra-cluster copying. It uses MapReduce to affect its distribution, error handling recovery and reporting. It expands a list of files and directories into input to map tasks, each of which will copy a partition of the files specified in the source list.”⁵

This section provides a simplified example of a data migration from HDFS to S3A by using the DistCp tool.

⁵ <https://hadoop.apache.org/docs/current/hadoop-distcp/DistCp.html>

To use the DistCp tool to move data, complete the following steps:

1. To achieve this migration, use Hive external tables. Configure a Hive external table that points to a .csv file that is named ok.csv, and store it in your HDFS. You should be able to migrate this data to S3 while still being able to query the data from Hive with minimal changes.

Example 4-12 demonstrates the creation of a Hive external table for reference. The table points to the countries folder within HDFS, where a CSV file named ok.csv is. This CSV file contains data about countries, including their capitals and populations.

Benefits of external tables: This approach offers a couple of advantages:

- ▶ Minimal Hive schema changes: Because the data is in S3, modifications to the Hive table schema are likely minimal when migrating from HDFS.
- ▶ Flexibility and scalability: External tables provide flexibility because the underlying data location (S3 in this case) can be changed without altering the Hive table definition.

Example 4-12 Creating a Hive external table for reference

```
$ hadoop fs -cat hdfs://localhost:9000/countries/ok.csv
1,US,Washington,328
2,France,Paris,67
3,Spain,Madrid,47
4,Russia,Moscow,145
5,Indonesia,Jakarta,267
6,Nigeria,Abuja,196
```

```
hive> create external table if not exists countries_list(
  > CountryID int, CountryName string, Capital string, Population string)
  > comment 'List of countries'
  > row format delimited
  > fields terminated by ','
  > stored as textfile
  > location 'hdfs://linux1:9000/countries/';
OK
Time taken: 0.136 seconds
```

```
hive> select * from countries_list ;
OK
1 US Washington328
2 FranceParis67
3 SpainMadrid47
4 RussiaMoscow145
5 IndonesiaJakarta267
6 NigeriaAbuja196
Time taken: 1.533 seconds, Fetched: 6 rows
```

2. Copy the data set from HDFS to S3. Create a bucket that is named bucketdata on the IBM Storage Ceph S3 by using the Hadoop user. This bucket is the destination target in the **distcp** command (Example 4-13 on page 63).

Example 4-13 Copying the data set from HDFS to S3

```
$ aws --endpoint https://s3.cephlabs.com --profile hadoop s3 mb s3://bucketdata/
--region default
make_bucket: bucketdata
```

3. Run **distcp** to copy the data from the HDFS source to the S3 destination. The **distcp** utility can be tuned depending on how much data that you must copy and the available resources to use more copy threads and reduce the data migration time to completion (Example 4-14).

Example 4-14 Running distcp to copy the data from the HDFS source to the S3 destination

```
$ hadoop distcp hdfs://linux1:9000/countries/ s3a://bucketdata/
2024-04-03 03:51:30,292 INFO mapreduce.Job: Running job: job_1712130643917_0001
2024-04-03 03:51:39,488 INFO mapreduce.Job: Job job_1712130643917_0001 running in
uber mode : false
2024-04-03 03:51:39,489 INFO mapreduce.Job: map 0% reduce 0%
2024-04-03 03:51:49,608 INFO mapreduce.Job: map 50% reduce 0%
2024-04-03 03:51:50,631 INFO mapreduce.Job: map 100% reduce 0%
2024-04-03 03:51:53,659 INFO mapreduce.Job: Job job_1712130643917_0001 completed
successfully
2024-04-03 03:51:53,743 INFO mapreduce.Job: Counters: 43
```

4. When our data set is migrated, you can modify the location of the table to start using the new S3A location (Example 4-15).

Example 4-15 Modifying the location of the table to start using the new s3a location

```
hive> ALTER TABLE countries_list SET LOCATION "s3a://bucketdata/countries/";
OK
Time taken: 2.047 seconds
```

5. After the table is modified, you can check that your queries work against the data set, which is now stored in your IBM Storage Ceph Cluster and accessible through the S3 protocol (Example 4-16).

Example 4-16 Checking that the queries work against the data set

```
hive> select * from countries_list ;
OK
1 US Washington328
2 FranceParis67
3 SpainMadrid47
4 RussiaMoscow145
5 IndonesiaJakarta267
6 NigeriaAbuja196
Time taken: 0.144 seconds, Fetched: 6 rows
```

4.8.3 Application-layer tools for data movement

Although dedicated data migration tools offer a streamlined approach, there are alternative strategies to consider. One option is to leverage application-layer tools for data movement.

Consider the example in 4.8.2, “The DistCp tool” on page 61 to process external tables containing data in CSV files that are stored on HDFS. In this scenario, you can use [Presto](#), a powerful SQL query engine that is included in the [IBM watsonx.data](#) bundle. Presto directly queries the external CSV files in HDFS by using standard SQL statements. Then, you can use an `INSERT INTO` statement to efficiently load the results into a destination table that is stored within an S3 bucket.

When the SQL runs, this action migrates the data but also transforms it from a text-based CSV format to a more efficient columnar format like Parquet.

Benefits of application-layer migration:

- ▶ **Flexibility:** This approach leverages existing tools within your application ecosystem for data movement.
- ▶ **Transformation:** Data is transformed into a more efficient format (Parquet) during migration, potentially improving storage efficiency and query performance.

There is an example of this procedure in Chapter 8, “Transform: Staging and curated zones” on page 121, where we present a hands-on use case example.

4.9 A real-world HDFS to S3A migration example

A European retailer (see Chapter 6, “Retail use case” on page 79) wanted to empower its data scientists and engineers with a more flexible and cost-effective way to analyze data by using the Hadoop ecosystem for financial modeling and product planning. Their current solution, a traditional tightly coupled Hadoop architecture with storage and compute on the same system, lacked the scalability that they needed.

By implementing IBM Storage Ceph, a software-defined storage solution, they decoupled compute from storage. This approach offered several benefits:

- ▶ **Reduced costs:** Separating storage from compute resources enables independent scaling, which optimizes resource utilization and lowers overall costs.
- ▶ **Improved flexibility:** Independent scaling of storage and compute provides the flexibility to adjust resources based on specific needs.
- ▶ **Enhanced service:** This modernized data environment empowers data scientists and engineers with a more efficient and responsive platform for data analysis, ultimately leading to better service for key stakeholders who rely on data-driven insights.

4.9.1 The environment

This particular merchant employed several data scientists and engineers who used various Hadoop ecosystem tools, including Spark, Hive, and NoSQL to conduct data analysis against a diverse range of data gathered from numerous sources, such as financial, sales, products, and shopper behavior.

The duration of this analysis can vary from a few minutes to several days and is repeated based on the season, store forecast, and future promotions.

The results of this analysis assist the retailer in making informed future financial and operational decisions. Some of these decisions can significantly impact future buying and stocking choices. Therefore, it is crucial to collect and analyze the correct information promptly and accurately.

4.9.2 Current solution

The existing data solution consists of proprietary hardware that provides appliance-style servers with integrated compute and storage. The solution also supports several Hadoop ecosystem tools.

The existing solution presents the retailer with several distinct challenges:

- ▶ Scaling storage and compute independently is not possible. Therefore, the retailer purchased more servers to meet increasing storage demands when they already had sufficient compute resources, which resulted in further and unnecessary financial costs to the business.
- ▶ The existing solution dictates that only one version of the Hadoop ecosystem could be available to all platform users. This situation frustrates the data users because the data scientists typically look to use the latest versions. In contrast, the data engineers want older versions of tools because they offer stability.
- ▶ With a 24-hour-a-day business, maintenance windows are becoming impossible to manage because any updates to hardware or software require the whole data system to be offline for an extended period.
- ▶ The maintenance costs are becoming excessive because their existing vendor wants to maintain the whole data estate regardless of the Hadoop ecosystem tools that are used by the retailer. This situation resulted in the retailer having to pay for support on tools they either did not use or needed.

To better serve internal data users and the overall business, the customer needs a high-performance, highly scalable storage solution that is flexible in meeting their needs while handling large amounts of unstructured data.

4.9.3 S3A Ceph-based solution

The customer chose IBM Storage Ceph for their underlying storage solution.

The IBM Storage Ceph-based solution consists of the following elements:

- ▶ The current Hadoop community ecosystem tools continue to be used by the retailer to provide limited internal support and enable users to choose which version of the tool they want to use.
- ▶ Commodity x86 servers and disks are purchased from their existing hardware provider for IBM Storage Ceph.
- ▶ IBM Storage Ceph provides the underlying storage layer for the unstructured data lake.
- ▶ Red Hat Single Sign-on (RHSSO) is used to integrate authentication and authorization within IBM Storage Ceph to existing customer systems.
- ▶ Red Hat Service Registry defines the data structure of the ingress and egress messages to the Hadoop applications.

- ▶ RHEL is the underlying operating system (OS) for the IBM Storage Ceph system.
- ▶ The existing hardware supplier provides hardware support for the physical assets.
- ▶ Ceph Object Storage multisite replication is used to replicate all the data to a different site to comply with the DR requirements of the applications.

This design offers several advantages:

- ▶ Independent scaling: Compute resources (Hadoop) can be scaled independently of storage (IBM Storage Ceph) by adding servers. IBM Storage Ceph automatically rebalances data across the cluster, which ensures efficient utilization.
- ▶ Multi-Hadoop version support: Different versions of Hadoop can access the same data set so that data engineering and data science teams can manage their preferred Hadoop versions independently, which fosters flexibility and innovation.
- ▶ Improved efficiency and cost savings: This approach enhances functions and performance while reducing storage costs.



IBM Storage Ceph with IBM watsonx.data

To help businesses unlock the true potential of their data, IBM created IBM watsonx. This enterprise-grade AI and data platform is designed to maximize the impact of AI across your entire organization.

IBM Storage Ceph and IBM watsonx.data work together seamlessly to help you manage and analyze your data.

This chapter introduces the IBM watsonx platform.

This chapter has the following sections:

- ▶ Introducing IBM watsonx.data
- ▶ IBM watsonx.data infrastructure components
- ▶ Typical use cases for IBM watsonx.data
- ▶ IBM watsonx.data UI

5.1 Introducing IBM watsonx.data

Data is the lifeblood of artificial intelligence (AI) and analytics. AI models can generate valuable insights only if they are trained on clean, accurate data. This situation has always been the case from the early days of business intelligence (BI) to the rise of machine learning (ML), and it is even more crucial with the emergence of generative AI.

However, most organizations struggle with fundamental data challenges, such as the ever-growing volume of data, the various formats and types of data, and the many different locations, on-premises or in private or public clouds, in which data is stored. There is also a challenge to get data ones who need it, with the uses and users of data becoming more varied than before.

To help organizations manage making an efficient use of their data, IBM designed the watsonx platform, which is an enterprise-ready AI and data platform that can multiply the impact of AI across an enterprise's business.

The watsonx platform is composed of three components: *IBM watsonx.ai* for new foundation models, generative AI, and ML; the *IBM watsonx.data* fit-for-purpose data store that provides the flexibility of a data lake with the performance of a data warehouse; and *IBM watsonx.governance* to enable AI workflows that are built with responsibility, transparency, and explainability.

Figure 5-1 shows the IBM watsonx platform.

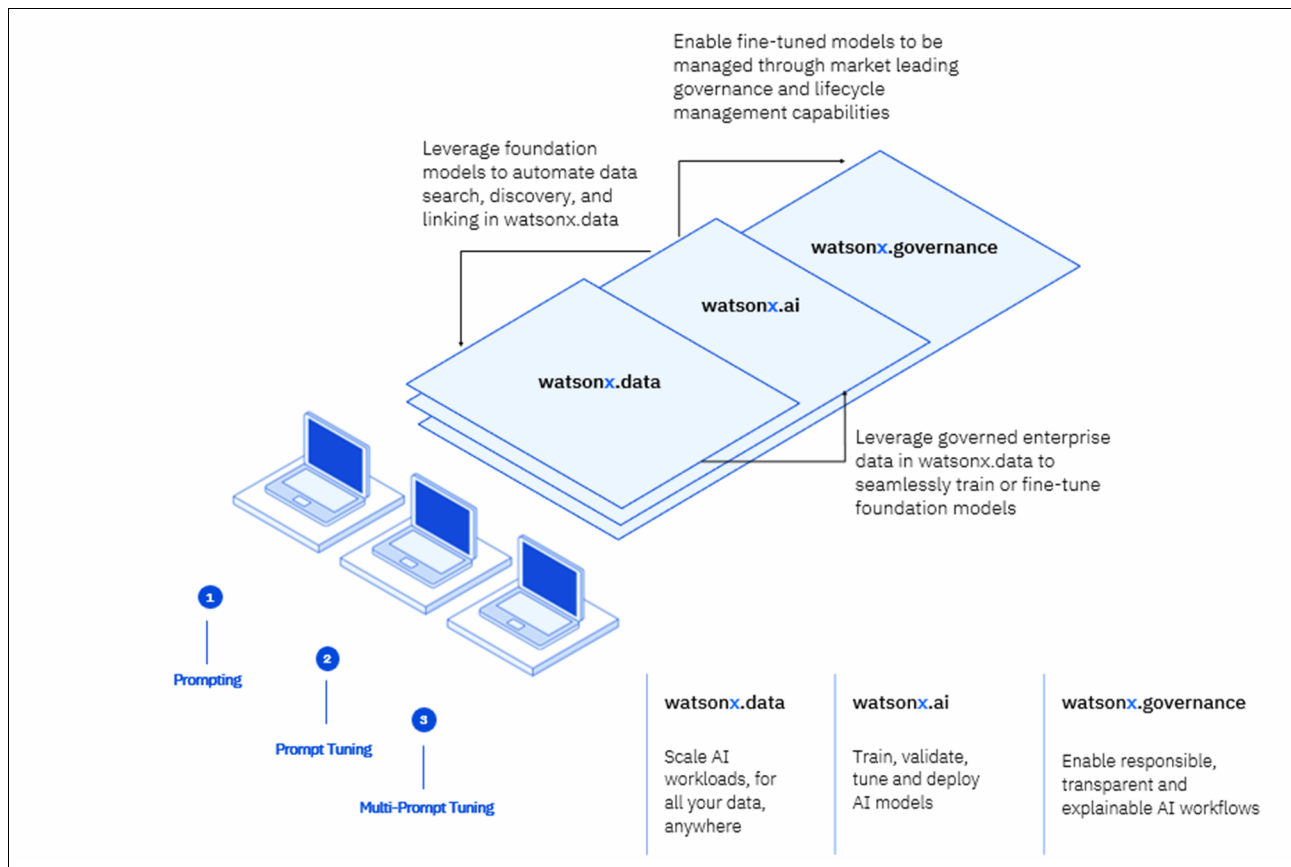


Figure 5-1 IBM watsonx platform

The watsonx.data component empowers enterprises to scale analytics and AI initiatives. It achieves this goal through a unified data repository that is built on an open lakehouse architecture. This architecture supports querying, data governance, and open data and table formats, enabling seamless data access and sharing.

With the IBM Cloud Pak® for Data AI and data platform as the foundation of watsonx.data, it integrates seamlessly with existing data and data fabric services within the platform. This integration accelerates and simplifies the process of scaling AI workloads across your organization.

Figure 5-2 shows the IBM watsonx platform together with IBM Data and Data Fabric services.

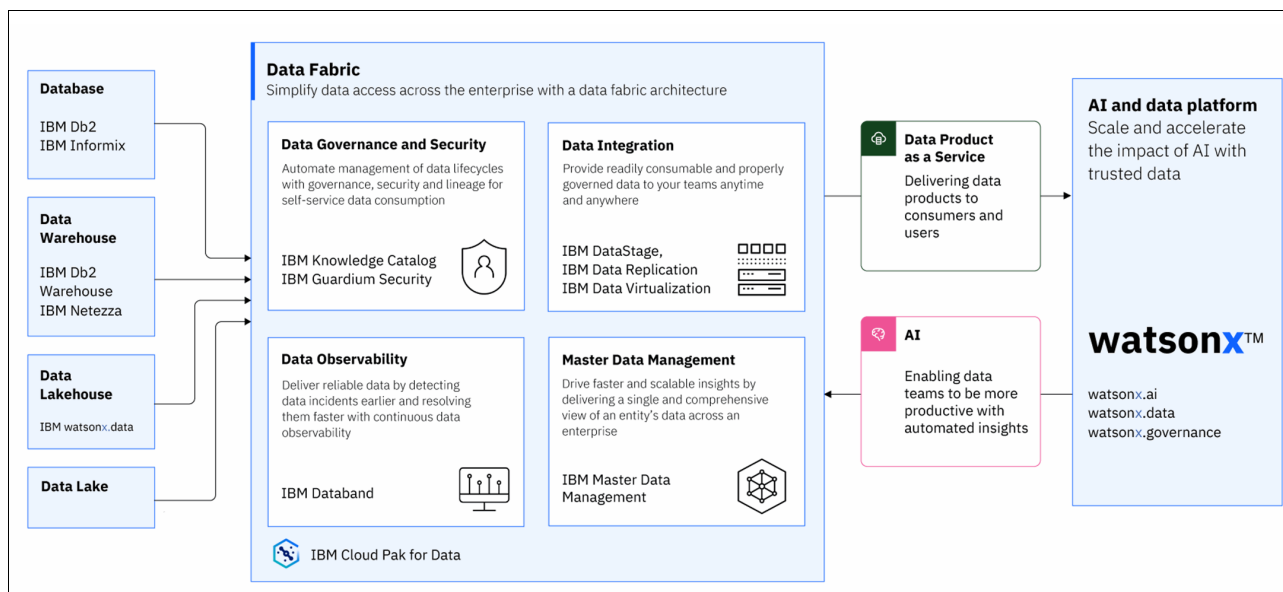


Figure 5-2 IBM watsonx platform together with IBM Data and Data Fabric services

The data teams define, organize, manage, and deliver trusted data to train and tune AI models by using data fabric services. The AI teams leverage generative AI and ML models that are tuned with responsible, transparent, and explainable data workloads to improve the productivity with automated insights.

5.2 IBM watsonx.data infrastructure components

Built on Red Hat OpenShift, watsonx.data is cloud-neutral, and supports deployment on-premises or any major cloud provider like IBM Cloud, Amazon Web Services (AWS), Azure, and Google Cloud Platform (GCP).

There are four infrastructure components that can be configured in watsonx.data:

- ▶ **Engines:** A query engine is used to run workloads against data in watsonx.data. Different engines are supported, including Presto, Spark, Db2, and IBM Netezza®.
- ▶ **Catalogs:** Metadata catalogs are used to manage table schemas and metadata for the data in watsonx.data.

- **Buckets:** IBM watsonx.data stores data in object storage at a fraction of the cost compared to the traditional block storage that is found in a high-performance data warehouse. Specifically, data is stored in Amazon Simple Storage Service (S3)-compatible buckets, which are identified storage areas within object storage, similar to file folders. S3, IBM Storage Ceph, IBM Cloud Object Storage, and MinIO object storage are supported.
- **Databases:** External databases such as IBM Db2, PostgreSQL, and MySQL can be registered and accessed by watsonx.data.

Figure 5-3 shows the IBM watsonx.data lakehouse architecture and its value proposition.

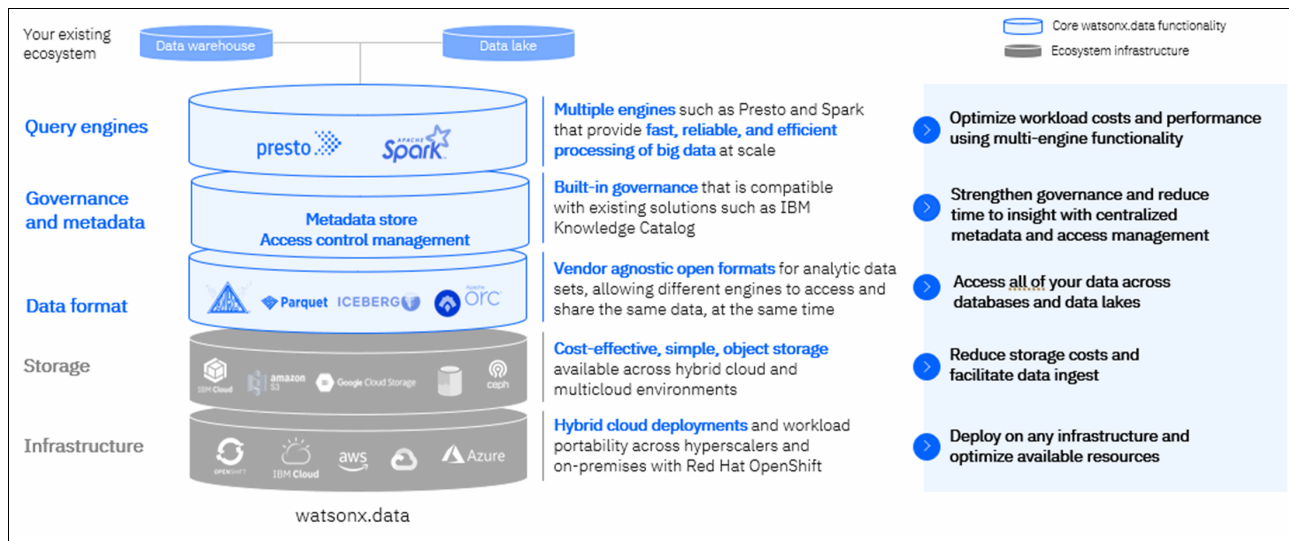


Figure 5-3 IBM watsonx.data lakehouse architecture and its value proposition

5.3 Typical use cases for IBM watsonx.data

There are many ways that enterprises can put their data to work with watsonx.data:

- **Data warehouse optimization:** Optimize workloads from the data warehouse by choosing the right engine for the right workload at the right cost; replace *extract, transform, and load (ETL)* jobs; and reduce the costs of your data warehouse through workload optimization.
- **Data lake modernization:** Augment Hadoop data lakes by using watsonx.data and access better performance, security, and governance without migration or ETL. Decoupling of compute and storage for better and more granular scalability.
- **Realtime analytics and BI:** Connect existing data with new data in minutes and unlock new insights. Integrate with IBM Cognos® and other third-party BI and dashboard tools to visualize data and insights.
- **Rapid analytics with data virtualization:** Query data in place with data virtualization in Presto, which has 35+ connectors to various external database and data store vendors.
- **Mainframe data for analytics:** Virtualize or replicate data to Iceberg tables for analytics.

- ▶ **Data store for Generative AI:** Unify, curate, and prepare data efficiently for AI models and applications. Also, integrated vectorized embedding capabilities enable Retrieval-Augmented Generation (RAG) use cases at scale across large sets of your trusted, governed data.
- ▶ **Generative AI-powered data insights:** Leverage generative AI that is infused in watsonx.data to find, augment, and visualize data and unlock new data insights through a conversational interface (no SQL is required). Unleash cryptic structured data by using auto-generated semantic metadata in natural language for self-service access to data.

5.4 IBM watsonx.data UI

Administration of the watsonx.data environment is primarily done through the watsonx.data UI, also referred to as the console.

One of the first things that you do when working with watsonx.data is to point watsonx.data to one or more object storage buckets. These buckets can already host existing data files that you want to surface as tables in watsonx.data, or they can be empty buckets in which you want to store data for new tables that you create.

Figure 5-4 shows the home page of the IBM watsonx.data console.

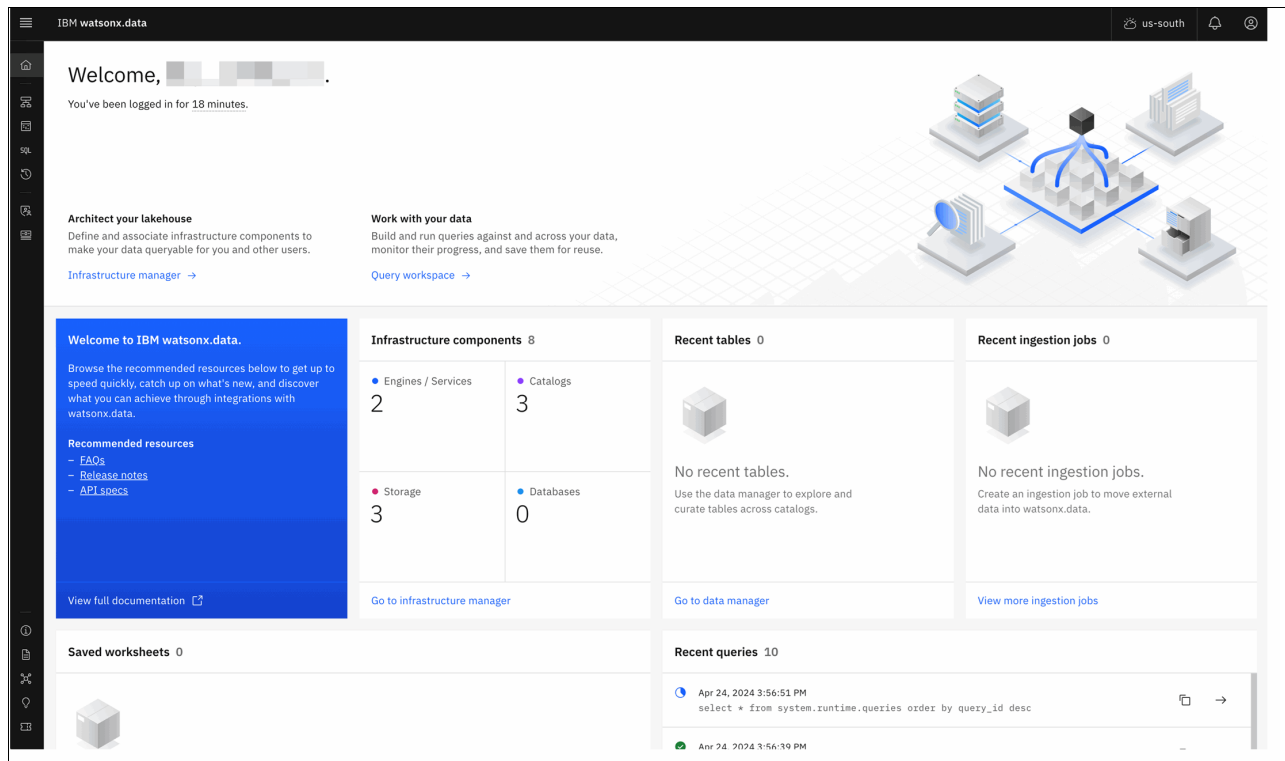


Figure 5-4 IBM watsonx.data console home

This home page contains the following pane:

- ▶ Welcome: Introductory information, including a link to documentation.
- ▶ Infrastructure components: A summary of the engines, catalogs, buckets, and databases that are registered with watsonx.data.
- ▶ Recent tables: Tables that recently were explored.
- ▶ Recent ingestion jobs: Frequently run queries that are saved as worksheets for reuse.
- ▶ Recent queries: Queries that recently ran or are running.

Tip: The left menu provides access to the different console pages.

5.4.1 Infrastructure manager

Adding buckets is done through the Infrastructure manager window. This window includes a graphical canvas view of the different infrastructure components that are defined in your watsonx.data environment, which can include engines (like Presto), catalogs, buckets, and databases.

A watsonx.data environment comes pre-configured with a Presto engine, object storage buckets in an embedded object storage service, and catalogs corresponding to those buckets.

Figure 5-5 shows the Infrastructure manager window.

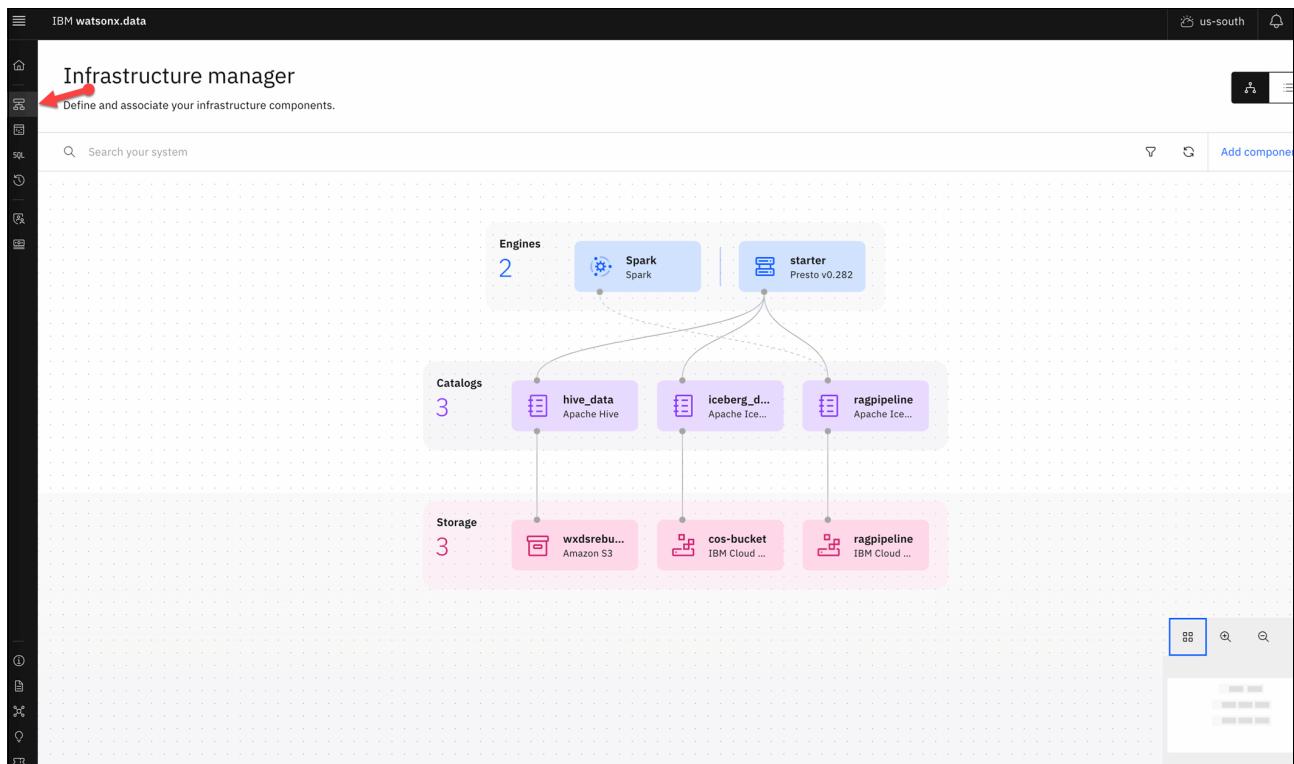


Figure 5-5 Infrastructure manager

Each bucket is associated with a catalog (with a 1:1 mapping). When a bucket is added to watsonx.data, a dedicated catalog is created too. The catalog is how you reference the bucket within the UI and in SQL statements.

In addition to adding buckets, you can also add databases. This action grants Presto access to various other data sources within your organization in-place without having to physically move or duplicate data from those data sources into watsonx.data. This process is commonly referred to as *data federation*. Supported data sources include MongoDB, MySQL, PostgreSQL, SingleStore, Snowflake, Db2, Netezza, and many others.

When you add a database, a dedicated catalog is created too, similar to adding a bucket. With this catalog, you can reference that database data within the watsonx.data UI and from SQL queries.

Catalogs can be Apache Iceberg, Apache Hive, or Delta Lake. Generally, tables that are created and populated with watsonx.data should use the Iceberg open table format because they inherit beneficial capabilities like transactional consistency, schema and partition evolution, snapshots for time travel queries and rollback, and improved query efficiency. A Hive catalog-managed bucket can be used for data that is not using the Iceberg table format. Hive catalog-managed bucket can also be converted to Iceberg catalog-bucket by using Presto and the **"CREATE TABLE AS SELECT"** SQL statement.

5.4.2 Data manager

Within the watsonx.data UI, you can use the Data manager window to explore and curate your data, which includes creating schemas and tables, dropping schemas and tables, and loading data into tables.

The Data manager window includes a data objects navigation pane on the left of the window, which uses a navigable hierarchy of **catalog** → **schema** → **table**. The pane on the right displays information such as a table's schema and a sample of its data.

Figure 5-6 shows the Data manager window.

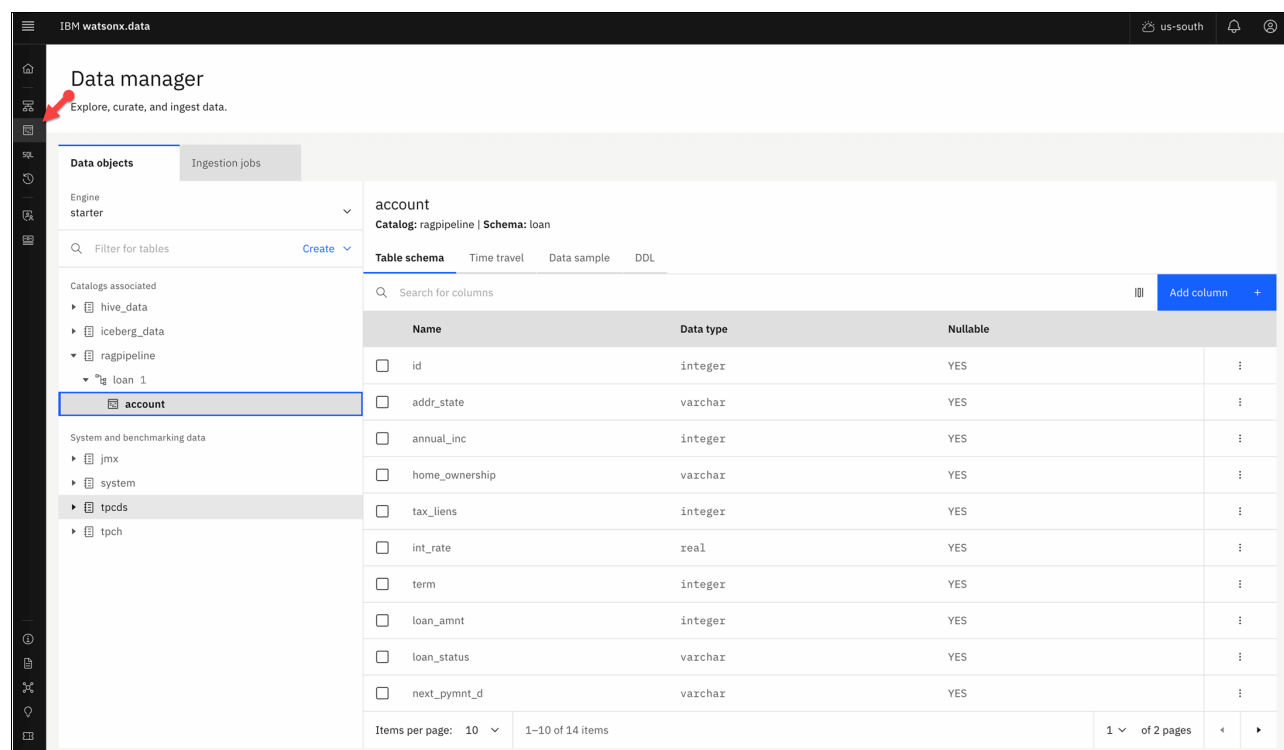


Figure 5-6 Data manager window

Catalogs correspond to the data sources that are added to watsonx.data. Specifically, these data sources include the object storage buckets and remote data sources that were added on the Infrastructure manager window. In Presto, tables can be referenced in queries and other SQL statements by specifying the catalog, schema, and table name, which are separated by periods. For example:

```
SELECT COUNT(*) FROM DATA_CATALOG.SALES_SCHEMA.CUSTOMER_TABLE
```

Using the top-level navigation point, you can select the engine to work with. In Figure 5-6 on page 73, only one Presto engine is available. However, in environments with multiple Presto engines that are defined, administrators can choose any of them, and non-administrators can select any engine that they have access to.

After the engine is selected, you can browse through the catalogs that are associated with the engine. When a catalog contains a data set, you can browse schemas and tables.

You can use the Data manager window to create schema and upload a data file to define and populate it.

The Iceberg open table format unlocks a powerful feature: *time travel*. With this feature, you can view a table's state at any point in the past. This capability offers significant advantages:

- ▶ Auditing: Need to track changes or verify historical data? Iceberg can instantly query past versions, simplifying audits and compliance tasks.
- ▶ Instant disaster recovery (DR): Accidental data corruption happens. With time travel, you can quickly restore a table to a known good state, minimizing downtime and data loss.

The Iceberg time travel feature leverages snapshots by capturing the exact state of a table at a specific moment. Whenever data is added, changed, or removed, a new snapshot is created. This approach keeps the original data files intact, minimizing wasted storage space while efficiently adding ones. Also, metadata files are updated to reflect the changes.

To keep storage efficient, Iceberg offers maintenance tasks that clean up old snapshots and their associated data files when they are no longer necessary.

5.4.3 Query workspace

The watsonx.data console's Query workspace provides a powerful environment for building and running SSQL statements and scripts. Beyond basic data manipulation, you can use the workspace to create schemas and tables, insert data, query tables, and potentially visualize results or integrate with other tools within the console. You can write or copy your own SQL statements, or leverage templates to help build new ones.

Here are some features of the Query workspace:

- ▶ Build schemas and tables: Define the structure for your data by using SQL commands.
- ▶ Manage data: Effortlessly insert, update, and query data within tables.
- ▶ Craft complex queries: Write or copy your own SQL statements or leverage built-in templates for assistance.

Similar to the Data manager, the Query workspace offers a left menu pane. With this menu, you can easily select objects of interest and generate the corresponding SQL statements. For example, you can quickly create a **SELECT** statement to retrieve data from a specific table.

The right pane features the SQL editor pane. Here, you can create and edit your SQL statements, run them by using Presto, and view the resulting data.

Figure 5-7 shows the Query workspace.

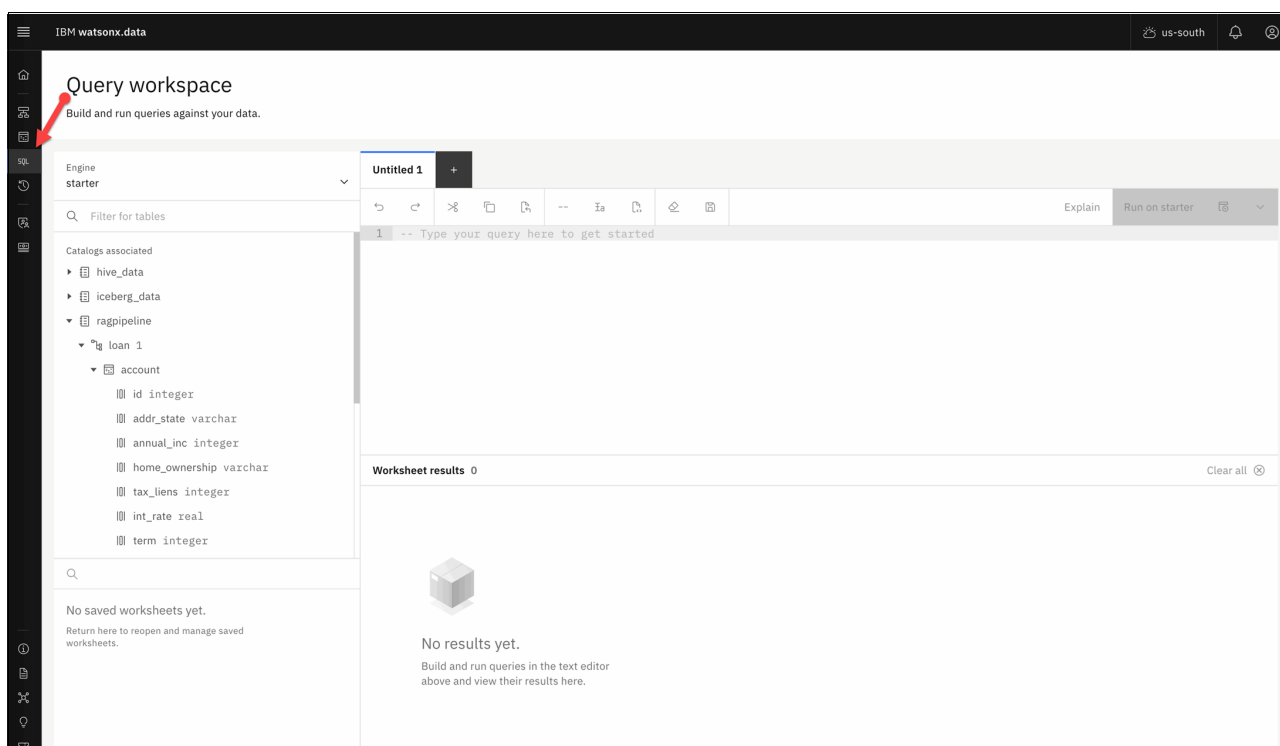


Figure 5-7 Query workspace

Although the Query workspace offers a convenient way to run SQL statements, watsonx.data provides more options for interacting with your data by using Presto:

- ▶ **Presto CLI:** For advanced users, Presto offers its own terminal-based CLI for running SQL queries directly.
- ▶ **Java Database Connectivity (JDBC) driver:** Developers can leverage Presto's JDBC driver to integrate “homegrown” and third-party applications seamlessly with watsonx.data tables and data. Your applications can directly interact with your data by using SQL.

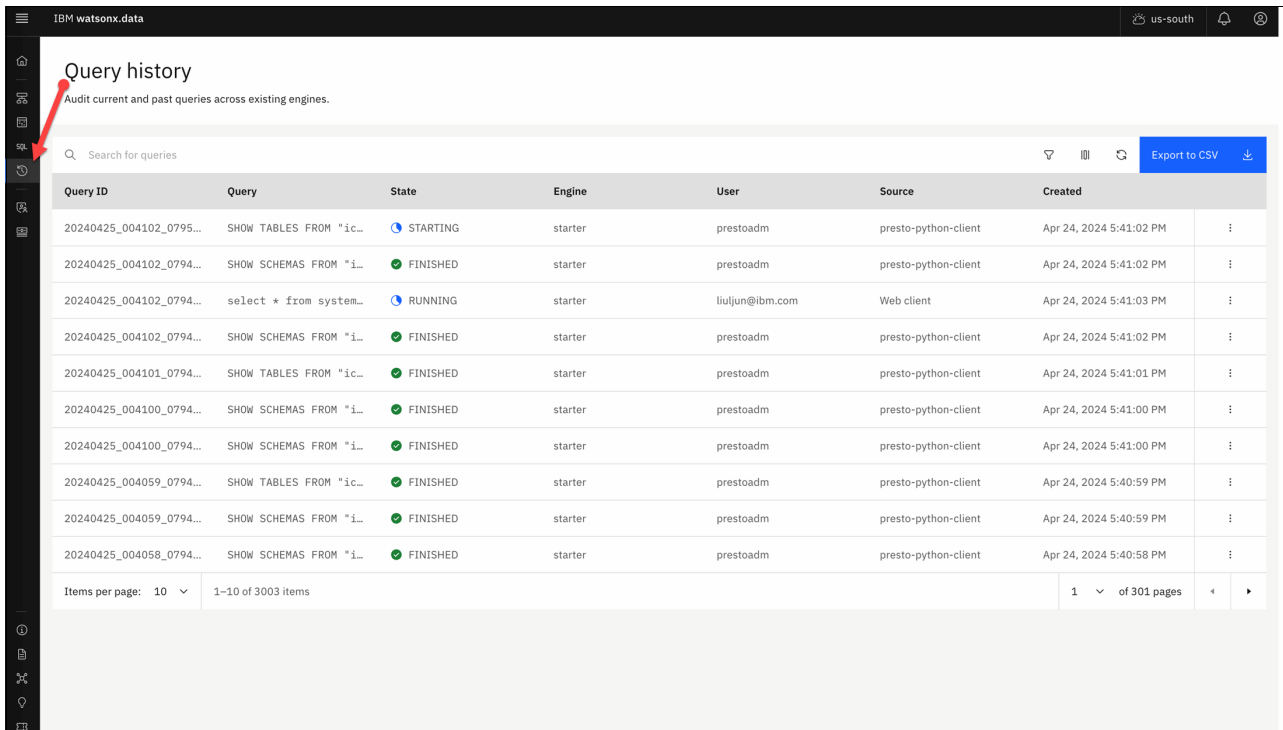
Note: You can also run SQL statements by using Spark SQL.

5.4.4 Query history

Using the Query history window, you can audit currently running queries and queries that ran in the past, across all the engines that are defined in the environment. These queries include ones that users have explicitly run, and queries that are used in the internal management and functions of the environment.

This window features a search bar that you can use to filter your queries based on various criteria. You can search for specific queries by keyword, filter the list by their status (FAILED, FINISHED, or RUNNING), by the engine that was used to run them (for example, Presto), or the user who submitted them.

Figure 5-8 shows the Query history window.



Query history

Audit current and past queries across existing engines.

Search for queries

Export to CSV

Query ID	Query	State	Engine	User	Source	Created
20240425_004102_0795...	SHOW TABLES FROM "ic...	STARTING	starter	prestoadm	presto-python-client	Apr 24, 2024 5:41:02 PM
20240425_004102_0794...	SHOW SCHEMAS FROM "i...	FINISHED	starter	prestoadm	presto-python-client	Apr 24, 2024 5:41:02 PM
20240425_004102_0794...	select * from system...	RUNNING	starter	liujun@ibm.com	Web client	Apr 24, 2024 5:41:03 PM
20240425_004102_0794...	SHOW SCHEMAS FROM "i...	FINISHED	starter	prestoadm	presto-python-client	Apr 24, 2024 5:41:02 PM
20240425_004101_0794...	SHOW TABLES FROM "ic...	FINISHED	starter	prestoadm	presto-python-client	Apr 24, 2024 5:41:01 PM
20240425_004100_0794...	SHOW SCHEMAS FROM "i...	FINISHED	starter	prestoadm	presto-python-client	Apr 24, 2024 5:41:00 PM
20240425_004100_0794...	SHOW SCHEMAS FROM "i...	FINISHED	starter	prestoadm	presto-python-client	Apr 24, 2024 5:41:00 PM
20240425_004059_0794...	SHOW TABLES FROM "ic...	FINISHED	starter	prestoadm	presto-python-client	Apr 24, 2024 5:40:59 PM
20240425_004059_0794...	SHOW SCHEMAS FROM "i...	FINISHED	starter	prestoadm	presto-python-client	Apr 24, 2024 5:40:59 PM
20240425_004058_0794...	SHOW SCHEMAS FROM "i...	FINISHED	starter	prestoadm	presto-python-client	Apr 24, 2024 5:40:58 PM

Items per page: 10 1-10 of 3003 items 1 of 301 pages

Figure 5-8 Query history window

Displayed columns can be modified to add Analysis time to determine the duration of each request to audit and tune queries by analyzing run time.

5.4.5 Access control

Traditionally, security often comes at the expense of user experience. Implementing complex security mechanisms can be a cumbersome task. IBM watsonx.data takes a different approach by offering a tiered security system that safeguards your data while maintaining an intuitive experience. This approach has the following features:

- ▶ **Infrastructure access control:** Defines role-based access control (RBAC) to resources, such as engines, catalogs, buckets, and external databases.
- ▶ **Engines:** Represent compute engines (only Presto at the time of writing) that are configured and sized for a set of data access use cases. These engines are associated with *catalogs* that describe the data artifacts that are inside physical storage, that is, buckets and external databases.
- ▶ **Catalogs:** Represent the core *container* of data that is accessible through engines that use SQL. Catalogs also identify further organization into schema tables and columns.

- **Object storage buckets:** Physical locations where data files are maintained. Typically, they are also organized by using specific table formats (Apache Iceberg is the primary one). These table formats and file formats, such as Parquet, Avro, or ORC, also include extra metadata about the contents of these tables. Buckets are represented by a *logical catalog* for use through SQL within an engine. Buckets can be in different sites, regions, or networks from engines. The same bucket can be accessed by multiple engines concurrently.
- **Databases:** IBM watsonx.data goes beyond object storage buckets. You can also connect to external databases by using federated SQL so that you can Also, these external databases are represented as logical catalogs, which can be attached to multiple engines simultaneously. You can leverage the processing power of different engines to query data from both internal and external sources.

Figure 5-9 shows the Infrastructure Access control window.

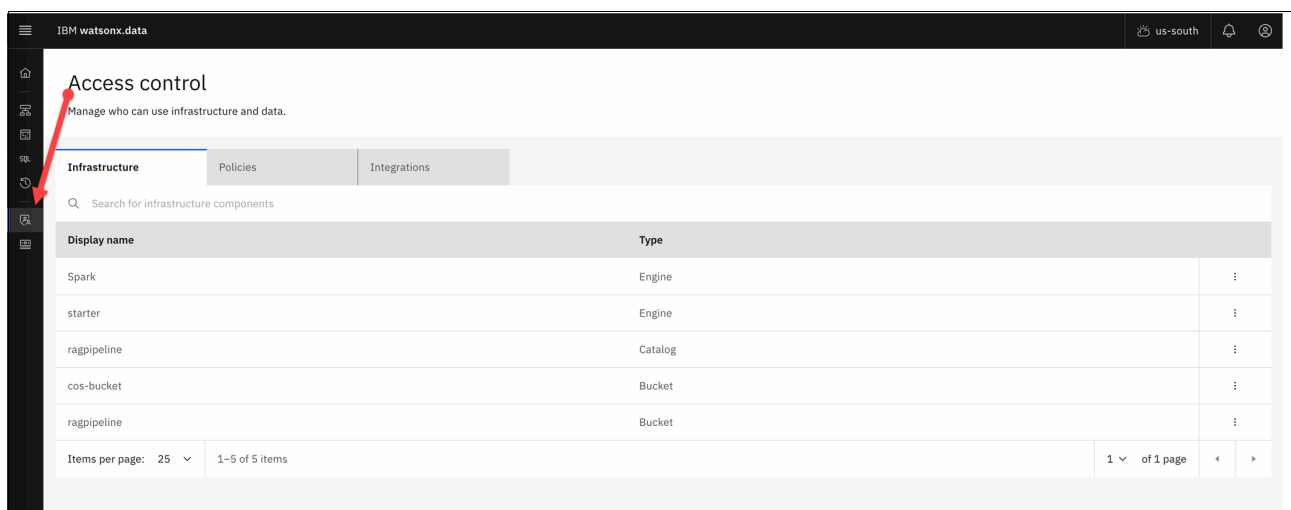


Figure 5-9 Infrastructure Access control window

With IBM watsonx.data, you can define access policies that govern what data users can see and manipulate. These policies determine the following items:

- **Schema access:** Specifies which schemas users are authorized to access.
- **Table permissions:** Control which tables users can view or modify data within.
- **Column visibility:** Defines which data columns are visible to specific users.

To effectively manage access policies, it is helpful to understand these data structures:

- **Schema:** A collection of tables. A schema acts like a container, organizing related tables under a single name.
- **Tables:** Store your data organized in rows and columns. Each row represents a single data record, and columns define the specific attributes that are associated with that data.
- **Columns:** Specific attributes that are present in each table.

Figure 5-10 shows the Access policies window.

IBM watsonx.data

us-south

Create access control policy

Create a policy to grant or limit access to the data objects in the system.

- Details
- Data objects
- Rules
- Summary

Summary

Policy name: account Policy status: active

Policy description:

Data objects

ragpipeline / loan / account / id | annual_inc | home_ownership | tax_liens

Rules ⓘ

Search rule

Rule type	Actions	User/group	User name/group ID
Allow	select	User	chrisr@ca.ibm.com

Cancel Back Create

Figure 5-10 Access policies

All access to these resources is controlled through a combination of authorization roles and policies. Infrastructure resources also include administrative responsibilities, for which privileges are granted through roles. Data access for SQL schema, tables, and columns are managed by policies for fine-grained control. Roles in catalogs support coarse-grained access primitives and helps with the abstraction of the logical (SQL) and physical artifacts.



Retail use case

This chapter dives into a practical example of a data pipeline leveraging an Amazon Simple Storage Service (S3) data lake. In this use case, we use IBM Storage Ceph Object Storage as the central repository for all the analytical data sets.

This chapter guides you through the pipeline's stages: *Ingest*, *Transform*, and *Consume*. Hands-on examples are provided at each step to illustrate the process.

This chapter explores the various zones and their functions within a data lake architecture. It demonstrates how IBM Storage Ceph capabilities address the key requirements of modern data lakes. By showcasing the integration between IBM Storage Ceph and IBM watsonx.data, it unveils how this powerful combination effectively stores, manages, and unlocks the potential of your analytical data.

This chapter has the following section:

- Use case introduction

6.1 Use case introduction

This chapter explores how a leading European-based retail company with a global presence (both physical and online stores) leverages a modern data lake to unlock the power of customer data.

By analyzing customer feedback and purchase behavior, the company aims to achieve the following goals:

- ▶ Deepen its understanding of customer behavior: Gain granular insights into what drives customer choices and preferences.
- ▶ Personalize the shopping experience: Deliver tailored recommendations and offers that resonate with individual customers.
- ▶ Optimize marketing strategies: Develop data-driven marketing campaigns that maximize customer engagement and return on investment.

This data-centric approach empowers the company to make strategic decisions that shape the future of their business.

6.1.1 Objectives

The objective is to develop a data-driven strategy by analyzing client purchases by using the following tools:

- ▶ Dashboards: Visualize data to help executives make better decisions.
- ▶ SQL query: Provide curated table views per business unit line.
- ▶ Online app: Provide more innovative customer e-commerce with personalized offers.

In this use case, we use basic data sets to make the examples simple to follow and understand. These simplified retail data sets are generated daily from the branch's physical stores in CSV format. There are two types of CSV files that are ingested from the branches:

- ▶ Personally identifiable information (PII)
- ▶ General purchase information that does not include personal details

Figure 6-2 on page 82 presents a diagram that provides a high-level overview of the data lake architecture.

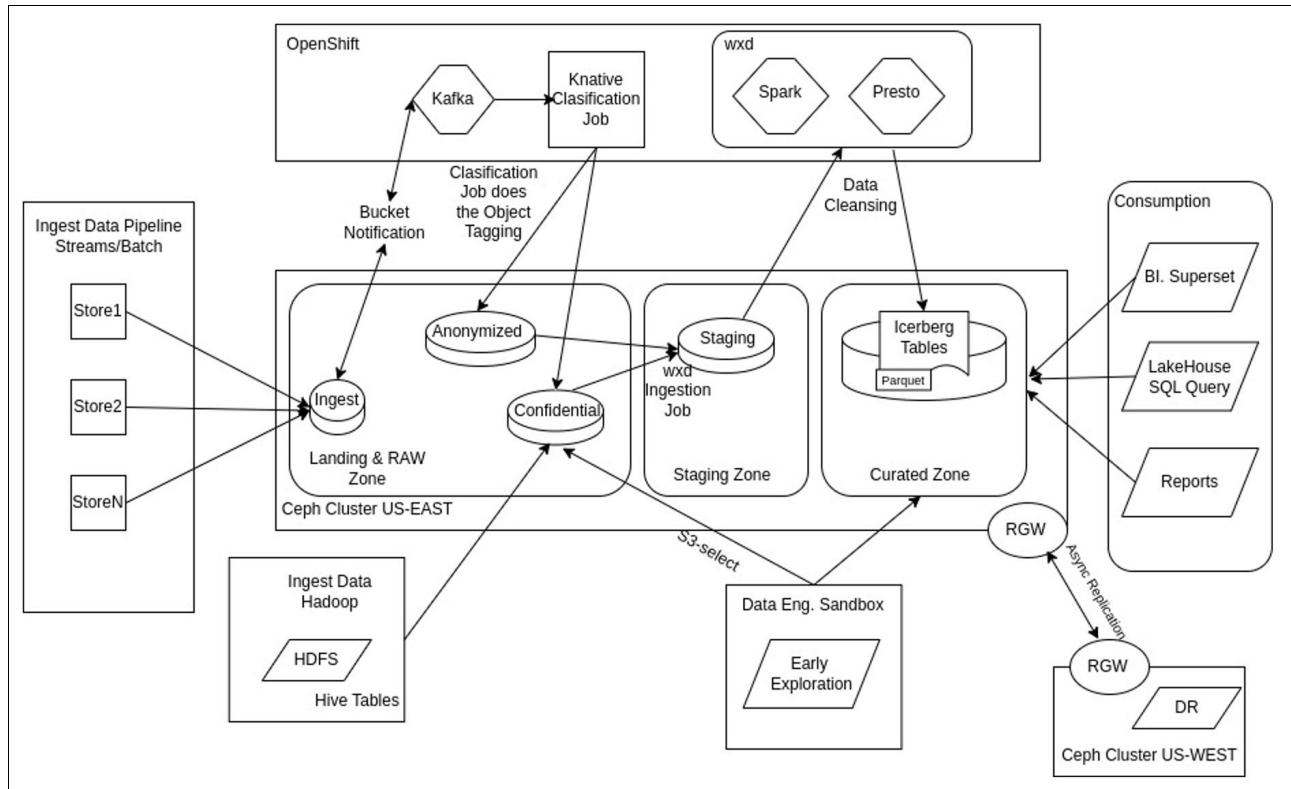


Figure 6-1 Point of sales example use case for the modern data lake

From left to right are the following sections:

- Ingest: Landing and raw zone:
 - Ingest from different sources:
 - Point-of-sale systems: The retail branch stores end-of-the-day batch jobs.
 - Inventory systems: Legacy Hadoop Hive tables. Data about stock levels, product arrivals, and distributions from warehouses.
 - E-commerce platform: Online transactions, browsing logs, and user session data are generated here.
 - Customer feedback: Data that is collected from in-store kiosks, online surveys, and feedback forms.
 - The characteristics of the ingested data are as follows:
 - Sales records.
 - Product information.
 - Customer details.
 - Inventory status.
 - Online browsing logs.
 - The central site uses an IBM Storage Ceph S3 bucket as a temporary landing zone to store daily data that is received from branches.
 - Ingested data is classified based on its confidentiality and stored in buckets within the raw zone.

- The raw zone confidential bucket ingests data from Hive tables that are available on Hadoop:
 - Some of this data is copied by running **distcp** into the confidential bucket directly.
 - Presto in watsonx.data seamlessly transfers other Hive tables into the curated zone by using efficient **INSERT INTO** operations.
- Data engineers require early exploration of data in its raw format (CSV).
- Transform: Staging and curated zone:
 - Using the buckets and data sets in the raw zone, you leverage the two engines that are available in watsonx.data, Spark and Presto to run Data Quality Assurance and Refinement.
 - Spark performs data transformations, aggregations, and cleaning.
 - Presto empowers data enrichment by efficiently joining tables from the staging zone data set. You can create views or tables in the curated zone, combining and transforming data for deeper analysis.
 - After the import and data cleansing process finishes, you end up with curated and optimized Iceberg data tables that tailored for each business line. These tables, which are stored in Parquet format within curated zone buckets, ensure efficient querying and analysis for data-driven decision making.
- Consume:
 - C-levels can consume the data through visual reporting tools like Tableau, Power BI, Superset, and others.
 - Retail departments can leverage the watsonx.data SQL client to independently query their curated business line data to gain faster insights and make data-driven decisions without relying on IT.

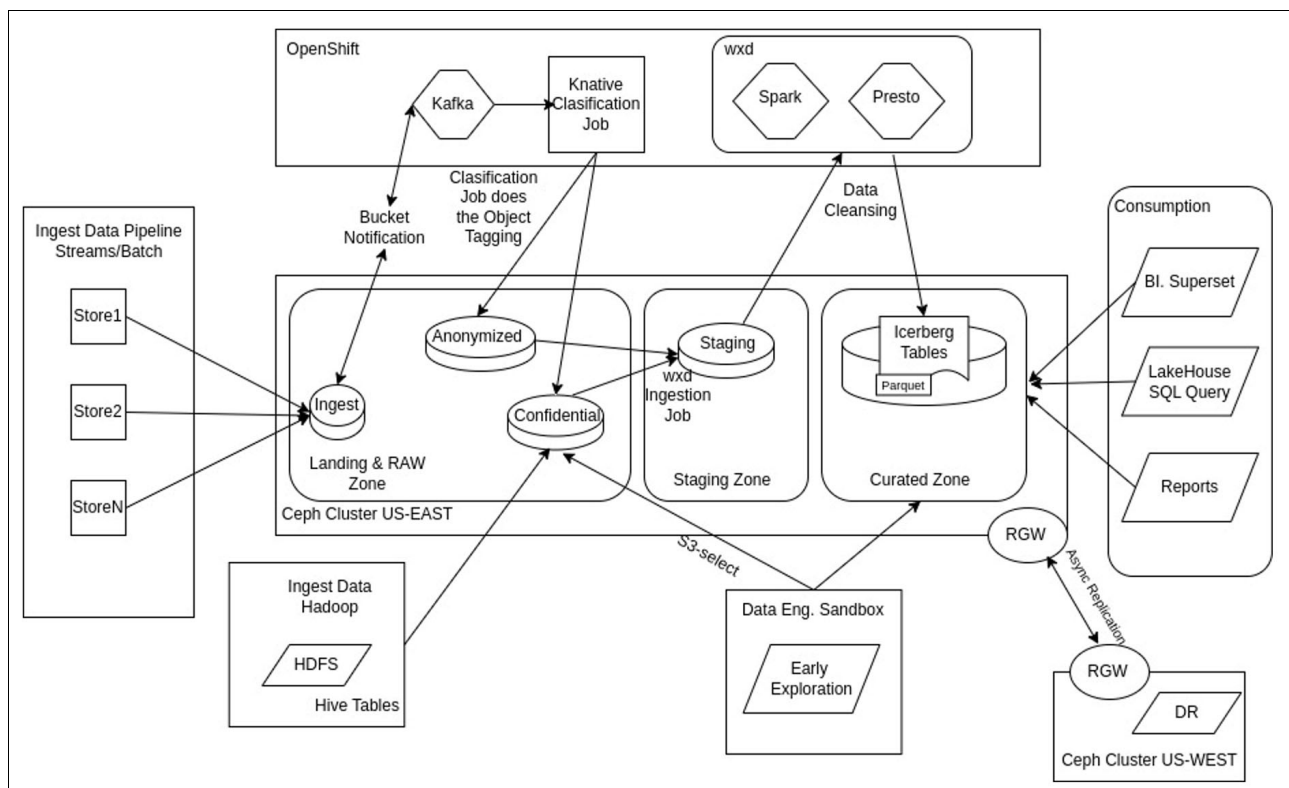


Figure 6-2 Point of sales example use case for the modern data lake



Ingest: Landing and raw zones

Landing zones and raw zones are the first point of entry for data in a data lake. They are where incoming data, regardless of format (structured, semi-structured, or unstructured) is deposited in its original state. Acting as a temporary storage area, the raw data is untouched and unaltered before being processed and transformed further downstream within the data lake.

This chapter describes how data from the different retail sources (physical stores, e-commerce, and Hadoop (Surveys)) is ingested into the data lake.

This chapter has the following sections:

- ▶ Ingestion from stores or branches to the landing zone
- ▶ Raw zone: Branch stores classification data pipeline

7.1 Ingestion from stores or branches to the landing zone

Here is a breakdown of how the system ingests data from the branches:

1. Daily batch jobs: Each store runs a batch job at the end of the day to collect its data.
2. Centralized transfer: Between 8 PM and 10 PM, these data files are sent to a central location.
3. Temporary landing zone: The data is initially stored in a temporary bucket that is called ingest. This bucket acts as a landing zone for the data pipeline.
4. Automatic deletion: Because ingest is a temporary storage location, the data in the bucket is automatically deleted after 24 hours after processing by using the Amazon Simple Storage Service (S3) lifecycle policy feature.

Stores, whether physical or online, can securely access the landing zone bucket by using Single Sign-On (SSO) authentication. This process leverages the existing Active Directory (AD) credentials through a service account. On successful authentication, the system assigns the stores an Identity and Access Management (IAM) role with write permissions specifically for the landing zone S3 bucket.

Figure 7-1 shows the landing zone diagram for our scenario.

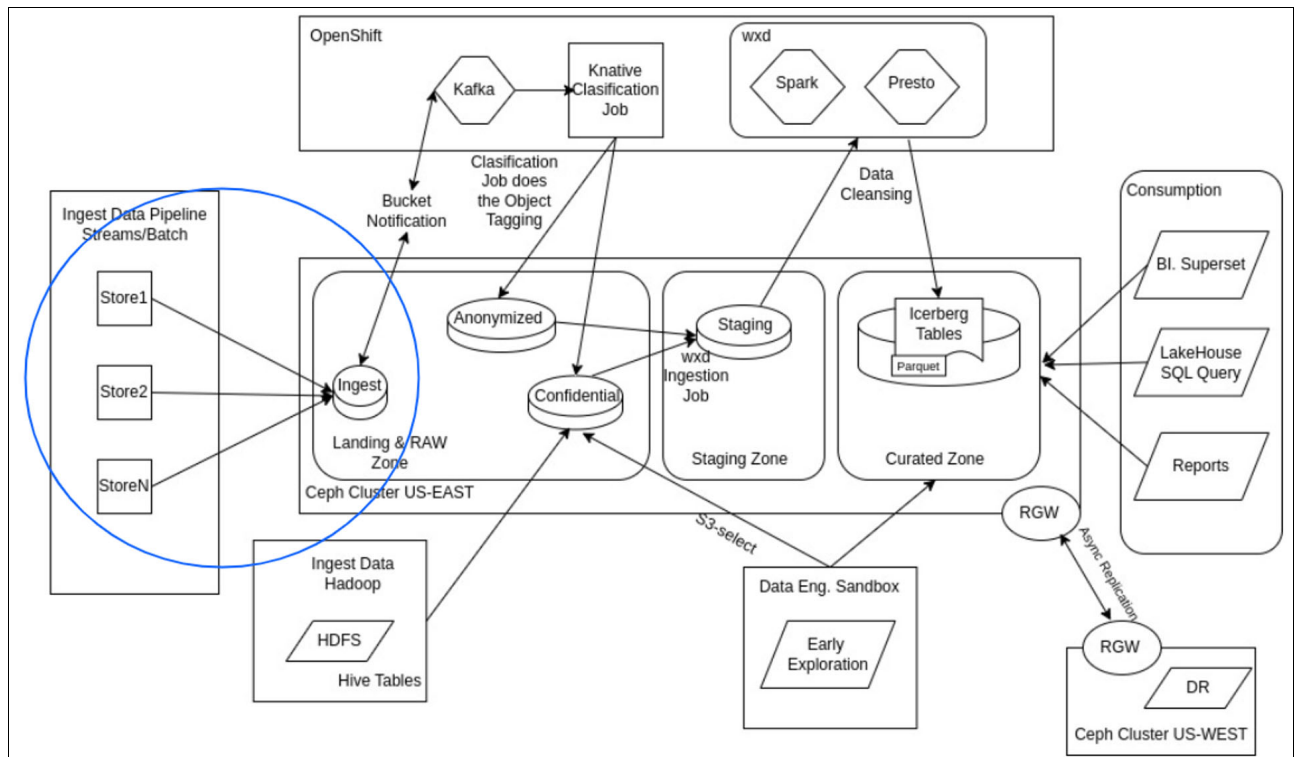


Figure 7-1 Landing zone diagram

Here is a step-by-step walk-through of this scenario:

1. Start with a practical example by setting up the landing zone bucket.

We have a local Object Gateway user who is the administrator for the physical store ingestion data pipelines (Example 7-1).

Example 7-1 Local Object Gateway user

```
# radosgw-admin user info --uid shoppingest | jq .keys
{
  "user": "shoppingest",
  "access_key": "D1YOUYZ8EPY98VAIKHW6",
  "secret_key": "Ct5Xx3eiU0fFW1h5pajJzqvM8PZSPh3mHi16QVav"
}
```

2. We leverage this user to create the ingest bucket and set up the lifecycle policy that deletes all contents of the bucket every 24 hours.

Configure the Amazon Web Services (AWS) CLI for the ingest user so that we can start getting work done. The AWS region equates to the zonegroup that we configured in IBM Storage Ceph (Example 7-2).

Example 7-2 Configuring the AWS CLI for the ingest user

```
$ aws --profile ingestuser configure set endpoint_url https://s3.cephlabs.com
# aws --profile ingestuser configure
AWS Access Key ID [None]: D1YOUYZ8EPY98VAIKHW6
AWS Secret Access Key [None]: Ct5Xx3eiU0fFW1h5pajJzqvM8PZSPh3mHi16QVav
Default region name [None]: default
Default output format [None]: json
```

3. We call our landing zone bucket ingest (Example 7-3).

Example 7-3 Landing zone bucket ingest

```
$ aws --profile ingestuser s3 mb s3://ingest
make_bucket: ingest
```

7.1.1 Ingest: Lifecycle policy expiration

Now, apply the lifecycle policy to delete the contents of buckets that are older than 24 hours by completing the following steps.

1. Run the command that is shown in Example 7-4.

Example 7-4 Deleting the contents of buckets that are older than 24 hours

```
$
$ cat << EOF > expiration_ingest_bucket.json
{
  "Rules": [
    {
      "Expiration": {
        "Days": 1
      },
      "ID": "Delete objects that are older than 24 hours",
      "Filter": {
        "Tag": {
```

```

        "Key": "processed",
        "Value": "true"
    },
    {
        "Status": "Enabled"
    }
]
}
EOF
$ aws --profile ingestuser s3api put-bucket-lifecycle-configuration
--lifecycle-configuration file://expiration-ingest-bucket.json --bucket ingest

```

2. You can check that the lifecycle configuration for bucket ingest is in place and still in a not initialized state because it has not run yet. After it runs, you see it in the COMPLETED state.

Example 7-5 Checking the lifecycle configuration for bucket “ingest”

```

# radosgw-admin lc get --bucket ingest | jq .rule_map[0]
{id: "Delete objects that are older than 24 hours",
  "rule": {
    "id": "Delete objects that are older than 24 hours",
    "prefix": "",
    "status": "Enabled",
    "expiration": {
      "days": "1",
      "date": ""
    }
  },
}
# radosgw-admin lc list
{
  "bucket": ":ingest:fcabdf4a-86f2-452f-a13f-e0902685c655.47553.1",
  "shard": "lc.0",
  "started": "Thu, 01 Jan 1970 00:00:00 GMT",
  "status": "UNINITIAL"
}

```

3. To ensure that the lifecycle is working as expected, run a quick test. Use the **rgw_lc_debug_interval** parameter and change the interval to 60 in the Ceph RADOS Gateway (RGW) configuration database, followed by an object gateway service restart. To expedite testing of the ingest bucket's lifecycle configuration, temporarily enable the debug interval. In this mode, one day within the lifecycle configuration is treated as 60 seconds, so you can verify whether the lifecycle is functioning correctly within a minute.

Note: Revert this change after the test is complete.

Example 7-6 Changing the interval to 60 in the Ceph RGW configuration database and an object gateway service restart

```

# ceph config set client.rgw.default rgw_lc_debug_interval 60
# ceph orch restart rgw.default
Scheduled to restart rgw.default.ceph02.gydyix on host 'ceph02'
Scheduled to restart rgw.default.ceph03.buayin on host 'ceph03'
Scheduled to restart rgw.default.ceph04.hrtkwx on host 'ceph04'
Scheduled to restart rgw.default.ceph06.xedtyh on host 'ceph06'

```

4. Upload a file that is called `test1c` without the S3 tag set to `processed=true` so that the file will not be deleted (Example 7-7).

Example 7-7 Uploading a file that is called “test1c” without the S3 tag set to `processed=true`

```
$ aws --profile ingestuser s3 cp /etc/hosts s3://ingest/test1c
upload: ../../etc/hosts to s3://ingest/test1c
$ aws --profile ingestuser s3 ls s3://ingest/
2024-04-05 12:30:35      821 test1c
```

5. Upload an object that is called `shop4_02_04_2024.csv` with the tag set to `processed=true` (Example 7-8).

Example 7-8 Uploading an object that is called “shop4_02_04_2024.csv” with the tag set to `processed=true`

```
$ aws --profile shopingest s3 ls s3://ingest/
2024-04-17 09:42:36    554622 shop4_02_04_2024.csv
2024-04-05 12:30:35      821 test1c
$ aws s3api get-object-tagging --bucket ingest --key shop4_02_04_2024.csv
  "TagSet": [
    {
      "Key": "processed",
      "Value": "true"
    }
  ]
```

6. After waiting a minute (due to the temporary debug mode), confirm that the lifecycle job successfully finished. You should see that the object `shop4_02_04_2024.csv` was deleted from the bucket. The lifecycle job identified the object based on the “processed” tag and performed the configured deletion. The object that is named `test1c` should remain in the bucket unchanged. The lifecycle job correctly identified objects without the “processed” tag and excluded them from deletion (Example 7-9).

Example 7-9 Confirming that the lifecycle job successfully finished

```
$ aws --profile shopingest s3 ls s3://ingest/
2024-04-17 09:44:07      821 test1c
```

7. After successfully testing the lifecycle configuration, remove the temporary debug interval setting (`rgw_lc_debug_interval`) to ensure that the lifecycle process operates with its normal timing parameters (Example 7-10).

Example 7-10 Testing the lifecycle configuration

```
$ radosgw-admin lc list
{
  "bucket": ":ingest:fcabdf4a-86f2-452f-a13f-e0902685c655.47553.1",
  "shard": "lc.0",
  "started": "Fri, 05 Apr 2024 10:32:25 GMT",
  "status": "COMPLETE"
}
# ceph config rm client.rgw.default rgw_lc_debug_interval
# ceph orch restart rgw.default
Scheduled to restart rgw.default.ceph02.gdyix on host 'ceph02'
Scheduled to restart rgw.default.ceph03.buayin on host 'ceph03'
Scheduled to restart rgw.default.ceph04.hrtkwx on host 'ceph04'
Scheduled to restart rgw.default.ceph06.xedyth on host 'ceph06'
```

7.1.2 Ingest: Data at rest encryption server-side encryption with Amazon Key Management Service

Our data encryption strategy leverages IBM Guardium Key Lifecycle Manager (GKLM), which is a Key Management Interoperability Protocol (KMIP)-compliant key manager, which is integrated with IBM Storage Ceph. This integration ensures encryption at rest for the data sets. Example 7-11 shows the IBM Storage Ceph configuration options that we are using to integrate GKLM with IBM Storage Ceph.

Example 7-11 IBM Storage Ceph configuration options that we are using to integrate IBM Guardium Key Lifecycle Manager with IBM Storage Ceph

```
# ceph config dump | grep rgw_crypt
client.rgw                                advanced rgw_crypt_kmip_addr
10.251.0.35:5696
client.rgw                                advanced
rgw_crypt_kmip_client_cert                /var/run/ceph/rgw.s3.cephlabs.com.cer
client.rgw                                advanced rgw_crypt_kmip_client_key
/var/run/ceph/rgw.s3.cephlabs.com.key
client.rgw                                advanced rgw_crypt_require_ssl
false
client.rgw                                advanced rgw_crypt_s3_kms_backend
kmip
```

Configure an S3 bucket encryption policy to automatically encrypt all objects that are uploaded to the bucket, ensuring data privacy for users (Example 7-12).

Example 7-12 Configuring an S3 put-bucket-encryption policy

```
$ cat kms-config.json
"Rules": [
  {
    "ApplyServerSideEncryptionByDefault": {
      "KMSEMasterKeyID": "rgw00dc42a9b000000000",
      "SSEAlgorithm": "aws:kms"
    }
  }
]
$ aws --profile shopingest s3api put-bucket-encryption --bucket ingest
--server-side-encryption-configuration file://kms-config.json
```

7.1.3 Ingest: Object storage identity provider authentication with single sign-on

Our central landing zone bucket is now ready to receive data. To enable secure data transfer, configure authentication and authorization for the branch stores so that their ingest applications can access the bucket and upload daily data files.

Here is the existing setup:

- ▶ There is an AD group that is named stores that includes a service user for each branch.
- ▶ The goal is to enable the branch applications running at each store to authenticate with the central S3 endpoint that is provided by IBM Storage Ceph. After they are authenticated, these applications should be authorized to upload their daily generated CSV files to the ingest bucket.

- Red Hat Single Sign-On (RHSSO) (built on the Keycloak project) is running on Red Hat OpenShift. We federated the RHSSO with our enterprise identity provider (IDP) (AD), where all our company users are managed. we have an AD group that is called stores that is now accessible within RHSSO (Figure 7-2).

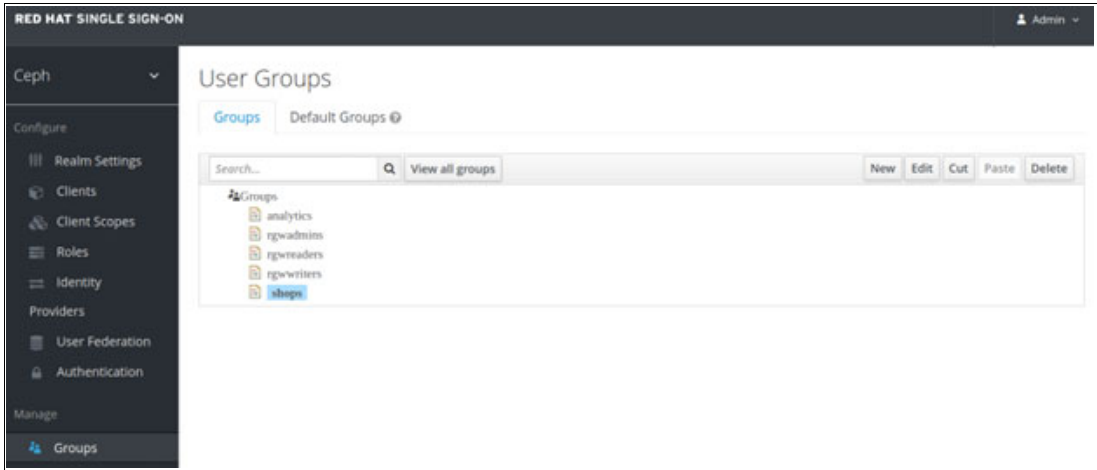


Figure 7-2 Red Hat Single Sign-On

Each branch has an AD user that is part of the shop's group (Figure 7-3).

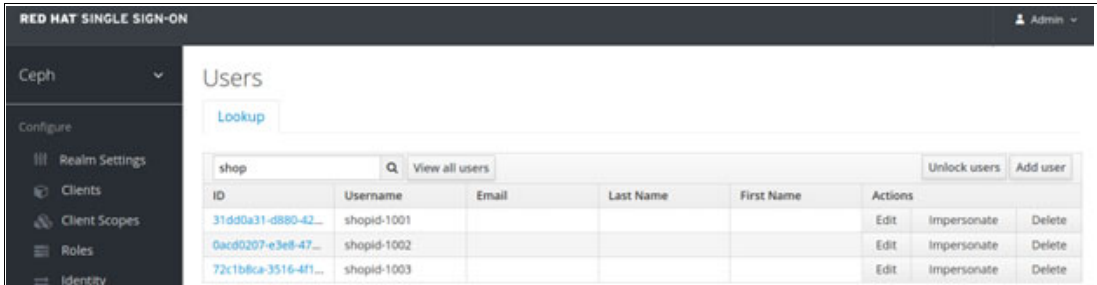


Figure 7-3 User lookup

For a step-by-step guide about setting up RHSSO as an OpenID Connect (OIDC) provider for IBM Storage Ceph, see Chapter 7, “Amazon S3 enterprise user authentication and authorization”, of *IBM Storage Ceph Solutions Guide*, REDP-5715.

IBM Storage Ceph supports other SSO solutions through OIDC. [Ceph ISV integration using Open ID Connect](#) provides an example of setting up the IBM Storage Ceph Object Authentication/Authorization with IBM Security Verify.

After RHSSO is set up, you can see that the group membership of the users is part of the JSON Web Token (JWT) (Example 7-13).

Example 7-13 Group membership of the users is part of the JWT token

```
# bash check_token.sh shopid-1001 Shopid-1001 | jq .groups "shops"
[
  "shops"
]
```

Note: All scripts that are referenced in this section are available in Chapter 7, “Amazon S3 enterprise user authentication and authorization”, of *IBM Storage Ceph Solutions Guide*, REDP-5715. Also, all code examples like bucket policies, IAM roles, lifecycle, and others are available at this [GitHub repository](#).

7.1.4 Ingest: IAM role-based access control-based authorization

Create an IAM role that enables users that belong to the shop group to upload objects (**PUT** permissions) to the ingest bucket. This task ensures that only authorized users within the shop group can interact with the designated bucket.

There are two main options for managing roles and policies in this scenario:

- ▶ Ceph admin control: You can take full control by creating and managing roles and policies directly by using the `radosgw-cli` or RESTful API as a Ceph administrator.
- ▶ The shoppingest user permissions: Alternatively, you can grant the shoppingest user-specific permissions to manage roles and policies that are relevant to their tasks.

In this example, we use the second approach by completing the following steps:

1. Grant the shoppingest user full access to managing roles (Example 7-14).

Example 7-14 Granting the shoppingest user full access to managing roles

```
# radosgw-admin caps add --uid="shoppingest" --caps="roles=*"

```

2. Create the IAM role with its policy doc to enable any user that belongs to the group shops to assume this IAM role (Example 7-15).

Example 7-15 Creating the IAM role with its policy doc

```
# cat iam_role_policy_shops.json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": [
          "arn:aws:iam:::oidc-provider/keycloak-sso.ocp-eu-redbook-ce869a544b369f1dfd24beec10027762-i000.eu-es.containers.appdomain.cloud/auth/realms/ceph"
        ]
      },
      "Action": [
        "sts:AssumeRoleWithWebIdentity"
      ],
      "Condition": {
        "StringLike": {
          "keycloak-sso.ocp-eu-redbook-ce869a544b369f1dfd24beec10027762-i000.eu-es.containers.appdomain.cloud/auth/realms/ceph:groups": "shops"
        }
      }
    ]
  ]
}

# aws --profile ingestuser iam create-role --role-name shop-ingest
--assume-role-policy-document file://iam_role_policy_shops.json

```



```
# aws --profile ingestuser iam get-role --role-name shop-ingest
"Role": {
  "Path": "/",
  "RoleName": "shop-ingest",
  "RoleId": "eb0af92d-fd78-4b54-b3f4-bd158ea17f64",
  "Arn": "arn:aws:iam::role/shop-ingest",
  "CreateDate": "2024-04-08T07:35:03.318000+00:00",
  "AssumeRolePolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Federated": [

"arn:aws:iam::oidc-provider/keycloak-sso.ocp-eu-redbook-ce869a544b369f1dfd24beec1
0027762-i000.eu-es.containers.appdomain.cloud/auth/realms/ceph"

        ],
        "Action": [
          "sts:AssumeRoleWithWebIdentity"
        ],
        "Condition": {
          "StringLike": {

"keycloak-sso.ocp-eu-redbook-ce869a544b369f1dfd24beec10027762-i000.eu-es.container
s.appdomain.cloud/auth/realms/ceph:groups": "shops"

        }
      }
    ]
  },
  "MaxSessionDuration": 3600
}
```

Note: To learn more about the configuration options and values within our IAM role JavaScript Object Notation (JSON) policies, Chapter 7, “Amazon S3 enterprise user authentication and authorization”, of *IBM Storage Ceph Solutions Guide*, REDP-5715.

3. Now that you established the IAM role, configure its policy. This policy is a crucial element that dictates what resources users can access after assuming the role (Example 7-16).

Example 7-16 Configuring the IAM policy

```
$ cat iam_policy_shops_write.json
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:ListBucket",
      "s3:PutObjectTagging"
    ],
    "Resource": [
```

```

        "arn:aws:s3:::example-bucket/ingest/*",
        "arn:aws:s3:::example-bucket/ingest",
        "arn:aws:s3:::example-bucket/ingest/"
    ]
}
]
$ aws --profile ingestuser iam put-role-policy --role-name shop-ingest
--policy-name shop-ingest-write --policy-document
file:///iam_policy_shops_write.json

```

4. With the IAM role configuration complete, test the entire workflow. This step involves a shop user assuming the role and using it to write data to the ingest bucket (Example 7-17).

Example 7-17 Testing the entire workflow

```

$ ./test-assume-role.sh shopid-1001 Shopid-1001 shop-ingest
Getting the JWT from SS0...
Trying to Assume Role shop-ingest using provided JWT token.
Exporting AWS ENV variables to use the AWS CLI with the STS credentials provided
by RGW.

```

5. Perform a quick test to confirm successful object uploads while verifying that deletion is restricted (Example 7-18).

Example 7-18 Checking that we can upload objects but not delete them

```

$ aws s3 cp /etc/hosts s3://ingest/test1
upload: ../etc/hosts to s3://ingest/test1
$ aws s3 rm s3://ingest/test1
delete failed: s3://ingest/test1 argument of type 'NoneType' is not iterable

```

7.2 Raw zone: Branch stores classification data pipeline

This section describes the automated data pipeline for ingesting data from branch stores, classifying the data, and routing the data to the appropriate storage location within the raw zone. It describes the following topics:

- ▶ Landing zone and S3 bucket notifications:
 - Data from branch stores is uploaded to a designated landing zone bucket, which serves as a temporary storage area.
 - S3 bucket notifications are configured on the landing zone bucket. These notifications trigger actions whenever a new file is uploaded.
- ▶ Kafka topic and Knative integration:
 - The landing zone bucket is linked to a specific Kafka topic.
 - On receiving a notification (triggered by a new file upload), Knative, which is a serverless platform, automatically spins up a container to run a classification job.
- ▶ Data classification and routing:
 - The classification job analyzes the uploaded data to determine whether it contains personally identifiable information (PII).
 - PII Data: Data that contains PII is routed to the confidential bucket in the raw zone and tagged with “red” for identification.

- Non-PII Data: Data without PII is directed to the anonymized bucket in the raw zone and tagged with “green”.
- Legal hold with S3 object lock (Optional): The data pipeline can incorporate an S3 object lock legal hold function (optional). This feature can be activated in customer disputes that are related to specific purchase transactions.

Key points:

- This automated pipeline streamlines data ingestion and classification, which ensures efficient data storage and retrieval based on its sensitivity.
- S3 bucket notifications and Knative integration enable real-time data processing on upload.
- Data classification and tagging facilitate data organization and security within the raw zone.

In this example, we focus the components inside the large, blue circle in Figure 7-4.

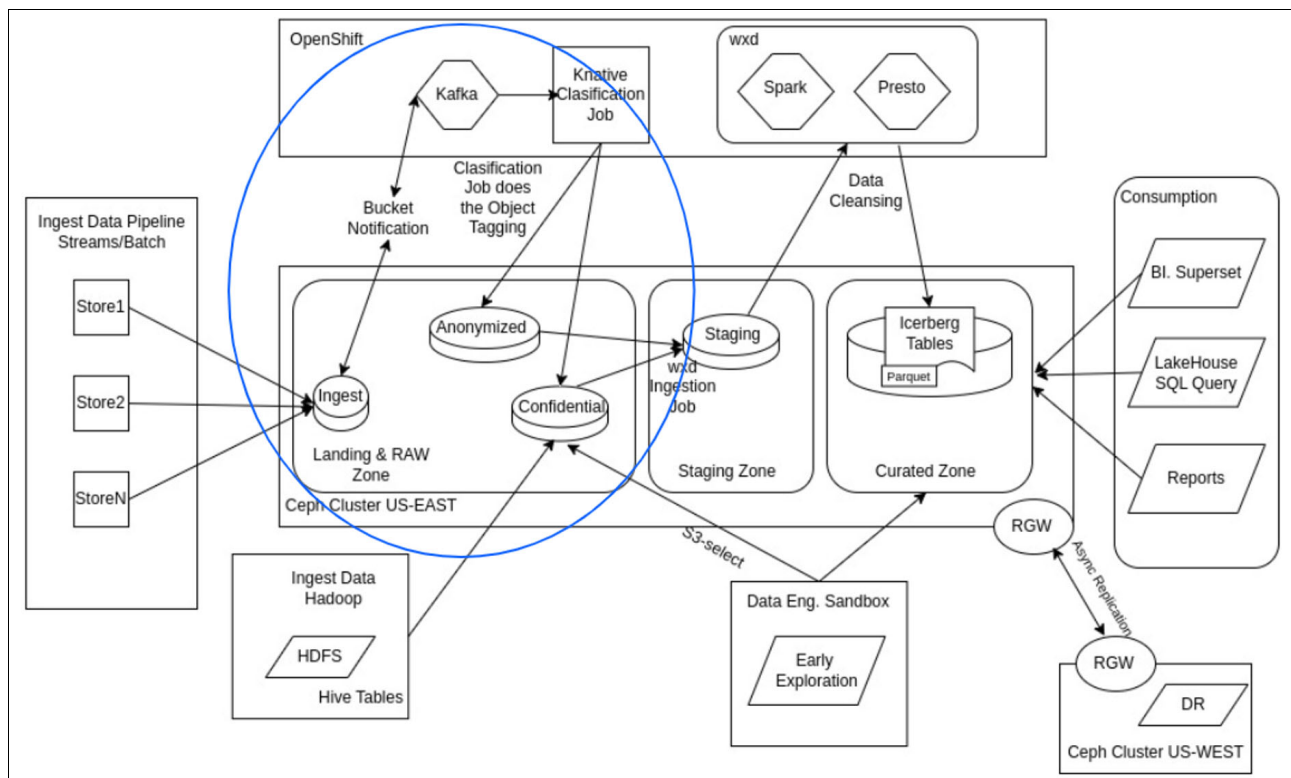


Figure 7-4 Data pipeline

We explain in detail, with hands-on examples, every step of the process.

We begin by configuring S3 bucket notifications on the ingest buckets. The goals of this step are as follows:

- ▶ Create a Kafka Topic that is called shop-ingest-pipe.
- ▶ Configure a topic on the bucket ingest that points to the Kafka deployment running on Red Hat OpenShift Container Platform.
- ▶ Configure a bucket notification on the ingest bucket to create an event for every object for the shop-ingest-pipe Kafka topic.

Chapter 5, “Amazon S3 bucket notifications for event-driven architectures”, in *IBM Storage Ceph Solutions Guide*, REDP-5715 provides a detailed step-by-step description about how to set up Kafka and S3 bucket notifications from scratch.

7.2.1 Raw zone: Red Hat OpenShift and Kafka setup

To illustrate, consider a basic example of a Kafka deployment running on Red Hat OpenShift Container Platform. This deployment listens through an Red Hat OpenShift Container Platform node port (Example 7-19).

Example 7-19 Kafka deployment running on Red Hat OpenShift Container Platform

```
# oc get pods -n kafka
```

NAME	READY	STATUS	RESTARTS	AGE
kafka-entity-operator-6854b788b9-jxs57	3/3	Running	3 (10d ago)	10d
kafka-kafka-0	1/1	Running	0	10d
kafka-kafka-1	1/1	Running	0	10d
kafka-kafka-2	1/1	Running	0	10d
kafka-zookeeper-0	1/1	Running	0	10d
kafka-zookeeper-1	1/1	Running	0	10d
kafka-zookeeper-2	1/1	Running	0	16d

```
# oc get svc kafka-kafka-routeplain-bootstrap-nkafka
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
kafka-kafka-routeplain-bootstrap	NodePort	172.21.169.227	<none>

```
9095:30131/TCP 17d
```

Create shop-ingest-pipe by using a Custom Resource (CR). A CR is a custom object in Kubernetes that you can use to describe and manage non-standard resources that are specific to your application (Example 7-20).

Example 7-20 Creating shop-ingest-pipe

```
# cat topic.yaml
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaTopic
metadata:
  generation: 1
  labels:
    strimzi.io/cluster: kafka
  name: shop-ingest-pipe
  namespace: kafka
spec:
  config:
```

```

    retention.ms: 604800000
    segment.bytes: 1073741824
    partitions: 10
    replicas: 3
# oc create -f topic.yaml -n kafka
# oc get kafkatopics.kafka.strimzi.io shop-ingest-pipe -n kafka
NAME                                CLUSTER  PARTITIONS  REPLICATION FACTOR
READY
shop-ingest-pipe    kafka    10          3
True

```

7.2.2 Raw zone: S3 bucket notification configuration

Complete the following steps:

1. Continue configuring the ingest bucket and adding a Simple Notification Service (SNS) topic with the Red Hat OpenShift Container Platform Kafka configuration details (Example 7-21).

Example 7-21 Configuring the ingest bucket and adding an SNS topic

```

# cat topic.json
{
  "push-endpoint": "kafka://10.251.0.43:30131",
  "verify-ssl": "False",
  "kafka-ack-level": "broker",
  "persistent": "true"
}
# aws --profile ingestuser sns create-topic --name shop-ingest-pipe
--attributes=file://topic.json --region default
# aws --profile ingestuser sns list-topics
{
  "Topics": [
    {
      "TopicArn": "arn:aws:sns:default::shop-ingest-pipe"
    }
  ]
}

```

2. Configure the notification event on the ingest bucket (Example 7-22).

Example 7-22 Configuring the notification event on the ingest bucket

```

# cat notification.json | jq .
"TopicConfigurations": [
  {
    "Id": "ingestnotificationknative",
    "TopicArn": "arn:aws:sns:default::shop-ingest-pipe",
    "Events": [
      "s3:ObjectCreated:*"
    ]
  }
]
# aws --profile ingestuser s3api put-bucket-notification-configuration --bucket
ingest --notification-configuration file://notification.json

```

```
# aws --profile ingestuser s3api get-bucket-notification-configuration --bucket
ingest | jq .
  "TopicConfigurations": [
    {
      "Id": "ingestnotificationknative",
      "TopicArn": "arn:aws:sns:default::shop-ingest-pipe",
      "Events": [
        "s3:ObjectCreated:*"
      ]
    }
  ]
```

3. Verify that the configuration for the Kafka topic and S3 bucket notification integration is working as expected by completing the following steps:
 - a. Start the Kafka consumer: Run the **kafka-console-consumer.sh** script with the necessary parameters to subscribe to the relevant Kafka topic.
 - b. Upload a test object: Upload a new object to the ingest bucket.
 - c. Monitor for notifications: Observe the output of the **kafka-console-consumer.sh** script. If the integration is functioning correctly, you should see a message appear within the console after uploading the object. This message indicates that the bucket notification triggered successfully and that the message was published to the Kafka topic.
4. Upload a test object to the ingest bucket by using a **PUT** request (Example 7-23).

Example 7-23 Using a PUT request on a test object to the `ingest` bucket

```
# aws --profile ingestuser s3 cp /etc/hosts s3://ingest/test1c8
upload: ../../etc/hosts to s3://ingest/test1c8
```

5. Confirm that the notification event was successfully sent to Kafka (Example 7-24).

Example 7-24 Checking for the notification event in Kafka

```
# oc exec -it kafka-kafka-0 -n kafka bash
$ cd /opt/kafka/bin
$ ./kafka-console-consumer.sh --bootstrap-server kafka-kafka-brokers:9092 --topic
shop-ingest-pipe --from-beginning
{"Records":[{"eventVersion":"2.2","eventSource":"ceph:s3","awsRegion":"default","e
ventTime":"2024-04-08T10:01:09.305906Z","eventName":"ObjectCreated:Put","userIdent
ity":{"principalId":"shopingest"},"requestParameters":{"sourceIPAddress":""},"resp
onseElements":{"x-amz-request-id":"fcabdf4a-86f2-452f-a13f-e0902685c655.57602.7475
548873844447640","x-amz-id-2":"e102-default-default"},"s3":{"s3SchemaVersion":"1.0
","configurationId":"ingestnotificationknative","bucketingest"},"ownerId
entity":{"principalId":"shopingest"},"arn":"arn:aws:s3:default::ingest","id":"fcab
df4a-86f2-452f-a13f-e0902685c655.47553.1"},"object":{"key":"test1c8","size":821,"e
Tag":"4fcb965c685a8c871c2a706f4249b779","versionId":"","sequencer":"65C01366CD3F8F
13","metadata":[{"key":"x-amz-content-sha256","val":"UNSIGNED-PAYLOAD"}, {"key":"x-
amz-date","val":"20240408T100109Z"}],"tags":[]},"eventId":"1712570469.328155.4fcb
965c685a8c871c2a706f4249b779","opaqueData":{""}]}
```

7.2.3 Raw zone: Setting up the raw zone buckets

Now, we respond to event notifications that we receive from the Kafka topic whenever a store uploads a new .csv file. The objective is to initiate a serverless process by using Knative for every event that is triggered. The process checks the uploaded object for any personal information and stores it in either the anonymized or confidential buckets of the raw zone.

To achieve this goal, complete the following steps:

1. Create and configure the anonymized and confidential buckets:
 - a. Enable object lock on the bucket, and enable S3 versioning (enabled with an object lock).
 - b. Set up S3 SSE with Amazon Key Management Service (SSE-KMS) encryption at rest.
 - c. Move cold data from the raw zone buckets to the hard disk drive (HDD) tier after 120 days.
 - d. Set up multi-factor authentication (MFA) delete: Deleting objects from the raw zone is only possible with MFA authentication.
 - e. Add S3 tags that represent the security level to the confidential and anonymized buckets:
 - Confidential tag: Security=Red
 - Anonymized tag: Security=Green
2. Create an IAM role with the required permissions to read data from the ingest bucket and write the processed data into the raw zone buckets.
3. Create a serverless Knative service to run the Python script for every incoming bucket notification on the Kafka topic.

7.2.4 Raw zone: S3 object lock and object versioning configuration

Now, create the two buckets for storing classified data:

- confidential: This bucket stores data containing PII.
- anonymized: This bucket holds data without PII.

Complete the following steps:

1. To manage object lifecycle and legal hold requirements, continue using the shoppingest user. This user has Ceph Object Storage admin privileges for both the landing and raw zones. Also, enable object locking on buckets containing confidential and anonymized data so that the classification script can initiate legal holds on specific objects when flagged by the legal team (Example 7-25).

Example 7-25 Creating the two buckets for storing classified data

```
$ aws --profile shoppingest s3api create-bucket --bucket confidential
--object-lock-enabled-for-bucket
$ aws --profile shoppingest s3api create-bucket --bucket anonymized
--object-lock-enabled-for-bucket
$ aws --profile shoppingest s3api get-object-lock-configuration --bucket test | jq
.
  "ObjectLockConfiguration": {
    "ObjectLockEnabled": "Enabled"
  }
}
```

2. Perform a quick test to verify that uploads of objects to this bucket do not have an object lock retention policy (Example 7-26).

Example 7-26 Verifying that uploads of objects to this bucket do not have an object lock retention policy

```
$ aws --profile shopingest s3api head-object --bucket test --key lock1 | jq .
  "AcceptRanges": "bytes",
  "LastModified": "2024-04-17T21:21:04+00:00",
  "ContentLength": 821,
  "ETag": "\"4fcb965c685a8c871c2a706f4249b779\"",
  "VersionId": "j81o6r09eU0lNEb9Fa0yy0TP0h8Xfnc",
  "ContentType": "binary/octet-stream",
  "Metadata": {}
$ aws --profile shopingest s3api put-object-legal-hold --bucket test --key lock1
--legal-hold Status=ON
$ aws --profile shopingest s3api head-object --bucket test --key lock1 | jq .
  "AcceptRanges": "bytes",
  "LastModified": "2024-04-17T21:21:04+00:00",
  "ContentLength": 821,
  "ETag": "\"4fcb965c685a8c871c2a706f4249b779\"",
  "VersionId": "j81o6r09eU0lNEb9Fa0yy0TP0h8Xfnc",
  "ContentType": "binary/octet-stream",
  "Metadata": {},
  "ObjectLockLegalHoldStatus": "ON"
```

3. Because object locking is now functioning as intended, now we describe object versioning. When you create a bucket with the **object-lock-enabled-for-bucket** parameter, S3 object versioning is automatically enabled on that bucket by default (Example 7-27).

Recommendation: Object versioning is not necessary in all object lock cases. Object lock already ensures data immutability. However, if you require the ability to revert to previous versions of objects, use object versioning.

Example 7-27 Creating the bucket with the parameter object-lock-enabled-for-bucket

```
$ aws --profile shopingest s3api get-bucket-versioning --bucket confidential
  "Status": "Enabled",
  "MFADelete": "Disabled"
$ aws --profile shopingest s3api get-bucket-versioning --bucket anonymized
  "Status": "Enabled",
  "MFADelete": "Disabled"
```

4. If you do not enable object lock and need only object versioning, you can enable it by running the command that is shown in Example 7-28.

Example 7-28 Enabling object versioning

```
$ aws --profile shopingest s3api put-bucket-versioning --bucket confidential
--versioning-configuration Status=Enabled
$ aws --profile shopingest s3api put-bucket-versioning --bucket anonymized
--versioning-configuration Status=Enabled
```

7.2.5 Raw zone: S3 SSE-KMS encryption at rest

To safeguard data confidentiality within the IBM Storage Ceph storage cluster, configure encryption at rest. Leverage IBM Security Key Lifecycle Manager, formerly known as GKLM, which is already integrated with our example IBM Storage Ceph environment.

This configuration uses *S3 PutBucketEncryption*. With this setting in place, every object that is uploaded to the designated bucket is transparently encrypted for the user. The encryption key itself is securely managed by GKLM, ensuring robust protection for your data at rest (Example 7-29).

Example 7-29 Configuring encryption at rest

```
$ cat kms-config.json
"Rules": [
  {
    "ApplyServerSideEncryptionByDefault": {
      "KMSEncryption": {
        "KMSMasterKeyID": "rgw00dc42a9b000000000",
        "SSEAlgorithm": "aws:kms"
      }
    }
  }
]
$ aws --profile shopingest s3api put-bucket-encryption --bucket anonymized
--server-side-encryption-configuration file://kms-config.json
$ aws --profile shopingest s3api put-bucket-encryption --bucket confidential
--server-side-encryption-configuration file://kms-config.json
$ aws s3api head-object --bucket anonymized --key shop3_22_03_2024.csv
  "AcceptRanges": "bytes",
  "LastModified": "2024-04-11T20:49:48+00:00",
  "ContentLength": 415589,
  "ETag": "\"4a59353820d1cbf9b3bcb7ca95af2bfa\"",
  "VersionId": "S4mQBMz09jb3NHPuHKYZWXQjRkMOuVa",
  "ContentType": "binary/octet-stream",
  "ServerSideEncryption": "aws:kms",
  "Metadata": {},
  "SSEKMSKeyId": "rgw00dc42a9b000000000"
```

7.2.6 Raw zone: S3 Multi-Factor Authentication delete

The data sets that are stored in the raw zone are valuable because they represent the *unaltered source of truth*. To protect these data sets from any mistakes, whether made by humans or automation, avoid granting users the ability to delete files from these buckets. Instead, configure the deletion of objects through lifecycle policies, and in cases where manual deletion is required, implement MFA delete, as shown in this example use case.

To further bolster the security of the data sets, enable MFA specifically for deletion actions. This extra layer of verification helps prevent unauthorized access and accidental data loss.

In this example, we use the OAuth tools package to implement *Time-based One-Time Password (TOTP)* as the MFA method. However, you may leverage any TOTP application, such as Google Authenticator, IBM Security Verify, and Authy.

Complete the following steps:

1. Install the OAuth tools on your Red Hat Enterprise Linux (RHEL) operating system (OS) server, which is designated as linux1. As a best practice, use the *Dandified YUM (DNF)* package manager, which is used to manage software on RHEL systems to install the required OAuth tools (Example 7-30).

Example 7-30 The dnf install oathtool command

```
$ dnf install oathtool -y
```

2. Create a seed by using the system's "urandom" module (Example 7-31).

Example 7-31 Creating a seed by using the system's "urandom" module

```
$ head -10 /dev/urandom | sha512sum | cut -b 1-30
2ed8dcdd34fe31b2d9174c6b25db98
```

3. Because the shopingest user is responsible for occasional deletions requiring MFA verification, configure a TOTP entry for their account. We name the totp-serial MFAdatalake, and we add the previously created totp-seed (Example 7-32).

Example 7-32 Configuring a TOTP entry for the account

```
$ radosgw-admin mfa create --uid=shopingest --totp-serial=MFAdatalake
--totp-seed=2ed8dcdd34fe31b2d9174c6b25db98
$ radosgw-admin user info --uid shopingest | jq .mfa_ids
"MFAdatalake"
$ radosgw-admin mfa list --uid shopingest
  "entries": [
    {
      "type": 2,
      "id": "MFAdatalake",
      "seed": "2ed8dcdd34fe31b2d9174c6b25db98",
      "seed_type": "hex",
      "time_ofs": 0,
      "step_size": 30,
      "window": 2
    }
  ]
```

4. Using the oathtool and the seed that we created, we can create TOTP Personal Identification Numbers (PINs) for the MFA. We can validate that our configuration is working by running the **radosgw-admin mfa check** command. We first get a PIN from the oathtool and then use it with the **radosgw-admin** command. We should get an "ok" response if the TOTP configuration is in sync (Example 7-33).

Example 7-33 The radosgw-admin mfa check command

```
$ oathtool -d6 --totp 2ed8dcdd34fe31b2d9174c6b25db98
032406
$ radosgw-admin mfa check --uid=shopingest --totp-serial=MFAdatalake
--totp-pin=032406
ok
```

5. After confirming a successful MFA setup, enable the S3 MFA delete feature on the confidential bucket. Because we do not have lifecycle expirations that are configured, we allow only object deletions through Multi-Factor Authentication delete.

Create a TOTP PIN with the oathtool (Example 7-34).

Example 7-34 Creating a TOTP PIN with the oathtool

```
$ oathtool -d6 --totp 2ed8dcd34fe31b2d9174c6b25db98
709585
```

6. Use this PIN with the MFA ID to enable the **MFADelete** option on the confidential bucket (Example 7-35).

Example 7-35 Enabling the MFADelete option on the confidential bucket

```
$ aws --profile shopingest s3api put-bucket-versioning --bucket confidential
--versioning-configuration Status=Enabled,MFADelete=Enabled --mfa "MFAdatalake
709585"
$ aws --profile shopingest s3api get-bucket-versioning --bucket confidential | jq
{
  "Status": "Enabled",
  "MFADelete": "Enabled"
}
```

7. Perform a quick test to verify that MFA is in place and that you cannot delete a file through the standard S3 API. Upload a test object that is called `mfa_delete_test` (Example 7-36).

Example 7-36 Verifying that MFA is in place

```
$ aws --profile shopingest s3api put-object --bucket confidential --key
mfa_delete_test --body /etc/hosts
{"ETag": "\"4fcb965c685a8c871c2a706f4249b779\"",
 "ServerSideEncryption": "aws:kms",
 "VersionId": "AXWdF4MbrvckwbygERv6uD4jCCiTu9q",
 "SSEKMSKeyId": "rgw00dc42a9b000000000",
 "SSEKMSEncryptionContext":
 "eyJhd3M6czM6YXJ1IjoieXJ1OmF3czpzMzo6c2hvcGluZ2VzdDpjbj25maWRlbnRpYWVwbWZhX2R1bGV0Z
V90ZXN0In0="}
```

8. Attempt to delete the object by using a standard `s3api delete-object` call; it fails (as expected) (Example 7-37).

Example 7-37 Trying to do a normal s3api delete-object

```
$ aws --profile shopingest s3api delete-object --bucket confidential --key
mfa_delete_test --version-id AXWdF4MbrvckwbygERv6uD4jCCiTu9q
argument of type 'NoneType' is not iterable
$ aws --profile shopingest s3 ls s3://confidential | grep mfa_delete_test
2024-04-12 09:39:01      821 mfa_delete_test
```

9. To enforce MFA for object deletion, implement the `--mfa` flag with the `s3api delete-object` call. This call requires a valid TOTP code to be provided for successful deletion. As shown in Example 7-38, using the correct MFA TOTP code allowed us to delete the test object.

Example 7-38 Adding the --mfa option

```
$ aws --profile shopingest s3api delete-object --bucket confidential --key
mfa_delete_test --version-id AXWdF4MbrvckwbygERv6uD4jCCiTu9q --mfa "MFAdatalake
460586"
$ aws --profile shopingest s3 ls s3://confidential | grep mfa_delete_test
```

7.2.7 Raw zone: Data tiering configuration - creating a cold storage class

To manage storage costs effectively, leverage data tiering for the less frequently accessed cold data in the raw zone buckets. To do so, use the following techniques:

- ▶ Lifecycle policy configuration: Configure a lifecycle policy that automates the migration of objects to a more cost-effective storage class. This policy is applied to all objects within the anonymized and confidential buckets.
- ▶ Migration to the HDD tier: After a predefined period of inactivity (for example, 90 days), objects that are migrated from the standard storage class might be retained for compliance or historical purposes.
- ▶ Tag-based filtering (optional): Similar to the filtering approach that is used with the ingest bucket example, lifecycle policies can also leverage tags for more granular control. Imagine your ingest jobs that process data from raw to curated zones set a tag to identify successfully imported objects. You can configure the lifecycle policy to migrate only tier objects with a specific tag (indicating processed data) to the HDD storage class. This approach ensures frequently accessed, recently ingested data remains in the higher-performing standard tier.

Complete the following steps:

1. Each of our Ceph object storage daemon (OSD) hosts has two Non-Volatile Memory Express (NVMe) OSDs and four HDD OSDs. This setup is represented in IBM Storage Ceph with device classes. IBM Storage Ceph detects device types and marks the device class. You can also configure custom device classes when creating the OSDs.

As an example, Example 7-39 shows a snippet of the output of the **ceph osd tree** command.

Example 7-39 The ceph osd tree command

```
# ceph osd tree
ID  CLASS  WEIGHT  TYPE NAME                STATUS  REWEIGHT  PRI-AFF
-1             4.56857  root default
-3             4.56857      datacenter madrid1
-2             0.76143        host ceph01
  0    ssd  0.13640            osd.0      up     1.00000  1.00000
  1    ssd  0.09769            osd.1      up     1.00000  1.00000
  2    hdd  0.09769            osd.2      up     1.00000  1.00000
 18    hdd  0.09769            osd.18     up     1.00000  1.00000
 28    hdd  0.16599            osd.28     up     1.00000  1.00000
 30    hdd  0.16599            osd.30     up     1.00000  1.00000
```

2. [CRUSH introduction](#) offers in-depth details about the Ceph Controlled Replication Under Scalable Hashing (CRUSH) algorithm. At a high level, CRUSH rules define data placement within the IBM Storage Ceph cluster based on the CRUSH map configuration. In Example 7-40, we are configuring our failure domain to host and setting the device class to use to a solid-state drive (SSD).

Example 7-40 Configuring the failure domain to the host and setting the device class to an SSD

```
# ceph osd crush rule dump replicated_rule
{
  "rule_id": 0,
  "rule_name": "replicated_rule",
  "type": 1,
  "steps": [
    {
      "op": "take",
```

```

        "item": -1,
        "item_name": "default~ssd"
    },
    {
        "op": "chooseleaf_firstn",
        "num": 0,
        "type": "host"
    },
    {
        "op": "emit"
    }
]

```

3. In our case, the data RGW pool, which stores all S3 objects, uses the `replicated_rule` you saw earlier. This rule plays a role in determining where your data is placed.

By default, the STANDARD storage class uses the `replicated_rule`. Your S3 objects within the STANDARD class are likely stored in the SSD storage tier, ensuring fast access and retrieval.

Key point: With IBM Storage Ceph, you can define different storage classes with varying performance and cost characteristics. You can choose the appropriate class based on your specific data needs.

Now, create an RGW datapool that consumes the four HDD drives that are available on each host by completing the following steps:

1. Create the rule that uses the HDD device class (Example 7-41).

Example 7-41 Creating the rule that uses the HDD device class

```
$ ceph osd crush rule create-replicated cold default host hdd
```

2. Create a pool that uses the cold CRUSH rule (Example 7-42).

Example 7-42 Creating a pool that uses the cold CRUSH rule

```
$ ceph osd pool create zone1.rgw.hdd.storage.class.buckets.data 32 32 cold
pool 'zone1.rgw.hdd.storage.class.buckets.data' created
$ ceph osd pool application enable zone1.rgw.hdd.storage.class.buckets.data rgw
enabled application 'rgw' on pool 'zone1.rgw.hdd.storage.class.buckets.data'
```

3. When the pool is ready, create a storage class that is named COLD.

Note: For compatibility with AWS software development kits (SDKs), it is a best practice to use storage classes names that are available in AWS S3.

4. Our zone uses a single placement target with a single configured storage class that is named STANDARD. This class leverages pools that are composed of NVMe drives (Example 7-43).

Example 7-43 STANDARD storage class

```
#radosgw-admin zone get | jq .placement_pools
{
  "key": "default-placement",
  "val": {
```

```

    "index_pool": "default.rgw.buckets.index",
    "storage_classes": {
      "STANDARD": {
        "data_pool": "default.rgw.buckets.data"
      }
    },
    "data_extra_pool": "default.rgw.buckets.non-ec",
    "index_type": 0,
    "inline_data": true
  }
}

```

5. Add the storage class to the zonegroup (Example 7-44).

Example 7-44 Adding the storage class to the zonegroup

```

$ radosgw-admin zonegroup placement add --rgw-zonegroup default --placement-id
default-placement --storage-class COLD
{
  "key": "default-placement",
  "val": {
    "name": "default-placement",
    "tags": [],
    "storage_classes": [
      "COLD",
      "STANDARD"
    ]
  }
}

```

6. Configure the datapool that we use for the COLD storage class (Example 7-45).

Example 7-45 Configuring the datapool for the COLD storage class

```

$ radosgw-admin zone placement add --rgw-zone default --placement-id
default-placement --storage-class COLD --data-pool
zone1.rgw.hdd.storage.class.buckets.data
$ radosgw-admin zone get | jq .placement_pools
{
  "key": "default-placement",
  "val": {
    "index_pool": "default.rgw.buckets.index",
    "storage_classes": {
      "COLD": {
        "data_pool": "zone1.rgw.hdd.storage.class.buckets.data"
      },
      "STANDARD": {
        "data_pool": "default.rgw.buckets.data"
      }
    },
    "data_extra_pool": "default.rgw.buckets.non-ec",
    "index_type": 0,
    "inline_data": true
  }
}
$ radosgw-admin period update --commit

```

This configuration optimizes storage costs and performance. It provides two storage classes:

- **STANDARD:** Uses high-performance, all-flash drives for frequently accessed data.
- **COLD:** Leverages cost-effective HDDs for less frequently accessed data.

A lifecycle policy (Figure 7-5) automatically migrates objects in raw zone buckets from STANDARD to COLD storage after 90 days of inactivity. This migration ensures that frequently used data remains on high-performance storage while inactive data is moved to a more cost-efficient tier.

Example 7-46 Lifecycle policy rule

```
$ cat lc_rawzone_transition_rule.json
{
  "Rules": [
    {
      "ID": "Move to COLD after 90 days",
      "Prefix": "",
      "Status": "Enabled",
      "Transition": {
        "Days": 90,
        "StorageClass": "COLD"
      }
    }
  ]
}

$ aws --profile shopingest s3api put-bucket-lifecycle-configuration
--lifecycle-configuration file://lc_rawzone_transition_rule.json --bucket
anonymized

$ aws --profile shopingest s3api put-bucket-lifecycle-configuration
--lifecycle-configuration file://lc_rawzone_transition_rule.json --bucket
confidential
```

7.2.8 Raw zone: Authentication - IAM role configuration for the raw zone

This section describes creating an IAM role that grants read access to the ingest bucket and write access for processed data to the raw zone buckets. To accomplish this task, complete the following steps:

1. Configure user access at the IDP level. In our case, the IDP is AD. We create a user that is named user-ops-01 who belongs to the rawzoneadmin Lightweight Directory Access Protocol (LDAP)/AD group. Because our RHSSO federates with AD, these users are automatically mapped to RHSSO, as shown in Figure 7-5.

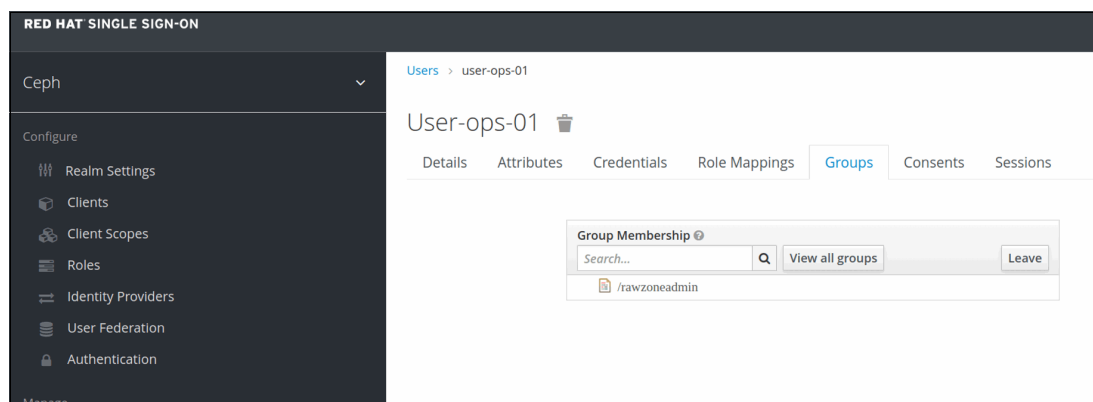


Figure 7-5 User map

2. In the same fashion as with the IAM setup that we used for the stores, we create an IAM role that enables the following actions:
 - Source bucket ingest: Read and Set Object Attributes
 - Destination buckets:
 - Confidential: Write(No Delete) and Set Object Attributes
 - Anonymized: Write(No Delete) and Set Object Attributes
3. The trust policy that is associated with our IAM role specifies that only members of the LDAP group rawzoneadmin can assume this role (Example 7-47).

Example 7-47 IAM role doc policy

```
$ cat iam_role_policy_shop_read.json
{"Version": "2012-10-17",
 "Statement": [
   {
     "Effect": "Allow",
     "Principal": {
       "Federated": [
         "arn:aws:iam:::oidc-provider/keycloak-sso.ocp-eu-redbook-ce869a544b369f1dfd24beec10027762-i000.eu-es.containers.appdomain.cloud/auth/realms/ceph"
       ]
     },
     "Action": [
       "sts:AssumeRoleWithWebIdentity"
     ],
     "Condition": {
       "StringLike": {
         "keycloak-sso.ocp-eu-redbook-ce869a544b369f1dfd24beec10027762-i000.eu-es.container
s.appdomain.cloud/auth/realms/ceph:groups": "rawzoneadmin"
       }
     }
   }
 ]
}
```

```
$ radosgw-admin role create --role-name raw-zone-admin
--assume-role-policy-doc=$(jq -rc . /root/iam_role_policy_shop_read.json)
```

4. Now that we established the IAM role for data ingestion, we define the specific permissions that it has. IAM roles can be assigned to one or more policies, enabling granular control over user access. We start by configuring a policy that is specific to accessing the ingest bucket (Example 7-48).

Example 7-48 Configuring a policy that is specific to accessing the ingest bucket

```
$ cat iam_policy_shops_read.json
{"Version": "2012-10-17",
 "Statement": [
   {
     "Effect": "Allow",
     "Action": [
       "s3:ListBucket",
       "s3:Get*",
       "s3:PutObjectTagging"
     ]
   }
 ],
}
```

```

        "Resource": [
            "arn:aws:s3:::ingest/*",
            "arn:aws:s3:::ingest",
            "arn:aws:s3:::ingest/"
        ]
    }
]
$ radosgw-admin role policy put --role-name=raw-zone-admin
--policy-name=raw-read-from-ingest --policy-doc=$(jq -rc .
/root/iam_policy_shops_read.json)

```

5. We add a second policy that covers the **PUT** access to the raw zone buckets (Example 7-49).

Example 7-49 Adding a second policy

```

$ cat iam_policy_raw_zone_write.json
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:ListBucket",
            "s3:Get*",
            "s3:PutObjectTagging"
        ],
        "Resource": [
            "arn:aws:s3:::anonymized/*",
            "arn:aws:s3:::anonymized",
            "arn:aws:s3:::anonymized/",
            "arn:aws:s3:::confidential/*",
            "arn:aws:s3:::confidential",
            "arn:aws:s3:::confidential/"
        ]
    }
]
$ radosgw-admin role policy put --role-name=raw-zone-admin
--policy-name=raw-write-to-rawbuckets --policy-doc=$(jq -rc .
/root/iam_policy_raw_zone_write.json)

```

6. Now that we created the IAM role raw-zone-admin, we perform a quick test to ensure that it functions as intended. We assume the role and attempt to access relevant resources (Example 7-50).

Example 7-50 Testing the raw-zone-admin

```

# source ./test-assume-role.sh user-ops-01 User-ops-01 raw-zone-admin
Getting the JWT from SS0...
Trying to Assume Role raw-zone-admin using the provided JWT token.
Export AWS ENV variables to use the AWS CLI with the STS creds.

```

7. We test the ingest bucket (Example 7-51).

Example 7-51 Testing the ingest bucket

```
[root@ceph01 ~]# aws s3 ls s3://ingest
2024-04-11 05:13:04      379997 shop1_22_03_2024.csv
[root@ceph01 ~]# aws s3 cp s3://ingest/shop1_22_03_2024.csv /tmp
download: s3://ingest/shop1_22_03_2024.csv to ../tmp/shop1_22_03_2024.csv
[root@ceph01 ~]# aws s3 cp /etc/hosts s3://ingest/
upload failed: ../etc/hosts to s3://ingest/hosts argument of type 'NoneType' is
not iterable
```

8. We test the anonymized and confidential buckets (Example 7-52).

Example 7-52 Testing the anonymized and confidential buckets

```
# aws s3 ls s3://anonymized && aws s3 ls s3://confidential
2024-04-11 05:13:09      405002 shop1_22_03_2024.csv
2024-04-11 02:12:47      235464 shop3_28_03_2024.csv
# aws s3 rm s3://anonymized/shop1_22_03_2024.csv
delete failed: s3://anonymized/shop1_22_03_2024.csv argument of type 'NoneType' is
not iterable
```

Note: The roles and policies that are used in this scenario are available at this [GitHub repository](#).

7.2.9 Raw zone: Serverless data pipeline workflow

This section describes creating a serverless Knative service to run a Python script for every incoming bucket notification on the Kafka topic.

Our goal is to run a Python classification script that processes ingest data from branch stores. This script categorizes CSV files based on data confidentiality. In our example, files containing personal information are classified as confidential and stored in the designated bucket. Conversely, files without personal information are classified as non-personal and stored in the anonymized bucket. Also, the script tags objects with “red” for personal data and “green” for non-personal data. Finally, to preserve data for potential legal issues, the script leverages the S3 object lock to apply a “Legal Hold” on objects containing customer complaints about transactions.

Benefits of using a classification script:

- ▶ Improved data organization: Automated classification ensures that data is stored securely based on confidentiality.
- ▶ Enhanced search and access: Data tagging facilitates identification and retrieval of specific data types.
- ▶ Legal compliance: S3 object lock safeguards sensitive data, potentially aiding in legal matters.

Figure 7-6 on page 109 provides a high-level description of the workflow.

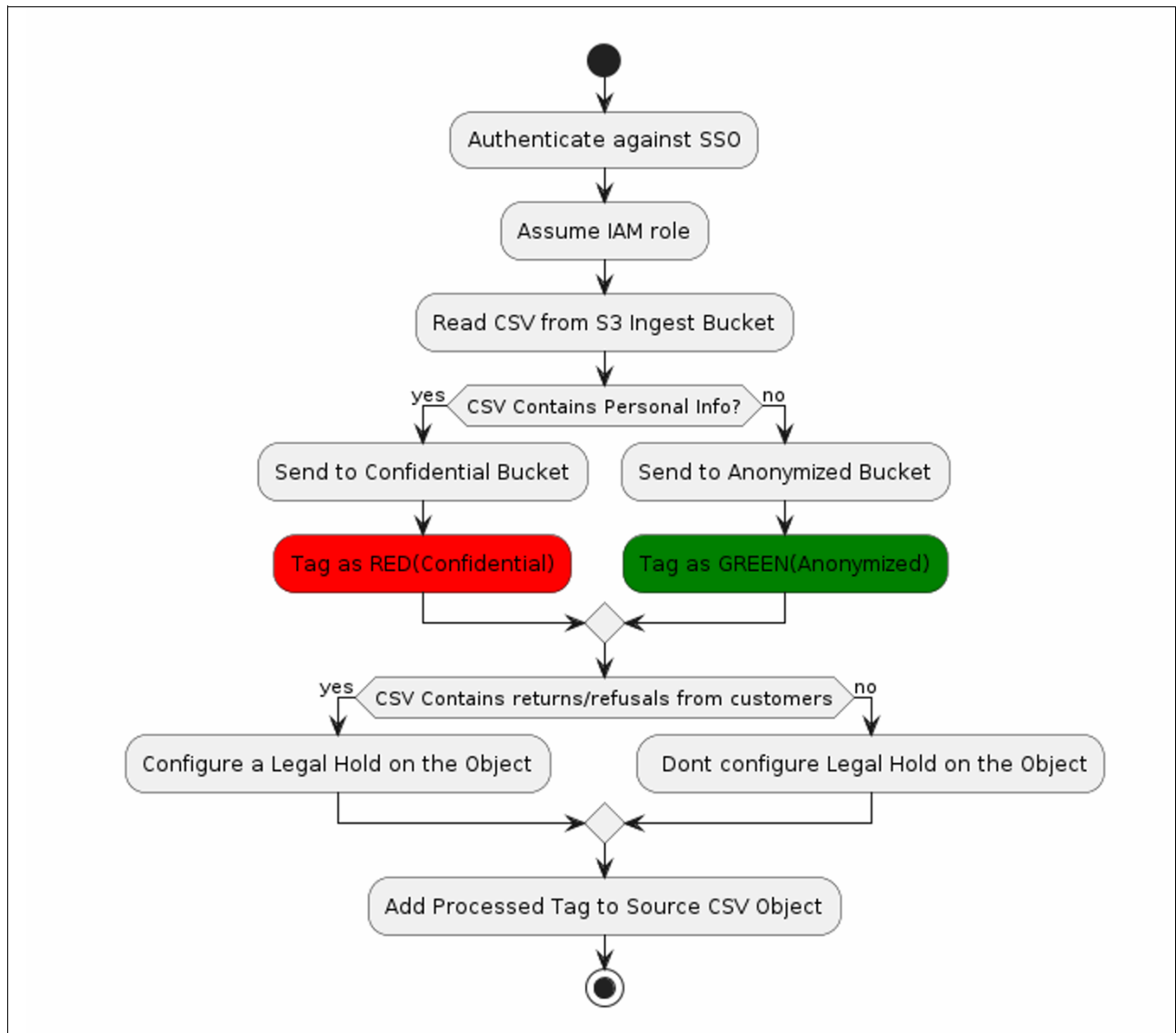


Figure 7-6 High-level description of the ingest data workflow

The Python script can be found at this [GitHub repository](#).

Our Python classification script is ready for deployment. The next step is to integrate the script within the pipeline. Here is the workflow of the pipeline (Figure 7-7):

1. The branch stores upload the end-of-day CSV files.
2. An S3 bucket notification triggers on upload.
3. The notification sends an event to Kafka.
4. The Kafka event triggers a Knative serverless job.
5. This job runs the script, classifies the data, and moves it to the appropriate bucket (confidential or anonymized).
6. Also, the script tags objects based on data type (“red” for personal, and “green” for non-personal) and applies legal holds to relevant files based on customer complaints.

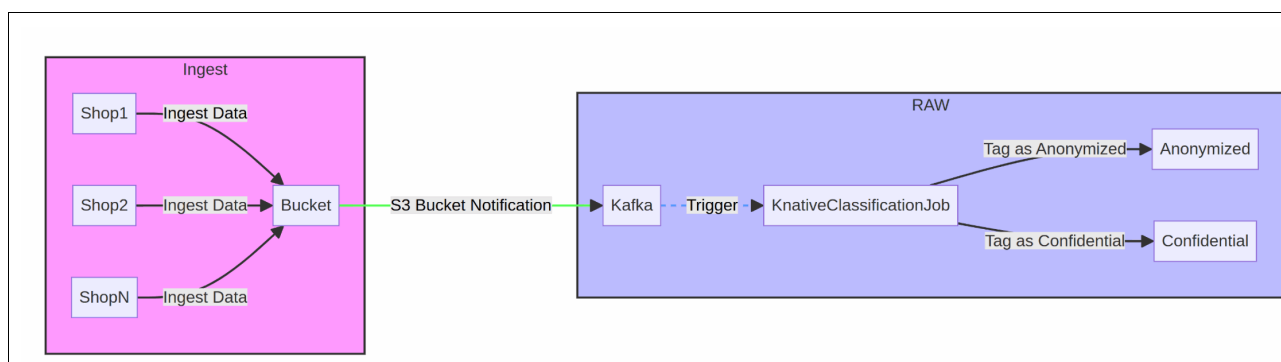


Figure 7-7 Data corresponding bucket flow

With a clear understanding of the workflow, begin the implementation process. We already have the ingest bucket with S3 bucket notifications that are configured against our Kafka topic that is called shop-ingest-pipe. So, we must work on the serverless (Knative) implementation.

To ensure smooth execution by the serverless function, our Python script/app must run within a container. This containerization approach provides isolation and a consistent environment, regardless of the serverless platform's underlying infrastructure.

The Dockerfile containing the instructions to build the container image for our Python script/app is available in this [GitHub repository](#). From that Dockerfile, we created a container image and uploaded it to the [quay.io image registry](#).

7.2.10 Raw zone: Knative and Kafka service deployment

The next step is to configure a Knative service by completing the following steps:

1. Knative operators are deployed and running on the Red Hat OpenShift Container Platform cluster (Example 7-53).

Example 7-53 Knative operators that are deployed and running on our Red Hat OpenShift Container Platform cluster

```
# oc get csv -n openshift-serverless
serverless-operator.v1.32.1      Red Hat OpenShift Serverless    1.32.1
serverless-operator.v1.32.0      Succeeded
```

2. Deploy the Knative Eventing, Serving, and Kafka operators by using their default configurations. These operators offer customization options for advanced configurations if needed (Example 7-54).

Example 7-54 Deploying the Eventing, Serving, and Knative Kafka operators

```
$ oc get crd knativeeventings.operator.knative.dev
NAME                                CREATED AT
knativeeventings.operator.knative.dev 2024-04-08T13:45:29Z
$ oc get crd knativeservings.operator.knative.dev
NAME                                CREATED AT
knativeservings.operator.knative.dev 2024-04-08T13:45:26Z
$ oc get crd knativekafkas.operator.serverless.openshift.io
NAME                                CREATED AT
knativekafkas.operator.serverless.openshift.io 2024-04-08T13:45:26Z
```

3. After successfully deploying the operators, create a Knative serving service, as shown in Example 7-55. This service is responsible for the following items:

- Instantiation: On receiving a request, the service automatically creates an instance of the Python classification application.
- Data classification: The instantiated application processes the incoming data and performs the classification task.

This approach ensures efficient resource utilization by dynamically provisioning application instances based on incoming data.

Example 7-55 Creating a Knative serving service

```
cat service-knative.yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: ingest-to-raw
spec:
  template:
    metadata:
      annotations:
        autoscaling.knative.dev/target: "1"
        revisionTimestamp: ""
    spec:
      containers:
      - name: ingest-to-raw
        image: quay.io/dparkes/ingest_to_raw:latest
        env:
        - name: SOURCE_ROLE_ARN
          value: "arn:aws:iam:::role/raw-zone-admin"
        - name: DESTINATION_ROLE_ARN
          value: "arn:aws:iam:::role/raw-zone-admin"
        - name: OIDC_PROVIDER_URL
          value: "https://keycloak-sso.ocp-eu-redbook-ce869a544b369f1dfd24beec10027762-i000.eu-es.c
ontainers.appdomain.cloud/auth/realms/ceph/protocol/openid-connect"
        - name: OIDC_CLIENT_SECRET
          value: "Secret"
        - name: OIDC_CLIENT_ID
          value: "ceph"
```

```

- name: OIDC_USERNAME
  value: "user-ops-01"
- name: OIDC_PASSWORD
  value: "XXXX"
- name: S3_ENDPOINT_URL
  value: "https://s3.cephlabs.com"
- name: STS_ENDPOINT_URL
  value: "https://s3.cephlabs.com"
$ oc create -f service-knative.yaml
$ oc get service.serving.knative.dev/ingest-to-raw
NAME          URL
LATESTCREATED  LATESTREADY    READY  REASON
ingest-to-raw
https://ingest-to-raw-rawtest.ocp-eu-redbook-ce869a544b369f1dfd24beec10027762-i000
.eu-es.containers.appdomain.cloud ingest-to-raw-00001 ingest-to-raw-00001
True

```

4. Every time that the Knative service gets a request to <https://ingest-to-raw-rawtest.ocp-eu-redbook-ce869a544b369f1dfd24beec10027762-i000.eu-es.containers.appdomain.cloud>, it creates a pod with the container image and processes the data. For example, when we run a `curl` command that issues a `GET` request, you can see how the two pods run immediately (Example 7-56).

Example 7-56 The `oc get deployment` command

```

$ oc get deployment
NAME                                READY  UP-TO-DATE  AVAILABLE  AGE
ingest-to-raw-00001-deployment      0/0    0           0          27h
$ curl -q
https://ingest-to-raw-rawtest.ocp-eu-redbook-ce869a544b369f1dfd24beec10027762-i000
.eu-es.containers.appdomain.cloud
Health OK%
$ oc get deployment
NAME                                READY  UP-TO-DATE  AVAILABLE  AGE
ingest-to-raw-00001-deployment      2/2    2           2          27h

```

Our Python classification script relies on CloudEvents to receive data and initiate the classification process. Here is how it works:

- ▶ **S3 bucket notifications:** When a new object arrives in the shop-ingest-pipe Kafka topic (triggered by changes in the ingest bucket), a notification is published.
- ▶ **Knative bridge with KafkaSource custom resource definition (CRD):** We defined a CRD that is called `KafkaSource`. This CRD acts as a bridge, seamlessly transforming the Kafka messages (like the S3 bucket notifications) into CloudEvents.
- ▶ **Delivery to Knative Service (Sink):** The Knative integration delivers these CloudEvents to the Knative service that we created earlier (See Example 7-55 on page 111). This Knative service acts as the sink in this scenario.

Key takeaways:

- ▶ The Knative Kafka integration, through the KafkaSource CRD, bridges the gap between Kafka messages and CloudEvents.
- ▶ CloudEvents provide a standardized format for event data, simplifying communication between different systems like Kafka and our Python script.
- ▶ The Flask route acts as a listener, enabling the script to receive and process the CloudEvents containing S3 bucket notification details.

Figure 7-8 shows a visual representation of the workflow.

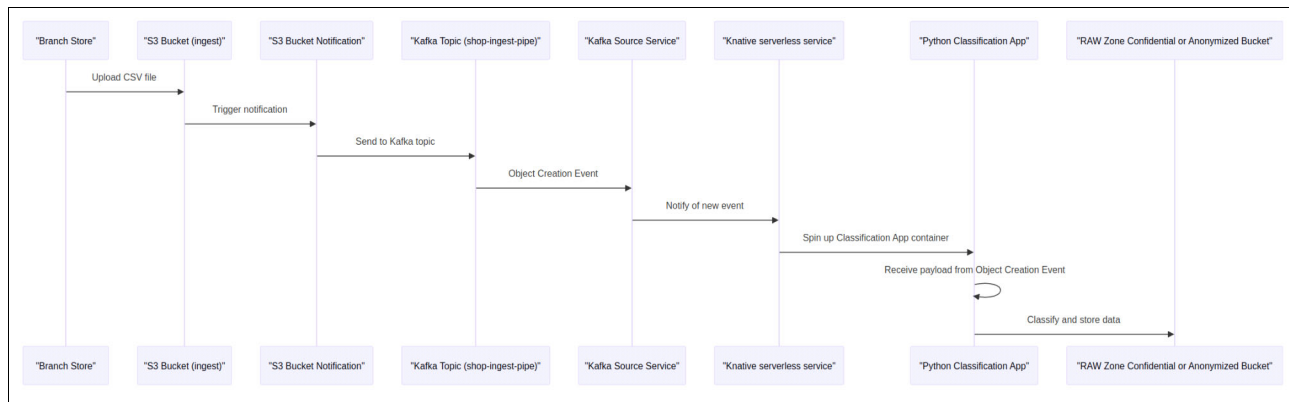


Figure 7-8 Visual representation of the workflow

Our Python classification script is shown in Example 7-57.

Example 7-57 Python classification script

```
$ cat kafka_knative_ingest_to_raw.yaml
apiVersion: sources.knative.dev/v1beta1
kind: KafkaSource
metadata:
  name: ingest-to-raw
spec:
  consumerGroup: ingest-to-raw
  bootstrapServers:
    - kafka-kafka-bootstrap.kafka.svc.cluster.local:9092
  topics:
    - shop-ingest-pipe
  sink:
    ref:
      apiVersion: serving.knative.dev/v1
      kind: Service
      name: ingest-to-raw
$ oc create -f kafka_knative_ingest_to_raw.yaml

$ oc get KafkaSource
NAME          TOPICS          BOOTSTRAPSERVERS
READY  REASON  AGE
ingest-to-raw  ["shop-ingest-pipe"]
["kafka-kafka-bootstrap.kafka.svc.cluster.local:9092"]  True  1m
```

7.2.11 Raw zone: Ingest data pipeline verification

Now, we have our full workflow setup. The raw zones buckets are created and configured. Our serverless classification app is waiting for new .csv files to be ingested by the shop branches. Let us ingest a .csv file from shop1 and follow the data pipeline flow:

1. We have a sample data set generator in the [GitHub repo](#) that we use to create an example CSV file and upload it to the ingest bucket, starting the pipeline (Example 7-58).

Example 7-58 Sample data set generator

```
$ python /home/user/./
IBM-Storage-Ceph-as-a-Data-Lakehouse-platform-for-IBM-watsonx.data-and-Beyond/raw_
zone_processing/fake_data_generation/dataset_generator.py shop1 01-04-2024 6000
--include-personal-data
INFO:root:Fake data generated and saved to 'shop1_01_04_2024.csv'
INFO:botocore.credentials:Found credentials in shared credentials file:
~/aws/credentials
INFO:root:Uploaded shop1_01_04_2024.csv to S3 bucket ingest
```

2. If we peek at the Kafka topic, we can see that the “object created” event arrived (Example 7-59).

Example 7-59 Object created event arrived

```
./kafka-console-consumer.sh --bootstrap-server kafka-kafka-brokers:9092 --topic
shop-ingest-pipe
{"Records":[{"eventVersion":"2.2","eventSource":"ceph:s3","awsRegion":"default","e
ventTime":"2024-04-12T13:59:01.469136Z","eventName":"ObjectCreated:Put","userIdent
ity":{"principalId":"shopid-1001"},"requestParameters":{"sourceIPAddress":""},"res
ponseElements":{"x-amz-request-id":"fcabdf4a-86f2-452f-a13f-e0902685c655.58499.660
0692889286183978","x-amz-id-2":"e483-default-default"},"s3":{"s3SchemaVersion":"1.
0","configurationId":"ingestnotificationnative","bucket":{"name":"ingest","ownerI
dentity":{"principalId":"shopid-1001"},"arn":"arn:aws:s3:default::ingest","id":"fc
abdf4a-86f2-452f-a13f-e0902685c655.47553.1"},"object":{"key":"shop1_01_04_2024.csv
","size":665078,"eTag":"0f735216f002184af224946f8d451018","versionId":"","sequen
ceNumber":"253E196640A8371D","metadata":[{"key":"x-amz-content-sha256","val":"UNSIGN
ED-PAYLOAD"}, {"key":"x-amz-date","val":"20240412T135900Z"}, {"key":"x-amz-security-token
","val":"B5CXfb0uiSSiFPST8k6fs9rncQfzK47fdnxSIDE00uwJr1AqJyS4S100yu7LudYIN8/8cMUuc
Ob9QhRsAv0VqVat6HrU6Vf7Vh1g8y0cKtNz83A1/1q9MsMmFjSfJw8sc61RhQ7WGmsExkWHQdv1CdF+FHQ
sxj+y/SuVb8CM5RQ9+XhW/BLr4jJxz/vt4rx5S0tgf9TG6LzhTs5wzkw0wyjX129xohsQbVsinSEarhq
ISdpWfcMnx/BuhdMPDaWh3P8QesPj/5xZ16auIdFszEGxyrAEuLK8L4BFnM1r0LGn1FA5SIsm2K+c2e/+u
wNyut8mLciIdm5LHYST6WzmuwP+We3tCgBh5dcb00+Ej760JXYfJjy1A5ZL0kX111S0j4Hs1vXwDHU919J
jE7mApusVsmkMwR932pmKV53P1k5m9sWcA5LwVpGL0Tr1OFHxc3v2+8BQZNtDicUhFvdGJc9GL4G9C7aCu
ooAvu8DrUpkwd+pSnU3Be9Sc0HpMYAprqM6SVb0xmM3n0NhTgK1s9h4EFAnc4wBbbwJKURwi++bvKMy1Xc
sfWPZVPd7pT"}]},"tags":[]},"eventId":"1712930341.490186.0f735216f002184af224946f8d
451018","opaqueData":""}]}
```

3. By examining the application pod logs, we can confirm that the serverless application initiated file processing (Example 7-60 on page 115).

Example 7-60 Application pod logs

```
$ oc logs ingest-to-raw-00001-deployment-6dfd748f9b-5npqs
INFO:werkzeug:Press CTRL+C to quit
INFO:root:ingest shop1_01_04_2024.csv
INFO:root:Modified CSV uploaded to S3: shop1_01_04_2024.csv
INFO:root:Object tagged as processed: shop1_01_04_2024.csv
```

4. Now, we verify that the CSV file containing personal information arrived at its secure destination. We use the head-object call to the confidential bucket by using the object name as the key. This call confirms the file's presence without downloading it. Because the confidential bucket uses SSE-KMS, the data remains encrypted at rest (Example 7-61).

Example 7-61 Checking whether the .csv file landed at its final destination

```
# aws s3api head-object --bucket confidential --key shop1_01_04_2024.csv
  "AcceptRanges": "bytes",
  "LastModified": "2024-04-12T13:59:06+00:00",
  "ContentLength": 695083,
  "ETag": "\"2aa674fca996e406c6d04f7bea9e28f2\"",
  "VersionId": "jy2qDVQjr4r8mId-fMS9aB-QvuVTuCz",
  "ContentType": "binary/octet-stream",
  "ServerSideEncryption": "aws:kms",
  "Metadata": {},
  "SSEKMSKeyId": "rgw00dc42a9b0000000000"
```

5. We verify that the object has an S3 tag, which is a label that is used to categorize data. Because data contains PII, the tag value for the data security classification should be red (Example 7-62).

Example 7-62 Checking whether the object has the S3 tag for the data security classification

```
$ aws s3api get-object-tagging --bucket confidential --key shop1_01_04_2024.csv
  "TagSet": [
    {
      "Key": "DataClassification",
      "Value": "red"
    }
  ]
```

6. Confirm whether S3 Object Legal Hold is activated for the ingested CSV files that contain legal conflicts from the stores (Example 7-63).

Example 7-63 Final check

```
$grep legal shop4_03_06_2024.csv
46806,22067,Briefcase,5,03-06-2024
12:08,80.98,33590,Italy,Cash,Accessories,246-40-6771,krista38@example.net,legal
$ aws --profile shopingest s3api head-object --bucket test --key
shop4_03_06_2024.csv | jq .
  "AcceptRanges": "bytes",
  "LastModified": "2024-04-17T21:21:04+00:00",
  "ContentLength": 821,
  "ETag": "\"4fcb965c685a8c871c2a706f4249b779\"",
  "VersionId": "j81o6r09eU01NEb9Fa0yy0TP0h8Xfnc",
  "ContentType": "binary/octet-stream",
  "Metadata": {},
  "ObjectLockLegalHoldStatus": "ON"
```

7.2.12 Raw zone: Hadoop data ingest into the raw zone

We established the raw zone for ingesting data from branch stores. We also have a Hadoop cluster with archived data that we want available for comparison in our curated zone data. So, we must configure an on-demand ingest of Hadoop Distributed File System (HDFS) data into our data lake. By enabling on-demand ingest, we empower final users at the consumer layer to query this historical data alongside our curated zone data sets. This flexibility enables users to leverage various tools for analysis, including watsonx.data and other SQL engines.

Chapter 5, “IBM Storage Ceph with IBM watsonx.data” on page 67 offers a comprehensive hands-on example of Amazon S3A File System (S3A) driver/connector usage for data movement between HDFS and IBM Storage Ceph Object. After the data is within a Ceph bucket, you can import it into your watsonx.data curated collection by following the same process that is used for ingested CSV files from your stores.

7.2.13 Raw zone: E-commerce platform data ingest

E-commerce platforms are the backbone of modern retail: The browsing logs mixed with the transaction and purchase information form an invaluable source of data. Browsing logs typically include data on user sessions, page views, clicks, interaction times, cart additions, and other user actions on the website. These logs are generated by web servers, application servers, and client-side tracking systems, such as JavaScript running in the user's browser. This browser data must be recollected and sent to the centralized data lake for further processing. Here are some examples about how the data may be transmitted to the main site:

- ▶ Direct upload: Data is sent directly from the source (for example, web servers or client-side browsers) to the data lake.
- ▶ Log shipping: Server logs are periodically collected and sent to the data lake by using automated scripts.
- ▶ Data streaming: Real-time data streaming that uses technologies like Apache Kafka enables the live streaming of log data to the data lake, enabling near real-time data processing and analytics.

In our scenario, we leverage log shipping to periodically collect browser log data sets. These data sets are transformed into CSV files and uploaded to a designated bucket that is named “ecommandlogs” within our storage system. This upload triggers an automated data pipeline that is facilitated by bucket notifications and Kafka, which performs some initial data processing.

Figure 7-9 shows the data pipeline architecture for e-commerce.

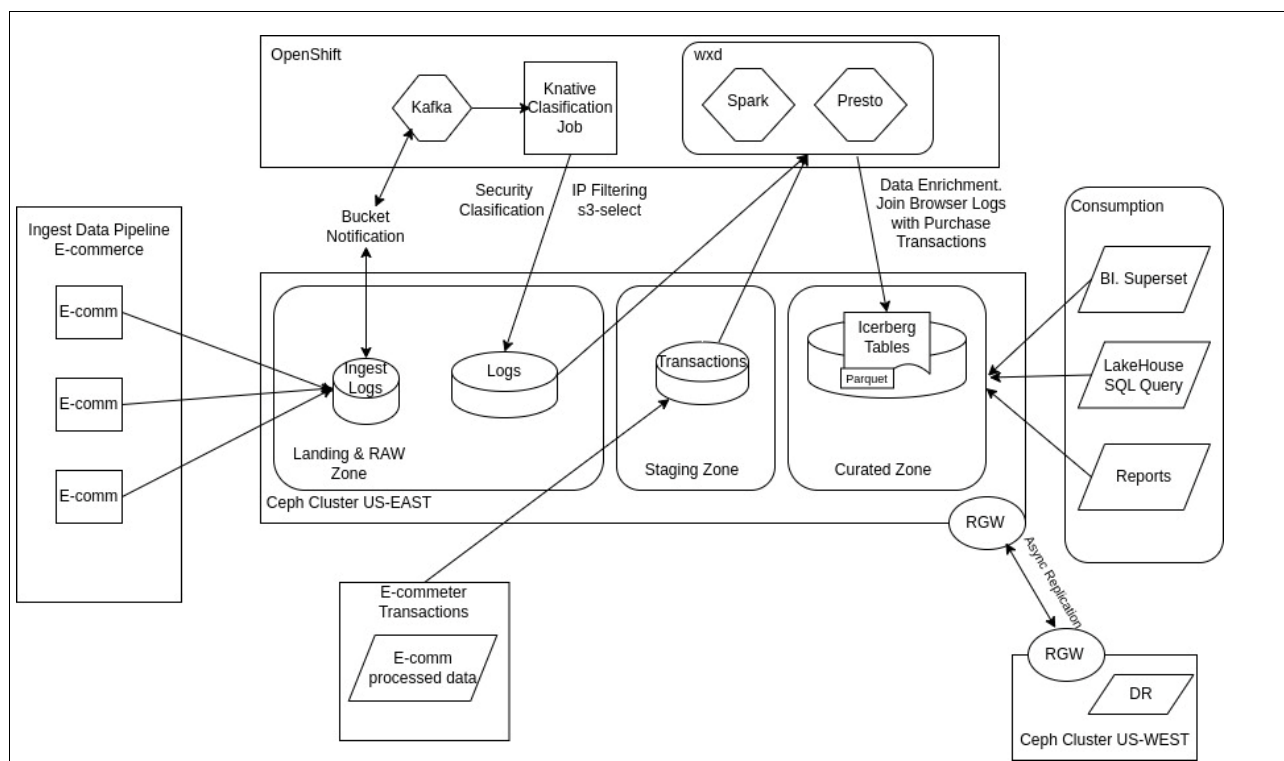


Figure 7-9 Data pipeline architecture for e-commerce

To avoid repetition, we skip the detailed workflow explanation because it mirrors the physical stores use case that is described in 7.2, “Raw zone: Branch stores classification data pipeline” on page 92. For more information, the Red Hat OpenShift deployment details for the e-commerce pipeline, including the [Knative serverless application example code](#) and sample [Red Hat OpenShift Custom Resources](#), are available in the [GitHub repository](#).

7.2.14 Early raw zone data filtering with S3 Select

As e-commerce platforms flourish, the volume of generated browsing logs inevitably grows exponentially. Although this data offers many insights into customer behavior, it can also contain unwanted elements such as activity from malicious IP addresses. Processing and storing such massive data sets can strain storage resources and extend processing times and costs during the analytics phase.

The Ceph *S3 Select* feature offers a strategic advantage to address these challenges. S3 Select enables querying and retrieving specific subsets of data directly within S3 objects that are stored on the Ceph layer. By applying S3 Select before data reaches the data lake's raw zone, retailers can filter out browsing logs that are associated with known malicious IP addresses. Discarding irrelevant or harmful data early reduces the volume of data that is written to and read from the data lake.

S3 Select: Faster data retrieval for IBM Storage Ceph: S3 Select is a powerful tool that optimizes data retrieval from IBM Storage Ceph storage. It uses two key techniques:

- ▶ **Predicate pushdown:** Instead of transferring all the data to the client and then filtering it, S3 Select pushes the filtering logic (predicates) to IBM Storage Ceph itself. IBM Storage Ceph identifies and transfers only the relevant data, which reduces the amount of data that is transferred and improves the overall performance.
- ▶ **SQL-like engine:** S3 Select leverages an SQL-like engine, enabling you to use familiar SQL syntax for filtering and selecting data directly within IBM Storage Ceph. This approach simplifies querying complex data sets and streamlines the data retrieval process.

Benefits:

- ▶ **Reduced network traffic:** By filtering data at the source, S3 Select minimizes the amount of data that is transferred, which leads to faster query run times.
- ▶ **Improved efficiency:** Using the IBM Storage Ceph processing power for filtering reduces the workload on the client, which improves overall system efficiency.
- ▶ **Simplified queries:** The SQL-like interface enables querying of complex data sets that are stored in IBM Storage Ceph.

The full Python script is available at this [GitHub repository](#). Example 7-64 provides a code snippet that highlights the usage of S3 Select.

Example 7-64 Code snippet

```
def s3_select_query(bucket_name, object_key, query, s3_client):
    logging.info(f"Executing S3 Select on Bucket: '{bucket_name}', Key:
    '{object_key}', Query: '{query}'")
    try:
        response = s3_client.select_object_content(
            Bucket=bucket_name,
            Key=object_key,
            ExpressionType='SQL',
            Expression=query,
            InputSerialization={'CSV': {"FileHeaderInfo": "USE"}},
            OutputSerialization={'CSV': {}},
        )
        result_data = ''
        for event in response['Payload']:
            if 'Records' in event:
                result_data += event['Records']['Payload'].decode('utf-8')
            elif 'Stats' in event:
                stats = event['Stats']['Details']
                logging.info(f"Statistics: {stats}")
            elif 'End' in event:
                logging.info("Reached end of the data stream.")
        return result_data
    except Exception as e:
        logging.error("Error during S3 Select: %s", str(e))
        return None

def trigger_processing():
    .....
    # Construct the S3 Select SQL expression based on CIDR range
```

```

query_parts = cidr_range.split('|')
condition = " OR ".join([f"ip LIKE '{part}'" for part in query_parts])
query = f"SELECT * FROM S3object WHERE NOT ({condition});"
logging.info(f"Executing S3 Select with query: {query}")
filtered_data = s3_select_query(source_bucket, object_key, query, s3_source)
.....

```

Using the S3 Select query **"SELECT * FROM S3object WHERE NOT (ip LIKE '20.%' OR ip LIKE '12.%');"**, we can filter out all entries in our data set that contain IP addresses on known malicious Classless Inter-Domain Routing (CIDR) network ranges. This query is a basic one for our example use case. You can perform advanced SQL queries with the help of the S3 Select feature that is available with IBM Storage Ceph.

If we check the logs for our running e-commerce example app, we can see S3 Select in action (Example 7-65).

Example 7-65 E-commerce example app log

```

* Serving Flask app 'process_ingest_to_raw_ecommerce'
* Debug mode: off
2024-05-08 10:25:13,844 - INFO - Script Version 1.2.0 - WARNING: This is a
development server. Do not use it in a production deployment. Use a production
WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8080
* Running on http://192.168.123.125:8080
2024-05-08 10:25:13,844 - INFO - Script Version 1.2.0 - Press CTRL+C to quit
2024-05-08 10:25:19,956 - INFO - Script Version 1.2.0 - Executing S3 Select on
Bucket: 'ecommingest', Key: 'browsing_data_20240507202409.csv', Query: 'SELECT *
FROM S3object WHERE NOT (ip LIKE '45.%' OR ip LIKE '210.%');'
2024-05-08 10:25:32,499 - INFO - Script Version 1.2.0 - Statistics:
{'BytesScanned': 14088969, 'BytesProcessed': 14088969, 'BytesReturned': 4677325}
2024-05-08 10:25:32,499 - INFO - Script Version 1.2.0 - Reached end of the data
stream.
2024-05-08 10:25:32,562 - INFO - Script Version 1.2.0 - Object tagged as
processed: browsing_data_20240507202409.csv

```

We can see that an S3 bucket notification triggered a new event on the pipeline, and a new browsing log file that is named "browsing_data_20240507202409.csv" arrived. Our Knative serverless application triggers on data arrival and initiates processing. One part of the processing script uses S3 Select, which runs an SQL query on the data set:

```
'SELECT * FROM S3object WHERE NOT (ip LIKE '45.%' OR ip LIKE '210.%')
```

Note: Crucially, the entire filtering process runs on the IBM Storage Ceph Object Gateways themselves and returns only the results to the client. This approach offloads processing from the client application and improves overall efficiency.

You can see this result on the next output line of the log:

```
"{'BytesScanned': 14088969, 'BytesProcessed': 14088969, 'BytesReturned':
4677325}"
```

It scanned 14,088,969 bytes, but after the filtering, it returned only 4,677,325 bytes to the application. For more information about the IBM Storage S3 Select feature, see [TPC-DS Benchmark using Trino with IBM Storage Ceph Object Storage S3 Select feature](#).

After processing, the data is stored in a raw zone bucket that is named `ecommrw` with the same configuration (server-side encryption (SSE) encryption, versioning, and others) as the buckets that are used for the physical branch stores that are described in 7.2, “Raw zone: Branch stores classification data pipeline” on page 92.

Two different data sets are stored: one for the e-commerce transactions (`ecommtans`), and the other for the e-commerce logs (`ecommlogs`).

- ▶ E-commerce logs: After data arrives in the raw zone, a Spark job takes over to perform essential cleaning and transformation tasks:
 - a. Data cleansing: The Spark job removes invalid or duplicate entries to ensure data quality and consistency. This step eliminates data that might lead to inaccurate results in downstream analytics.
 - b. Format conversion: The job efficiently converts the data from its original CSV format to Parquet format. Parquet is a columnar storage format for big data analytics. This conversion optimizes data storage efficiency and accelerates future processing within the data lake.
 - c. Transfer to staging zone: The cleansed and transformed data is transferred to the staging zone. This zone serves as a temporary storage area for further enrichment before the data is made available for analytics workloads.
- ▶ E-commerce transactions: This data set reaches the data lake, which is already cleansed and in Parquet format, so it gets stored in the staging zone.

In Chapter 8, “Transform: Staging and curated zones” on page 121, we enrich the e-commerce data set by joining browsing logs and transaction data sets in the staging zone. By combining these data sets, retailers gain a holistic view of customer behavior:

- ▶ Data-driven culture: Retailers can leverage these deep customer insights to foster a data-driven culture within the organization.
- ▶ Strategic decision making: Data-driven insights empower businesses to make informed strategic decisions.
- ▶ Enhanced competitive edge: By understanding customer behavior and trends, retailers can optimize their offerings and gain a competitive advantage.



Transform: Staging and curated zones

Chapter 7, “Ingest: Landing and raw zones” on page 83 described how the different retail sources (physical stores, e-commerce, and Hadoop (Surveys)) ingest data into the data lake.

This chapter continues to describe this topic by providing examples and detailed hands-on instructions about how you can transform and enrich your data sets so that they can provide higher value for your users at the consume layer.

This chapter has the following sections:

- ▶ Points of sale (physical stores)
- ▶ E-commerce

8.1 Points of sale (physical stores)

In-store customer purchases generate valuable data, including potentially confidential information like email addresses, customer numbers, and invoice numbers. To leverage this data for analytics while protecting privacy, this data must be split into two different types of data, such as anonymized and confidential.

Anonymized data covers non-confidential data, such as the items that are sold from a specific store and the item details, such as their type, category, payment type, and other details.

To analyze the different categories of data, create separate buckets for anonymized and confidential information with separate access policies so that only the approved users can work on their respective buckets.

8.1.1 Assigning bucket policies and object tags for effective S3 access control

This section dives into two essential mechanisms for managing access and data classification within Amazon Simple Storage Service (S3) buckets: bucket policies and object tags. This section explores how to assign bucket policies to control user access and leverage object tags to categorize your data effectively.

To illustrate these concepts, we create two example buckets:

- Confidential bucket: This bucket stores user-related confidential data, which requires strict access controls.
- Anonymized bucket: This bucket stores anonymized, non-confidential customer data. We use object tags to further classify this data within the bucket.

When we establish these buckets, we delve in to the commands for creating and assigning bucket policies and object tags.

To create the buckets, use the commands that are shown in Example 8-1.

Example 8-1 Creating the buckets

```
aws --profile confidential --endpoint https://s3.cephlabs.com s3api create-bucket  
--bucket confidential
```

```
aws --profile anon --endpoint https://s3.cephlabs.com s3api create-bucket --bucket  
anonymized
```

After you create the buckets, assign bucket policies to these buckets to limit access to them. An example bucket policy is shown in Example 8-2.

Example 8-2 Assigning a bucket policy to the confidential bucket

```
[root@ceph01 ~]# cat conf_bucket_policy.json  
{  
  "Version": "2012-10-17",  
  "Statement": [{  
    "Effect": "Allow",  
    "Principal": {"AWS":  
      ["arn:aws:iam:::user/confidential","arn:aws:iam:::user/shoppingest"]},  
    "Action": "s3:GetObject",  
    "Resource": [  
      "arn:aws:s3:::confidential/*"
```



```
    ]  
  }]  
}
```

Tip: You can assign policies for multiple users at once. For example, because the `shopingest` user also needs access, you grant permission to that user.

With this policy, only the `confidential` user may retrieve objects from the `confidential` bucket. Because you use another bucket to store anonymized user data, you must assign a bucket policy to it to ensure consistent access control across all the data storage buckets (Example 8-3).

Example 8-3 Assigning a bucket policy to the anonymized bucket

```
root@ceph01 ~]# cat anon_bucket_policy.json  
{  
  "Version": "2012-10-17",  
  "Statement": [{  
    "Effect": "Allow",  
    "Principal": {"AWS":  
["arn:aws:iam::user/anon", "arn:aws:iam::user/shopingest"]},  
    "Action": "s3:GetObject",  
    "Resource": [  
      "arn:aws:s3::anonymized/*"  
    ]  
  }]  
}
```

After you create the buckets and assign policies, add them to watsonx.data for ingesting the data inside those buckets. To do so, complete the following steps:

1. Go to the Infrastructure manager dashboard of watsonx data and select **Add component** → **Add bucket** (Figure 8-1).

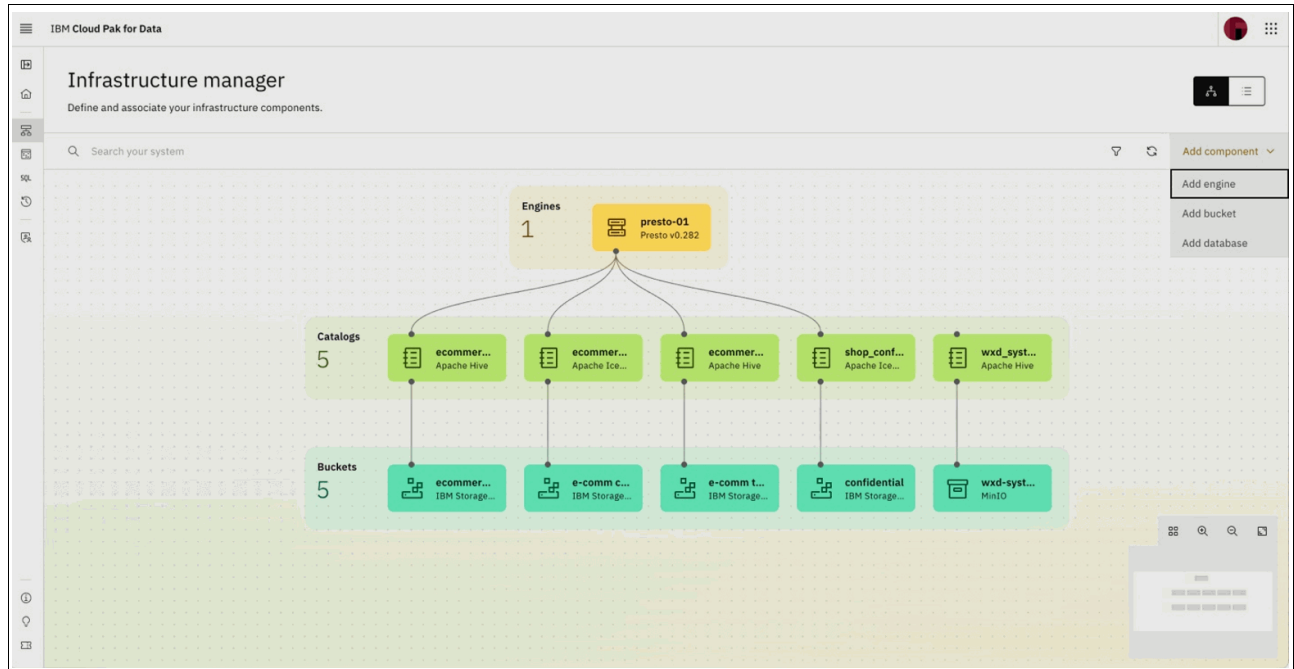


Figure 8-1 Add bucket option

2. Select **IBM Storage Ceph** as the Bucket type and enter the details for accessing your IBM Storage Ceph bucket. An example is shown in Figure 8-2 on page 125.

Add bucket

Register existing, externally managed object storage.

Bucket details

Bucket type

IBM Storage Ceph

Bucket name

confidential

Display name

confidential

Endpoint

https://s3.cephlabs.com

Access key

.....

Secret key

.....

Connection status

Successful

Retest

Associated catalog ⓘ

Catalog type

Apache Iceberg

Catalog name

shop_confidential

Cancel

Register

Figure 8-2 Adding a bucket

- When the connection test is successful, click **Register** and add the bucket to your `watsonx.data` environment. The Access key and Secret key values belong to the `confidential` user on IBM Storage Ceph.

4. After you add the buckets, integrate them with the Presto engine running on watsonx.data to ingest and consume the data. To do this task, hover your cursor over the bucket that you want to add and click **Manage associations** (Figure 8-3).



Figure 8-3 Manage associations

5. From the options menu, select **Presto**. The Presto engine restarts. This process might take a few moments (Figure 8-4 on page 127).

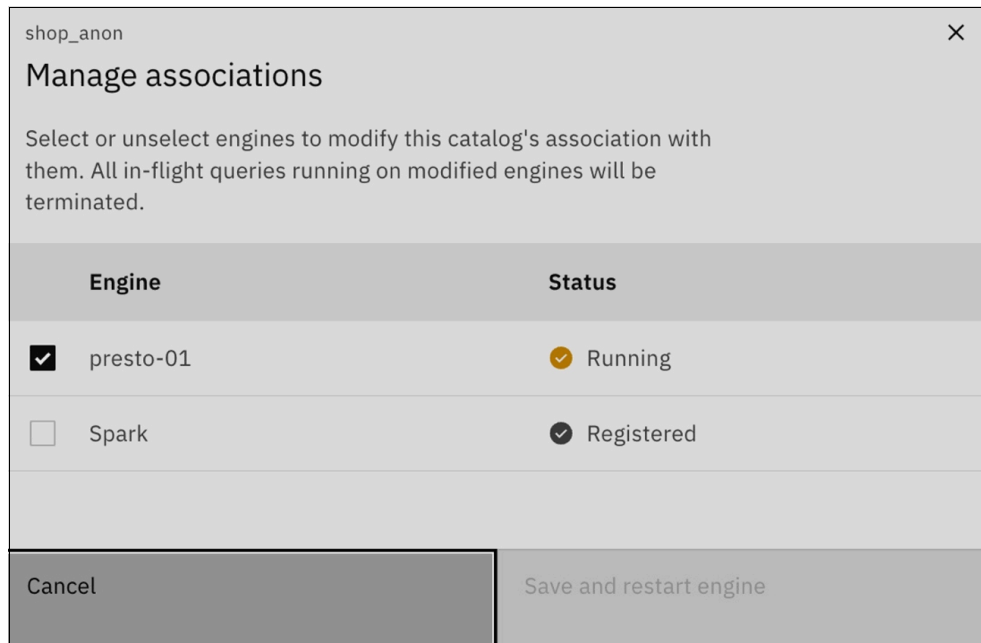
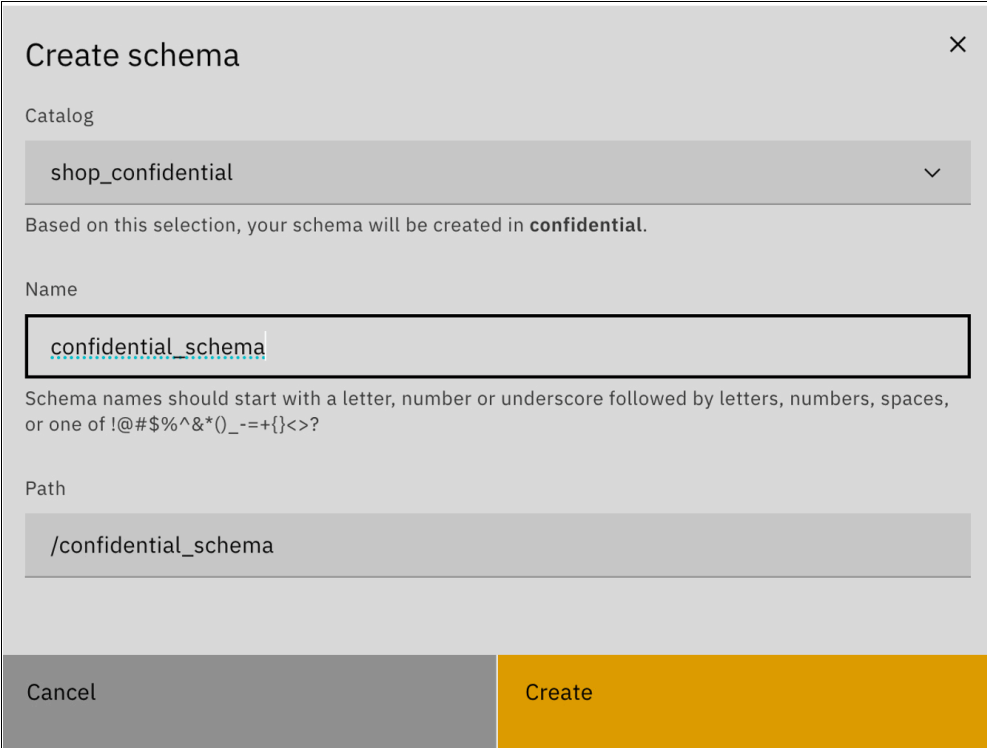


Figure 8-4 Presto engine restarts

6. After you integrate the confidential bucket, integrate the anonymized bucket by using the `watsonx.data` CLI. To do this task, you must install the [watsonx.data command-line tool](#).
7. After you complete the integration, create schemas and tables under the catalogs that were created after integrating the buckets with Presto. To create a schema, go to **Data manager** in the left menu and select **Create schema** → **Create**.

8. Enter the details for the schema that you want to create and click **Create** (Figure 8-5).

A dialog box titled "Create schema" with a close button (X) in the top right corner. It contains three main sections: "Catalog" with a dropdown menu showing "shop_confidential"; a text line stating "Based on this selection, your schema will be created in **confidential**."; "Name" with a text input field containing "confidential_schema" and a validation message below it: "Schema names should start with a letter, number or underscore followed by letters, numbers, spaces, or one of !@#\$%^&*()_-=+{}<>?"; and "Path" with a text input field containing "/confidential_schema". At the bottom are two buttons: "Cancel" on the left and "Create" on the right.

Create schema

Catalog

shop_confidential

Based on this selection, your schema will be created in **confidential**.

Name

confidential_schema

Schema names should start with a letter, number or underscore followed by letters, numbers, spaces, or one of !@#\$%^&*()_-=+{}<>?

Path

/confidential_schema

Cancel Create

Figure 8-5 Create schema

9. Create tables under the schemas that were created in the previous steps. To do this task, go to your schemas and select **Create table from** in the three-dot menu for your schema (Figure 8-6).

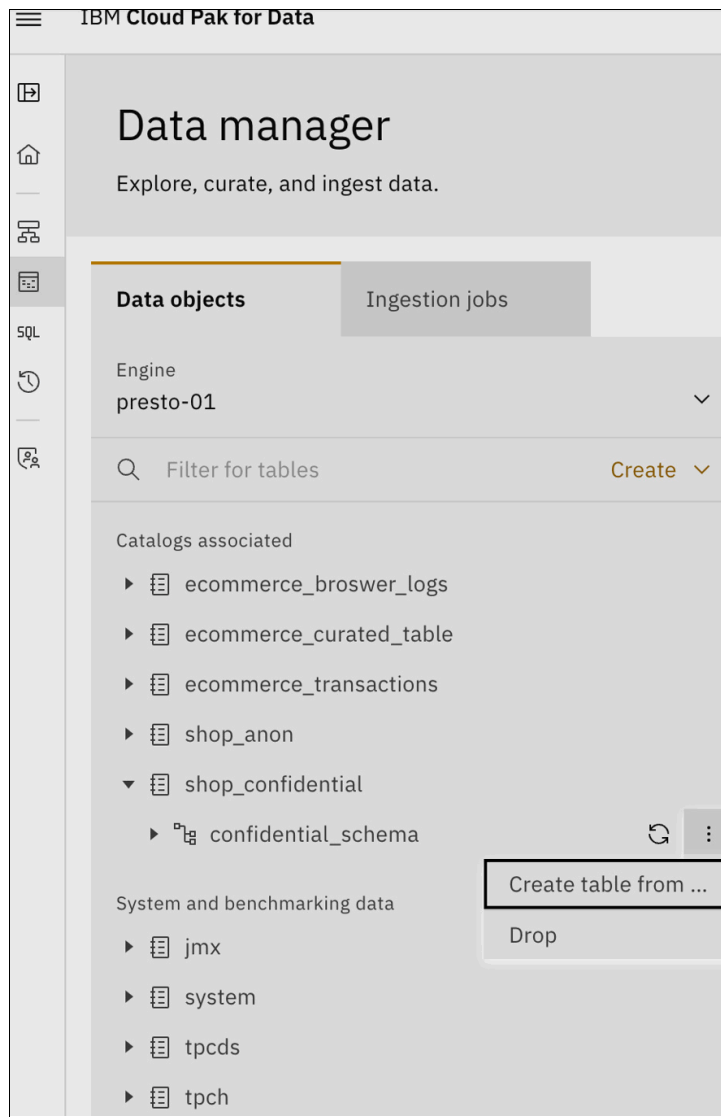


Figure 8-6 Create table from

10. Upload a sample file from your computer to watsonx.data to create the table layout and click **Next** to proceed with the table you have uploaded (Figure 8-7).

Create table from file
Generate, configure, and run DDL.

Source
Upload a file to define your new table's initial columns and rows.

Upload file
Supported file types: .csv, .parquet, .json, .txt
File encodings supported: UTF-8
Max. file size: 2 MB

Drag and drop a file or click to upload

Text / CSV file configuration
These configurations will be applied to .csv and .txt files.

Encoding: UTF-8 | Escape character: \ | Field delimiter: , | Line delimiter: \n

☒ Has header

Table schema configuration
Confirm the data types to apply to each column.

shop1_28_03_2024.csv

shop1	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	Price	CustomerID	Country	SSN	Email
varchar	int	int	varchar	int	varchar	real	int	varchar	varchar	varchar
shop1	78544	31779	Charger	7	28-03-2024 22:01	73.73	41391	Spain	317-89-6263	larrytaylor@exa
shop1	31812	94564	Tablet	19	28-03-2024 06:40	41.27	73310	Spain	491-87-3274	erikmendoza@e
shop1	22904	11002	Sandals	16	28-03-2024 22:06	89.74	18879	Spain	525-55-1565	richard14@exa

Cancel Back Next

Figure 8-7 Uploading a sample file from your computer to watsonx.data

11. Select your target schema and give your table a name. Click **Next** to continue to the summary window and table creation (Figure 8-8).

Summary
Ensure all details are correct before creating your table.

Source

Name	Value
File	shop1_28_03_2024.csv

Target

Name	Value
Catalog	shop_confidential
Schema	confidential_schema
Table name	shop_confidential_table
Table format	Apache Iceberg
Data format	Parquet

DDL preview
Review the DDL that will be run to create your table.

```
1 - CREATE TABLE "shop_confidential"."confidential_schema"."shop_confidential_table" (
2   "shop1" VARCHAR,
3   "InvoiceNo" INT,
4   "StockCode" INT,
5   "Description" VARCHAR,
6   "Quantity" INT,
7   "InvoiceDate" VARCHAR,
8   "Price" REAL,
9   "CustomerID" INT,
10  "Country" VARCHAR,
11  "SSN" VARCHAR,
12  "Email" VARCHAR,
13  "Description" VARCHAR
```

Cancel Back Create

Figure 8-8 Summary window

12. Your table will be ready in a while and you will get a notification when it is ready. You can review the table and ingest more data by clicking **Ingest data** on your table (Figure 8-9 on page 131).


<div> <div>Name</div> <div>shop_confidential_table</div> </div> <div> <div>Catalog: shop_confidential Schema: confidential_schema</div> </div>				
<div> <div>Table schema</div> <div>Time travel</div> <div>Data sample</div> <div>DDL</div> </div>				
<div> <div>Q Search for columns</div> <div> <div> </div> <div>+</div> <div>Ingest data</div> <div>↺</div> </div> </div>				
	Name	↕	Data type	Nullable
<input type="checkbox"/>	shop1		varchar	YES
<input type="checkbox"/>	InvoiceNo		integer	YES
<input type="checkbox"/>	StockCode		integer	YES
<input type="checkbox"/>	Description		varchar	YES
<input type="checkbox"/>	Quantity		integer	YES
<input type="checkbox"/>	InvoiceDate		varchar	YES
<input type="checkbox"/>	Price		real	YES
<input type="checkbox"/>	CustomerID		integer	YES

Figure 8-9 Ingest data

13. You can also review the uploaded data from the **Data sample** tab on the same window. Clicking **Ingest data** opens a dialog where you can choose your preferred ingestion mode. Select your ingest mode and click **Next**. In this example, we select **Iceberg copy loader** (Figure 8-10).

Mode


Specify how you'd like to ingest data.



Iceberg copy loader

Leverage watsonx.data internal serverless technology to ingest .csv or .parquet files.

Supported target table format: **Apache Iceberg**



IBM Analytics Engine (Spark)

Configure and run an ingestion job on your registered IBM Analytics Engine (Spark).

Supported target table format: **Apache Iceberg**

Figure 8-10 Selecting Iceberg copy loader

14. Select the files within the confidential bucket that you want (Figure 8-11).

Note: We select the files in the confidential bucket to use analytics capabilities for user-confidential files. If you want to expand your analysis with other files in other buckets, you can select from other buckets too.

Figure 8-11 Selecting the files within the confidential bucket

15. Click **Next** after selecting the files to use and specify your target for the ingestion. You can use the same table or create one during this step (Figure 8-12).

Figure 8-12 Specifying your target for the ingestion

16. You see the Summary window, where you review the data. Click **Ingest** to continue with data ingestion (Figure 8-13 on page 133).

Ingest data

Select files, configure, and ingest.

Mode

Source

Target

Summary

Summary

Ensure all details are correct before ingesting your files.

Source

Name	Value
Job id	ingestion-1715245243736
Bucket	confidential
Folder and files selected	4
Encoding	UTF-8
Escape character	\\
Field delimiter	,
Line delimiter	\n
Has header	true

Cancel

Back

Ingest

Figure 8-13 Summary window

17. You are notified when data ingestion is complete. You can review the status by clicking the **Ingestion jobs** tab on the Data Manager window.

18. You can query your data tables with the SQL option on the left menu of watsonx.data. You can see a simple SQL query and the result set as an example in Figure 8-14.

Query workspace

Build and run queries against your data.

Engine

presto-01

Untitled 1

+

Filter for tables

ecommerce_transactions

shop_anon

anon_schema 0

shop_confidential

confidential_schema 1

shop2_26_03_2024

shop2 varchar

InvoiceNo integer

StockCode integer

Description varchar

Quantity integer

```

1 SELECT
2   "InvoiceNo"
3 FROM
4   "shop_confidential"."confidential_schema"."shop2_26_03_2024"
5 LIMIT
6   10;

```

Worksheet results 3

Clear all

^ SELECT "InvoiceNo" FROM "shop_confidential"."confidential_schema"."shop2_26_03_2024" LIMIT 10

Run time: 1.748s

Result set

Details

InvoiceNo

96854

48767

31396

40070

99567

20030

No saved worksheets yet.

Return here to reopen and manage saved worksheets.

Figure 8-14 Query workspace

8.2 E-commerce

E-commerce platforms are the backbone of modern retail, and they generate massive amounts of data that can be valuable. A key source of this data is browsing logs, which capture user interactions on the website. These browsing logs are a window into customer behavior. They typically include details like the following ones:

- ▶ User sessions (how long users stay on the site).
- ▶ Page views (which pages users visit).
- ▶ Clicks (what elements users interact with).
- ▶ Time that is spent on specific pages.
- ▶ Shopping cart activity (items that are added and removed).
- ▶ Other user actions (searches, filters used, and so forth).

This rich data is generated from various sources:

- ▶ Web servers: Track overall website traffic and user requests.
- ▶ Application servers: Capture activity within the e-commerce platform itself.
- ▶ Client-side tracking (JavaScript): Monitor user interactions directly in the user's browser.

Figure 8-15 shows the data pipeline architecture for e-commerce.

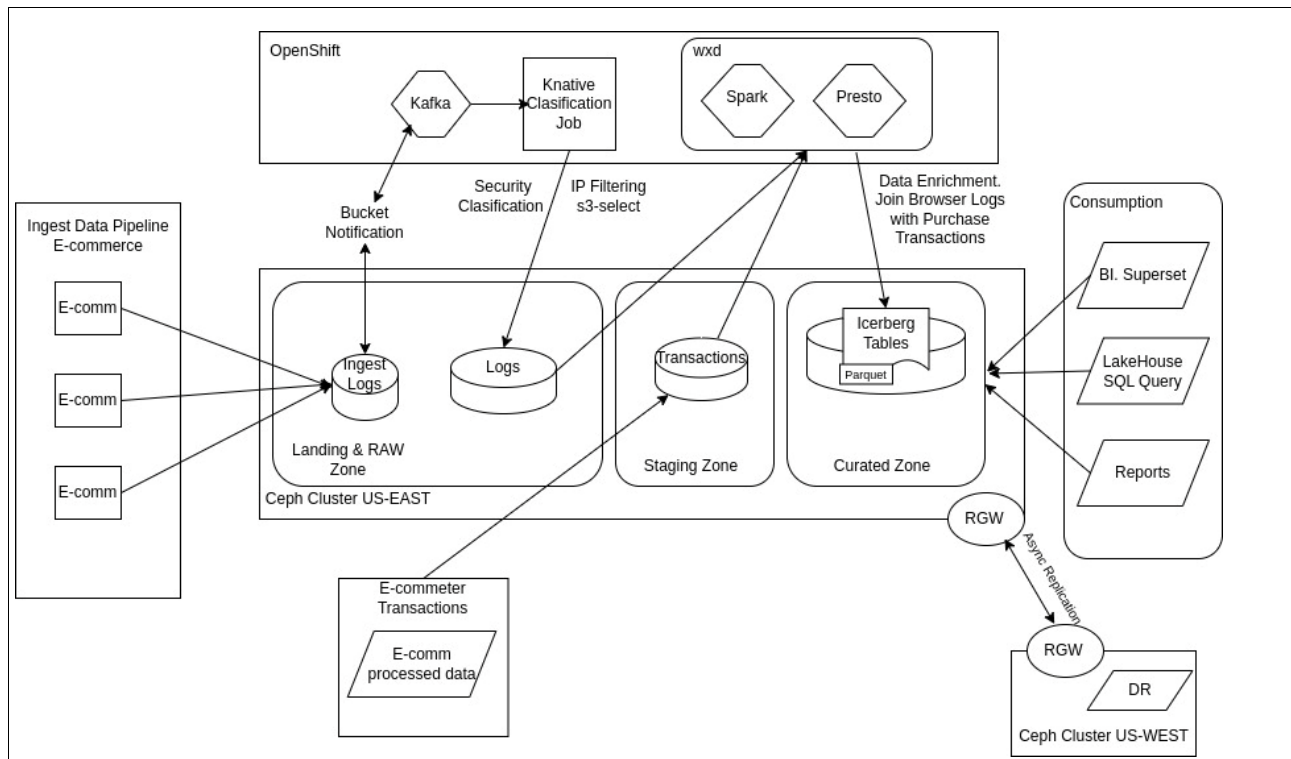


Figure 8-15 Data pipeline architecture for e-commerce

8.2.1 E-commerce: Ingest

To understand the upcoming data processing steps, let us follow the path of our example e-commerce data before it lands in the raw zone. We examine two example data sets: user browsing logs and customer online purchase transactions:

- *Browser logs* from various web servers are sent to the data lake and ingested as CSV files within a landing zone bucket. This process triggers a processing event that scans all customer IP addresses within the logs. Any IP addresses that are identified as malicious based on predefined IP address ranges are filtered out and removed from the data set.

When processed, the logs are stored on a raw zone bucket that is called `ecommlogs`.

- *Transaction logs* reach the data lake processed and converted into Parquet format. For optimized storage and retrieval, we store them in a dedicated bucket that is named `ecomtrans`.

For more information about the ingest e-commerce pipeline, see 8.2, “E-commerce” on page 134.

8.2.2 E-commerce: Browser log workflow

When you store CSV browser logs in the `ecommlogs` bucket, a Spark cleansing job automatically runs. This job removes duplicates or corrects invalid data; transforms the data set into a Parquet format; and stores the resulting Parquet objects in a date-partitioned structure. This format ensures consumption by any SQL engine, like Presto (`watsonx.data`).

Figure 8-16 shows the configured high-level workflow.

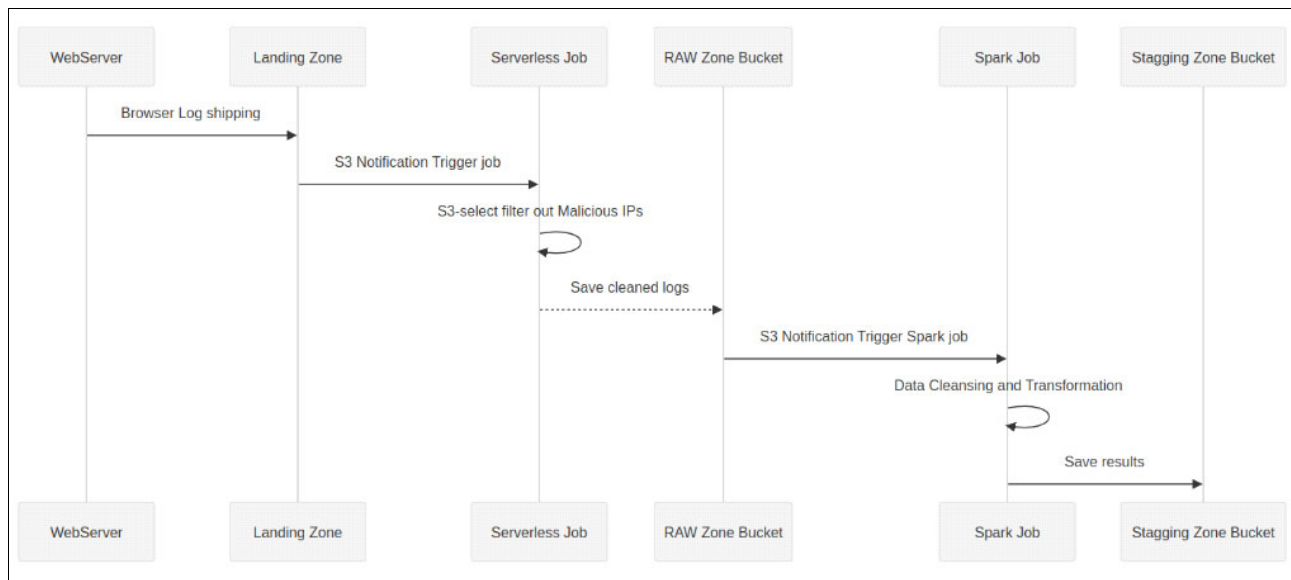


Figure 8-16 E-commerce ingest high-level workflow

8.2.3 E-commerce: Transactions workflow

Unlike browser logs, e-commerce transaction data arrives in the data lake and is processed and stored in Parquet format within the staging zone's `ecommttrans` bucket. This streamlined approach is possible because of the following features:

- ▶ External processing: The processing of transaction logs occurs outside the data lake pipelines. This approach might involve a separate system that is dedicated to transaction data management.
- ▶ Pre-optimized format: By arriving in Parquet format, which is a columnar storage format that is optimized for data lakes, the data is ready for efficient storage and retrieval within the data lake. This approach eliminates the need for further processing on landing in the staging zone.

8.2.4 E-commerce: Initial data cleansing

There is a configured bucket that is called `ecommlogs` that is part of the raw zone. Data arrives in the `ecommlogs` bucket, which is the designated landing zone within the raw zone for our e-commerce browser logs. Example 8-4 shows a sample of the data set.

Example 8-4 Sample data set

```
ip,ts,tz,verb,resource_type,resource_fk,response,browser,os,customer,d_day_name,i_
current_price,i_category,i_description,c_preferred_cust_flag,ds
51.167.165.55,2024-02-24
03:14:39,UTC,GET,Item,06e4ad90-9ac0-4d4d-a91c-6e3a75b19629,200,Safari,Linux,345187
,Saturday,380.33,Accessories,Scarf,False,2024-02-24
71.243.146.206,2024-01-29
20:38:33,UTC,GET,Item,cbbcbace-6a26-41c1-b0b0-41fc383583b1,200,Edge,iOS,169502,Mon
day,416.29,Jewelry,Tiara,True,2024-01-29
```

Whenever a new CSV file is uploaded to the `ecommlogs` bucket, an automated notification process is triggered. Here is how it works:

1. S3 bucket notification: The upload triggers an S3 bucket notification. This notification acts as an alert mechanism that something changed within the bucket.
2. Kafka topic delivery: The S3 bucket notification is delivered to a designated Kafka topic.
3. Spark job activation: The message within the Kafka topic triggers the running of a Spark job.
4. CSV object processing: The Spark job receives the name of the uploaded CSV object as an argument from the Kafka message. This argument enables the Spark job to identify and process the specific new data that must be handled.

The example Spark job code is available at [GitHub](#). The Spark Job uses the Amazon S3A File System (S3A) connector to access the buckets that contain the data sets. We use [S3A Identity and Access Management \(IAM\) AssumeRole](#) integration to access the source and destination buckets. For this example, we use `AssumeRole`. At the time of writing, Hadoop/S3A lacks built-in support for `AssumeRoleWithWebIdentity` authentication. However, this function can be achieved by implementing a [Custom Credentials Provider](#).

To overcome this limitation, the IBM Storage Ceph Object Engineering team developed a Custom Hadoop Credential provider that uses the `AssumeRoleWithWebIdentity` authentication method so that it can be used from any product that uses the S3A connector like Spark, Hadoop, Dremio, and others. The code for the Custom Credentials Provider is at [GitHub](#).

To proceed with the initial data cleansing, complete the following steps:

1. Create an IAM role with a policy that enables the Spark job to access the bucket in the raw zone that is called `ecommraw`. Store the processed objects on a bucket that is called `ecommlogs`.

If it does not exist, create a bucket that is named `ecommlogs` within the raw zone. This bucket is owned by the `ecommadmin` RADOS Gateway (RGW) user. The `ecommadmin` user is our S3 IBM Storage Ceph Object administrator for the e-commerce raw and staging zones.

Because the `ecommadmin` user is managing the creation and maintenance of the IAM roles, we grant the user the roles capability so that they can manage the required IAM roles (Example 8-5).

Example 8-5 Granting the user the roles capability

```
$ radosgw-admin user create --uid ecommadmin --display-name 'ecommadmin'
$ radosgw-admin caps add --uid="ecommadmin" --caps="roles=*"
$ aws --profile ecommadmin s3 mb s3://ecommlogs
make_bucket: ecommlogs
```

2. Before proceeding, confirm the existence of the `ecommraw` bucket within the raw zone (Example 8-6). This bucket serves as the landing zone for our browser log data.

As described in 8.2.4, “E-commerce: Initial data cleansing” on page 136, CSV files containing browsing activity are delivered to this bucket from the web server's HTTP logs. These logs capture all HTTP requests that are made by users on our website. This data arrives through an event-driven data pipeline, as described in 8.2.4, “E-commerce: Initial data cleansing” on page 136.

Example 8-6 Confirming the existence of the ecommraw bucket within the raw zone

```
$ aws --profile ecommadmin s3 ls s3://ecommraw
2024-04-25 09:05:04      1566462 browsing_data_20240424183838.csv
```

3. We use the IAM AssumeRole call to access a specific IAM role that grants the required access to the `ecommraw` and `ecommlogs` buckets. The assumed role call checks whether a local user from the RGW user list can assume a specific role. The first step for this setup is creating a local RGW user that we use to assume the IAM role and gain access to the S3 buckets through a set of temporary secure token service credentials (Example 8-7).

Example 8-7 Creating a local RGW user

```
# radosgw-admin user create --uid wxdecomm --display-name 'wxdecomm'
```

The IAM role that we assume grants the necessary permissions for our Spark job to process e-commerce data efficiently. Here is a breakdown of the specific permissions:

- Read access to the `ecommraw` bucket: Enables the Spark job to retrieve data files from the raw zone for processing.
- Write access to the `ecommlogs` bucket: Enables the Spark job to write the processed data files to the appropriate location within the staging zone.

With these permissions in place, the Spark job can seamlessly perform the following activities:

- Access the raw browsing log data that is stored in the `ecommmraw` bucket.
 - Process the data as needed (for example, cleaning or transforming).
 - Write the processed data to the designated `ecommlogs` bucket within the staging zone for further use.
4. Our JavaScript Object Notation (JSON) IAM role doc policy specified who is allowed to access and assume our new “spark-ecomm” role. In the policy, we allow g the local rgw user “wxdecomm” to assume the role (Example 8-8).

Example 8-8 JSON IAM role doc policy

```
# cat iam_role_ecommerce_spark.json
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::user/wxdecomm"
      },
      "Action": "sts:AssumeRole"
    }
  ]

# radosgw-admin role create --role-name spark-ecomm --assume-role-policy-doc=$(jq
-rc . /root/iam_role_ecommerce_spark.json)
  "RoleId": "5c0cb7e5-5595-4265-80c0-8b1c7ff53a9f",
  "RoleName": "spark-ecomm",
  "Path": "/",
  "Arn": "arn:aws:iam::role/spark-ecomm",
  "CreateDate": "2024-04-22T15:06:13.430Z",
  "MaxSessionDuration": 3600,
  "AssumeRolePolicyDocument":
  "{ \"Version\": \"2012-10-17\", \"Statement\": [{ \"Effect\": \"Allow\", \"Principal\": { \"AWS\": \"arn:aws:iam::user/wxdecomm\" }, \"Action\": \"sts:AssumeRole\" } ] }"
```

5. Although the `radosgw-admin` CLI offers IAM role management, the S3 API is another option. For example, Example 8-9 demonstrates listing recently created roles by using the S3 API.

Example 8-9 Listing the recently created role by using the S3 API

```
$ aws --profile ecommadmin iam get-role --role-name spark-ecomm
  "Role": {
    "Path": "/",
    "RoleName": "spark-ecomm",
    "RoleId": "5c0cb7e5-5595-4265-80c0-8b1c7ff53a9f",
    "Arn": "arn:aws:iam::role/spark-ecomm",
    "CreateDate": "2024-04-22T15:06:13.430000+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::user/wxdecomm"
          }
        }
      ]
    }
  }
```



```

        },
        "Action": "sts:AssumeRole"
    }
]
},
"MaxSessionDuration": 3600
}

```

6. When the IAM role is successfully created, attach the role policies that allow or restrict access for this role to specific S3 resources.

An IAM role can be assigned multiple policies to grant specific permissions. In this case, we use two separate policies for different buckets:

- First role policy: This policy grants the role access to perform **GET** and **LIST** operations on objects within the `ecommrw` bucket (Example 8-10).
- Second role policy (Optional): Also, a separate policy can be created to allow the role to put object tags on the processed objects in the `ecommrw` bucket. These tags help identify processed objects and prevent redundant processing.

Example 8-10 First role policy

```

# cat iam_policy_ecommerce_spark_read_from_ecommrw.json
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:PutObjectTagging",
      "s3:Get*",
      "s3:ListBucket",
      "s3:ListBucketMultipartUploads"
    ],
    "Resource": [
      "arn:aws:s3:::ecommrw/*",
      "arn:aws:s3:::ecommrw",
      "arn:aws:s3:::ecommrw/"
    ]
  }
]
# radosgw-admin role policy put --role-name=spark-ecomm
--policy-name=read-from_ecommrw --policy-doc=$(jq -rc .
/root/iam_policy_ecommerce_spark_read_from_ecommrw.json)
Permission policy attached successfully

```

7. Our second role policy grants the Spark job the required access to the `ecommlogs` tagging zone bucket (Example 8-11).

Example 8-11 Second role policy

```

# cat iam_policy_ecommerce_spark_write_to_ecommlogs.json
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:PutObjectTagging",

```

```

        "s3:Get*",
        "s3:Delete*",
        "s3:Put*",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:PutObjectTagging",
        "s3:AbortMultipartUpload"
    ],
    "Resource": [
        "arn:aws:s3:::ecommlogs/*",
        "arn:aws:s3:::ecommlogs",
        "arn:aws:s3:::ecommlogs/"
    ]
}
]
# radosgw-admin role policy put --role-name=spark-ecomm
--policy-name=write_to_ecommlogs --policy-doc=$(jq -rc .
/root/iam_policy_ecommerce_spark_write_to_ecommlogs.json)
Permission policy attached successfully

```

8. Now that our IAM role is ready, we can use the “wxdecomm” credentials (Access and Secret keys) to assume the “spark-ecomm” role and get a set of temporary credentials to access the S3 data set (Example 8-12).

Example 8-12 Assuming the “spark-ecomm” role

```

$ aws --profile=wxdecomm sts assume-role --role-arn arn:aws:iam::role/spark-ecomm
--role-session-name testecomuser | jq .
  "Credentials": {
    "AccessKeyId": "SnFcrc5Mch5C3tymA1W",
    "SecretAccessKey": "VSGPR3W1KESNIWPG1HYHLUFWFYJKM7AY9YYTO6B",
    "SessionToken":
"iyrTmKgTK0ezePM6QsYeCikCwrcAoJ27vHNaBA3w719KY1Lwv3KccK3GLwRIH0EJI+SInTYMFmhLGrkPA
PkeaiYs0mxv5KcLJK736LqDJcyQub3eEPBsuEoujpfUHYIgTs3Q1v4X7PwogdKy4jASiDJEzCoofJcfaTm
ls9mFljpZe5+iEydNjTtuELn1ve4MY59axk29Pz0UqGmB+Ijfhf6rtZQ3632AsSP7uJsyhVY3VbKypQhe
R29lznK7R16gWTBhsKSbJ4BtC1WuXrgtK/BchgSP28z76j+2a+5ihpjXai5x4L6atsskVlhGxghgYwUMNo
Wu15QEdSmXR2wvA=",
    "Expiration": "2024-04-25T12:44:31.236872+00:00"
  },
  "AssumedRoleUser": {
    "Arn": "arn:aws:sts::assumed-role/spark-ecomm/testecomuser"
  },
  "PackedPolicySize": 0
$ aws --profile=wxdecomm s3 ls s3://ecommlogs
PRE browsing/

```

9. Example 8-13 shows a snippet of the Spark job S3A configuration that we have in our script. As you can see, we are using environment variables to provide the required input for the S3A configuration options.

Example 8-13 Snippet of the Spark job S3A configuration

```

spark = SparkSession.builder \
    .appName("Data Processing with IAM Role Assumption") \
    .master(spark_master_url) \
    .config("spark.driver.host", os.getenv('SPARK_DRIVER_HOST')) \
    .config("spark.hadoop.fs.s3a.endpoint", os.getenv('S3_ENDPOINT')) \

```

```

        .config("spark.hadoop.fs.s3a.aws.credentials.provider",
"org.apache.hadoop.fs.s3a.auth.AssumedRoleCredentialProvider") \
        .config("spark.hadoop.fs.s3a.access.key", os.getenv('AWS_ACCESS_KEY_ID')) \
        .config("spark.hadoop.fs.s3a.secret.key",
os.getenv('AWS_SECRET_ACCESS_KEY')) \
        .config("spark.hadoop.fs.s3a.assumed.role.arn",
os.getenv('SOURCE_ROLE_ARN')) \
        .config("spark.hadoop.fs.s3a.assumed.role.sts.endpoint",
os.getenv('STS_ENDPOINT')) \
        .config("spark.hadoop.fs.s3a.assumed.role.sts.endpoint.region",
os.getenv('STS_REGION')) \
        .config("spark.hadoop.fs.s3a.assumed.role.session.duration",
os.getenv('SESSION_DURATION')) \
        .config("spark.hadoop.fs.s3a.assumed.role.session.name", "sparkSession") \
        .config("spark.hadoop.fs.s3a.assumed.role.credentials.provider",
"org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider") \
        .config("spark.hadoop.fs.s3a.impl",
"org.apache.hadoop.fs.s3a.S3AFileSystem") \
        .config("spark.hadoop.fs.s3a.path.style.access", True) \
        .getOrCreate()

```

10. We use environment variables to configure our Python script. For illustration purposes, we set `S3_OBJECT_KEY` to a test value. However, when the full pipeline is triggered, the key names are dynamically obtained from the Kafka event payload. The IP ENV is for our deployed Spark Master.

To securely access S3 buckets within the data processing pipeline, we leverage IAM roles and temporary credentials. We configure the following environment variables:

- Source and destination buckets: The names of the S3 buckets that are involved (for example, `ecommraw` and `ecommlogs`).
- IAM role: The role to assume for accessing the buckets (“`spark-ecomm`”).
- RGW Security Token Service (STS) endpoint: The endpoint that is used to obtain temporary credentials.
- STS session duration: The lifespan of the temporary credentials that are retrieved for accessing the buckets.

By assuming the “`spark-ecomm`” role, the Spark job acquires temporary credentials with a limited validity period (session duration). These credentials can be used only to access the specified S3 buckets within that time frame (Example 8-14).

Example 8-14 Leveraging a combination of IAM roles and temporary credentials

```

export DESTINATION_ROLE_ARN='arn:aws:iam::role/spark-ecomm'
export S3_ENDPOINT='https://s3.cephlabs.com'
export STS_ENDPOINT='https://s3.cephlabs.com'
export SPARK_MASTER_URL=spark://10.88.0.6:7077
export SOURCE_BUCKET=ecommraw
export DESTINATION_BUCKET=ecommlogs
export SOURCE_ROLE_ARN='arn:aws:iam::role/spark-ecomm'
export AWS_ACCESS_KEY_ID='TV9FXNXXTABN7SSMG56M'
export AWS_SECRET_ACCESS_KEY='BOUQ6XXXXMDc4xDQ4hxtzrz7oDFtg7BuGtIdt6V'
export STS_REGION='default'
export SESSION_DURATION='3600'
export S3_OBJECT_KEY='browsing_data_20240421012010.csv'

```

```
export SPARK_DRIVER_HOST=10.88.0.6
export AWS_REGION='default'
```

11. With everything in place, we can run our Spark job and start our script. The script performs data cleansing, converts the data to Parquet format, and optimizes the S3 object layout for efficient querying by partitioning based on log dates (Example 8-15).

Example 8-15 Running the Spark job

```
$ spark-submit spark_data_cleansing_from_raw_ecommerce.py
24/04/25 14:34:13 INFO SparkContext: Running Spark version 3.5.1
24/04/25 14:34:13 INFO SparkContext: OS info Linux, 5.14.0-383.el9.x86_64, amd64
24/04/25 14:34:13 INFO SparkContext: Java version 17.0.10
24/04/25 14:34:13 INFO ResourceUtils:
=====
24/04/25 14:34:28 INFO ShufflePartitionsUtil: For shuffle(0), advisory target
size: 67108864, actual target size 1048576, minimum partition size: 1048576
24/04/25 14:34:29 INFO ParquetUtils: Using default output committer for Parquet:
org.apache.parquet.hadoop.ParquetOutputCommitter
24/04/25 14:34:29 INFO SparkContext: Starting job: parquet at
NativeMethodAccessorImpl.java:0
24/04/25 14:34:29 INFO DAGScheduler: Got job 1 (parquet at
NativeMethodAccessorImpl.java:0) with 1 output partitions
24/04/25 14:34:29 INFO DAGScheduler: Final stage: ResultStage 2 (parquet at
NativeMethodAccessorImpl.java:0)
24/04/25 14:34:29 INFO DAGScheduler: Parents of final stage: List(ShuffleMapStage
1)
24/04/25 14:34:47 INFO DAGScheduler: Job 1 finished: parquet at
NativeMethodAccessorImpl.java:0, took 17.416162 s
24/04/25 14:34:47 INFO FileFormatWriter: Start to commit write Job
2380ab6d-9ba6-4d40-8802-427873771e14.
224/04/25 14:35:09 INFO FileFormatWriter: Finished processing stats for write job
2380ab6d-9ba6-4d40-8802-427873771e14.
24/04/25 14:35:09 INFO SparkContext: SparkContext is stopping with exitCode 0.
24/04/25 14:35:09 INFO StandaloneSchedulerBackend: Shutting down all executors
24/04/25 14:35:09 INFO SparkContext: Successfully stopped SparkContext
24/04/25 14:35:10 INFO ShutdownHookManager: Shutdown hook called
24/04/25 14:35:10 INFO MetricsSystemImpl: s3a-file-system metrics system shutdown
complete.
```

12. After the Spark job completes, we can perform some data verification checks. One approach is to verify whether a specific tag is applied to the processed objects within the ecommraw bucket. This tag can serve as an indicator that the data was successfully processed by the Spark job (Example 8-16).

Example 8-16 Data verification checks

```
$ aws --profile ecommadmin s3api get-object-tagging --bucket ecommraw --key
browsing_data_20240424183838.csv | jq .
"TagSet": [
  {
    "Key": "processed",
    "Value": "true"
  }
]
```

13. We can also check the staging zone `ecommlogs` bucket, where the data is saved in Parquet format with the partitioning “by-date” scheme in place (Example 8-17).

Example 8-17 Data verification checks

```
$aws--profileecommadmins3lss3://ecommlogs/browsing/
main
PRE ds=2024-01-01/
PRE ds=2024-01-02/
PRE ds=2024-01-03/
PRE ds=2024-01-04/
PRE ds=2024-01-05/
PRE ds=2024-01-06/
PRE ds=2024-01-07/
PRE ds=__HIVE_DEFAULT_PARTITION__/_
2024-04-25 16:35:09      0 _SUCCESS
```

14. Our e-commerce browser logs are in the `ecommlogs` bucket in Parquet format and partitioned by date. So, we finished processing the browser logs in the staging zone. We also have the e-commerce transaction data set in Parquet format in the `ecommtans` bucket. This transaction data set is stored in the staging zone by a different system that is not part of our data analytical workloads (Example 8-18).

Example 8-18 Transaction data set

```
$aws--profileecommadmins3lss3://ecommtans/transactions/
main
2024-04-20 20:21:40      794937 transaction_data_20240420201736.parquet
```

15. We enrich the data by running SQL queries that join and correlate the transaction data set with the browser log data set. The resulting JOIN queries create views or new tables in the curated zone, which provide high-quality data for final users to consume.

Our data processing pipeline involves two key steps:

- Staging zone setup: We configure `watsonx.data` to recognize the staging zone buckets `ecommlogs` and `ecommtans`.
- Catalog and Presto integration: We create a catalog within `watsonx.data` and connect it to the Presto engine so that we can run queries on the data that is stored in these buckets.

We do not cover these steps in detail because they are in 8.1, “Points of sale (physical stores)” on page 122. Follow the same steps.

We show how our catalogs (Figure 8-17) and buckets (Figure 8-18) look like after they are configured from the watsonx.data UI interface.

Infrastructure manager
Define and associate your infrastructure components.

Search your system

Engines 1 | **Catalogs 6** | Buckets 6 | Databases 0

Name	Status	Type	Engines associated	Bucket associated
ecommerce_brower_logs	Queryable	Apache Hive	1	ecommerce_brower_logs
ecommerce_curated_table	Queryable	Apache Iceberg	1	e-comm curated data bucket
ecommerce_transactions	Queryable	Apache Hive	1	e-comm transactions

Figure 8-17 Infrastructure manager: Catalogs

Infrastructure manager
Define and associate your infrastructure components.

Search your system

Engines 1 | Catalogs 6 | **Buckets 6** | Databases 0

Display name	Status	Type	Catalog associated
ecommerce_brower_logs	Queryable	IBM Storage Ceph	ecommerce_brower_logs
e-comm curated data bucket	Queryable	IBM Storage Ceph	ecommerce_curated_table
e-comm transactions	Queryable	IBM Storage Ceph	ecommerce_transactions

Figure 8-18 Infrastructure manager: Buckets

Figure 8-19 on page 145 shows the Data manager window.

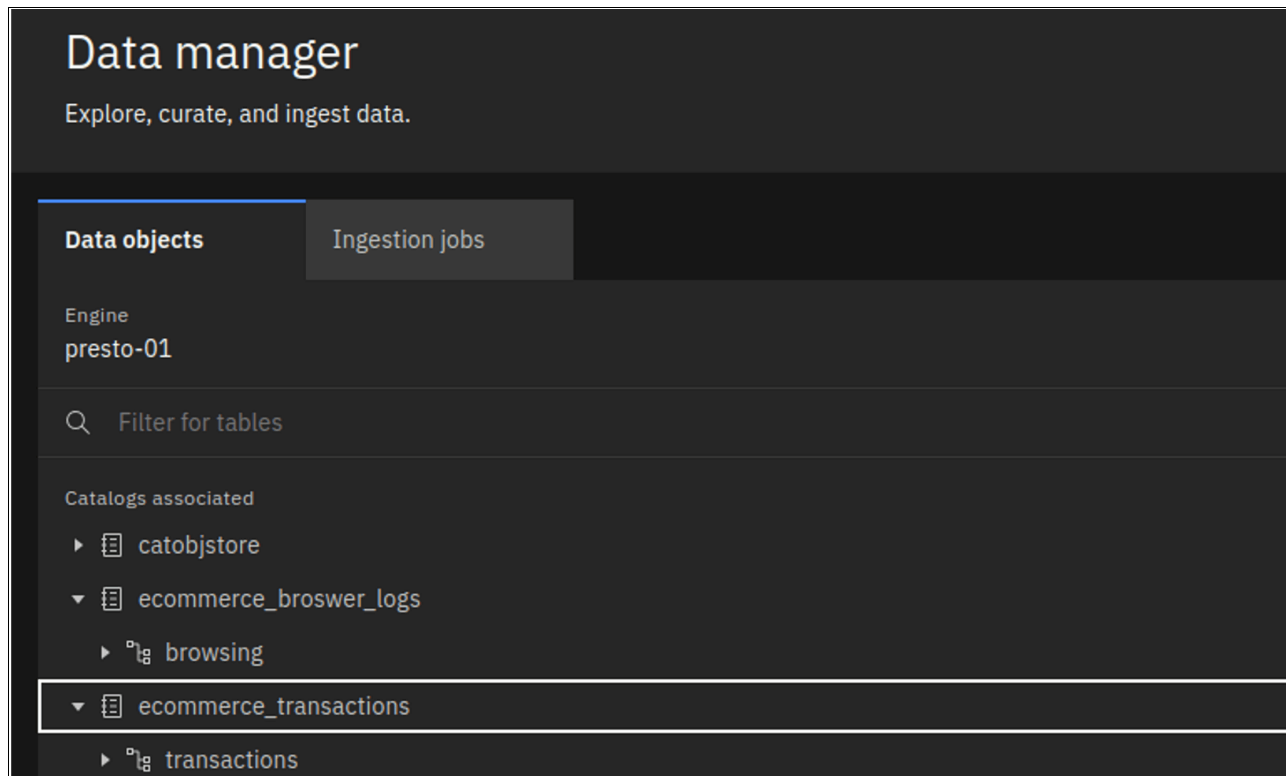


Figure 8-19 Data manager objects

Note: Although watsonx.data offers a powerful SQL query interface, for this demonstration, we continue working from the presto CLI. This choice provides a couple of benefits:

- ▶ Improved readability: Presto CLI outputs the query results directly in the terminal window. This format can be simpler to read and navigate compared to screen captures from a GUI like watsonx.data.
- ▶ Code focus: With the CLI environment, you can focus solely on the SQL code itself, potentially improving clarity for users familiar with CLI tools.

Connecting to a Presto instance

To connect to an IBM Cloud Pak for Data (CP4D) watsonx.data Presto instance, follow the detailed instructions at [Connecting to a Presto server](#). The process can be summarized in the following steps:

1. Download and install the watsonx.data client package. For more information, see Appendix B, “Configuring the command-line tools for IBM watsonx.data” on page 187.
2. Get the Presto Route from Red Hat OpenShift Container Platform (if connecting from outside Red Hat OpenShift Container Platform) (Example 8-19).
3. Authenticate against the Presto server by using the `presto-cli` binary file.

Example 8-19 Connecting to a CP4D watsonx.data Presto instance

```
$ oc get route -A | grep -i presto
cpd-operands          ibm-lh-lakehouse-presto-01-presto-svc
ibm-lh-lakehouse-presto-01-presto-svc-cpd-operands.ocp-eu-redbook-ce869a544b369f1d
fd24beec10027762-i000.eu-es.containers.appdomain.cloud
```

```
ibm-lh-lakehouse-presto-01-presto-svc 8443 reencrypt
None
$ ./presto-cli --server
https://ibm-lh-lakehouse-presto-01-presto-svc-cpd-operands.ocp-eu-redbook-ce869a54
4b369f1dfd24beec10027762-i000.eu-es.containers.appdomain.cloud --user cpadmin
Password:
presto> show catalogs;
      Catalog
-----
ecommerce_broswer_logs
ecommerce_transactions
Query 20240425_172833_00037_3yss9, FINISHED, 1 node
Splits: 19 total, 19 done (100.00%)
[Latency: client-side: 0:01, server-side: 395ms] [0 rows, 0B] [0 rows/s, 0B/s]
```

Running SQL queries with the Presto CLI

The Presto CLI provides a terminal-based interactive shell to run queries. This section shows some examples.

To run an SQL query by using the Presto CLI, complete the following steps:

1. Select a default catalog and schema on the Presto CLI for the session (Example 8-20).

Example 8-20 Selecting a default catalog and schema

```
./presto-cli --server
https://ibm-lh-lakehouse-presto-01-presto-svc-cpd-operands.ocp-eu-redbook-ce869a54
4b369f1dfd24beec10027762-i000.eu-es.containers.appdomain.cloud --user cpadmin
--schema transactions --catalog ecommerce_transactions
Password:
presto:transactions> show tables ;
      Table
-----
(0 rows)
Query 20240425_181246_00057_3yss9, FINISHED, 1 node
Splits: 19 total, 19 done (100.00%)
[Latency: client-side: 0:01, server-side: 0:01] [0 rows, 0B] [0 rows/s, 0B/s]
presto:transactions> CREATE TABLE transactions ( client_id BIGINT, transaction_id
VARCHAR, item_id VARCHAR, item_description VARCHAR, category VARCHAR, quantity
INTEGER, total_amount DOUBLE, credit_card_number VARCHAR, transaction_date
TIMESTAMP ) WITH ( format = 'PARQUET', external_location =
's3a://ecomtrans/transactions/' );
CREATE TABLE
```

2. Before proceeding, confirm that the data is accessible for querying through Presto. This approach ensures that you can interact with the data by using Presto commands. Run a select query on the transactions database to check valid entries (Example 8-21).

Example 8-21 Confirming that our data is accessible for querying through Presto

```
presto:transactions> select * from transactions limit 10;
 client_id | transaction_id | item_id
| item_description | category | quantity | total_amount |
credit_card_number | transaction_date
```


[illegible]

Example 8-22 Creating the browser log table

```

i_description VARCHAR, c_preferred_cust_flag BOOLEAN, ds DATE ) WITH ( format =
'Parquet', external_location = 's3a://ecommlogs/browsing/', partitioned_by =
ARRAY['ds'] );
CREATE TABLE
presto:browsing> show tables ;
      Table
-----
browsing_data_partitioned
(1 row)

```

- When table creation is complete, issue a **SELECT** statement to fetch data from the newly created table. By doing so, you verify the table schema and ensure data population. This approach is common with partitioned tables that must update the schema metadata (Example 8-23).

Example 8-23 Issuing a SELECT statement to fetch data from the newly created table

```

presto:browsing> select * from browsing_data_partitioned LIMIT 5;
 ip | ts | tz | verb | resource_type | resource_fk | response | browser | os |
customer | d_day_name | i_current_price | i_category | i_description |
c_preferred_cust_flag | ds
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----
(0 rows)
Query 20240425_203132_00094_3yss9, FINISHED, 1 node
Splits: 1 total, 1 done (100.00%)
[Latency: client-side: 0:01, server-side: 307ms] [0 rows, 0B] [0 rows/s, 0B/s]
presto:browsing>

```

- You can run a **CALL** command to synchronize or refresh the table partition metadata. Afterward, you can see that the **SELECT** query provides the expected database entries (Example 8-24).

Example 8-24 Running a CALL command to synchronize or refresh the table partition metadata

```

presto:browsing> CALL
ecommerce_browser_logs.system.sync_partition_metadata(schema_name => 'browsing',
table_name => 'browsing_data_partitioned', mode => 'ADD');
presto:browsing> select * from browsing_data_partitioned LIMIT 5;
      ip      |      ts      | tz | verb | resource_type |
resource_fk   | response | browser | os  | customer | d_day_name |
i_current_price | i_category | i_description | c_preferred_cust_flag | ds
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
---
72.126.54.198 | 2024-03-20 17:00:47.000 | UTC | GET | Item          |
8b0402e7-346f-4903-aa47-a63b90e702b5 | 500 | Opera  | Windows | 178584 |
Wednesday | 969.97 | Jewelry | Pendant | true
| 2024-03-20
111.21.31.49  | 2024-03-20 22:32:38.000 | UTC | GET | Item          |
fa43e8d2-c512-402b-b7f4-f0580c2afef4 | 200 | Edge   | Android | 247478 |
Wednesday | 416.9 | Accessories | Briefcase | false
| 2024-03-20
172.98.136.85 | 2024-03-20 09:04:33.000 | UTC | GET | Item          |
194b15fa-52dd-426c-94e8-91294e15d62a | 200 | Opera  | iOS     | 884882 |

```

```

Wednesday |          120.07 | Jewelry      | Anklet        | false
| 2024-03-20
15.218.133.222 | 2024-03-20 01:39:02.000 | UTC | GET | Item |
be9d78f4-b352-472e-8596-1536f35d5455 | 200 | Opera | Windows | 574703 |
Wednesday |          241.46 | Footwear     | Shoes         | false
| 2024-03-20
9.11.144.150 | 2024-03-20 08:21:17.000 | UTC | GET | Item |
33cca06e-a966-422c-90ac-453264a25cee | 500 | Chrome | iOS | 989507 |
Wednesday |          850.84 | Clothing     | Blouse        | false
| 2024-03-20
(5 rows)
Query 20240425_204056_00099_3yys9, FINISHED, 1 node
Splits: 250 total, 22 done (8.80%)
[Latency: client-side: 0:02, server-side: 0:02] [42 rows, 580KB] [24 rows/s,
337KB/s]

```

6. When both of the e-commerce tables are ready, you can perform a **JOIN** SQL operation that combines the detailed transaction records with the corresponding browsing behavior to create a unified data view. By linking these two data sets, you can get a complete view of the customer journey from initial interest (browsing) to purchase (transaction).

By combining browsing data with transaction records, you can uncover valuable patterns that would be hidden in isolated data sets. This enriched view allows for data-driven decision making:

- Understanding purchase behavior: Analyze which items were browsed the most before purchasing, which reveals customer interest and potential buying triggers.
- Purchase journey insights: Explore how long customers took to make a purchase so that you can identify areas for streamlining the buying process.
- Targeted recommendations: Discover browsing habits that lead to higher sales. Leverage this knowledge to personalize product recommendations and optimize marketing strategies.

The **JOIN** operation enriches each transaction record by adding contextual information from the browsing data. For example, a transaction record might show what was purchased, but when joined with browsing data, it reveals what the customer considered before making that purchase. This enriched context provides a deeper understanding of customer preferences so that you can tailor your offerings and marketing efforts for greater impact.

Example 8-25 shows an example of a **JOIN** query between the transactions and browser log tables. By leveraging both tables, we can, for example, calculate the conversion rate for different product categories to see which are more effective at converting user browses into sales.

Example 8-25 JOIN query between the transactions and browser log tables

```

-> JOIN ecommerce_browser_logs.browsing.browsing_data_partitioned b
-> ON t.client_id = b.customer AND t.item_id = b.resource_fk
-> GROUP BY b.i_category
-> ORDER BY conversion_rate DESC
-> LIMIT 5;

```

i_category	purchases	browsed_items	conversion_rate
Clothing	56	1654	3.39
Accessories	72	2599	2.77
Footwear	29	1879	1.54

Strategies for complex SQL queries involving JOINS: Complex SQL queries involving **JOIN** statements can be challenging to write and maintain, and can lead to slow performance. There are two main strategies to address this challenge, each with its own advantages and disadvantages:

- ▶ Create a **VIEW** that makes accessing the complex **JOIN** query simple for the final users.
- ▶ Create a table out of the **JOIN** query, so further queries can be done on a single table.

Views and tables have their pros and cons.

A view is a virtual table that is based on the result set of an SQL statement. It contains rows and columns like a real table, but the fields in a view are fields from one or more real tables in the database.

- ▶ Pros:
 - Does not occupy physical space in the storage.
 - It always provides up-to-date data because it queries the underlying tables dynamically.
 - Simplifies complex SQL into simple queries for users.
- ▶ Cons:
 - Performance can be slower than querying from a physical table because the view must run the underlying SQL each time that it is accessed.
 - Views can become invalid if underlying tables are changed significantly.

A table is a database object that physically stores data. Because the data is directly accessible, it can be queried more quickly.

- ▶ Pros:
 - Faster query performance, especially with proper indexing.
 - It can be optimized with physical tweaks in the database, such as partitions.
- ▶ Cons:
 - Occupies physical space.
 - Data can become stale unless updated or synchronized regularly.

8. Example 8-27 shows an example of creating a view for reference. After you create a view in PrestoDB SQL, you can use it in queries like a regular table. A view acts as a virtual table that does not physically store data but dynamically generates it from the underlying tables on each query against the view. You can simplify complex data access logic and reuse it throughout your code.

Example 8-27 Creating a view

```
presto:browsing> CREATE VIEW conversion_rate_by_category AS
SELECT b.i_category, COUNT(DISTINCT t.transaction_id) AS purchases, COUNT(DISTINCT
b.resource_fk) AS browsed_items,
      ROUND(100.0 * COUNT(DISTINCT t.transaction_id) / COUNT(DISTINCT
b.resource_fk), 2) AS conversion_rate
FROM ecommerce_transactions.transactions.transactions t
JOIN ecommerce_browser_logs.browsing.browsing_data_partitioned b
ON t.client_id = b.customer AND t.item_id = b.resource_fk
GROUP BY b.i_category;
presto:browsing> select * from conversion_rate_by_category ORDER BY
conversion_rate DESC ;
i_category | purchases | browsed_items | conversion_rate
```

Clothing	56	1654	3.39
Accessories	72	2599	2.77
Footwear	29	1879	1.54
Jewelry	17	2401	0.71
Electronics	12	2807	0.43

Query 20240503_091637_00018_k5dz7, FINISHED, 1 node

Splits: 164 total, 164 done (100.00%)

[Latency: client-side: 0:05, server-side: 0:04] [309K rows, 1.07MB] [73K rows/s, 259KB/s]

9. Create an external table that holds this joined data. This table can be used for more efficient queries because it avoids repeatedly joining these tables and allows quicker access to the combined data set. This new table is stored in a new bucket that is part of the curated zone. This bucket is the final destination of our e-commerce data. We cleansed and enriched the data set, and this new destination bucket uses the Iceberg table format.

Before creating the table, create and configure the bucket in watsonx.data. As in the previous examples, the buckets are created by the `ecomadmin` user ID and we provide access to the `wxdecomm` user through bucket policies. Because there is no support in watsonx.data for IAM roles at the time of writing, we use bucket policies to control access to the buckets (Example 8-28).

Example 8-28 Creating and configuring the bucket in watsonx.data

```
$ aws --profile ecomadmin s3 mb s3://ecommcured
$ cat bucket_policy-for-wxd-to-use-ecommcured.json
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::user/wxdecomm"
      },
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3::ecommcured/*",
        "arn:aws:s3::ecommcured/",
        "arn:aws:s3::ecommcured"
      ]
    }
  ]
$ aws --profile ecomadmin s3api put-bucket-policy --bucket ecommcured --policy
file://bucket_policy-for-wxd-to-use-ecommcured.json
```

10. When we have the bucket and policy, we create the catalog for watsonx.data by using the Iceberg connector and connect it to the Presto engine through the UI (Figure 8-20 on page 153).

Infrastructure manager				
Define and associate your infrastructure components.				
<input type="text" value="Search your system"/>				
Engines 1	Catalogs 4	Buckets 4	Databases 0	
Name	Status	Type	Engines associated	Bucket associated
ecommerce_brower_logs	Queryable	Apache Hive	1	ecommerce_brower_logs
ecommerce_transactions	Queryable	Apache Hive	1	e-comm transactions
wxd_system_data	Queryable	Apache Hive	0	wxd-system
ecommerce_curated_table	Queryable	Apache Iceberg	1	e-comm curated data bucket
Items per page: 10 1-4 of 4 items				

Figure 8-20 Catalogs view

11. When the catalog and schema are defined, you have two options for using the joined data:

- Create a permanent table: Run a **CREATE TABLE** statement that is based on the join query. This statement creates a table that stores the combined results for future use (Example 8-29).
- Temporary analysis: If the joined data is needed only for immediate analysis, consider using the join query directly within Presto. This approach avoids creating a separate table.

Example 8-29 Creating a table from the JOIN statement

```

presto:browsing> CREATE TABLE
"ecommerce_curated_table"."joint_table_logs_trans".combined_browsing_transactions
WITH (
    format = 'PARQUET',
    partitioning = ARRAY['ds']
) AS
SELECT
    b.ip,
    b.ts,
    b.tz,
    b.verb,
    b.resource_type,
    b.resource_fk,
    b.response,
    b.browser,
    b.os,
    b.customer,
    b.d_day_name,
    b.i_current_price,
    b.i_category,
    b.i_description,
    b.c_preferred_cust_flag,
    b.ds,
    t."transaction id" AS transaction_id,
    t."item description" AS item_description,
    t.category,
    t.quantity,
    t."total amount" AS total_amount,
    t."credit card number" AS credit_card_number,

```

[illegible]

Benefits of using the Iceberg table format in PrestoDB: Using the Iceberg table format in PrestoDB provides several advantages over traditional table formats like those offered by the Hive connector:

- ▶ With Iceberg, you have robust schema evolution, which enables additions, deletions, and updates to table schemas without breaking old readers. Queries run correctly even as the table schema changes. Iceberg also offers hidden partitioning, query optimization, and atomicity, consistency, isolation, and durability (ACID) transactions, which ensure the ACID of data operations.
- ▶ Another benefit of using Iceberg is snapshot isolation, which enables readers to access a consistent snapshot of the data, even as the data is being updated, which provides a consistent view always.
- ▶ Lastly, Iceberg tables are designed to scale efficiently to petabyte-size tables and beyond without degradation in performance. The metadata size is kept small and distributed, avoiding bottlenecks that are associated with extensive metadata, as seen in some Hive table implementations.

Our data pipeline leverages Iceberg tables and S3 lifecycle policies to ensure efficient data management:

- ▶ **Data cleansing and removal:** After data is processed into curated Iceberg tables, the previously stored data in the staging zone is automatically cleaned and removed by using an S3 lifecycle policy. This approach optimizes storage utilization by eliminating redundant copies.
- ▶ **Immutable source of truth:** The raw zone remains the sole immutable source of truth for the data. This immutability ensures data integrity and enables rerunning the entire data pipeline if necessary. Therefore, all downstream views, tables, and other data products can be regenerated based on the clean and unaltered data in the raw zone.

Our example showcases a curated Iceberg table that is created by joining browser logs and transactions data. As new data arrives in the staging zone and updates occur in the source tables, it is crucial to refresh the curated table to capture these changes.

There are several methods to refresh our table, from the most straightforward approach that uses scheduled jobs that incrementally update the table data through SQL **INSERTS** to more complex scenarios for large data sets that use a data pipeline tool for the orchestration, like Apache Airflow.

During this walkthrough, you saw examples where data is copied across different data lake zones. Although this approach can be useful sometimes, it is not always necessary.

IBM watsonx.data empowers you to directly query data in its raw zone. This approach eliminates the need for creating extra copies of the data, which saves storage space and processing time. This strategy is beneficial for the following items:

- ▶ **Real-time decision making:** When immediate insights are required, querying the raw zone data directly enables faster access to the freshest information.
- ▶ **Complex analytical queries:** Complex analytical queries often benefit from accessing the most up-to-date data. IBM watsonx.data facilitates this task by enabling direct raw zone queries.



Consume

The *curated zone* acts as the bridge between the raw data you ingest and the valuable insights that you seek. By leveraging the curated zone, you can unlock the true potential of your data, transforming it from a collection of facts into a strategic asset that fuels informed decision-making.

This chapter shares some examples of consuming data from the curated zone through visualization tools like Apache Superset, and optimized SQL tables through the watsonx.data SQL query engine.

This chapter has the following sections:

- ▶ Introduction
- ▶ Visualization: E-commerce example
- ▶ Curated point-of-sale SQL queries

9.1 Introduction

The curated zone plays a critical role in transforming raw data into actionable insights within your data lake architecture. It serves as a layer where data is refined, consolidated, and optimized for use by various users and applications throughout the organization. It is essential to support high-quality data analysis, reporting, and decision-making.

Extensive processing and validation ensure that data in the curated zone meets the quality standards that are required for analytical purposes. This zone contains data that is cleansed, enriched, transformed, and cataloged to be easily accessible and queryable by data scientists, business analysts, and decision-makers.

The curated zone is the heart of your data lake, housing high-quality data that is ready for analysis. This data undergoes a rigorous cleaning process to eliminate inaccuracies, inconsistencies, and redundancies. It adheres to strict organizational data quality standards, which ensures reliable foundations for all downstream tasks.

To facilitate rapid data retrieval, the curated zone leverages techniques like indexing, partitioning, and caching. Unlike raw data in its original formats (CSV, JavaScript Object Notation (JSON), and TXT), curated data is typically structured and optimized for efficient querying and analysis. This process often involves transforming it into columnar formats like Parquet, Avro, or Optimized Row Columnar (ORC), which are designed for faster data access and processing.

The curated zone prioritizes data security and compliance. Rigorous controls are implemented to safeguard sensitive information and ensure adherence to relevant regulations. This robust security posture protects your valuable data assets.

The data in the curated zone is prepared for direct use in business intelligence (BI) tools, machine learning (ML) models, and other analytics applications without requiring further transformation. This chapter shares some examples of consuming data from the curated zone through visualization tools like Apache Superset, and optimized SQL tables through the watsonx.data SQL query engine.

9.2 Visualization: E-commerce example

In our e-commerce retail example, we processed the raw data that was obtained from the server logs in to the raw zone. The logs underwent multiple processes, including security checks to remove fake entries from malicious or non-human customers. The logs were also cleansed to remove duplicate and inconsistent entries and transformed into the Parquet file format. Now, the logs are available for table partitioning by date if required.

In the staging zone, the browser logs merge with the e-commerce transaction data set to enrich our collection by combining them into a single table. This table is accessible to different retail departments through dedicated views and tables.

Data visualization tools are crucial for data analysis. They transform complex data sets into clear and compelling visuals, empowering decision-makers to achieve these goals:

- ▶ Grasp difficult concepts at a glance.
- ▶ Spot hidden patterns with ease.
- ▶ Make faster and more informed decisions.

This section demonstrates an example of visualizing curated data by using [Apache Superset](#). For more information about the Superset functions, see the official documentation.

Apache Superset is a powerful, open-source web application for data exploration and visualization. It caters to enterprises by offering an interface for creating and sharing interactive dashboards. Superset supports various visualization types and can connect to many SQL-based data sources. Some of its key features include rich visualization types, integration with different databases, customizable dashboards with a simple drag interface, an integrated SQL editor, and robust authentication mechanisms and permission configuration.

There is a long list of visualization and BI-related tools that you can explore along with Apache Superset. Here are some of them:

- ▶ [Tableau](#) is a leading BI software that enables users to create and share interactive charts, graphs, dashboards, and complex visual stories. It has powerful data visualization capabilities and a simple interface.
- ▶ [Power BI](#) is a suite of business analytics tools from Microsoft that enables businesses to visualize their data and share insights across the organization, or embed them in an app or website. It has deep integration with other Microsoft products.
- ▶ [Looker](#), now part of Google Cloud, offers a data exploration and discovery BI platform. It provides powerful analytics and insights through its robust data modeling language.
- ▶ [Sisense](#) empowers builders to manage, analyze, and visualize complex data to deliver insights at the right time on the right device. It can be embedded into apps to deliver analytics at the point of decision.
- ▶ [Metabase](#) is an open-source BI tool that you can use to gain insight into your data, and displays answers in formats like a bar graph or a detailed table.
- ▶ [Redash](#) helps you make sense of your data. You can use it to connect and query your data sources, and build dashboards to visualize data and share them with your company. It supports querying multiple databases, including newer NoSQL databases.

9.2.1 Deploying and configuring Apache Superset

Apache Superset offers various deployment methods to suit your needs. These methods include local installations, containerization with Docker or Podman, and deployments on Kubernetes by using Helm charts.

In our example, we have a running Red Hat Enterprise Linux (RHEL) 9 operating system (OS) system with Podman available, where we spin up a container that is called `superset` based on the `"apache/superset"` container image. Also, we configure the Superset secret key by using an environment variable (Example 9-1).

Example 9-1 The podman run command

```
# cat /etc/redhat-release
Red Hat Enterprise Linux release 9.3 (Plow)

# podman run -d -p 8088:8088 -e SUPERSET_SECRET_KEY="oh-so-secret" --name superset
apache/superset
```

After the container is running, we use the **podman exec** command to connect to the running container and run commands within its environment. After we are inside the container, we run the superset **create-admin** command. This command guides you through setting up an administrator account, including defining a password that you use to access the Superset dashboard (Example 9-2).

Example 9-2 Setting up an administration account

```
# podman exec -it superset bash
$ superset fab create-admin
$ superset db upgrade
$ superset superset init
```

After running the **superset init** command, the Superset UI is available through a web browser on port 8088 of the RHEL operating system (OS) host (Example 9-3).

Example 9-3 Superset UI is available through a web browser on port 8088

```
# curl http://10.251.0.35:8088
<!doctype html>
<html lang=en>
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to the target URL: <a
href="/superset/welcome/">/superset/welcome/</a>. If not, click the link.
```

Figure 9-1 shows the Sign In window.

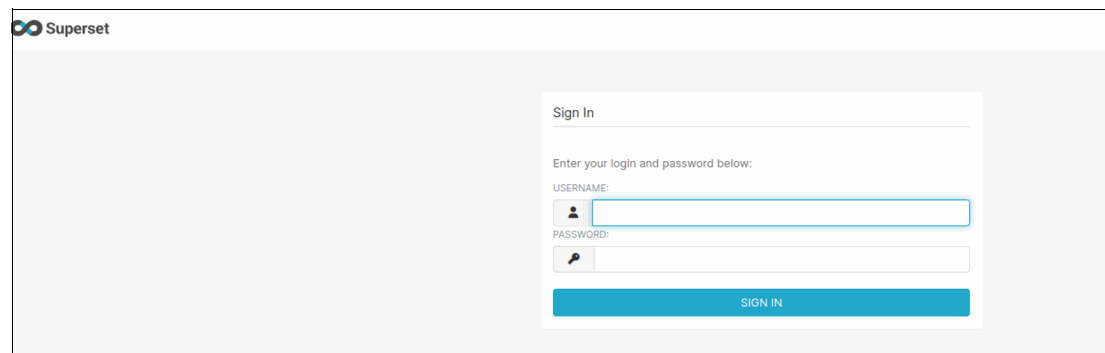


Figure 9-1 Sign In window

Now that we are logged in with our administrator credentials, we connect Superset to your data source. In this example, we configure Superset to access the PrestoDB SQL engine that we set up in `watsonx.data` (Figure 9-2 on page 161).

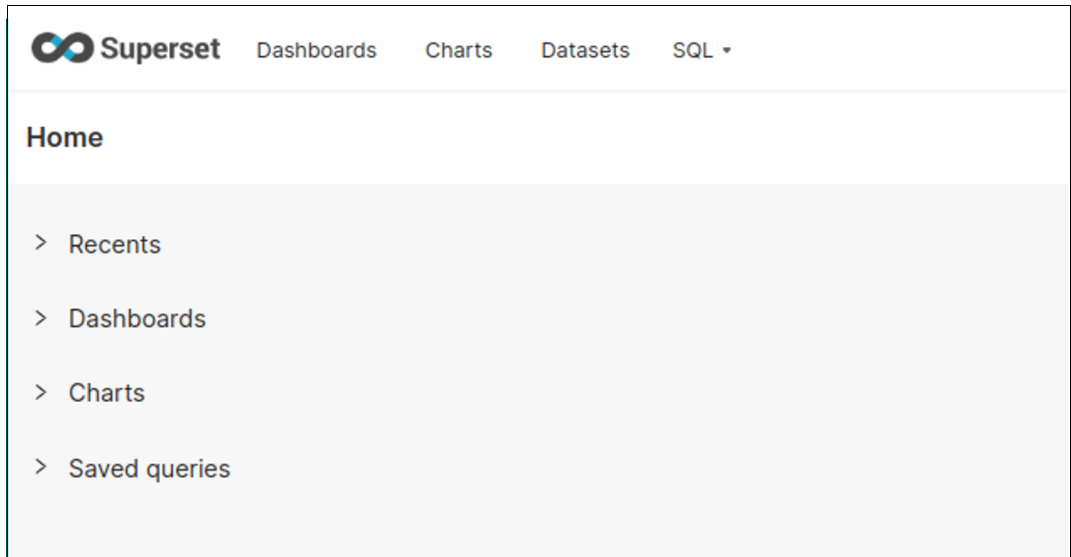


Figure 9-2 Configuring Superset to access the PrestoDB SQL engine

9.2.2 Connecting Apache Superset to PrestoDB (watsonx.data)

This section describes an example of connecting Apache Superset to a running PrestoDB instance. To do so, complete the following steps:

1. Select **Settings** → **Database Connections** (Figure 9-3).

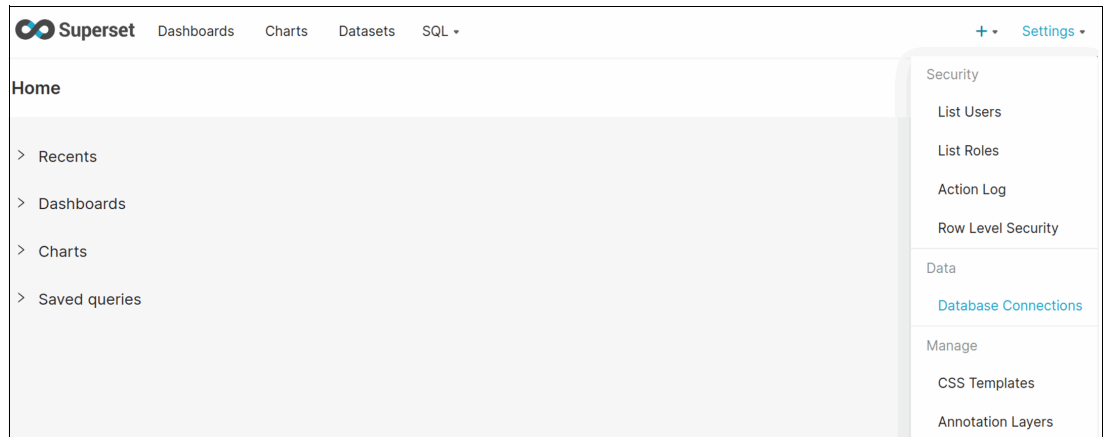


Figure 9-3 Selecting Settings > Database Connections

2. In the Databases section, click **+ DATABASE** to add a database connection (Figure 9-4).

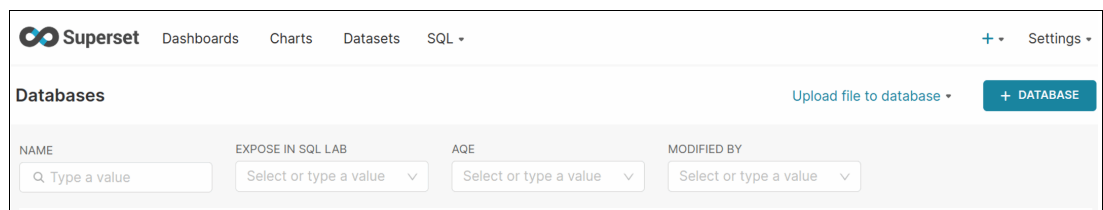


Figure 9-4 Adding a database connection

3. Click **Presto** (Figure 9-5 on page 162).

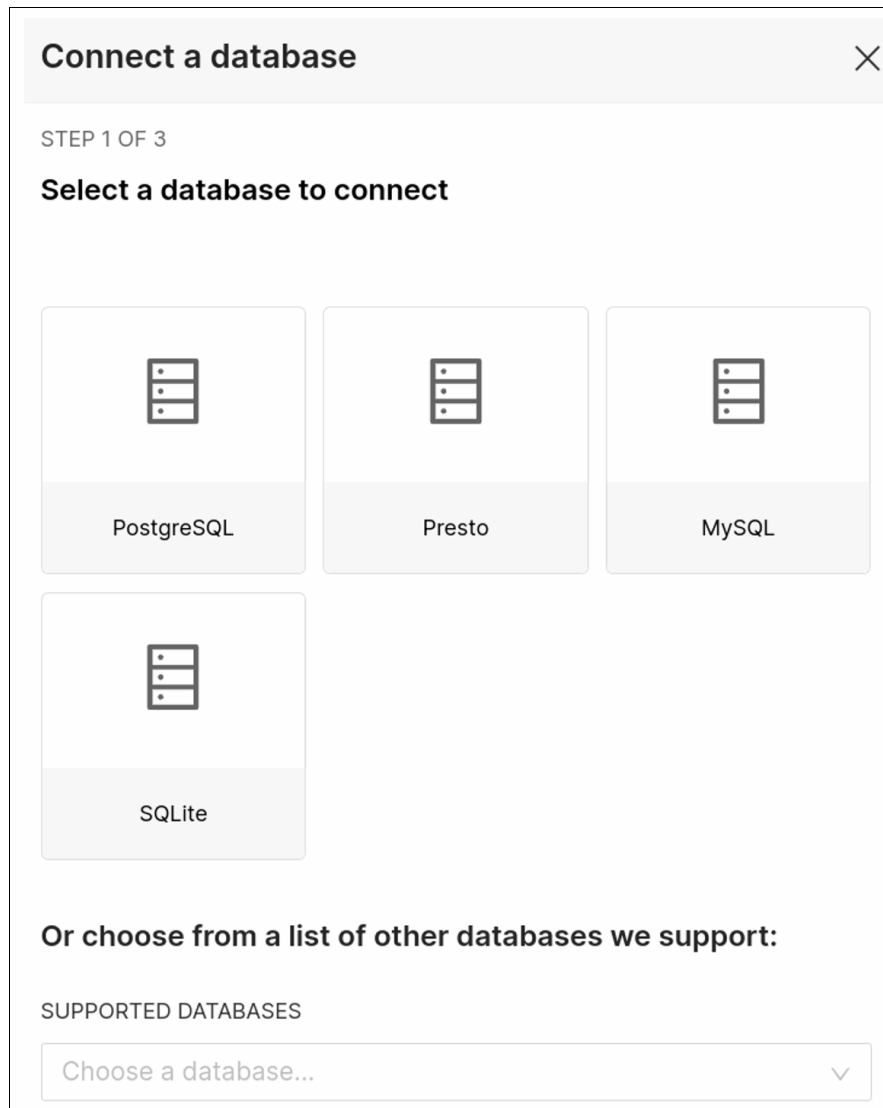


Figure 9-5 Selecting Presto from the list

4. In the **BASIC Configuration** tab, you can define key details for your data source. Provide a display name and the SQLAlchemy connection URI, which specifies how Superset connects to the database (Figure 9-6 on page 163).

PRESTO

WXD_curated_ecommerce_logs_transactions

BASIC

ADVANCED

DISPLAY NAME *

WXD_curated_ecommerce_logs_transactions

Pick a name to help you identify this database.

SQLALCHEMY URI *

presto://ecommerceadm:XXXXXXXXXX@ibm-lh-lakehouse-presto-0

Refer to the [SQLAlchemy docs](#) for more information on how to structure your URI.

TEST CONNECTION

Figure 9-6 BASIC Configuration tab

5. Connect to a curated data set in the catalog. This data set is in a bucket within the curated zone and combines the e-commerce browsing logs and transaction data in a single table for analysis.

The URI is to use for this example is shown in Example 9-4.

Example 9-4 URI that is used for this example

```
presto://ecommerceadm:XXXXXXXXXX@ibm-lh-lakehouse-presto-01-presto-svc-cpd-operand
s.ocp-eu-redbook-ce869a544b369f1dfd24beec10027762-i000.eu-es.containers.appdomain.
cloud:443/ecommerce_curated_table
```

6. For this example, we use a username and password combination to access the PrestoDB catalog that is named `ecommerce_curated_table`.

Since we use SSL for the PrestoDB connection, but the Superset deployment does not trust the PrestoDB SSL certificate, we must add the entry `"verify: false"` into the Advanced security settings (Figure 9-7).

Tip: In production, ensure that your Superset deployment trusts the SSL certificate that is used by your PrestoDB instance by installing the trusted certificate authority (CA) certificate or configuring mutual Transport Layer Security (TLS) authentication. Bypassing this check (`verify: false`) introduces security vulnerabilities and should be avoided in a production environment.

WXD_curated_ecommerce_logs_transactions

BASIC

ADVANCED

SQL Lab

Adjust how this database will interact with SQL Lab.

Performance

Adjust performance settings of this database.

Security

Add extra connection information.

SECURE EXTRA

1

2

3

4

5

6

7

```
{
  "connect_args":
  {"protocol": "https",
   "requests_kwargs":{"verify":false}}
}
```

JSON string containing additional connection configuration. This is used to provide connection information for systems like Hive, Presto and BigQuery which do not conform to the username:password syntax normally used by SQLAlchemy.

Figure 9-7 Adding the entry `"verify: false"` in the Advanced security settings

7. After saving your configuration, click **Test connection** to confirm that Superset can successfully connect to your PrestoDB instance on Red Hat OpenShift.

8. Now, we can go into the Superset SQL Lab to start creating our charts (Figure 9-8).

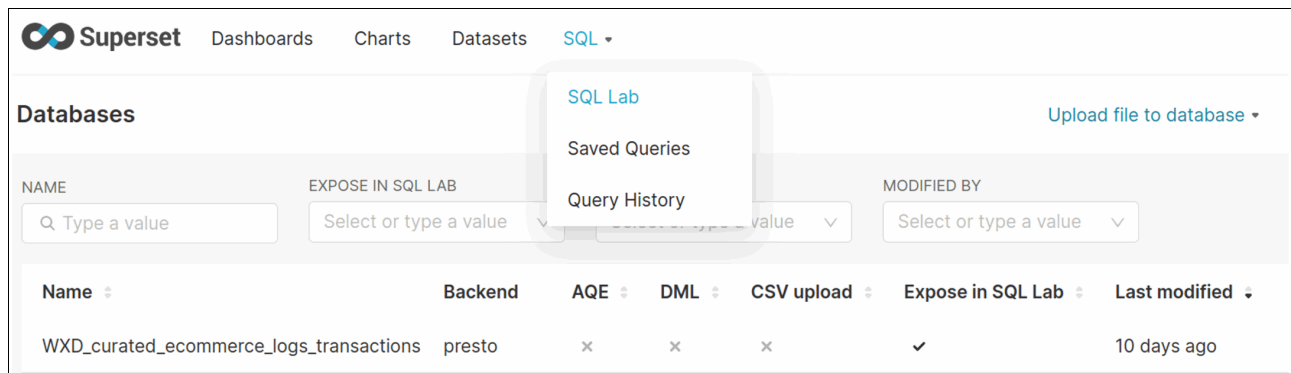


Figure 9-8 Superset SQL Lab

9.2.3 Creating a graph or data set from an SQL query

This section describes an example of creating a chart from an SQL query. In this example, you populate our e-commerce visualization dashboard with this chart.

Complete the following steps:

1. In the Superset SQL Lab, select the database or catalog, schema, and table. In this case, choose the combined_browsing_transactions table that was created in Chapter 8, “Transform: Staging and curated zones” on page 121.

- Superset automatically displays the table schema and provides a detailed overview of the data structure. Also, you can preview a sample of the data to get a quick glimpse into its contents (Figure 9-9).

The screenshot shows the Apache Superset web interface. On the left, the 'DATABASE' is set to 'presto' and the 'SCHEMA' is 'WXD_curated_ecommerce_logs_tr...'. The 'SEE TABLE SCHEMA' section shows the table 'combined_browsing_transactions'. Below this, the table schema is displayed as a list of columns and their data types:

Column Name	Data Type
ip	VARCHAR
ts	TIMESTAMP
tz	VARCHAR
verb	VARCHAR
resource_type	VARCHAR
resource_fk	VARCHAR
response	INTEGER
browser	VARCHAR
os	VARCHAR
customer	BIGINT
d_day_name	VARCHAR
i_current_price	FLOAT
i_category	VARCHAR
i_description	VARCHAR
c_preferred_cust_flag	BOOLEAN
ds	DATE
transaction_id	VARCHAR
item_description	VARCHAR

On the right, the 'RESULTS' tab is active, showing a preview of the data. The preview shows 100 rows of data. The first few rows are:

ip	ts	tz	verb	resource_type
25.179.76.191	2024-03-16 04:10:47.000	UTC	GET	Item
153.152.70.39	2024-04-06 00:28:49.000	UTC	GET	Item
151.143.81.126	2024-01-20 23:31:56.000	UTC	GET	Item
218.64.50.178	2024-03-09 01:07:57.000	UTC	GET	Item
72.147.69.26	2024-04-17 14:24:15.000	UTC	GET	Item
179.198.49.255	2024-03-09 13:07:18.000	UTC	GET	Item
57.187.30.158	2024-03-20 14:11:56.000	UTC	GET	Item

Figure 9-9 The table schema: Displaying a detailed overview of the data structure

- Insert a query to see the top 10 selling items (Example 9-5).

Example 9-5 Query to see the top 10 selling items

```
SELECT
    item_description,
    SUM(quantity) AS total_quantity_sold,
    SUM(total_amount) AS total_revenue
FROM
    combined_browsing_transactions
GROUP BY
    item_description
ORDER BY
    total_quantity_sold DESC
LIMIT 10;
```

4. After running the query in SQL Lab, you see the results. Superset provides an option to “Create a chart” from the query data. Leverage this feature to create visual representations of your analysis (Figure 9-10).

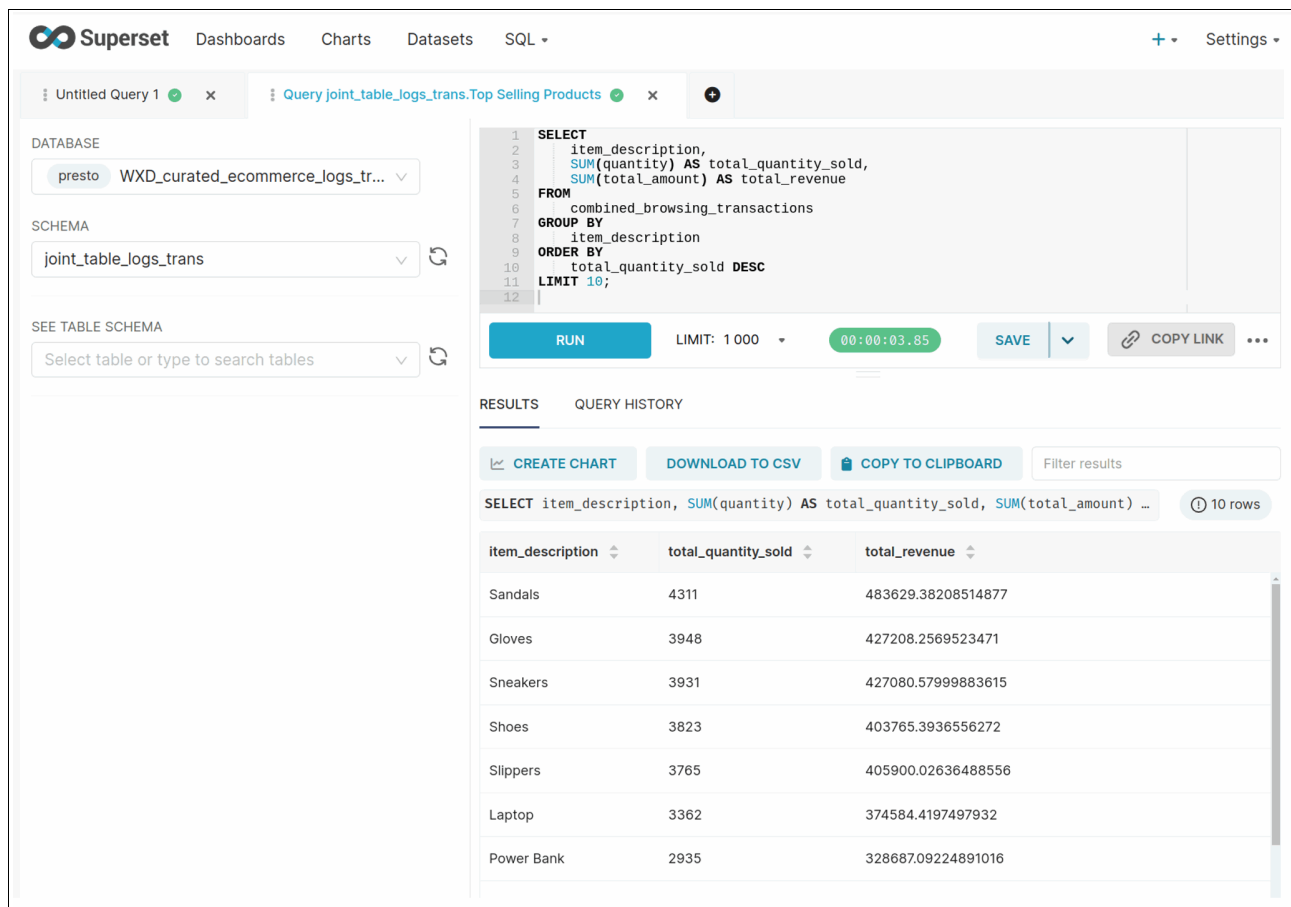


Figure 9-10 Superset displays the results

5. In the Charts section, use a bar chart to visualize the data. Configure the chart's elements as follows:
 - X-axis: Assign the `Item_description` field to the X-axis. This element displays individual item descriptions along the horizontal axis of the chart.
 - Sorting: To prioritize the most frequently sold items, sort the data by `Total_quantity_sold` in descending order. Items with the highest sales volumes appear at the top or right of the chart.
 - Metric: Use the `SUM(total_quantity_sold)` metric to aggregate the total quantity that is sold for each item. You have a clear view of sales performance across different items.

Figure 9-11 shows the configured elements for the chart.

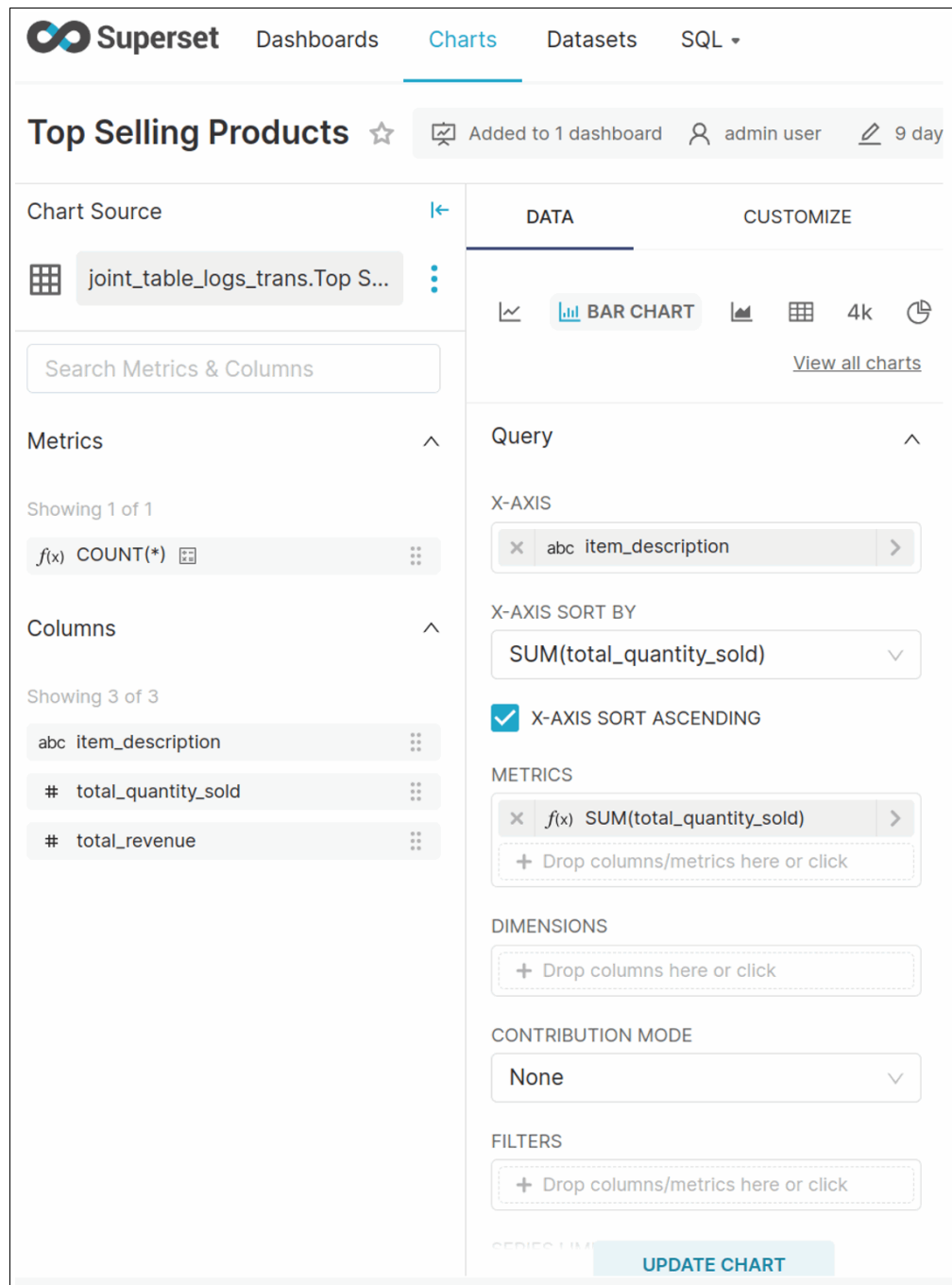


Figure 9-11 Visualizing the data

Now, you have a visualized graph that shows the top-selling e-commerce products (Figure 9-12).

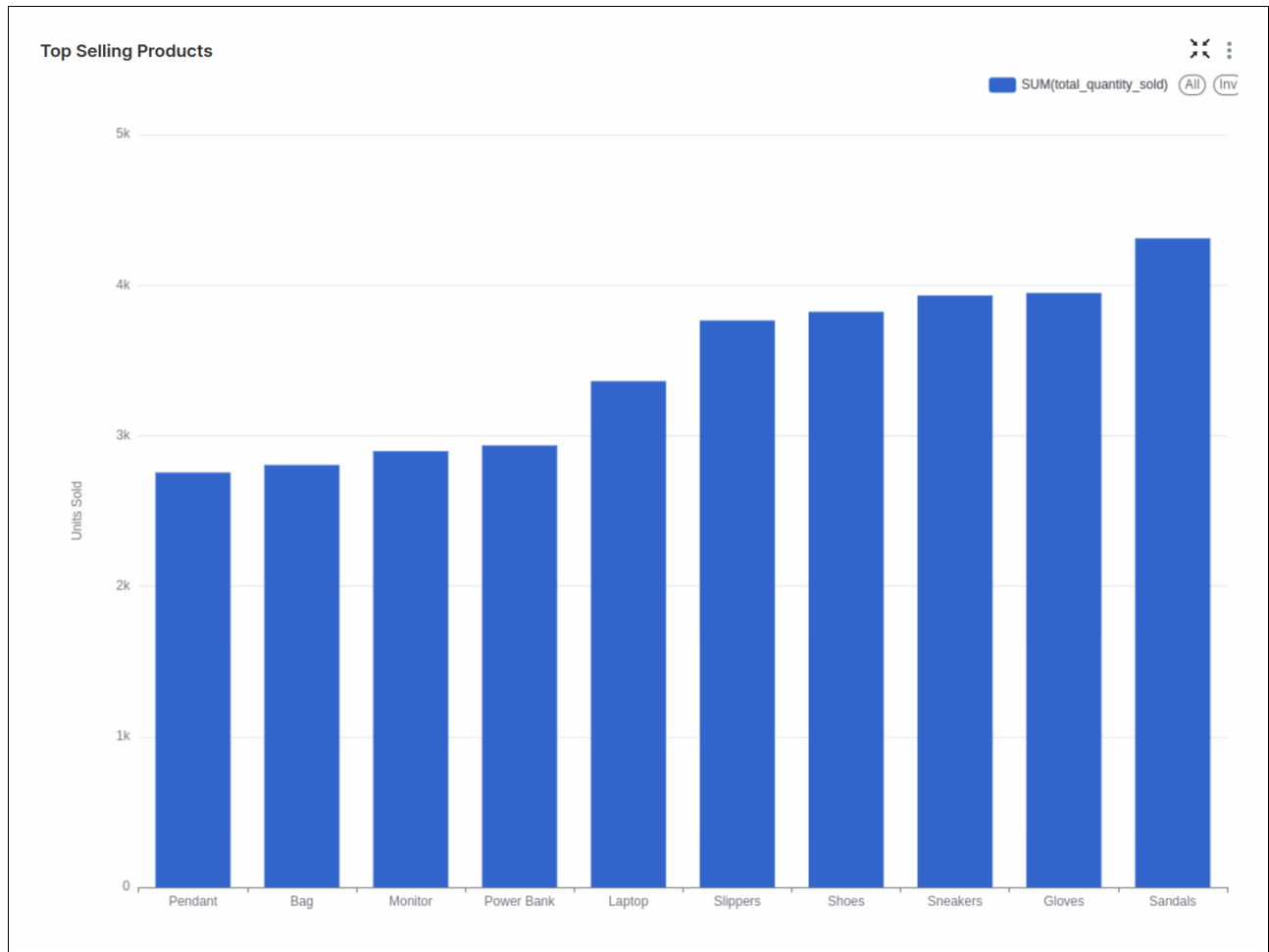


Figure 9-12 Top-selling e-commerce products

6. After you craft a chart that effectively visualizes your findings, click **Save**. You can choose to directly save the chart to a dashboard. In this example, we start building a dedicated e-commerce dashboard that we call “E-commerce Sales Insight” (Figure 9-13).

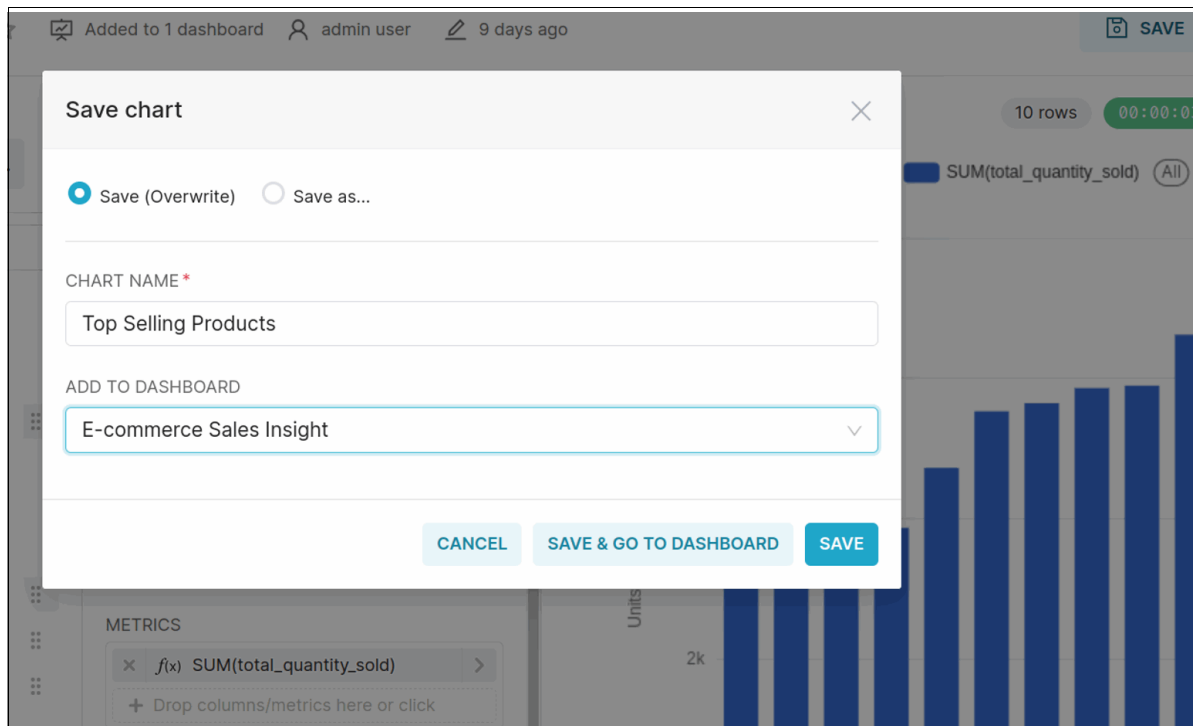


Figure 9-13 Save chart

7. This example demonstrates how to create a single chart in Superset. The next section of the dashboard leverages the same principles to build more charts. These charts delve deeper into your e-commerce data, which provides consumable, high-value data around your e-commerce business (Figure 9-14 on page 171).

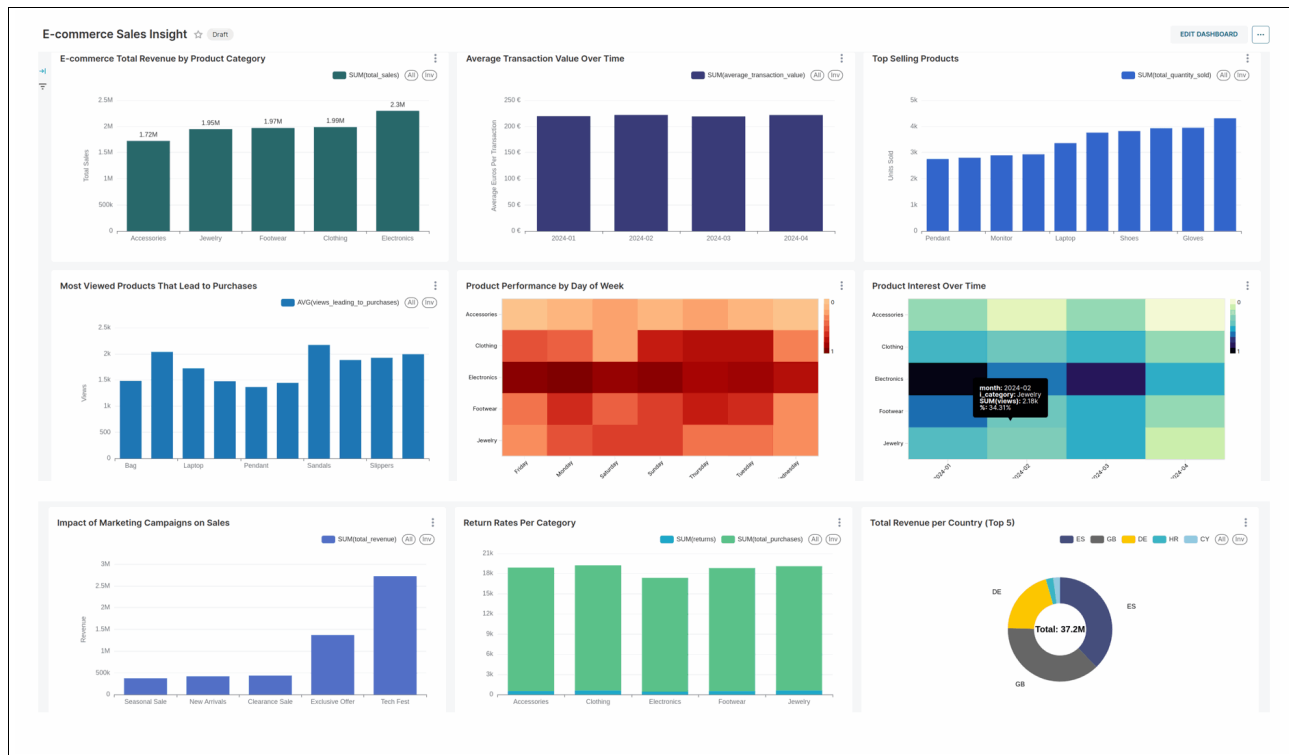


Figure 9-14 Building more charts

- When you are satisfied with the dashboard setup and configuration, move it from draft to published (Figure 9-15).

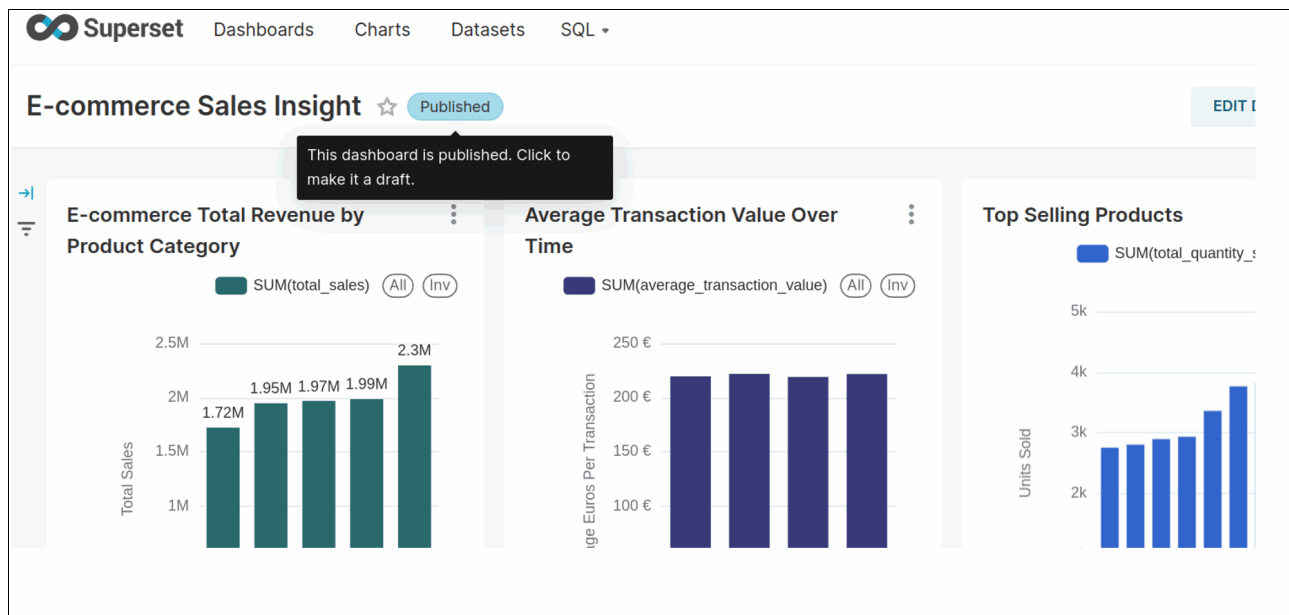


Figure 9-15 Moving the chart from draft to published

Apache Superset implements a robust role-based access control (RBAC) model. This authorization system offers granular control that you can use to define permissions for individual users and roles. It even extends to row-level access for the data sets that you configure, which provides maximum flexibility in data security.

With the Superset RBAC system, you may define permissions for dashboards and data sets, which ensure that users see only the data that is relevant to their roles:

- ▶ **Owner access:** Assign “owner” permissions to users that are responsible for maintaining and updating dashboards.
- ▶ **View access:** Grant “view” access to specific teams that require insights for their business unit.

With this granular control, you can share data securely. For example, the marketing team can view customer behavior data while the finance team sees sales data that is relevant to their department.

For more information about what you can achieve regarding authorization with Apache Superset, see the official [documentation](#).

9.3 Curated point-of-sale SQL queries

Curated point-of-sale SQL queries empower you to extract valuable information from your data quickly and efficiently. This section provides some examples.

9.3.1 IBM watsonx.data Single Sign-On authentication setup

You can bring your own Single Sign-On (SSO) to the watsonx.data platform. The IBM watsonx.data platform supports Lightweight Directory Access Protocol (LDAP), Security Assertion Markup Language (SAML), and OpenID Connect (OIDC) providers that can be integrated with your instance.

In this example, you integrate a Keycloak OIDC to the watsonx.data instance by completing the following steps:

1. In the left pane, click **Identity providers** (Figure 9-16 on page 173).

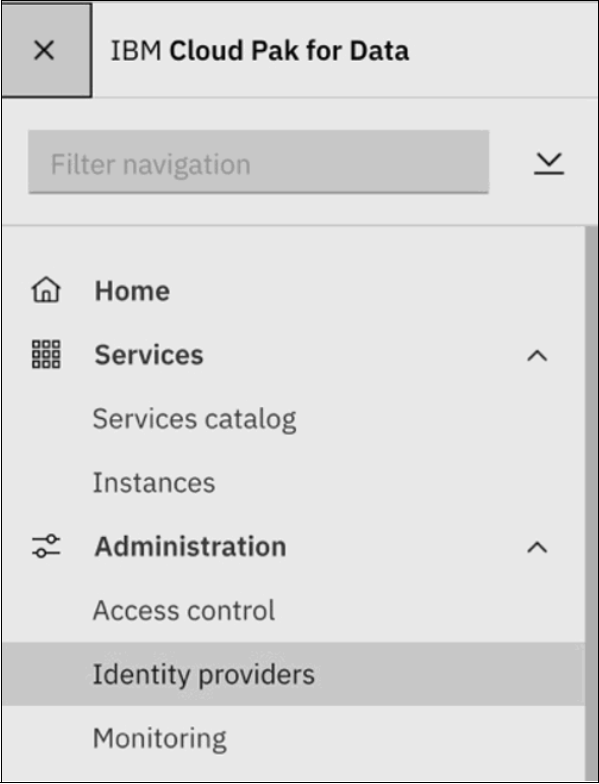


Figure 9-16 Identity providers option

2. Click **New connection** (Figure 9-17).

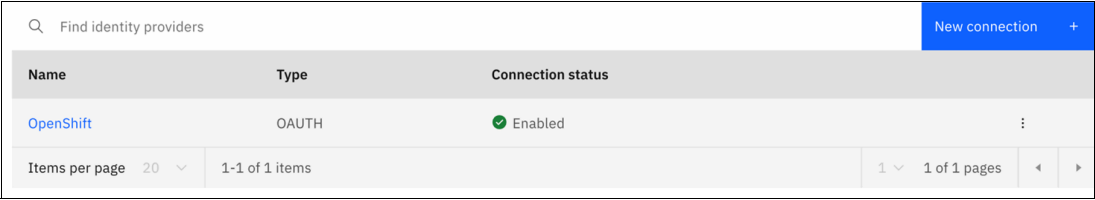
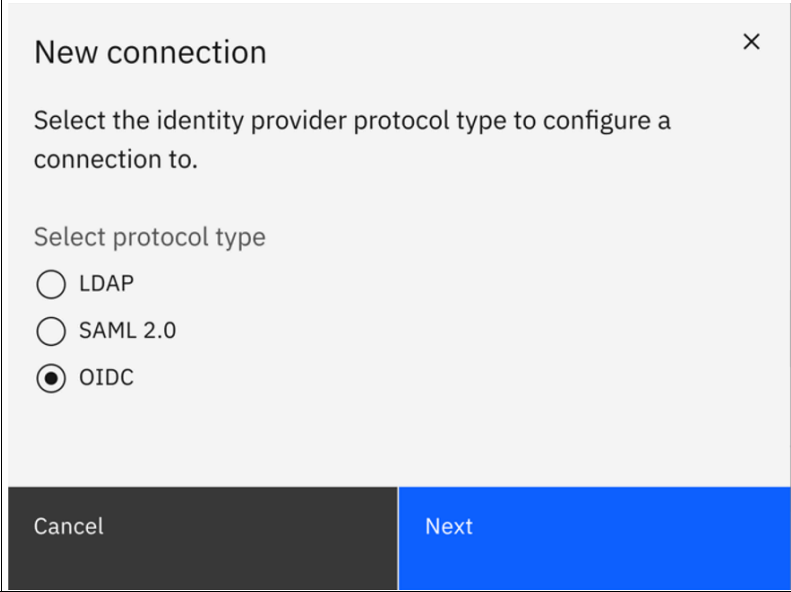


Figure 9-17 New connection

3. You are prompted to choose a protocol. For this example, use **OIDC** (Figure 9-18).

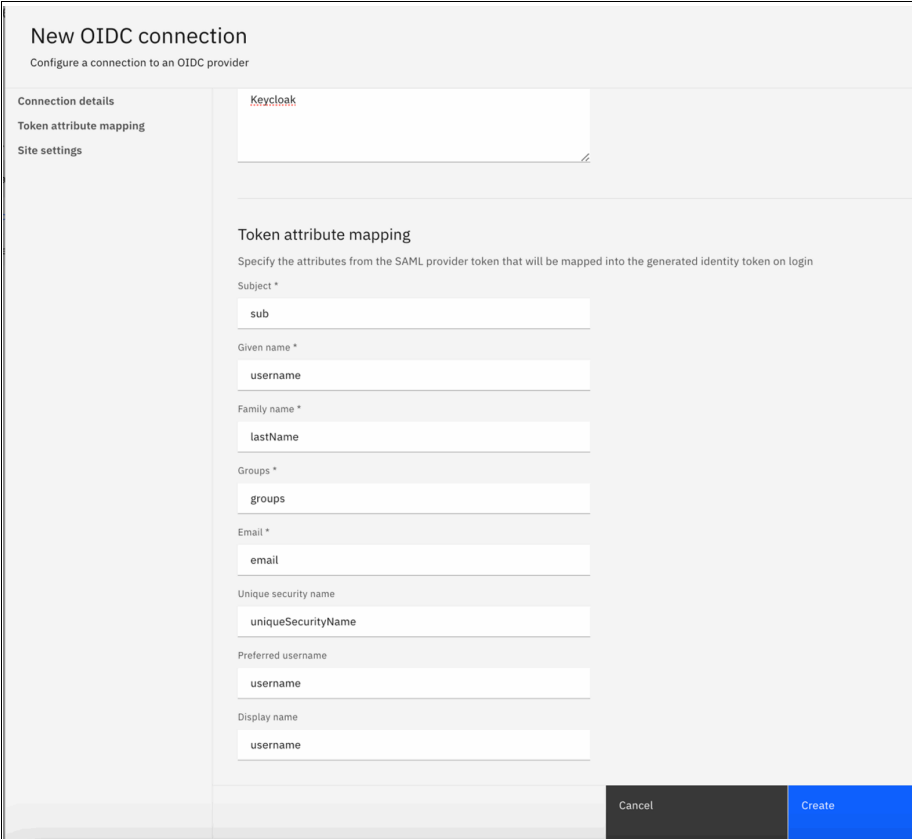


A dialog box titled "New connection" with a close button (X) in the top right corner. The text inside says "Select the identity provider protocol type to configure a connection to." Below this, it says "Select protocol type" followed by three radio button options: "LDAP", "SAML 2.0", and "OIDC". The "OIDC" option is selected with a black dot. At the bottom, there are two buttons: "Cancel" on the left and "Next" on the right.

Figure 9-18 OIDC option

4. Provide the following Keycloak credentials (See Figure 9-19):

- The token details
- The configuration URL



A web form titled "New OIDC connection" with the subtitle "Configure a connection to an OIDC provider". On the left is a sidebar with three tabs: "Connection details", "Token attribute mapping", and "Site settings". The "Token attribute mapping" tab is active. The main content area has a text input field at the top containing "Keycloak". Below this is a section titled "Token attribute mapping" with the instruction "Specify the attributes from the SAML provider token that will be mapped into the generated identity token on login". It contains several labeled input fields: "Subject *" with value "sub", "Given name *" with value "username", "Family name *" with value "lastName", "Groups *" with value "groups", "Email *" with value "email", "Unique security name" with value "uniqueSecurityName", "Preferred username" with value "username", and "Display name" with value "username". At the bottom right are "Cancel" and "Create" buttons.

Figure 9-19 Entering the details

5. Click **Create** to save your Keycloak configuration. The connection with your SSO provider will be verified, and on success, the settings are saved. For more information about identity provider (IDP) configurations in watsonx.data, see [IBM Documentation](#).
6. Add users from your SSO provider directly within watsonx.data. Go to the **Access control** section in the left pane, click **Add User**, and search for users by using your SSO credentials (Figure 9-20).

Add users
Authorize user access to the platform.

☒ Profile information
☐ Platform access
☐ Summary

Profile information
Search for the users you want to add from your connected identity providers.

Search: poyraz

1 result returned

Name	Username	Email
[REDACTED]	[REDACTED]	[REDACTED]@ibm.com Not available

Buttons: Cancel, Back, Next

Figure 9-20 Add users

7. After selecting the user, click **Next** to set the platform access type for the user (Figure 9-21).

Add users
Authorize user access to the platform.

☐ Profile information
☒ Platform access
☐ Roles
☐ Summary

Platform access
Privileges on the platform are controlled by permissions. Users are administered permissions through role assignment. Users can be assigned roles directly or inherit them from groups they are added to.

☒ Assign roles directly
☐ Add to user group

Buttons: Cancel, Back, Next

Figure 9-21 Platform access

8. There are two ways to grant users access in watsonx.data: direct assignment or group membership. In this example, you focus on directly assigning roles to the specified user in this example (Figure 9-22).

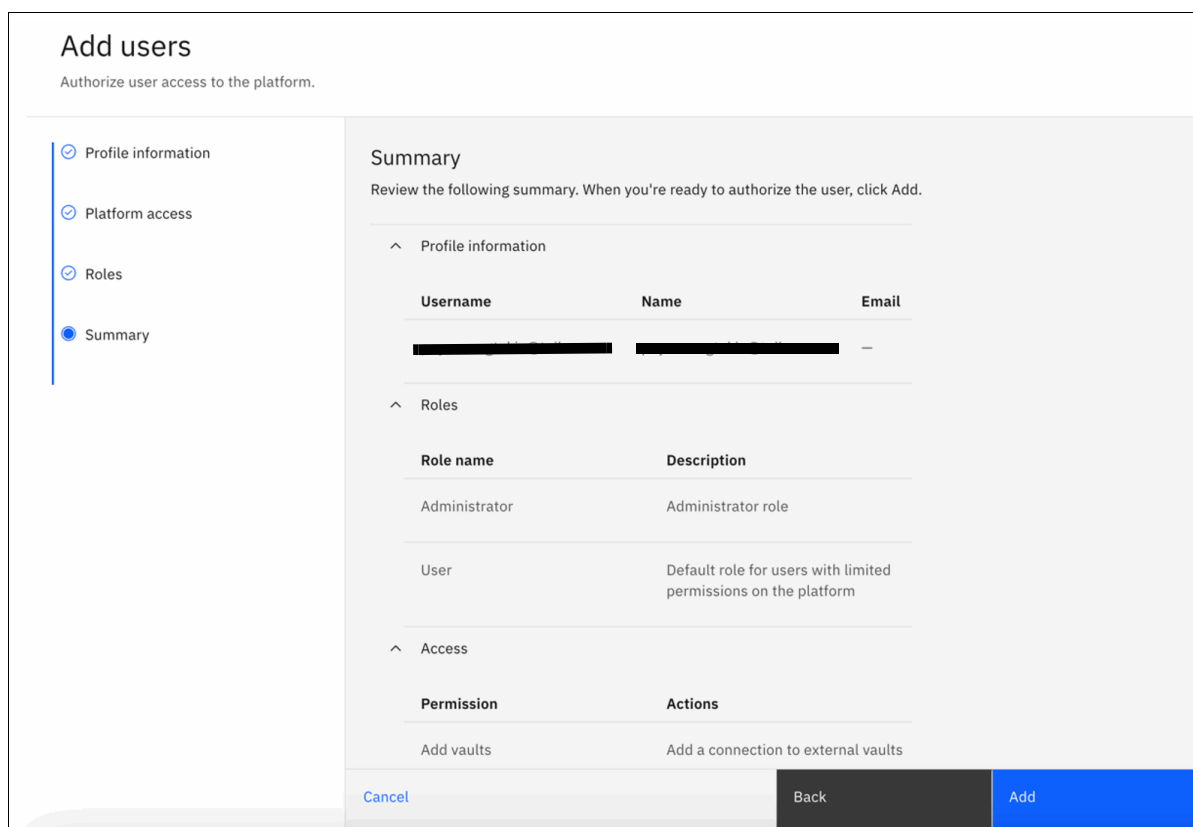
The screenshot shows the 'Add users' interface with the following components:

- Header:** 'Add users' and 'Authorize user access to the platform.'
- Left Sidebar:** A list of tabs: 'Profile information', 'Platform access', 'Roles' (selected), and 'Summary'.
- Role Selection:** Two radio buttons: 'Administrator' (checked) and 'User'.
- Role Details (Administrator):**
 - Description:** Administrator role
 - Modified on:** May 21, 2024 1:36 AM
 - Permissions:** 12 permissions, 90 actions (with an 'Expand all' link)
 - Actions:**
 - Add vaults (dropdown)
 - Administer platform (highlighted with a blue box)
 - Create platform roles
 - Edit platform roles
 - Delete platform roles
 - Configure connection to SMTP server (with an info icon)
 - Configure alert forwarding
- Footer:** 'Cancel', 'Back', and 'Next' buttons.

Figure 9-22 Assigning roles

9. You can assign administrative and user-specific roles to a single user on the watsonx.data platform. You can read more about the user roles in [IBM Documentation](#).

10. Click **Next** to view a summary of the roles that you assigned to your user and then click **Add** to add the user to your watsonx.data instance (Figure 9-23).



Add users

Authorize user access to the platform.

- Profile information
- Platform access
- Roles
- Summary**

Summary

Review the following summary. When you're ready to authorize the user, click Add.

^ Profile information

Username	Name	Email
[REDACTED]	[REDACTED]	—

^ Roles

Role name	Description
Administrator	Administrator role
User	Default role for users with limited permissions on the platform

^ Access

Permission	Actions
Add vaults	Add a connection to external vaults

[Cancel](#)[Back](#)[Add](#)

Figure 9-23 Summary

The added user can now benefit from secure SSO for logging in to the watsonx.data platform UI.

9.3.2 Consuming the curated data from watsonx.data

With IBM watsonx.data, you can run SQL queries with the Presto engine by using the GUI, which means that you can run your SQL queries without needing any additional setup. To do this task, complete the following steps:

1. To run the queries from the GUI, go to the **Query workspace** in the left pane. You see the query window with your data catalogs on the left and the SQL query terminal on the right, along with the results section at the bottom (Figure 9-24).

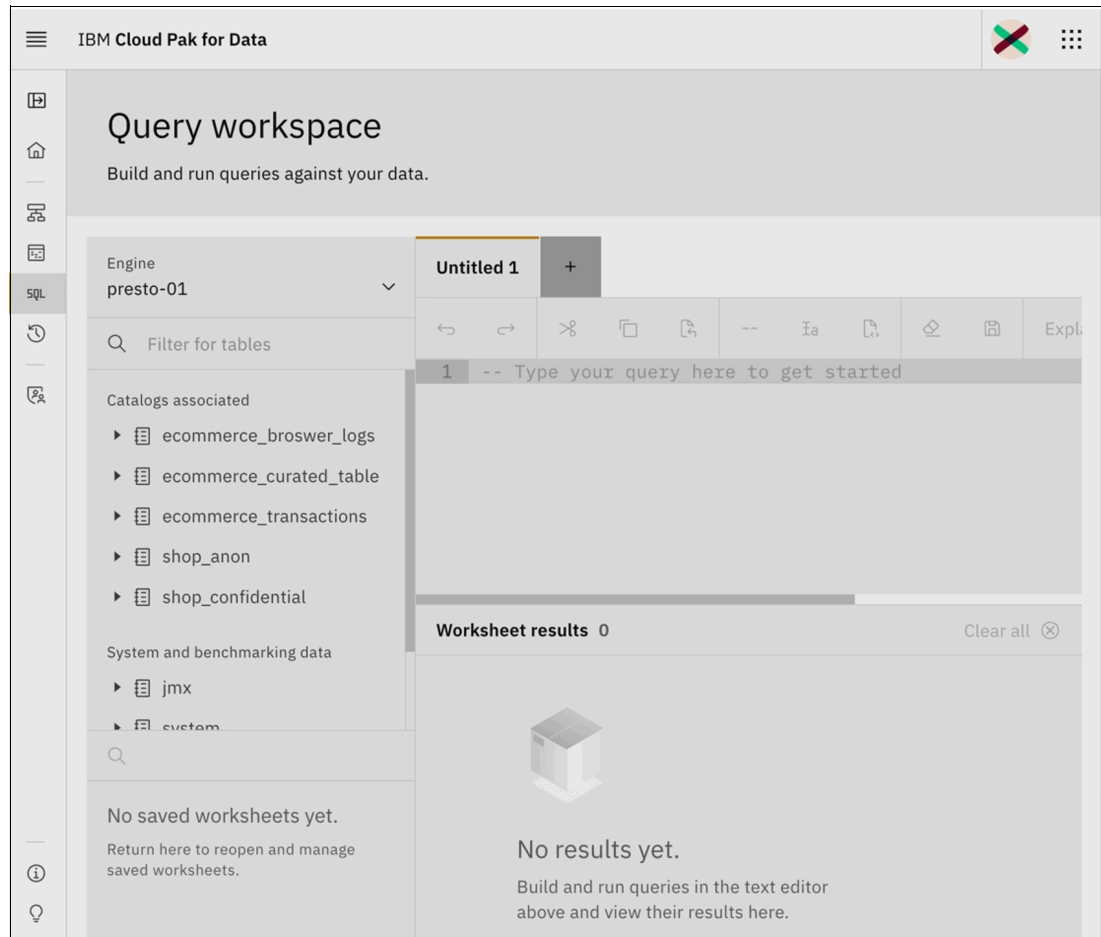


Figure 9-24 Query workspace

2. In the watsonx.data SQL Query workspace, suppose that you want to analyze detailed invoice information and total quantity that is sold per product for each customer. To target a specific table, you need its path. Right-click the correct table and select **Generate path**. This path automatically populates in the SQL query section.

Tip: You can also retrieve paths for individual columns within tables.

Figure 9-25 shows the **Generate path** option.

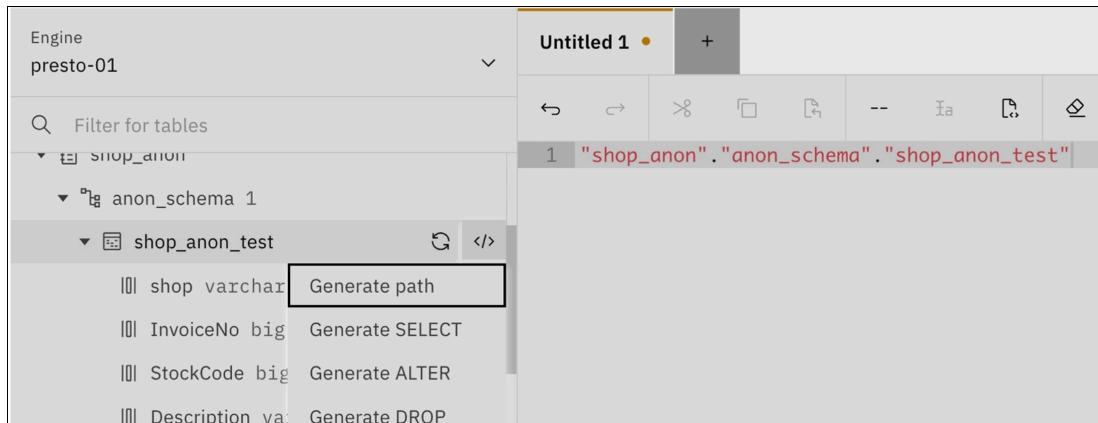


Figure 9-25 *Generate path option*

3. Now that you have your path, you can run the SQL query in Example 9-6 to retrieve the data.

Example 9-6 SQL query

```
SELECT
  shop,
  InvoiceNo,
  StockCode,
  Description,
  Quantity,
  InvoiceDate,
  Price,
  CustomerID,
  Country,
  SSN,
  Email,
  PaymentMethod,
  ProductCategory,
  SUM(Quantity) OVER (PARTITION BY CustomerID, StockCode ORDER BY InvoiceDate)
AS CumulativeQuantitySold
FROM
  "shop_confidential"."confidential_schema"."shop_confidential_test"
ORDER BY
  CustomerID,
  StockCode,
  InvoiceDate;
```

- When the query finishes, the results are displayed. Then, you can export them as a CSV file by clicking **Export to CSV** (Figure 9-26).

Worksheet results 1 Clear all												
Result set		Details Export to CSV										
shop	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	Price	CustomerID	Country	SSN	Email	PaymentMethod	CumulativeQuantitySold
shop4	34407	32052	Jeans	17	02-04-2024 19:41	83	10008	Italy	434-96-7453	aholland@example.org	Credit Card	Clothing
shop2	53677	62512	Socks	6	28-03-2024 06:28	3.76	10009	France	855-55-9148	christopher73@example.org	Cash	Clothing
shop1	27892	22229	Anklet	5	25-03-2024 08:12	11.65	10012	Spain	742-19-7529	adrianwallace@example.com	Credit Card	Jewelry
shop4	32164	49109	Slippers	17	02-04-2024 09:34	89.63	10012	Italy	705-89-9494	ronnie64@example.net	Credit Card	Footwear
shop5	38466	14345	T-shirt	5	28-03-2024 12:26	91.66	10014	Netherlands	326-53-7515	mharris@example.net	Cash	Clothing
shop4	28522	47979	Monitor	4	02-05-2024 14:04	78.64	10022	Italy	585-31-7153	ydavis@example.org	Credit Card	Electronics
shop4	73011	33650	Hoodie	5	02-05-2024 21:10	33.08	10023	Italy	428-58-2683	buckvincent@example.com	Cash	Clothing
shop5	39012	29098	Scarf	6	29-03-2024 16:56	35.14	10027	Netherlands	671-28-7041	warrentracy@example.net	Cash	Clothing
shop3	32649	67041	Sneakers	9	28-03-2024 15:00	34.01	10027	Germany	357-32-6864	ryanroberts@example.org	Credit Card	Footwear
shop4	43940	30414	Chain	17	02-05-2024 17:00	12.19	10029	Italy	443-73-9558	vcoleman@example.com	Cash	Jewelry

Figure 9-26 Worksheet results

9.3.3 Visual Explain

You can see the visual representation of the execution plan for your queries by clicking **Explain** to the left of **Run on Presto**.

With the **Explain** option, you can see the steps of your SQL query along with the resources that are required for each step (Figure 9-27).

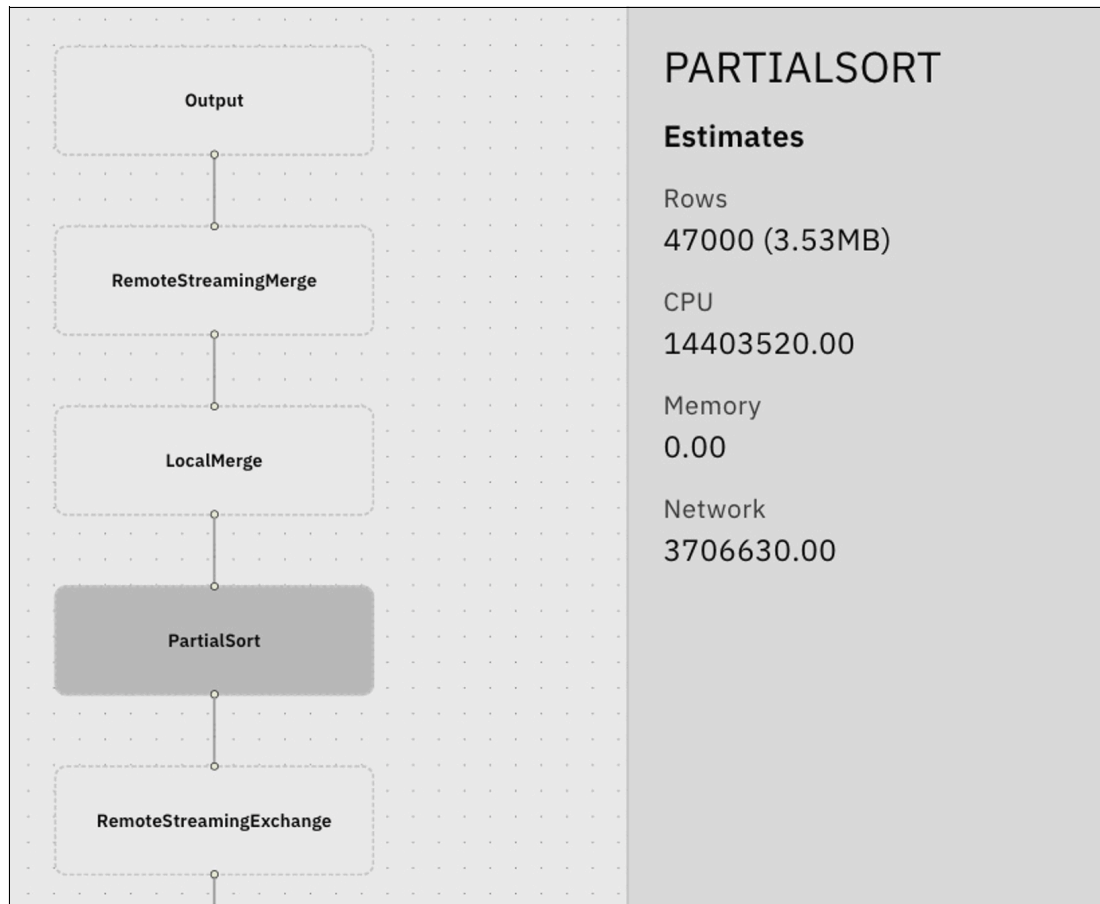


Figure 9-27 Visual Explain

For example, you can preview the estimated costs that are associated with the PartialSort operation when you run the SQL query against the Presto engine. IBM watsonx.data uses an **EXPLAIN SQL** statement on the query to create the corresponding graph. This tool can be used to analyze and improve the efficiency of your queries.

Alternatively, you can run the SQL queries directly by using the Presto CLI. To do so, you must have the Presto CLI installed on your environment.

To run the commands by using the Presto CLI, first connect to your Presto engine. An example command is shown in Example 9-7.

Example 9-7 Connecting to your Presto engine

```
[root@localhost wxd]# ./presto --server
https://ibm-lh-lakehouse-presto-01-presto-svc-cpd-operands.ocp-eu-redbook-ce869a54
4b369f1dfd24beec10027762-i000.eu-es.containers.appdomain.cloud --user cpadmin
--password
Password:
```

After you log in to Presto by using the CLI and provide your credentials, you can directly run SQL queries. For example, in Example 9-8, you explore a query that identifies the most popular payment methods by revenue for each product category.

Example 9-8 SQL query to determine the most popular payment methods

```
presto> SELECT
->     ProductCategory,
->     PaymentMethod,
->     TotalRevenue
-> FROM (
->     SELECT
->         ProductCategory,
->         PaymentMethod,
->         SUM(Quantity * Price) AS TotalRevenue,
->         RANK() OVER (PARTITION BY ProductCategory ORDER BY SUM(Quantity * Price)
DESC) AS RevenueRank
->     FROM
->         "shop_anon"."anon_schema"."shop_anon_test"
->     GROUP BY
->         ProductCategory,
->         PaymentMethod
-> ) ranked_methods
-> WHERE
->     RevenueRank = 1;
```

ProductCategory	PaymentMethod	TotalRevenue
Electronics	Cash	845072.8099999998
Jewelry	Credit Card	771207.8899999998
Accessories	Credit Card	771762.2500000002
Clothing	Cash	809484.77
Footwear	Cash	832541.4200000009

(5 rows)

```
Query 20240522_132030_00156_ut8z2, FINISHED, 1 node
Splits: 83 total, 83 done (100.00%)
[Latency: client-side: 0:02, server-side: 0:01] [15K rows, 93.8KB] [10.5K rows/s, 65.5KB/s]
```




A

Configuring RADOS Gateway by using the IBM Storage Ceph GUI

This appendix describes how to configure the RADOS Gateway (RGW) service within an IBM Storage Ceph cluster by using its GUI.

This appendix has the following sections:

- ▶ Configuring RADOS Gateway by using the IBM Storage Ceph GUI
- ▶ RADOS Gateway user creation
- ▶ Bucket creation

Configuring RADOS Gateway by using the IBM Storage Ceph GUI

The IBM Storage Ceph GUI provides a simple interface for creating and configuring RGW services. To begin, click **Object Gateway** in the left pane.

RADOS Gateway user creation

To create an RGW user, complete the following steps:

1. In the left pane, select **Object Gateway** → **Users**. Here, you can view existing users, create ones, and modify their access permissions (Figure A-1).

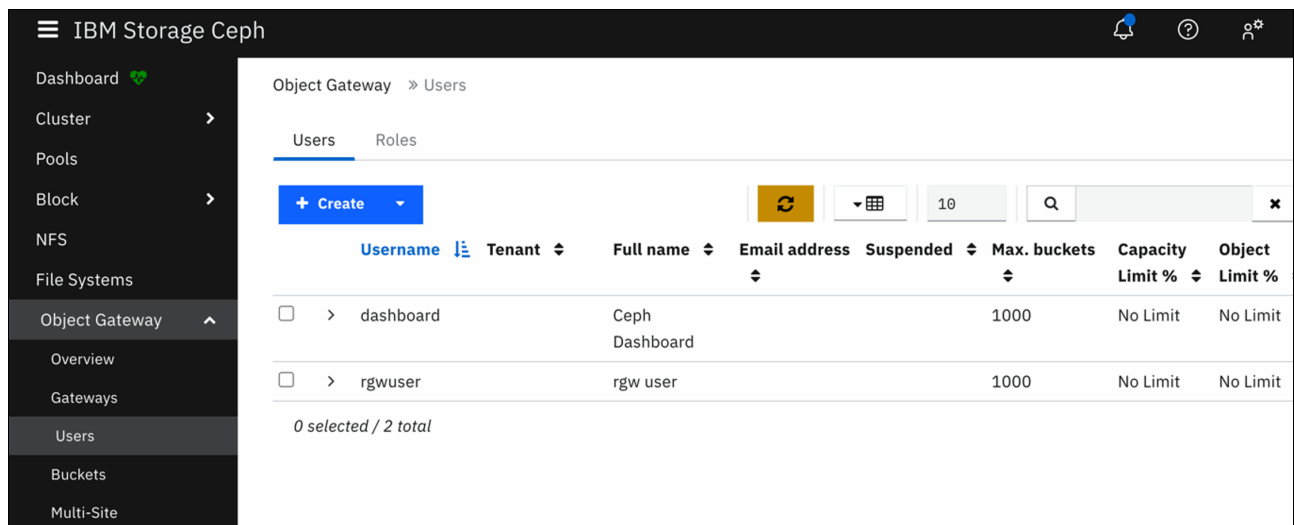


Figure A-1 Users option on the Object Gateway interface

2. To create an RGW user, click **Create** and complete the required information (Figure A-2). You can create a user with two options:
- Auto-generate secure Amazon Simple Storage Service (S3) credentials for the user.
 - Set storage quotas to limit their data usage.

Create User

User ID *

confidential

✓

☐ Show Tenant

Full name *

Confidential User

✓

Email address

confidential_user@cephlabs.com

✓

Max. buckets

Custom

▼

1000

✓

☐ Suspended ?

S3 key

☒ Auto-generate key

User quota

☐ Enabled

Bucket quota

☐ Enabled

Cancel

Create User

Figure A-2 Creating an RGW user

Bucket creation

You can create buckets and assign owners to these buckets by using the IBM Storage Ceph GUI. To create a bucket, select **Buckets** → **Create** in the left pane. You can lock and encrypt your bucket by using this form. An example of creating the confidential bucket by using the GUI is shown in Figure A-3.

Create Bucket

Name * confidential ✓

Owner * confidential ▼

Placement target * default-placement (pool: default.rgw.buckets.data) ▼

Locking

☐ Enabled ?

Security

☐ Encryption ?

Cancel Create Bucket

Figure A-3 Create Bucket



Configuring the command-line tools for IBM watsonx.data

This appendix guides you through configuring the **ibm-lh-client** utility to manage IBM watsonx.data from the command-line interface (CLI).

This appendix has the following sections:

- ▶ Configuring command-line tools for watsonx.data
- ▶ Installing the ibm-lh-client utility
- ▶ Setting up the ibm-lh utility
- ▶ Creating a schema
- ▶ Creating a table and ingesting data

Configuring command-line tools for watsonx.data

To interact with watsonx.data from the CLI, you need the **ibm-lh-client** utility. However, before installing it, ensure that your environment meets the prerequisite package requirements. The **ibm-lh-client** utility can be installed on the platforms that are listed in Table B-1.

Table B-1 The ibm-lh-client utility

Operating system (OS)	x86-64	Docker, Podman, or Colima installation instructions
Linux	Yes	Docker or Podman
Windows	Yes	Docker or Podman
MacOS x86	Yes	Docker or Podman
Mac with Apple Silicon with Rosetta Emulation	No	Docker or Colima

Installing the ibm-lh-client utility

To install the **ibm-lh-client** utility, complete the following steps:

1. Create an installation directory and browse to it (Example B-1).

Example: B-1 Creating an installation directory and browsing to it

```
mkdir Documents/wxd
cd Documents/wxd
```

2. Set up the environment variables, as shown in Example B-2.

Example: B-2 Setting up the environment variables

```
export LH_ROOT_DIR=Documents/wxd
export LH_RELEASE_TAG=latest
export
IBM_LH_TOOLBOX=cp.icr.io/cpopen/watsonx-data/ibm-lakehouse-toolbox:$LH_RELEASE_TAG
export LH_REGISTRY=cp.icr.io/cp/watsonx-data
export PROD_USER=cp
export IBM_ENTITLEMENT_KEY= <Your IBM Entitlement Key>
export IBM_ICR_IO=cp.icr.io
```

Example B-3 on page 189 shows the internal tag (represented by `$LH_RELEASE_TAG`) that is associated with each corresponding watsonx.data version.

Example: B-3 Release tags with Cloud Pak for Data versions

Cloud Pak for Data version	Service instance version
4.8.5	1.1.4 (latest)
4.8.4	1.1.3
4.8.3	1.1.2
4.8.1	1.1.1
4.8.0	1.1.0

3. If Docker is installed in your environment, run the following command to set an environment variable that is named **DOCKER_EXE** with the value docker:

```
export DOCKER_EXE=docker
```

4. Then, pull the watsonx.data client package and copy it into your environment (Example B-4).

Example: B-4 The watsonx.data client package

```
$DOCKER_EXE pull $IBM_LH_TOOLBOX
id=$(($DOCKER_EXE create $IBM_LH_TOOLBOX)
$DOCKER_EXE cp $id:/opt - > /tmp/pkg.tar
$DOCKER_EXE rm $id
id=
```

5. Extract the watsonx.data client pkg.tar file into the /tmp directory and compare the checksum value with the bom.txt file (Example B-5).

Example: B-5 The watsonx.data client pkg.tar file

```
tar -xf /tmp/pkg.tar -C /tmp
cat /tmp/opt/bom.txt
cksum /tmp/opt/*/*
tar -xf /tmp/opt/client/ibm-lh-client-*.tgz -C $LH_ROOT_DIR
```

6. Authenticate Docker with the IBM Container Registry (Example B-6).

Example: B-6 Authenticating Docker with the IBM Container Registry

```
$DOCKER_EXE login ${IBM_ICR_IO} \
--username=${PROD_USER} \
--password=${IBM_ENTITLEMENT_KEY}
```

7. Run the setup script:

```
$LH_ROOT_DIR/ibm-lh-client/bin/setup --license_acceptance=y
--runtime=$DOCKER_EXE
```

Setting up the ibm-lh utility

All **ibm-lh** related commands should be run from the `bin` folder of the installation directory. This directory is defined as `/Documents/wxd/` for this demonstration.

To configure your **ibm-lh** utility, go to your installation folder and start with adding the username and password for authentication (Example B-7).

Example: B-7 Adding the username and password for authentication

```
connect-lh --op=add \  
--name=wxd \  
--host=ibm-lh-lakehouse-presto-01-presto-svc.cpd-operands.svc.cluster.local \  
--port=8443 \  
--username=cpadmin \  
--password=<Your watsonx.data instance user password>
```

If you are using custom certificates, you can import your watsonx.data certificates into your **ibm-lh** tools by using the following command:

```
./cert-mgmt --op=add --host=<hostname> --port=<port>
```

To add your Cloud Pak for Data instance connection to your **ibm-lh** utility, run the following command:

```
./ibm-lh config add_cpd --name wxd --host  
https://ibm-lh-lakehouse-presto-01-presto-svc-cpd-operands.ocp-eu.eu-es.containers  
.appdomain.cloud --port 8443
```

Note: The host address is the public URL of your watsonx.data instance.

If you are running the developer edition of watsonx.data, you can replace `add_cpd` with `add_dev` in this command.

You can connect to your Presto instance scheme by running the following command:

```
./presto-cli --server <your instance URL> --user cpadmin --schema  
confidential_schema --catalog shop_confidential
```

Creating a schema

To create a schema, complete the following steps:

1. To create a schema on your watsonx.data instance, first log in by using **presto-cli** (Example B-8).

Example: B-8 Logging in by using presto-cli

```
./presto-cli --server  
https://ibm-lh-lakehouse-presto-01-presto-svc-cpd-operands.ocp-eu-redbook-ce869a54  
4b369f1dfd24beec10027762-i000.eu-es.containers.appdomain.cloud --user cpadmin  
--catalog shop_anon
```

2. After running this command, you are prompted to provide the cpadmin user's password. Then, you see the presto prompt:

```
presto>
```

3. Now, you can create your schema by using the Presto CLI. In this example, create the anonymized bucket's schema by using the CLI (Example B-9).

Example: B-9 Creating your schema by using the presto CLI

```
presto> create schema anon_schema with (location='s3a://anonymized/anon_schema');  
CREATE SCHEMA <-- This means that your schema has been created  
presto>
```

4. You can verify schema creation by asking Presto to show the created schemas (Example B-10).

Example: B-10 Showing the schemas

```
presto> show schemas;  
Schema  
-----  
anon_schema  
(1 row)  
  
Query 20240509_134258_00090_ut8z2, FINISHED, 1 node  
Splits: 19 total, 19 done (100.00%)  
[Latency: client-side: 0:01, server-side: 412ms] [1 rows, 16B] [2 rows/s, 38B/s]
```

Creating a table and ingesting data

To create a table inside your schema by using the presto CLI, you manually define the column names instead of importing a table from the GUI.

Example B-11 shows an example command to create a table.

Example: B-11 Creating a table

```
presto> use shop_anon.anon_schema;  
  
presto:anon_schema> CREATE table anon_table(shop varchar, InvoiceNo integer,  
StockCode integer, Description varchar, Quantity integer, InvoiceDate varchar,  
Price real, CustomerID integer, Country varchar, PaymentMethod varchar,  
ProductCategory varchar);
```

Define your Amazon Simple Storage Service (S3) credentials as environment variables before ingesting the data into your Presto engine. You can use the command in Example B-12 to define these variables.

Example: B-12 Defining your S3 credentials as environment variables

```
export  
SOURCE_S3_CREDS="AWS_ACCESS_KEY_ID=xxxxxxx,AWS_SECRET_ACCESS_KEY=yyyyyyyy,ENDPOINT  
_URL=https://s3.cephlabs.com,AWS_REGION=EU_ES,BUCKET_NAME=anonymized"
```



Additional material

This book refers to additional material that can be downloaded from the internet as described in the following sections.

Locating the GitHub material

The web material that is associated with this book is available in softcopy on the internet from the IBM Redbooks GitHub web page:

<https://github.com/IBMRedbooks/IBM-Storage-Ceph-as-a-Data-Lakehouse-platform-for-IBM-watsonx.data-and-Beyond>

Cloning the GitHub material

To clone the GitHub repository for this book, complete the following steps:

1. Download and install the **git** client (if it is not installed) from [Git](#).
2. Clone the GitHub repository by running the following command:

<https://github.com/IBMRedbooks/IBM-Storage-Ceph-as-a-Data-Lakehouse-platform-for-IBM-watsonx.data-and-Beyond.git>

Abbreviations and acronyms

ABAC	attribute-based access control	ITIL	Information Technology Infrastructure Library
ACID	atomicity, consistency, isolation, and durability	JDBC	Java Database Connectivity
AD	Active Directory	JSON	JavaScript Object Notation
AI	artificial intelligence	JWT	JSON Web Token
ASF	Apache Software Foundation	KMIP	Key Management Interoperability Protocol
AWS	Amazon Web Services	KMS	Key Management Service
BI	business intelligence	LDAP	Lightweight Directory Access Protocol
CA	certificate authority	MDS	Metadata Server
CephFS	Ceph Filesystem	MFA	multi-factor authentication
CFTC	Commodity Futures Trading Commission	ML	machine learning
CIDR	classless inter-domain routing	NFS	Network File System
CP4D	Cloud Pak for Data	NL-SAS	near-line SAS
CR	Custom Resource	NVMe	Non-Volatile Memory Express
CRD	custom resource definition	OIDC	OpenID Connect
CRUSH	Controlled Replication Under Scalable Hashing	ORC	Optimized Row Columnar
D3N	Datacenter-Data-Delivery Network	OS	operating system
DR	disaster recovery	OSD	object storage daemon
DRAM	dynamic random access memory	PCI-DSS	Payment Card Industry Data Security Standard
EBS	Elastic Block Store	PII	personally identifiable information
EC	erasure coded	PIN	personal identification number
EMR	Elastic MapReduce	POSIX	Portable Operating System Interface
EOL	end of life	QAT	Intel QuickAssist Technology
ETL	extract, transform, and load	RADOS	Reliable Autonomic Distributed Object Store
FINRA	Financial Industry Regulatory Authority	RAG	Retrieval-Augmented Generation
FIPS	Federal Information Processing Standards	RBAC	role-based access control
GA	general availability	RBD	RADOS Block Device
GCP	Google Cloud Platform	RDBMS	relational database management system
GDPR	General Data Protection Regulation	RGW	RADOS Gateway
GKLM	IBM Guardium Key Lifecycle Manager	RHEL	Red Hat Enterprise Linux
HDD	hard disk drive	RHSSO	Red Hat Single Sign-On
HDFS	Hadoop Distributed File System	RPO	recovery point objective
HIPAA	Health Insurance Portability and Accountability Act	S3	Simple Storage Service
HSM	Hardware Security Modules	S3A	Amazon S3A File System
IAM	Identity and Access Management	SAML	Security Assertion Markup Language
IDP	identity provider		

SDK	software development kit
SEC	US Securities and Exchange Commission
SNS	Simple Notification Service
SSD	solid-state drive
SSE	server-side encryption
SSE-C	Server-Side Encryption with Customer-Provided Keys
SSE-KMS	Server-Side Encryption with Amazon Key Management Service
SSE-S3	Server-Side Encryption with Amazon S3 Managed Keys
SSO	Single Sign-On
STS	Security Token Service
TCO	total cost of ownership
TLS	Transport Layer Security
TOTP	Time-based One-Time Password
WORM	Write Once Read Many

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 book.

IBM Redbooks

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

- ▶ *IBM Storage Ceph Concepts and Architecture Guide*, REDP-5721
- ▶ *IBM Storage Ceph Solutions Guide*, REDP-5715

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

Online resources

These websites are also relevant as further information sources:

- ▶ Community Ceph Documentation
<https://docs.ceph.com/en/latest/monitoring/>
- ▶ IBM Storage Ceph Documentation
https://www.ibm.com/docs/en/storage-ceph/7?topic=SSEG27_7/installation/con_core_ibm-storage-ceph.htm
- ▶ IBM watsonx.data Documentation
<https://www.ibm.com/docs/en/watsonx/watsonxdata/1.1.x?topic=watsonxdata-introduction>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



SG24-8563-00

ISBN 0738461539

Printed in U.S.A.

Get connected

