Draft Document for Review June 3, 2024 6:36 pm SG24-8563-00

Redbooks
ibm.com/redbooks

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 Redbooks

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

June 2024

SG24-8563-00

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

First Edition (June 2024)

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

This document was created or updated on May 31, 2024.

Contents

Figures

Tables

Examples

Notices

Trademarks

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 Ceph offers high-performance, scalable object storage that can handle large datasets efficiently. IBM watsonx.data and 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 IBM Storage Ceph's robust storage capabilities and IBM watsonx.data's 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 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 created for the book's scenarios are available on 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 on the IBM Storage Ceph architecture and technology that is behind the product, see the *IBM Storage Ceph Concepts and Architecture Guide*, REDP-5721 IBM Redpaper.

Authors

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

Daniel Parkes brings years of experience as an infrastructure enthusiast to the IBM Storage Ceph Product Management team. As a passionate open-source advocate with a keen eye for innovation, his focus lies on the IBM Storage Ceph Object Storage offering, and he actively participates in technical discussions to ensure customers achieve digital excellence with Ceph.

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

Franck Malterre is an information technology professional with a background of over 25 years in designing, implementing, andmaintaining 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 Ceph live demonstrations and proof of concept for IBMers and 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, Ceph, and Red Hat OpenShift. He takes part in the Upstream Community and downstream versions of 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 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 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 Ceph based offerings at IBM. He has spent a decade architecting Ceph based storage clusters and solutions. His current interests lie at the intersection of distributed storage, containers, data intensive applications, and machine learning.

Poyraz Sagtekin joined IBM in 2016 after completion of his Computer Enginering degree from Middle East Technical University. Since then, he has developed expertise on diverse areas including IoT systems, IBM Cloud, Kubernetes, RedHat Enterprise Linux, and IBM Power Systems. Currently, he works as a Hybrid Cloud Services Consultant in IBM Systems Expert Labs in Türkiye and he is an IBM recognized speaker and educator.

Tushar Agrawal leads a team of Customer Success Architects is the US National Market to drive adoption of Hybrid Cloud and Artificial Intelligence platforms provided by IBM. Tushar is passionate about integration of AI to augment our day-to-day lives to bring operational efficiency. Tushar focuses on ethical AI principles and collaborates with his customers to develop innovative solutions to address business transformation challenges. Tushar 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, an MBA degree and 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
IBM USA

Kelly Schlamb
IBM Canada

Shrinivas Kulkarni
IBM India

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

1

Introduction

This chapter dives into the origin story of IBM Storage Ceph, a Software-Defined Storage solution. We will explore the fundamental architectural concepts that underpin its functionality.

This chapter has the following sections:

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 called the Metadata Server that must be queried before accessing a file or a directory. Under heavy load, this Metadata Server 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, named CRUSH (Controlled Replication Under Scalable Hashing), leverages a sophisticated calculation to optimally place and redistribute data across the cluster, minimizing data movement when the cluster's state changes, all without the need for any additional metadata.

It is designed to distribute the data across all devices in the cluster to avoid the classic bias of favoring empty devices to write new data. This eliminates the potential bottleneck caused in other systems where empty devices get overloaded with new writes due to capacity balancing.

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

To enhance the data distribution, the solution was designed to allow the breakdown of large elements (for example, 100 GiB file) into smaller elements, each assigned a specific placement via the CRUSH algorithm. Therefore, reading a large file will leverage multiple physical disk drives rather than a single disk drive.

Sage Weil prototyped the new algorithm and doing so 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 it harbors given Santa Cruz, CA is a Pacific Ocean coastal town.

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 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	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, identified by an * with no corresponding downstream product are considered Ceph development versions with a short lifespan and are 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	GA	EOL	Extended Support 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
Red Hat Ceph Storage 1.3		2018-06-30	N/A

Product	GA	EOL	Extended Support Available until
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

Over the many years persistent storage has existed, it has always faced the same basic challenges:

1.3.1 Data keeps growing

As Information Technology has evolved, it moved from character-based data (text) to multimedia data (images, videos, audio). This not only accelerated data growth but also democratized data creation, allowing for data to be generated from various devices and applications. Consequently, the traditional concept of structured data has become increasingly complex.

1.3.2 Technology changes

As Information Technology lifecycle accelerates, the steady centralized datacenter has morphed into a more decentralized organization relying on application and server communication like never before when all Central Processing Units were in the same room with passive terminals requiring access to its processing capabilities.

1.3.3 Data organization, access and costs

Historically, data resided in centralized repositories using 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, utilized by diverse users and devices, its true value lies in the ability to analyze vast, multifaceted datasets. This requires processing information from multiple sources and formats, often through advanced techniques like Artificial Intelligence and Machine Learning. The key to unlocking this value hinges on real-time data accessibility, regardless of location, and free from human intervention for speed and efficiency.

Ceph addresses the challenges of modern data storage head-on. Designed from the ground up for high availability, it eliminates single points of failure. Ceph boasts exceptional scalability, allowing you to 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 the above challenges, Ceph relies on a layered architecture with an object store at its core. This core, known as the *Reliable Autonomic Object Store*, or *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 via a virtual block device solution known as *RBD (RADOS Block Device)*.
- File storage via a POSIX compliant shared filesystem solution known as *CephFS* or *Ceph Filesystem*.
- Object storage via a OpenStack Swift and Amazon S3 compatible gateway known as the *RADOS Gateway* or *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 provides the ability to created segmented pools of storage at the node level.

The different types of storage are identified and implemented as Ceph access methods on top of 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, Python wrapper for librados or librbd).

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

1.4 What is new in IBM Storage Ceph 7.0

On December 8th, 2023, IBM announced the general availability (GA) of IBM Storage Ceph Version 7.0. This major release marks a significant upgrade, migrating the upstream codebase from Quincy (used in IBM Storage Ceph 6) to Reef. This shift promises enhanced performance and stability for your storage system. Additionally, Ceph 7.0 introduces a range of new features and improvements, which we'll explore in detail below.

1.4.1 New features

This section delves into the some of the new features introduced with IBM Storage Ceph version 7.0. Refer to IBM Storage Ceph Version 7.0 Release Notes for more details.

Note: This book is based on IBM Storage Ceph version 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 allows IBM Product Managers to make data-driven decisions that benefit customers, while you gain a centralized view of all your deployed clusters through a user-friendly interface.

NFS over CephFS

This new release returns the ability to share data between CephFS native and Network File System (NFS) clients.

This provides non CephFS capable servers access to the data stored in the Ceph File System through new NFS export capabilities 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 can be configured via the integrated IBM Storage Ceph dashboard user interface.

WORM certification

Cohasset Associates, Inc. evaluated IBM Storage Ceph Object Lock against electronic records requirements mandated by various regulatory bodies. They concluded that IBM Storage Ceph, when configured and utilized with Object Lock, adheres to the electronic record keeping system stipulations of SEC and FINRA rules, as listed below:

- SEC 17a-4(f)
- SEC 18a-6(e)
- FINRA 4511(c)
- CFTC 1.31(c)-(d)

Note: U.S. Securities and Exchange Commission (SEC) stipulates record keeping requirements, including retention periods. Financial Industry Regulatory Authority (FINRA) rules regulate member brokerage firms and exchange member markets.

See the Cohasset certification for more information.

Object Storage Gateway

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

- Configure RADOS Gateway multi-site through the dashboard user interface.

- Multi-site asynchronous replication status and monitoring.
- Object level interaction through the dashboard user interface (For example, add or remove object).

CephFS

Improved health and performance monitoring of the Ceph File System together with a better capacity utilization tracking.

CephFS volume management is now integrated with the IBM Storage Ceph dashboard user interface:

- Create snapshot.
- Delete snapshot.
- Snapshot access and encryption settings.
- Snapshot management.

NVMe over Fabric (NVMe-oF) - Technology preview

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

Archive zone - Technology preview

The S3 object store deployed via the IBM Storage Ceph RADOS Gateway 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 functionality can prove itself 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. See 3.6.1, “Archive zone” on page 44 for more information on the archive zone feature.

NFS over S3 - Technology preview

An S3 bucket can be access using a Network File System (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 the some of the enhancements included in IBM Storage Ceph version 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.

IBM recommends your cluster to be sized with one extra node above the protection scheme to allow normal cluster healing and data recovery (For example, 2+2 Erasure Coding data protection is recommended with 5 IBM Storage Ceph nodes).

Object Storage Gateway

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

- CSV
- JSON
- Parquet

The new feature *Datacenter-Data-Delivery Network (D3N)* uses high-speed storage such as NVMe flash or DRAM to cache datasets 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 RADOS Object Storage Gateway multi-site asynchronous replication to reduce inter-site replication traffic and potentially reducing your Recovery Point Objective (RPO) by limiting the amount of data being replicated hence focusing on the critical data.

This version introduces *policy-based archive and migration strategies* into AWS S3 and AWS compatible S3 endpoints such as IBM Storage Ceph and IBM Cloud Object Storage. These policies can be managed by the end user via standard S3 bucket policies and follow your cost or security requirements or needs.

A *better parallelism* is introduced with this new version to provides faster asynchronous replication in Object Storage Gateway multi-site setups.

Security

TLS is now 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, Alert Manager as well as the Node Exporter.

Flexibility

Placing a host is maintenance mode is now protected via a flag to prevent placing a host in maintenance mode by mistake, causing all the Ceph daemons on this node to be stopped inadvertently.

Nodes can now be drained without deleting configuration information but can also be placed in `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 in need of a pre-validated hardware platforms to run our Ceph software offering. These ready nodes come in two flavors:

- Capacity optimized Ready Nodes (HDD based).
- Performance optimized Ready Nodes (Flash based).

Note: For more information see the Ceph Ready Nodes Data Sheet.

The IBM Storage Ceph RADOS Gateway supports *Intel QuickAssist Technology (QAT)* encryption and compression for enhanced efficiency on both type of operations in your

Lakehouse environment. The Performance Optimized nodes are fitted with the appropriate Intel components while the Capacity Optimized nodes will be fitted with it in their next update cycle.

2

The modern data lake architecture

Imagine a central location where you can store all your data, regardless of format or structure. That 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. Modern data lake architecture provides a flexible and scalable foundation for organizations to leverage the power of all their data.

In this chapter we introduce the modern data lake architecture and cover the following topics:

2.1 Introduction

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

Though Apache Hive paved the way for data lakes, it predated the widespread adoption of cloud services. Google's cloud offering would not arrive for another three years. As a result, early data lake adopters turned to Hive paired with 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 (S3) services offered unmatched ubiquity, scalability, and cost-effectiveness compared to HDFS. Unfortunately, Hive table organization had become 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 enabled polyglot data retrieval, processing, and analytics, reducing vendor lock-in and breaking down data silos.

Data warehouses were still alive and well in the confines of enterprise datacenters. While data lakes afforded great flexibility and economics, Hive tables did not map well to object storage and performance optimization was difficult. *If Hive was the precursor to data lakes, Apache Iceberg is the precursor to the data lakehouse*. Iceberg was designed from the ground up with object storage in mind, and in so doing addressed many of the performance and correctness deficiencies of Hive tables. Iceberg should be considered the gold 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 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 translates to several key benefits:

Improved data accessibility: Users can access and process data using various tools (SQL, machine learning) across all data types.

Simplified data exploration: Having all your data in one place allows for 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 traditional data warehouse infrastructure.

Stronger data governance: Centralized data management facilitates access controls and regulatory compliance.

For a deeper dive into data lakehouses visit this IBM website where you will find additional information and a video.

2.2 Data zones

If Iceberg is considered the gold standard, then it begs the question of what constitutes silver and bronze. Data strategies must acknowledge *polyglot persistence*, that is recognition that source data may be unstructured, or serialized in a structured, semi-structured, or multi-modal data formats. This is where an approach to organizing data that logically partitions data into a variety of zones comes into play. Zoning data provides isolation, and signals intended consumers data quality. The exact name and number of zones may vary depending on the organization, but generally there is one that is synonymous with *raw data*, another synonymous with *staging data* and another synonymous with *curated data*.

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

What is RAG?: RAG (Retrieval-Augmented Generation) 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, though. In between raw and curated is the *staging zone*, a place to store data between processing steps (See Chapter 8, “Transform: Staging and curated zones” on page 121). The final step is the *curated zone.* This is the high-quality data that constitutes a data product (See Chapter 9, “Consume” on page 155).

As stated in 2.1, “Introduction” on page 12, if the data is tabular in nature, it should be represented as Iceberg tables once it is in the curated zone. Other examples of curated data sets are flattened features serialized in multimodal data formats like TFRecord or Petastorm, or embeddings for RAG that have been indexed by vector databases like Milvus. Regardless of zone or type, all data needs to be persisted in cost effective, scalable object storage.

This is where IBM Storage Ceph takes the stage. IBM Storage Ceph is open source, software defined, runs on industry standard hardware, and has best-in-class coverage of the lingua franca of object storage that is the Amazon S3 API. It was designed from the ground up as an object store, in contrast with approaches that bolt-on S3 API servers to a distributed filesystem. With Ceph, data placement is by algorithm instead of by lookup. This allows Ceph to scale well into the billions of objects even on modestly sized clusters. Data stored in Ceph is protected with efficient erasure coding, through in-flight and at-rest checksums and encryption, and through robust access control that thoughtfully 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

Data lakehouses offer a powerful solution for managing all your data, but with great flexibility comes great responsibility. This section explains some strategies for organizing your data lakehouse.

2.3.1 Buckets

Buckets are a logical container to organize collections of objects. At the infrastructure level buckets are a resource administrated and owned by an account. There are 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, where others can be scoped to objects that match specified prefixes or tags.

Ceph buckets excel at handling massive object volumes, supporting hundreds of millions of objects with ease. This scalability stems from Ceph's 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 is true for data that represented in disparate formats. *Use separate buckets for tabular data with different structures, data with specific offline functionalities (data that has unique requirements for access or management when the Ceph cluster itself is offline), and objects related to vector databases*.

Note: A vector database is a type of database specifically designed to store and manage data as multidimensional vectors. These vectors are essentially 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 designed for managing tabular data. They act as registries, storing information about schemas. Each schema, in turn, defines the structure and organization of tables, including their offline features. These catalogs play a crucial role in commit protocols used during table updates, ensuring data consistency and integrity.

While both catalogs offer these functionalities, 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 functionality.

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

3

Building a scalable data lake with Ceph Object Storage

Built upon the highly scalable foundation of the RADOS, IBM Storage Ceph extends its capabilities with additional access methods integrated through the RADOS Gateway. This combination delivers a powerful S3-compatible object store that boasts exceptional compatibility with the Amazon Simple Storage Service (Amazon S3) protocol. As a result, users can leverage the familiar S3 interface while benefiting from the inherent scalability and flexibility of Ceph.

This chapter explores the S3-compatible object storage features offered by IBM Storage Ceph. We will delve into why these features make Ceph an ideal choice for data lake architectures. Additionally, we'll examine the growing popularity of the S3 protocol and its features among Data Engineering, AI, and Analytics teams worldwide.

In this chapter we cover the following:

- “Ceph Object Storage: An ideal platform for data lakes” on page 18
- “Security” on page 19
- “Scalability” on page 24
- “Efficiency” on page 27
- “Data management” on page 31
- “Region data resiliency in Ceph” on page 38

3.1 Ceph Object Storage: An ideal platform for data lakes

IBM Ceph Storage, based on the open-source Ceph Distributed Storage System (`http://ceph.io`), is a powerful and cost-effective solution available to organizations for harnessing full potential from their data assets. Ceph Storage should be used for creating data lakes or lakehouses because of the following key advantages.

- **Cost-effectiveness**: IBM Ceph Storage utilizes commodity hardware and open-source software to reduce the upfront infrastructure costs.
- **High scalability**: IBM Ceph allows horizontal scaling to accommodate large volumes of growing data in a data lake or lake house.
- **High flexibility**: IBM Ceph Storage can handle a variety of data types structured, semi-structured and unstructured data (For example, text, images, videos and sensor data and so forth.) making it versatile and appropriate for data lakes.
- **High availability**: IBM Ceph is designed to provide high durability and reliability for the data stored in data lake or lake house. Data is always accessible despite hardware failures or disruptions in the network by data replication across multiple geographic locations providing redundancy and fault tolerance to prevent data loss.
- **High performance**: IBM Ceph allows for 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 lake house. It also provides a cache data accelator (D3N) and Query pushdown (S3 Select) to improve the performance of analytical workloads. See 7.2.14, “Early raw zone data filtering with S3-Select” on page 117 for details of implementing S3 Select in Ceph.
- **Data governance**: IBM Ceph provides efficient management of metadata to enforce data governance policies, track data lineage, monitor data usage and provide valuable information about the data stored in the data lake such as format, data source and so forth.

Figure 3-1 shows the IBM Ceph Storage features.

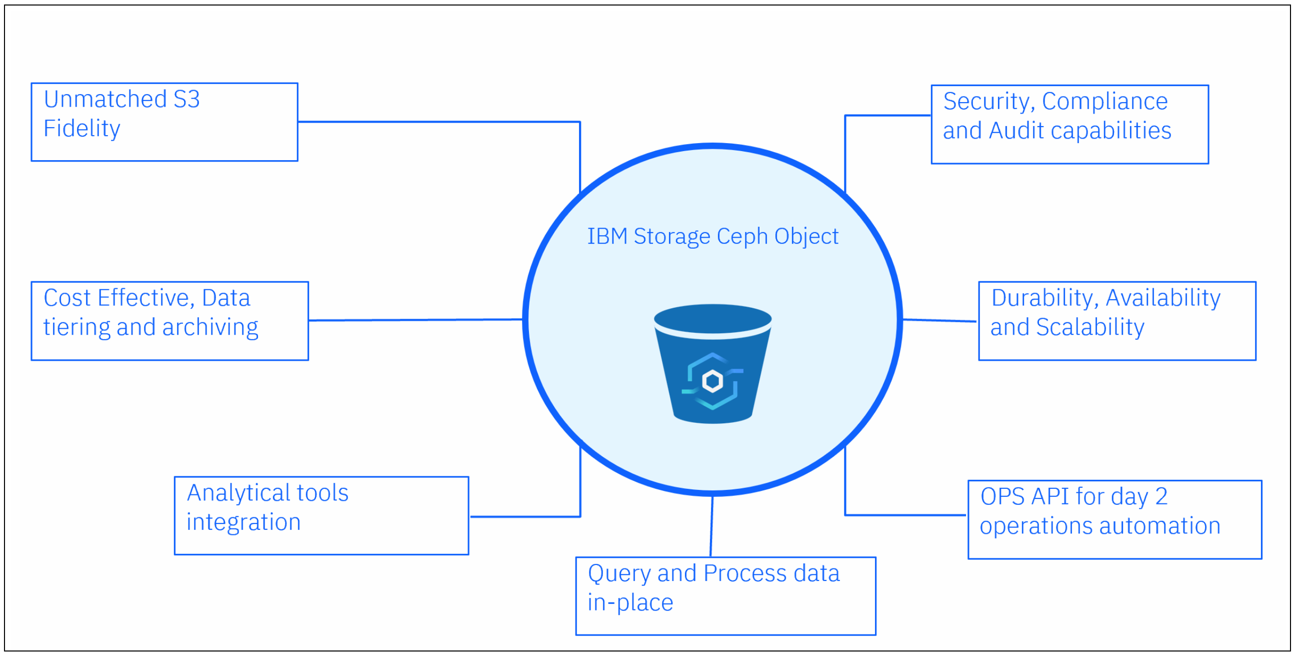

Unmatched S3 Fidelity
Security, Compliance and Audit capabilities
IBM Storage Ceph Object
Cost Effective, Data tiering and archiving
Durability, Availability and Scalability
Analytical tools integration
OPS API for day 2 operations automation
Query and Process data in-place

Figure 3-1 IBM Ceph Storage features

Some key features of IBM Ceph Storage are described in the sections below:

3.2 Security

Security is crucial for a data lake or data lake house implementations. IBM Ceph Storage consists of multiple security features that make it a robust and ideal choice for building data lakes or lake houses. Some of these key security features are encryption, access controls and audit logging. These features protect sensitive data stored in the data lake and help organizations stay in compliance to regulatory obligations like GDPR, HIPAA and PCI-DSS and so forth. The security features allow consumers to maintain trust in their data lake or lake house environments.

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

- **Data-level access management**: Control access at the data object level, not just the server. This approach minimizes the attack surface and focuses on protecting your valuable information.
- **Seamless IAM/SSO integration**: Integrate Ceph Storage with your existing Identity and Access Management (IAM) or Single Sign-On (SSO) systems. This simplifies user management and ensures consistent access control policies.
- **Granular control with RBAC/ABAC**: Implement Role-Based Access Control (RBAC) or Attribute-Based Access Control (ABAC) to grant users access based on their roles or specific attributes. This ensures only authorized users can access specific data within your data lakehouse.

Best practices for data lake security can be found in these blog posts:

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 Ceph Storage 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 residing within the storage cluster's data lake.

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

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

Additionally, IBM Ceph Storage seamlessly *integrates with various external identity providers (IDPs) and access management (IAM) systems*. This includes industry-standard protocols like LDAP (Lightweight Directory Access Protocol) and AD (Active Directory), along with compatible OpenID Connect (OIDC) providers such as IBM Secure Verify or Red Hat SSO.

This flexibility allows you to leverage existing authentication infrastructure for a unified access control experience.

Note: Additional details on how to configure authentication with SSO (OIDC) using IBM Security® Verify can be found here.

Ceph Storage empowers organizations to implement robust access management for their data lake, especially in large deployments. This is achieved through:

- **External identity integration**: Streamline access control by leveraging existing user identities 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 strengthens your data security by requiring an additional verification step before objects can be deleted. This safeguard helps prevent accidental or unauthorized deletion of critical data within your Ceph storage system. Refer to 7.2.6, “Raw zone: S3 Multi-Factor Authentication delete (MFA delete)” on page 99 for more information on how to implement MFA delete in Ceph.

- **Granular authorization policies**: Define which users or groups can perform specific actions (read, write, delete) on objects or buckets within the data lake. The Ceph Object Gateway (RADOSGW) enforces these policies, ensuring only authorized users can access data via APIs. See 8.1.1, “Assigning bucket policies and object tags for effective S3 access control” on page 122 for step by step instructions on implementing granular authorization policies in Ceph.
- **Simplified role-based access control (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.
- **Attribute-based access control (ABAC) for scalability**: Manage access rights based on tags assigned to resources. This simplifies access control, especially when adding new resources or modifying privileges for many users at once. ABAC allows for fewer, more concise permission policies.

Note: More details on ABAC can be found here.

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 ease of management and granular control over data access.

See Figure 3-2.

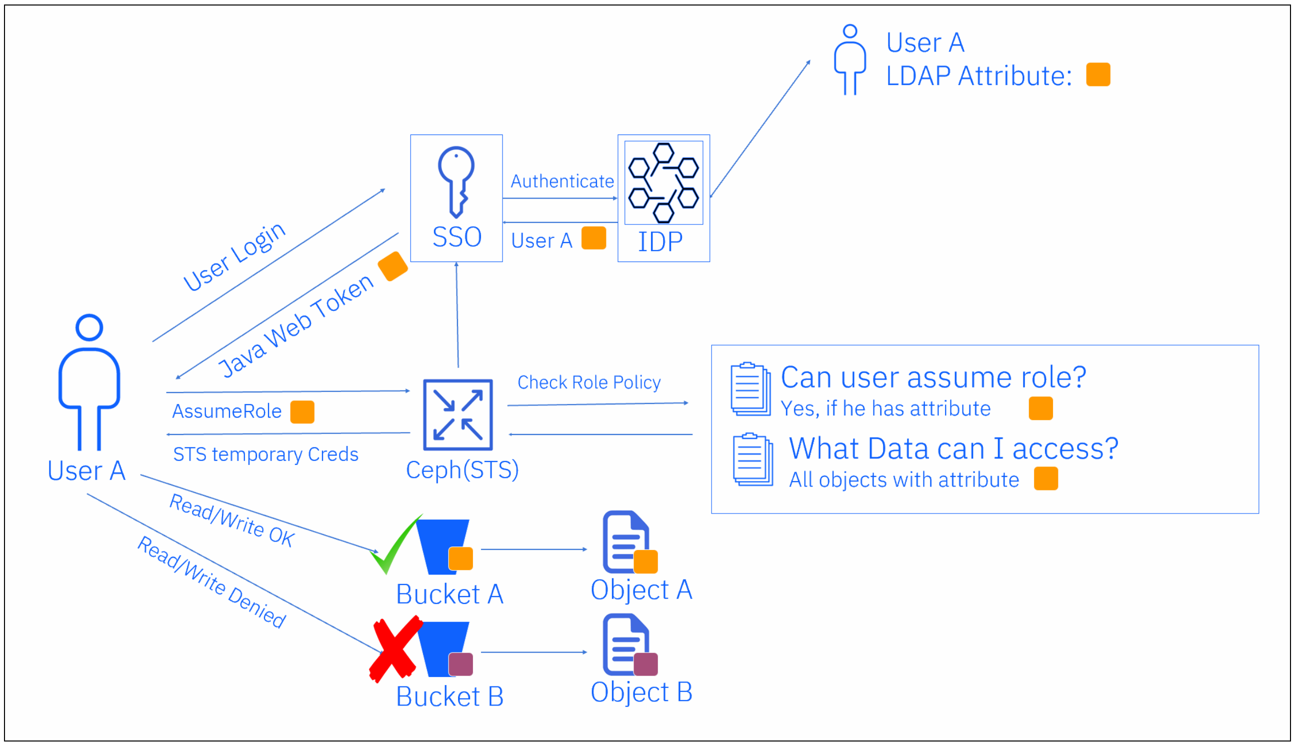


Figure 3-2 Attribute based access control example

What is STS?: RADOS Gateway implements a subset of functionalities from Amazon's Secure Token Service (STS). This subset is known as Ceph STS. It allows users and administrators to manage authentication and access control for object storage within a Ceph cluster. Users first authenticate against STS and, upon success, receive a short-lived S3 access and secret key that can be used in subsequent requests.

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

How to implement access control in Ceph: Refer to 7.1.3, “Ingest: Object storage IDP authentication with single-sign-on” on page 88 and 7.1.4, “Ingest: IAM role RBAC-based authorization” on page 90 for more information on how to implement access control in Ceph.

Note: More details on Audit Logging can be found here. Also, Ceph provides centralized logging out of the box. More information can be found in this blog post.

3.2.2 Encryption

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

Data-at-rest encryption (DARE)

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

Hardware Security Modules (HSM) or Key Management Systems (KMS) are typically responsible for generating, storing and distributing encryption keys. IBM Ceph manages encryption keys by integrating them with key management interoperability protocol (KMIP) compliant KMS like IBM Guardium® Key Lifecycle Manager (GKLM).

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

IBM Ceph Storage provides *built-in support* for various server-side encryption options like SSE-KMS, SSE-S3 and 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 using per bucket keys. This makes key management invisible to the calling principal.

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

How to implement encryption in Ceph: Refer to 7.2.5, “Raw zone: S3 SSE-KMS encryption at rest” on page 92 for more information on how to implement encryption in Ceph.

Data-in-motion encryption (DIME)

IBM Ceph Storage utilizes Transport Layer Security (TLS) to encrypt data transmitted over the network, ensuring a secure communication channel. Cephadm simplifies setup with automated TLS configuration for the ingress service.

Note: More details on TLS config automation with cephadm can be found here.

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

Note: More information on Messenger v2 protocol can be found here.

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 unauthorized/malicious activity. IBM Ceph Storage provides S3 object lock API that enables an *object lock* that prevents objects from being altered or deleted for a retention period. The following are the features of object lock:

- **Retention period:** When creating an object, you define a retention period during which it is locked. This timeframe guarantees 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. This provides additional control for sensitive data.
- **Granular access control:** IBM Ceph Storage restricts setting, modifying, or removing object lock settings to authorized administrators. This 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 allows organizations to adhere with regulations such as GDPR, HIPAA and other data retention mandates.

Key takeaway: S3 object lock with IBM Ceph Storage 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 Ceph: Refer to 7.2.4, “Raw zone: S3 object lock and object versioning configuration” on page 97 for more information on how to implement object lock in Ceph.

Note: Consulting company Cohasset concludes that IBM Storage Ceph, when properly configured and upon satisfying the additional considerations, meets the electronic record-keeping system requirements of SEC Rules 17a-4(f)(2), 18a-6(e)(2), and FINRA Rule 4511(c), as well as, 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 CFTC Rule 1.31(c)-(d). See the Cohasset certification for more information.

3.2.4 Versioning

IBM Ceph Storage supports the S3 Object versioning API. IBM Ceph Storage creates a new version of the object on every write action (for example, COPY, PUT, 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, allowing users to access older versions as needed for historical bookkeeping via S3 versioning APIs. This feature safeguards your data by allowing you to recover previous versions of the object in case of:

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

Note: More details on S3 Object versioning API can be found here.

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 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 or transition retained versions to a different storage class or delete older versions based on criteria such as age or last accessed date and so forth 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 Ceph: Refer to 7.2.4, “Raw zone: S3 object lock and object versioning configuration” on page 97 for more information on how to implement versioning in Ceph.

3.3 Scalability

When it comes to scalability, IBM Storage Ceph will provide some key features that are discussed in this section. Some of these features are specific to the RADOS Gateway while others are provided by RADOS:

- RADOS scalability to thousands of OSDs.
- CRUSH to distribute objects across OSDs.
- Highly available and scalable through multiple RADOS Gateways.
- Highly scalable bucket indexing with dynamic resharding.
- Compatible with many HTTP load balancers, proprietary or open source, hardware or software.
- Indexless buckets.

In extremely large object stores, a key design consideration is *bucket indexing*. Bucket indexes allow 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. For those comfortable with Linux environments, using a bucket index is similar to using the **`/bin/ls`** command to see a directory's contents.

In extremely large environments, the time it will take to list all the objects stored in a bucket will grow longer and longer and a naive approach to indexing might require locking the entire index for a single application or user uploading a tiny (1KiB) or very large (thousands of MiB) object.

To address this scalability challenge, Ceph utilizes a technique called *bucket sharding*. This approach splits each bucket index into smaller, manageable slices (shards) with a configurable upper threshold (default: 100,000 entries). Each shard is stored as key-value data on a distinct RADOS object (See the details in *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 version 7.0 of IBM Storage Ceph, each bucket starts with ten (10) index shards easily accommodating one million (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 RADOS Gateway.

IBM Storage Ceph allows you to set the following behaviors for dynamic resharding:

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

If required, the administrator can manually reshard a bucket index 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 using `radosgw-admin bi list --bucket={bucket_name} >bucket.backup`.

Note that you need to add quotes 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`.

Run the following command to view the outstanding resharding requests:

```
radosgw-admin reshard list
```

Run the following command to view the status of a reshard operation:

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

IBM and Red Hat conducted a series of scalability tests for Ceph storage. Their latest test focused on evaluating Ceph's ability to handle massive datasets. They successfully ingested over 10 billion objects into a Ceph cluster, demonstrating its exceptional scalability. You can view a video about this test here.

The scalability of Ceph described in the above video is made possible by the highly scalable and highly available nature of RADOS and the CRUSH algorithm that will efficiently distribute objects across all the Object Storage Devices (OSDs) available in your IBM Storage Ceph cluster.

Scalability and performance with 1 TB/s throughput: See this blog post for a detailed discussion on scalability and performance of Ceph with 1 TB/s throughput.

As we just discussed the scalability of the bucket indexes, it is important to note that it can only be paired with scalability in terms of bandwidth and number of operations per second. This can easily be achieved thanks to the combination of the RESTful nature of the S3 protocol and the ability to deploy multiple RADOS Gateways, placing all or a subset of them behind a HTTP load balancer. The supported deployment tool used by IBM Storage Ceph, **`cephadm`**, supports the deployment of an ingress service based on `HAProxy` for teams that do not have load balancers available. The built-in ingress service can also be paired with existing on-site traffic distribution mechanisms.

Key takeaways:

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

IBM Storage Ceph also offers *index-less buckets*, a feature suited for custom applications that can trade the ability to list buckets for higher write operations per second (indexless 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 RADOS Gateway 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 using database specific APIs or query languages.
- **Zonal separation**: While index-less and indexed buckets can coexist within a Ceph cluster, they must be placed in separate RADOS Gateway 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 together. By default, data and metadata are automatically synchronized across all zones within a zonegroup. This ensures redundancy and availability of your objects.

Analytical workloads in Ceph:

Ceph offers two features that can benefit analytical workloads:

- **D3N (Direct Data Placement Network):** D3N accelerates data access for large datasets by utilizing a local caching layer (RAM or NVMe) at the edge. This positions frequently accessed data closer to RADOS Gateways, significantly improving performance. Additionally, the overall infrastructure maintains a good Total Cost of Ownership (TCO) by leveraging cheaper, high-capacity HDD storage for the physical backend. See the Ceph Documentation for more information.
- **S3 Select:** S3 Select helps you build a more efficient pipeline between your S3 clients and your S3 endpoint by allowing to use an SQL-like syntax to selected subsets of the data stored in the S3 object store. This results in better data targeting and improves latency and throughput while reducing network usage for better TCO in the public cloud and on-premises. See the Ceph Documentation for more information.

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

3.4 Efficiency

Data lakes thrive on the ability to access massive datasets for comprehensive and nuanced insights. However, traditional hierarchical file systems, often reliant on directory lookups, struggle with scalability. This limitation becomes particularly evident as data volumes reach the exabyte range. Object storage, on the other hand, 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 and larger it is important to be able to 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 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 stored in a pool are fully replicated according to the `size` parameter 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 pools**:

- All objects stored in the pool benefit from *erasure coding*, a data protection technique 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 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.

As mentioned above, replicated pools create full copies of the data and as such offer a fixed raw to usable storage ration aligned with the pool `size` parameter:

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

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

Erasure coded (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 - 2 data chunks and 2 parity chunks (2 raw bytes used for every usable byte stored).
- 4+2 - 4 data chunks and 2 parity chunks
  1.5 raw bytes used for every usable byte stored.
- 8+3 - 8 data chunks and 3 parity chunks
  1.375 raw bytes used for every usable byte stored.
- 8+4 - 8 data chunks and 4 parity chunks
  1.5 raw bytes used for every usable byte stored.

With erasure coding you can not only get a better raw to usable storage ratio, but you can also significantly enhance the durability of your data so that it can survive up to 4 component failures without any data loss.

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

Key takeaway: 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 lake house 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 RADOS Gateway and helps you optimize storage utilization within your data lake by automatically compressing data as it enters the system. This not only reduces storage requirements but can also 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 on the fly.
- Potentially improves data processing efficiency due to smaller data sizes.

The supported compression plugins include the following:

- **Snappy:** This is a fast and lightweight algorithm, making it 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):** This is a widely used and well-established algorithm 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):** This is a modern algorithm 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 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, while media files like images or audio might benefit more from zstd's high compression capabilities.
- **Performance versus compression ratio:** Faster algorithms like Snappy offer a trade-off 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 could be preferable.
- **Processing power:** Decompressing data requires some processing overhead. Consider the resources 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 creation time of the RADOS Gateway placement or enabled afterward. Example 3-1 illustrates how you can update an existing placement target to enable compression via `zlib`.

Example 3-1 Update 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 will not compress the data already written unless it is updated or rewritten.
- Disabling compression will not decompress the data already written compressed unless it is updated or rewritten.

Note: All-flash IBM Storage Ready Nodes include hardware capable of Intel QAT (Quick Assist Technology) 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.

Refer to IBM Documentation for more information on IBM Storage Ready Nodes.

3.4.3 Life cycle management

IBM Storage Ceph RADOS Gateway supports many bucket life cycle management settings to make the best usage of your S3 compatible object store:

- Object transition
- Object expiration
- Versioning
- Object lock

Object transition provides you with the ability to transition 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 RADOS Gateway placement target. This target could utilize NVMe (Non-Volatile Memory Express) drives for exceptional speed.

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

Object expiration allows you to configure a bucket so that older data gets automatically deleted after a specific amount of time or on a specific date without any action from any S3 client application. This will lower the overall space usage within your IBM Storage Ceph cluster and improve your TCO over time.

Object versioning lets you keep older version of objects accessible within a bucket where this feature is enabled. By doing so, the application or an administrator does not need to require data restoration from a backup and restore software and requires a lower amount of time to get the old data back for reprocessing. This will help reduce the manpower needed to get access to older version of your data without any physical media handling.

Object locking lets you make sure that the data stored in a bucket cannot be deleted or overwritten. By doing so you detach yourself of the need to deploy a specific third-party software or hardware to guarantee that your data, once stored, is immutable and cannot be deleted.

To use any of the life cycle management features, you simply need to assign the appropriate policy to the bucket. The appropriate policy can even be assigned directly by the end user based on his or her level of permissions on the bucket (for example, the owner of the bucket can assign any bucket policy).

Lifecycle policies are documented here for IBM Storage Ceph.

Key takeaway: Bucket lifecycle configuration lets you automatically transition objects to more cost-effective storage classes, archive inactive data, or even delete it based on your specific needs. This helps you optimize storage costs and ensure your valuable data remains readily accessible when needed.

How to implement lifecycle policies in Ceph: Refer to 7.1.1, “Ingest: Lifecycle policy expiration” on page 85 for more information on how to implement lifecycle policies in Ceph.

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 allows for precise alignment with performance, availability, and cost objectives across various workloads. By optimizing storage utilization across the entire Ceph cluster, this adaptability enhances overall functionality and resource efficiency.

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

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

- **Bucket notifications**: Trigger automated actions in response to pre-defined events happening within a bucket. This enables real-time data management workflows.
- **Bucket inventory**: Facilitate efficient data organization, auditing, reporting, and compliance tasks. Bucket inventory streamlines essential data management activities.

We will review these features deeper in this section.

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 subsequently 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 NVMes, SSDs, or SAS/SATA drives, each distinguished by unique attributes relating to durability, replication, erasure coding, and more.

It is worth emphasizing that placement targets dictate the association of pools with individual buckets. Once a bucket is created, its placement target is fixed and cannot be altered.

By utilizing various placement targets within the Ceph Object Gateway, cluster administrators gain the flexibility to precisely dictate the location of their data. For instance, administrators can:

- Strategically allocate bucket data to different performance tiers by statically assigning placement targets:
  - Bronze tier: designate the placement target with a pool supported by rotational devices.
  - Silver tier: designate the placement target with a pool supported by SSDs.
  - Gold tier: designate the placement target with a pool supported by NVMe drives.
- Implement different data protection mechanisms for buckets based on their requirements:
  - Utilize replication for critical data, ensuring high redundancy.
  - Employ Erasure Coding for other buckets to achieve superior storage efficiency, albeit with slightly reduced redundancy.
- Tailor placement targets based on the nature of the data to be stored:
  - Archival data: assign to a placement target utilizing rotational devices for cost-effective long-term storage.
  - I/O sensitive workloads: assign to a placement target utilizing 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 tailored to specific data management needs. These targets might prioritize performance, optimize cost, or ensure high availability.

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

However, *it is important to remember that placement targets are fixed after a bucket is created*. Choosing the right target during bucket creation is crucial for optimal data segregation and management.

For more information, see Appendix A “Bucket creation” on page 183.

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 our data upon bucket creation. However, there are instances where it becomes advantageous to provide users with the capability to dynamically transition their data between various logical partitions based on predefined criteria, such as time. This functionality is facilitated by Storage classes in Ceph Object Gateway, which are governed by S3 Bucket Lifecycle rules, allowing for the automation of object transitions between Storage classes.

With Storage classes, users gain the flexibility to categorize their data and define lifecycle policies that automatically move objects between different storage tiers over time. This dynamic data management approach allows for optimized resource utilization and cost efficiency, as 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 transition data between Storage classes, ensuring that it 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 utilizing Storage classes in Ceph Object Gateway, users gain the ability to strategically determine the placement of their data based on time, enabling a variety of 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.
  - Utilize Erasure Coding 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 to a Storage class utilizing rotational devices for cost-effective long-term storage.
  - For I/O sensitive workloads, designate a Storage class utilizing 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 utilizing S3 Bucket Lifecycle policies to seamlessly transition objects between different Storage classes based on predefined transition rules, users also have the flexibility to override the default Storage class when creating an object. This 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, allowing them to specify the desired 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, irrespective 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 utilize source filters (prefix), enabling the transition or expiration of only desired objects.

It is imperative to recognize that when employing AWS S3 SDKs such as boto3, it is crucial to ensure that non-default storage class names are in accordance with those defined by AWS S3. Failure to do so will result in the SDK rejecting the request and generating an exception. This underscores the importance of adhering to AWS S3's storage class naming conventions to ensure seamless interaction with the SDK and prevent errors during data storage operations. Further information on the naming convention AWS uses to defined the non-default Storage classes can be found at the AWS Documentation.

In this example, we will demonstrate how to effectively utilize Storage classes in Ceph Object Gateway alongside S3 Bucket Lifecycle policies to manage object transitions and expiration based on the duration of storage within the Ceph cluster.

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

Additionally, after objects have been 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:

- Transition objects from the `STANDARD`. Storage class to the `STANDARD_IA` Storage class after 30 days of storage.
- Expire objects after one year of storage to ensure efficient resource utilization and storage management.

By implementing the comprehensive S3 Bucket Lifecycle policy as 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 Ceph: Refer to 7.2.7, “Raw zone: Data tiering configuration - creating a cold storage class” on page 101 for more information on how to implement Storage classes in Ceph.

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 instance, 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's 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. See 7.2.9, “Raw zone: Serverless data pipeline workflow” on page 108 for details of implementing an event-driven architecture with Ceph.
- **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 Ceph cluster, ensuring reliable event delivery and efficient processing.

- **Customizable actions**: Users can define custom actions to be executed in response to S3 bucket events, such as invoking external APIs, executing scripts, or triggering business logic.
- **Real-time monitoring and alerting**: S3 bucket notifications provide real-time visibility into object-related activities within Ceph clusters, enabling users to monitor, track, and analyze changes as they occur. This 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 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 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 detailed instructions on configuring the different providers, refer to the documentation provided by IBM Storage Ceph.

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	Sub-event
s3:ObjectCreated:	* (all sub-events)
	Put
	Post
	Copy
	CompleteMultipartUpload
s3:ObjectRemoved:	* (all sub-events)
	Delete
	DeleteMarkerCreated
s3:ObjectLifecycle:Expiration:	Current
	NonCurrent
	DeleteMarker
	AbortMultipartUpload
s3:ObjectLifecycle:Transition:	Current
	NonCurrent

Event	Sub-event
s3:ObjectSynced:	* (all sub-events)
	Create
	Delete
	DeletionMarkerCreated
s3:Replication:	* (all sub-events)
	Create
	Delete
	DeletionMarkerCreated

Notification filtering

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

- **Prefix/Suffix filtering**: This allows you to filter notifications based on the prefix (the beginning) or suffix (the end) of object keys. It is a simple yet effective way to filter objects based on their names.
- **Regular expression matching**: With regular expression filtering, you can define complex patterns using regular expressions to match object keys. This 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 providing additional 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 allows 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 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, catering 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. In this section, we delve into the expected performance and dependability for both synchronous and asynchronous notification methods.

Synchronous notifications happen right away, as part of the original operation. In this mode, the operation will not be considered complete (acknowledged) until the notification successfully reaches its destination (topic's endpoint). This means the overall time 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 (topic's endpoint) later. This means the initial operation only experiences a delay when the notification is saved.

To minimize this delay, consider placing the Ceph Object Gateway's "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 upon dispatch, ensuring prompt confirmation of notification delivery. Conversely, asynchronous notifications, while offering reduced latency for the original operation, may 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 the event of system failures or disruptions.

How to implement bucket notifications in Ceph: Refer to 7.2.2, “Raw zone: S3 bucket notification configuration” on page 95 for more information on how to implement bucket notifications in Ceph.

3.6 Region data resiliency in Ceph

We can ensure regional data resiliency in Ceph by utilizing the Ceph Object Gateway Multi-site feature, which enables the deployment of multiple instances of Ceph Object Gateway across different geographical locations while maintaining data consistency and accessibility.

- **Geographical distribution**: Using Ceph Object Gateway Multisite, you can deploy multiple Object Gateway instances in various geographical locations or data centers. This distribution ensures proximity to users and enables data redundancy and disaster recovery capabilities.
- **Bucket replication**: One of the key functionalities of Ceph Object Gateway Multisite is bucket replication. You can configure either all or specific buckets to be replicated across multiple 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 made to data are propagated across instances with some delay, and it may take time for all instances to synchronize fully.
- **Active-Active configuration**: Ceph Object Gateway Multisite supports active-active configurations, allowing all instances of 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.
- **Disaster recovery**: By replicating data across multiple geographical locations, Ceph Object Gateway Multisite enhances disaster recovery capabilities. In the event of 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, 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 version (released on 2016-04-01), it has been possible to use multisite replication in Ceph Object Gateway. It is important to note that when using traditional multisite replication, all objects in all buckets in our multi-zone configuration will be 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* allow us to define replication per bucket.

Utilizing multisite replication is highly effective for replicating our data across different locations worldwide. Performing this replication at the storage layer is the most efficient approach, as 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

This setup adopts a sophisticated structure. It comprises 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 Ceph Storage Cluster.

The inclusion of multiple zones within a specific zonegroup serves as a safeguard against disasters affecting individual zones. In the event of a substantial failure in one zone, the presence of other zones ensures continuity. Furthermore, each zone remains active, capable of receiving write operations. Such a multi-zone setup, with several active zones, not only strengthens disaster recovery measures but also serves as a solid basis for establishing content-delivery networks.

In the latest versions of IBM Ceph Storage, 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 may arise. The dynamic resharding feature identifies this condition and automatically increases the number of shards utilized by a bucket's index. Consequently, this reduces the number of entries in each shard.

Replication operates in an Active/Active mode for data access, allowing end users to read and write simultaneously from their nearest S3 endpoint location. This enables faster data access and minimizes downtime.

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

Multi-zone groups

The Ceph Object Gateway facilitates the use of multiple zonegroups, with each zonegroup comprising one or more zones. If two zones belong to the same zonegroup, and that zonegroup shares the same realm as another zonegroup, the objects stored within these zones share a unified object namespace. This 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 will be redirected to the zonegroup where the bucket is located.

The creation of multiple zonegroups can prove beneficial when aiming 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 necessitating 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 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 allows for the definition of distinct namespaces and configurations, granting flexibility wherein each realm can possess configurations independent of those in other realms. For instance, it is not possible to have duplicate bucket or user names within a single realm.

Figure 3-3 on page 40 shows multisite one-realm configuration.

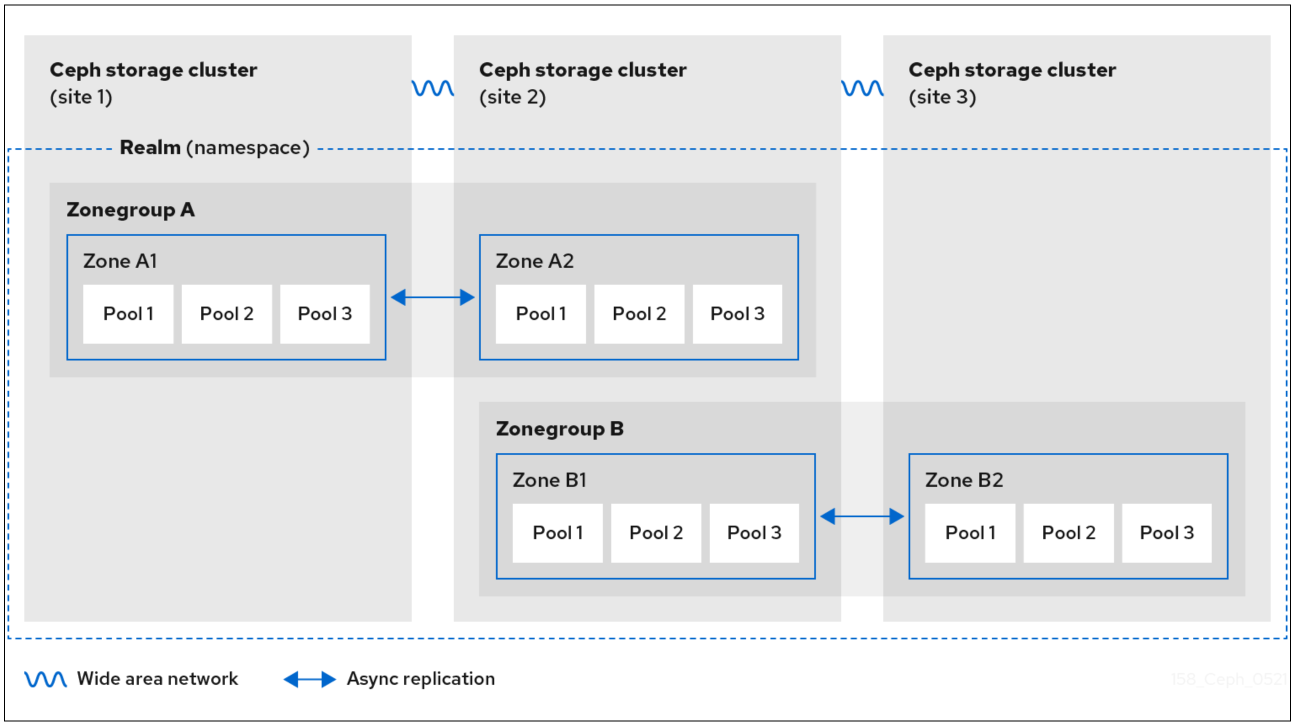


Figure 3-3 Multisite one-realm

A realm serves as a globally unique namespace encompassing one or more zonegroups. Zonegroups, in turn, comprise 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 41 multisite two-realms configuration.

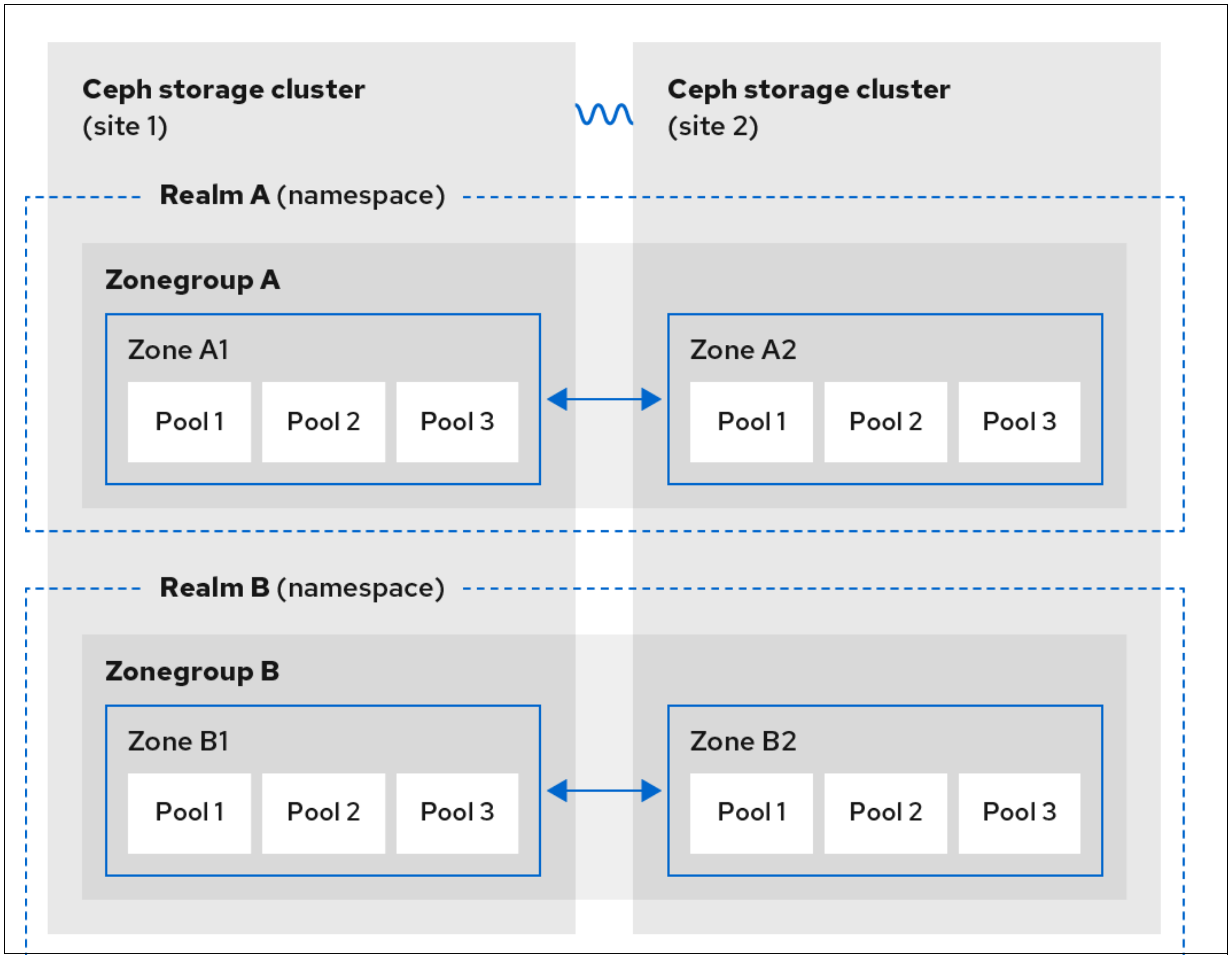


Figure 3-4 Multisite two-realms

With the flexibility offered by realms, zonegroups, and zones, the Ceph Object Gateway enables us to tailor our data lakes to the desired architecture, allowing for data replication across diverse global locations. This ensures data consistency and provides robust monitoring metrics to verify that our 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 recommended to allocate specific groups of Ceph Object Gateway instances to handle public S3 requests and multisite replication requests between two Ceph clusters.

While not mandatory, this approach offers several benefits:

- **Scalability**: With dedicated resource sets for public and replication tasks, we 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 allowing us to target specific Object Gateway instances based on the issue. This separation also ensures replication messages do not interfere with client logs or the other way around.
- **Enhanced security**: Utilizing different Object Gateway sets allows for the use of distinct networks with varying security levels, firewall rules, and OS security measures. For instance, public-facing Object Gateway instances could operate on Network A, while replication Object Gateway instances could utilize 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:

- Set the `rgw_run_sync_thread` parameter to false in the Object Gateway instance. This prevents the Object Gateway from transmitting multisite replication data.
- Additionally, to avoid receiving replication data, Object Gateway instances must be removed from the zonegroup and zone replication endpoints.

How to implement dedicated Ceph Object Gateway instances in Ceph: Refer to this blog post for more information on how to implement dedicated Ceph Object Gateway instances in Ceph.

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, allowing one bucket to retrieve data from others located in different zones. As a result, the synchronization process, which previously assumed that 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 utilize 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 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 refers to the current bucket instance. For disaster recovery entire zones can be mirrored without bucket-level configuration. For more granular control, configure synchronization pipes at the zone group level, then enable specific bucket synchronizing 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.

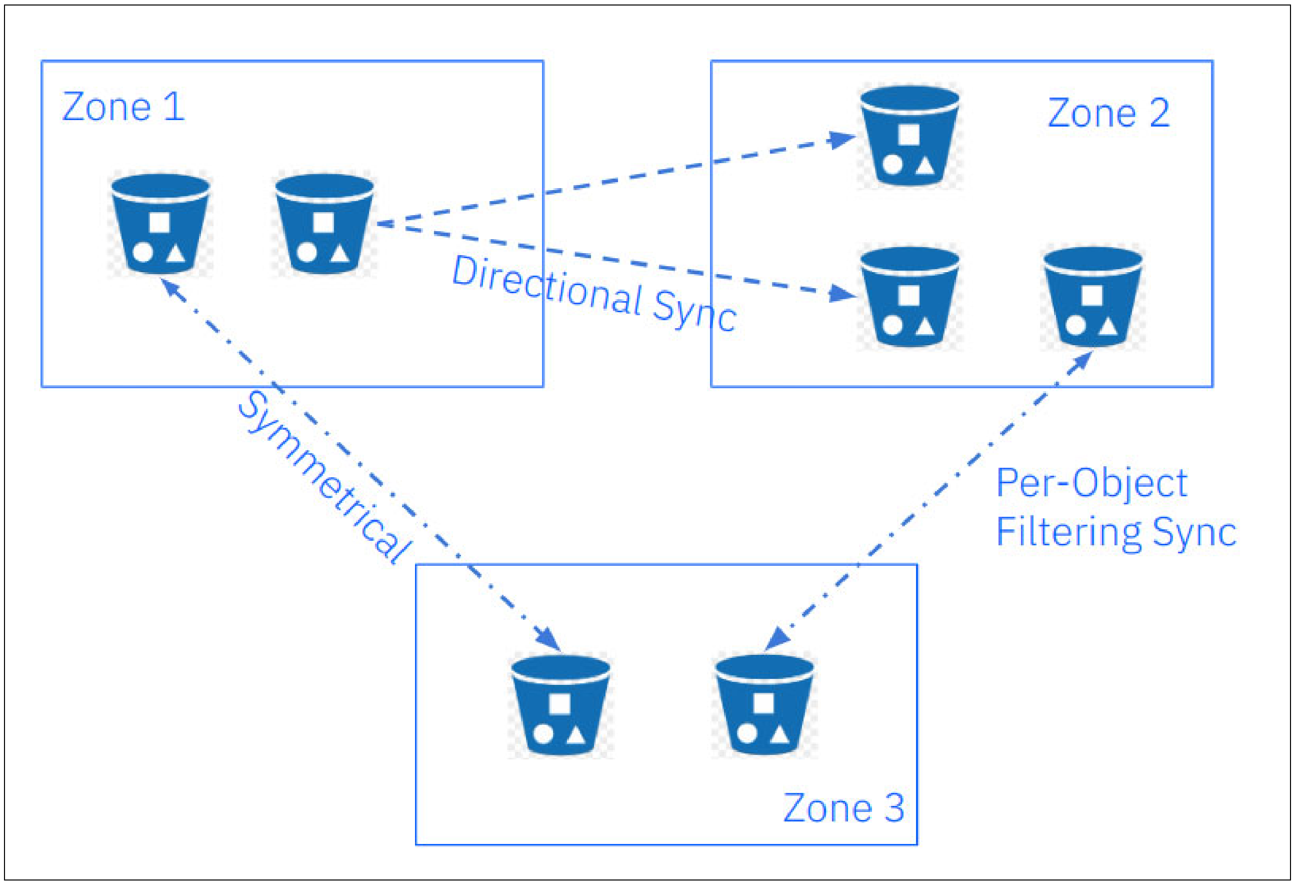


Figure 3-5 Multisite bucket-granularity

Multisite bucket-granularity sync policies offers 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 have the option to enable or disable synchronization for each individual bucket, allowing for precise control over replication processes.

It is important to note that currently buckets must be located in different zones. This means that users cannot enable replication between buckets located in the same zone.

A standout feature of this capability is the ability to utilize source filters, enabling the replication of only desired objects. This can involve replicating objects with specific prefixes or those tagged appropriately.

Utilizing multisite bucket-granularity sync policies in our data lakes allows us to selectively replicate desired buckets and specific objects to other zones. For example, if we have a bucket dedicated to ingesting data from our edge locations, we may prefer not to replicate this data until the objects are validated, organized, and cleaned. Once these tasks are completed, we can then move the objects to another bucket designated for replication, enabling our data scientists to extract and utilize the data to enhance our business. *The best part in this case is that this solution is built into IBM Storage Ceph; we do not need any external tools to replicate our data to other locations*..

How to implement multisite bucket-granularity sync policies in Ceph: Refer to this blog post for more information on how to implement multisite bucket-granularity sync policies in Ceph.

How to implement multisite replication in Ceph: Refer to this blog post series for more information on how to implement multisite replication in Ceph.

3.6.1 Archive zone

While many third-party backup solutions can store backups in AWS S3-compatible buckets via the S3 protocol, finding solutions that can back up the data within those AWS S3-compatible buckets has been much 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 timeframe), cost (storing a massive backup can be expensive), and choosing the appropriate destination for the backup data.

One of the main reasons 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 do offer excellent protection against hardware or zone failures. However, *these techniques are not backups*. For instance, any accidental deletion or a ransomware attack will be 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 Multi-Factor Authentication delete in their buckets. This is a valid option when dealing with a single zone. However, when replicating data to multiple locations, this approach becomes impractical. Enabling versioning features in each zone would consume more capacity and performance than necessary.

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

Archive zone feature: A few years ago, Red Hat collaborated with a Spanish customer to address their data replication needs. They explored various capabilities within IBM Ceph Storage (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 archive zone proved to be the ideal solution:

- The archive zone feature implements a specific archiving behavior 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 only be deleted through the gateways associated with the archive zone.
- This functionality proves beneficial in configurations where several non-versioned zones replicate their data and metadata through their zone gateways, ensuring high availability for end users, while the archive zone captures all data updates.

Figure 3-6 on page 45 shows a visual representation of how archive zones manage S3 object versions.

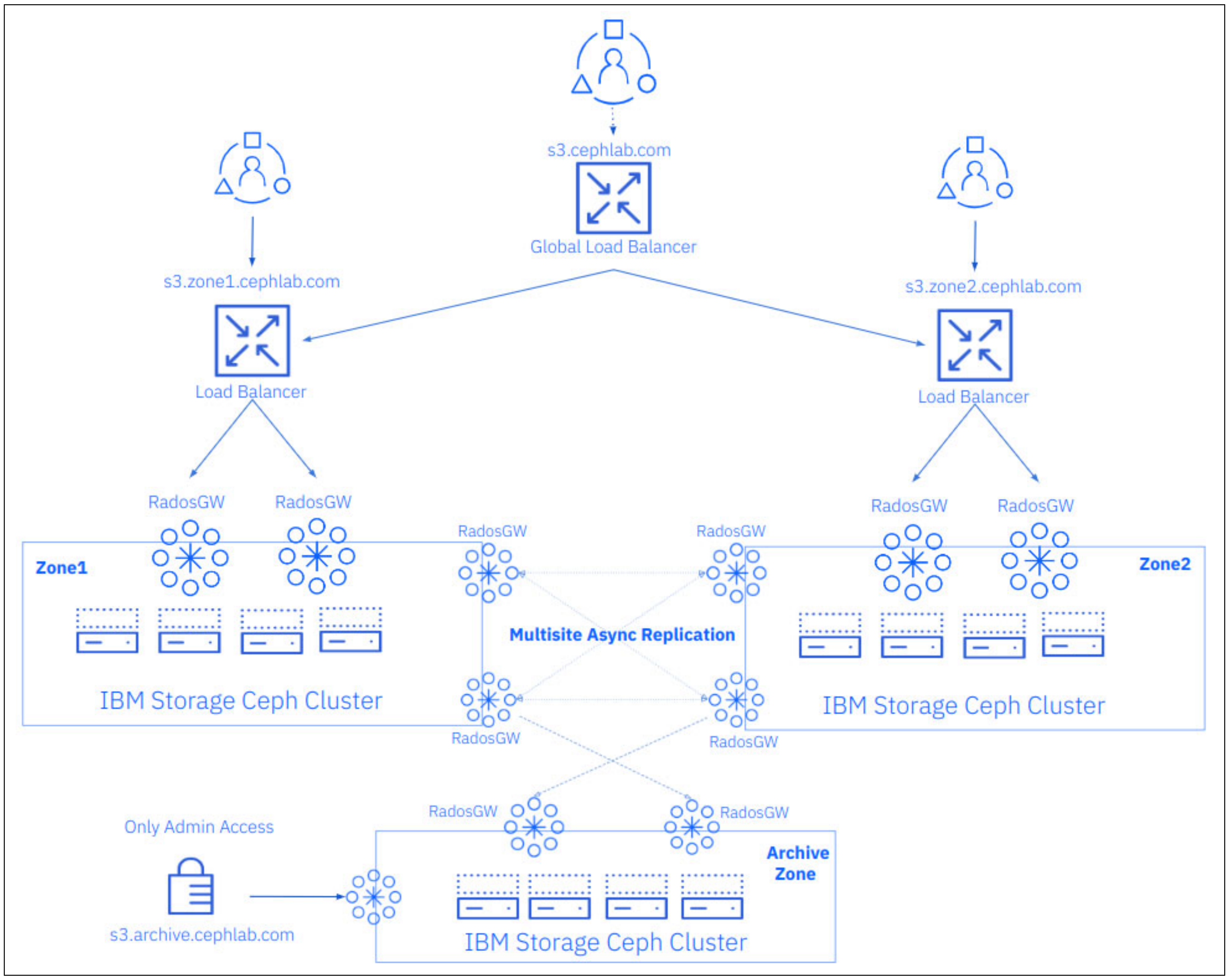


Figure 3-6 Visual representation of how archive zones manage S3 object versions

By leveraging the archive zone feature, we 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 made by legitimate users in non-archive zones.

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

Here is how archive zone safeguards your data:

- **Automatic versioning:** Every time a user object is updated, archive zone automatically creates a new version. This eliminates the need for scheduled backups and ensures all changes are captured promptly.
- **Near real-time resilience:** By continuously capturing data updates, 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 uploaded by a user, the archive zone seamlessly replicates the data asynchronously, providing immediate protection against data loss.
- Every modification made to objects within the production zone triggers the generation of a new object version within the archive zone, including delete operations. This ensures that a complete record of object changes is maintained for audit and recovery purposes.
- Immutability is guaranteed 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, it is important to note that while 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 via the S3 endpoint.
- Even if one of our users accidentally deletes the entire bucket, we 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.

We have the capability to utilize both the archive zone feature and multisite bucket-granularity sync policies concurrently, granting us precise control over which buckets or individual objects within our storage environment are sent or replicated to the archive zone. This flexibility empowers us 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, we can opt to exclude their objects from replication to the archive zone. By doing so, we effectively conserve storage capacity within the IBM Storage Ceph cluster and mitigate the transmission of unnecessary data across our 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 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. Given that 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 life-cycle Policies becomes crucial.

Life-cycle policies offer a strategic solution by enabling administrators to specify the desired number of versions to retain for each object within the IBM Storage Ceph cluster 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 archieve zone in Ceph: Refer to this blog post for more information on how to implement archieve zone in Ceph.

4

Replacing Hadoop Distributed File System (HDFS) with IBM Storage Ceph Object Storage

Ceph Object Storage can be a compelling alternative to 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 and has the following sections:

4.1 Hadoop introduction

Hadoop is an open-source framework that allows for the efficient storage and processing of large datasets across clusters of computers. The following is an excerpt from Hadoop project documentation.

"*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 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.*"

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 designed for distributed data processing across multiple computers. This enabled them to analyze massive datasets 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 functionalities – the MapReduce framework for distributed data processing and its file system – to the ASF. This marked the genesis of a new project, playfully 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 functionalities, 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. Other major companies like Facebook, LinkedIn, and Twitter followed suit, recognizing the power of distributed computing for handling their massive datasets.

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. Hadoop has been successful and popular due to several valuable features.

- Hadoop is an open-source project. This has resulted in a vast community of developers who contribute to the project, add new features, and provide support to users.
- Hadoop was a highly scalable solution compared to the other technologies 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 allows data to be processed in parallel across a cluster of computers, making it capable of handling massive datasets.

- 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 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. This includes tools like Hive, Pig, Spark, and HBase, which provide additional functionality and capabilities to Hadoop users.

Over the years, Hadoop and HDFS have faced challenges, some of which have been 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 a small number of large files for storing large data sets. If too many small files exist, the performance tends to drop. Here is a link that references the small file problem: 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, on the other hand, offers a more flexible approach. It utilizes Directed Acyclic Graphs (DAGs) to represent jobs, allowing intermediate results to stay in memory as long as possible. This in-memory processing significantly boosts performance, especially for large datasets. As a result, Spark has become 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 particularly daunting for smaller organizations lacking dedicated IT staff. The specialized skills required for implementation and management can add complexity for resource-constrained teams.
- Hadoop's MapReduce framework requires deep expertise, hindering developer productivity. While Hive and Pig provide higher-level abstractions for data access, they introduce additional complexity for managing workflows.

Key takeaways: While Hadoop and its ecosystem (MapReduce, Pig, Hive) played 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 (k8s) 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 Hadoop Distributed File System (HDFS) revolutionized big data storage when it arrived. This distributed file system, a core component of Apache Hadoop, excels at handling massive datasets across clusters. However, as data storage needs have evolved, object storage has emerged as a compelling alternative due to its inherent architectural advantages.

HDFS was designed to be 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 at all times.

Data locality was a crucial concept in Hadoop HDFS. Its purpose was to decrease the amount of data that needs 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. Here's a related article for further reading: What about locality?

For a deep dive into HDFS architecture, refer to this guide: link.

4.3 The rise of object storage

Traditionally, data resided in file systems built for structured information and limited storage needs. However, with the explosion of data, these systems became overwhelmed. This paved the way for object storage, a scalable and cost-effective solution 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*, leveraging object storage's strengths for data analysis.

Here is why object storage shines 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 your data remains secure.

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

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

- HDFS in the cloud often required overprovisioning of instances, particularly when compute needs grew slower than storage. Additionally, the constant need to store data limited dynamic scaling for cost optimization, as clusters typically ran 24/7.
- HDFS replication on instance store or EBS is significantly more expensive than 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 using erasure-coded S3. Network advancements made this feasible, eliminating the need for co-located 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 filesystems, 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, cloud forced Hadoop to evolve to embrace decoupled compute and object storage.

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

4.4 Accessing S3 Object Storage from Hadoop (S3A)

The rise of cloud storage, specifically Amazon's 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:** Following S3's launch, the initial attempt to bridge the gap was the S3 block filesystem, offering limited functionality.
- **Amazon S3 Native Filesystem (S3N):** This marked a significant step forward, allowing Hadoop to directly work with S3 data. It eventually merged with the Hadoop project.
- **Amazon's EMR and S3 Client:** Amazon's Elastic MapReduce (EMR) included its own S3 client, highlighting the growing need for seamless integration.
- **S3A:** S3A (Amazon S3A File System), an open-source S3 object storage connector, 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 filesystem clients for Hadoop reflects the increasing importance of object storage for big data processing.

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

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

From the Hadoop S3A documentation, S3A high-level feature description:

- Supports per-bucket granular configuration.
- Directly reads and writes S3 objects.
- Compatible with standard S3 clients.
- Compatible with files created by the older s3n:// client and Amazon EMR's s3:// client.
- Supports multipart uploads for multi-GB objects.

- Offers a high-performance random IO mode for working with columnar data such as Apache ORC and Apache Parquet files.
- Uses Amazon's Java V2 SDK with support for the latest S3 features and authentication schemes.
- Supports authentication via: environment variables, Hadoop configuration properties, the Hadoop key management store IAM roles and even custom credentials providers.
- Supports S3 "Server Side Encryption" for reading and writing: SSE-S3, SSE-KMS and SSE-C.
- Supports S3-CSE client-side encryption.
- Instrumented with Hadoop metrics.
- Actively maintained by the open-source community.

While S3A makes every attempt to map the HDFS API closely to the S3 API, there are some semantic differences inherent to object storage (Reference: Hadoop Wiki):

- **Data consistency**: Eventual consistency guarantees that all replicas of your data will eventually reflect the latest changes. However, there might be a temporary delay before updates are propagated across all locations. This means 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). Ceph utilizes 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.

For further information, check this great article by Kyle Bader The anatomy of the S3A filesystem. There is also detailed information on the S3A connector/client on the upstream documentation following this link.

4.5 IBM Storage Ceph Object Storage integration with Hadoop

Hadoop and Spark use the popular S3A interface to access data in an S3 object store like IBM Storage Ceph, instantly unlocking all of the 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 STS.
- Multi-cluster federated bucket namespaces.
- Scalability to billions of objects.
- Unmatched security:
  - FIPS 140-2 validated cryptography for in-transit and server-side encryption1[1]
  - IAM and bucket policy
  - STS with OIDC integration
  - Bucket versioning
  - Object lock
  - MFA delete

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

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

- With HDFS, connecting different versions of Hadoop and Spark to query the same dataset is impossible. With S3A and object storage, we can connect different tooling versions and query the same dataset, providing great flexibility to the data scientists.
- Decoupling of Storage and Compute: With Hadoop and HDFS, a tight coupling between storage and compute will not allow scaling the components individually. With the S3A integration and IBM Storage Ceph, each component, compute, and storage are decoupled and can be scaled independently of the others as needed.
- Notably, IBM Storage Ceph provides erasure-coded pools that offer higher efficiency than the initial HDFS replication scheme. This leads to a significant improvement in storage optimization and cost-effectiveness.
- IBM Storage Ceph can offer out-of-the-box multi-site 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, Server side encryption (`SSE-KMS,SSE-C,SSE-S3`), MFA delete, audit logs and object lock providing immutability.

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 using IBM Storage Ceph's Object Storage capabilities with S3A.

Joint efforts between IBM Storage Ceph Engineering and our customers have developed and extended the S3A functionalities in different areas, for example, A custom Identity Provider that enhances the integration of S3A with the Secure Token Service (STS) S3 feature. This enhancement allowed users to remove and transform their RBAC authorization framework into a new full-fledged Attribute-based access control provided by the STS/IAM features in IBM Storage Ceph Object.

The combination of Secure Token Service Authentication with the flexibility of attribute-based access control dramatically simplifies the application authentication policy.

To get detailed information on IBM Storage Ceph authentication and authorization schemes with STS and IAM roles, refer to the IBM Redpaper *IBM Storage Ceph Solutions Guide*, REDP-5715. For a hands-on example of implementing an ABAC policy, take a look at this article.

4.7 Example of connecting Hadoop with IBM Storage Ceph using S3A

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

To provide some context to the example environment, we set up a simple Hadoop environment on a RHEL node called “linux1.” We also have an IBM Storage Ceph Cluster up and running; our S3 endpoint is set, and the configured S3 restful API endpoint is `https://s3.cephlabs.com`.

1. Let us start by setting up Ceph for object storage. We will create a new user specifically for Hadoop access. This user will have a UID and display name of "Hadoop". See Example 4-1

Example 4-1 Create a new 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": "sxAILYNUgsXuUhIIDsVTCylAAJbaUf6EHNTAN5A1"
        }
    ],
    "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. We have already configured and integrated a KMIP-compliant key manager IBM Life Cycle Key manager(GKLM) with our IBM Storage Ceph deployment so that we can provide encryption at rest for our datasets:

Example 4-2 Provide encryption at rest for the datasets

```
# 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. In this step, we will leverage the AWS CLI to validate our S3 endpoint and ensure that it is operational as intended. We will use the access key and secret key provided when creating the Hadoop user for authentication. See Example 4-3.

Example 4-3 Validate the S3 endpoint and ensure it is operational as intended

```
$ grep -A 2 hadoop .aws/credentials
[hadoop]
aws_access_key_id = QN71C16RLCRTCQT6WEUC
aws_secret_access_key = sxAILYNUgsXuUhIIDsVTCylAAJbaUf6EHNTAN5A1
$ aws --endpoint https://s3.cephlabs.com --profile hadoop  s3 mb s3://hadoop/
--region default
make_bucket: hadoop
```

4. Our credentials (access key and secret key) successfully authenticated with the S3 endpoint. We created a new bucket called `hadoop` to store our objects. As a final check, let us upload an object and encrypt it at rest using SSE-KMS with the key ID “`rgw00dc42a9b000000000`”. See Example 4-4.

Example 4-4 Upload an object and use 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, we can configure an S3 bucket encryption policy. This policy automates the process, transparently encrypting every object uploaded to the bucket. This eliminates the need for manual encryption on each object upload, streamlining the workflow for users. See Example 4-5.

Example 4-5 Configure an S3 bucket encryption policy

```
$catkms-config.json
"Rules": [
    {
         "ApplyServerSideEncryptionByDefault": {
              "KMSMasterKeyID": "rgw00dc42a9b000000000",
              "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 could configure SSE-KMS encryption by using the S3A integration (`fs.s3a.encryption.algorithm,fs.s3a.encryption.key`). In this example, we have set the encryption at the level of the IBM Storage Ceph bucket. This means any object uploaded to this bucket will be automatically encrypted.

   Let us configure the Hadoop side first. The `hadoop-aws` module must be enabled in the `hadoop-env.sh` file. See Example 4-7 on page 58.

Example 4-6 Enable the hadoop-aws module

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

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

   Note: There are several ways to authenticate with S3A.

   - **Authentication providers:** These authentication providers offer pre-configured options streamline setup.
   - **Environment variables:** Set environment variables to provide your credentials directly.
   - **AWS credential files:** Use the standard AWS credential file format for authentication.

   For enhanced security, consider the Hadoop credentials provider. It creates an encrypted credentials provider to safeguard your access keys.

   We are also configuring Ceph Object Storage SSE-KMS (Ceph's built-in encryption), so all the data we upload to Hadoop using S3A will be encrypted at rest. See Example 4-7.

Example 4-7 Configuring CephObject 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>6N85CBHVHKEZ6EMOPYJH</value>
    </property>
    <property>
      <name>fs.s3a.secret.key</name>
      <value>svP1GMkpGO1gRkmdhask3fuVTvPulZZUk5lc5yTF</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, we restart the Hadoop services. Once they are up and running, we verify if we can access the `hadoop` bucket using S3A from our Hadoop application. See Example 4-8.

Example 4-8 Check if we can access the hadoop bucket 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 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 could be *partial migration*, where we decide to keep the old data in HDFS and only migrate the new incoming datasets into object storage. We could also do a *complete migration* of all of our data from HDFS to S3, and once we have finished, we can deprecate our HDFS storage nodes.

We will discuss three different techniques for HDFS to S3 migration in this section.

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 backends seamlessly. This is accomplished by specifying the appropriate protocol. In the case of S3, the protocol scheme is “`s3a://`”, and for HDFS, it is “`hdfs://`”.

By specifying the correct protocol at runtime, applications can easily transition between HDFS and S3 storage backends without complex modifications to the application's underlying code. This simplifies migration, enabling developers to update their applications and start using S3 easily.

We performed the following steps for HDFS to S3A migration in our environment:

1. First, we access the data on the HDFS filesystem with the "`hdfs://`" prefix. See Example 4-9 on page 60.

Example 4-9 Access the data on the HDFS filesystem

```
$ 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. Let us copy the `hosts` file from HDFS to S3. For S3, we just need to reference the "`s3a://`" prefix with our bucket name, as shown in Example 4-10.

Example 4-10 Copy 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 easy, as shown in Example 4-11. Simply specify a prefix to direct data to your preferred storage location. If you do not provide a prefix at runtime, the default storage location set in your `core-site.xml` file will be 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:

- While you can change the default storage to S3A as shown in Example 4.8.1, keep in mind that this approach requires specifying a bucket name for each operation. This 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 datasets to S3A while keeping HDFS for existing code. This allows you to leverage S3A's advantages incrementally while minimizing code changes.
- Two further migration strategies are presented in 4.8.2, “DistCp tool” on page 61 and 4.8.3, “Application-layer tools for data movement” on page 63.

4.8.2 DistCp tool

Hadoop provides a great tool to migrate or copy data between different sources called *DistCp*. DistCp can be used to move vast amounts of data between different storage backends, so it is a perfect match for migrating our 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.”

In this section, we will provide a simplified example of a data migration from HDFS to S3A using the DistCp tool.

1. We will utilize Hive external tables to achieve this migration. We configure a Hive external table pointing to a .csv file called `ok.csv`, stored in our HDFS filesystem. We want 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` resides. This CSV file contains data about countries, including their capitals and populations.

   Benefits of external tables: This approach offers several advantages:

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

Example 4-12 Creation of a Hive external table for reference

```
$ hadoop fs -cat hdfs://localhost:9000/countries/ok.csv
1,USA,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  USAWashington328
2  FranceParis67
3  SpainMadrid47
4  RussiaMoscow145
5  IndonesiaJakarta267
6  NigeriaAbuja196
Time taken: 1.533 seconds, Fetched: 6 row(s)
```

2. We will copy our dataset from HDFS to S3. We will create a bucket named `bucketdata` on the IBM Storage Ceph S3 using the Hadoop user. This bucket will be our destination target in the **`distcp`** command. See Example 4-13.

Example 4-13 Copy the dataset from HDFS to S3

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

3. Now, we can run the **`distcp`** to copy our data from HDFS source to S3 destination. Distcp can be tuned depending on how much data we need to copy and the available resources to use more copy threads and reduce the data migration time to completion. See Example 4-14.

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

```
$ hadoop distcp hdfs://linux1:9000/countries/ s3a://bucketdata/
024-04-03 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. Once our dataset has been migrated, we can modify the location of our table to start using the new S3A location. See Example 4-15.

Example 4-15 Modify 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. Once modified, we can check that our queries work fine against our dataset, which is now stored in our IBM Storage Ceph Cluster accessible through the S3 protocol. See Example 4-16.

Example 4-16 Check that the queries work fine against our dataset

```
hive> select * from countries_list ;
OK
1  USAWashington328
2  FranceParis67
3  SpainMadrid47
4  RussiaMoscow145
5  IndonesiaJakarta267
6  NigeriaAbuja196
Time taken: 0.144 seconds, Fetched: 6 row(s)
```

4.8.3 Application-layer tools for data movement

While 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.

Let us consider the example in 4.8.2, “DistCp tool” on page 61 to process external tables containing data in CSV files stored on HDFS. In this scenario, we can use Presto, a powerful SQL query engine included in theIBM watsonx.data bundle. Presto allows you to directly query the external CSV files in HDFS using standard SQL statements. You can then use an `INSERT INTO` statement to efficiently load the results into a destination table stored within an S3 bucket.

When the SQL executes, this action not only 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 allows you to leverage existing tools within your application ecosystem for data movement.
- **Transformation on the fly**: Data is transformed into a more efficient format (Parquet) during migration, potentially improving storage efficiency and query performance.

We will be sharing an example of this procedure in Chapter 8, “Transform: Staging and curated zones” on page 121, where we will 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 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 they needed.

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

- **Reduced costs**: Separating storage from compute resources allows for independent scaling, optimizing resource utilization and lowering 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 consisted of proprietary hardware that provided appliance-style servers with integrated compute and storage. The solution also supported several Hadoop ecosystem tools.

The existing solution presented the retailer with several distinct challenges:

Scaling storage and compute independently was not possible. Consequently, the retailer purchased additional 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 dictated that only one version of the Hadoop ecosystem could be available to all platform users. This frustrated the data users, as the data scientists would typically look to use the latest versions. In contrast, the data engineers wanted older versions of tools because they offered stability.

With a 24-hour-a-day business, maintenance windows were becoming impossible to manage as any updates to hardware or software would require the whole data system to be offline for an extended period.

The maintenance costs were becoming excessive as their existing vendor wanted to maintain the whole data estate regardless of the Hadoop ecosystem tools used by the retailer. This 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 needed a high-performance, highly scalable storage solution that could be flexible to meet their needs while handling large amounts of unstructured data.

4.9.3 S3A Ceph-based solution

The customer chose Ceph Storage for their underlying storage solution.

The Ceph-based solution consisted of the following elements:

- The current Hadoop community ecosystem tooling would continue to be used by the retailer, providing limited internal support but, most importantly, allowing users to choose which version of the tool they would like to use.
- Commodity x86 servers and disks would be purchased from their existing hardware provider for Ceph.
- Ceph Storage would provide the underlying storage layer for the unstructured data lake.
- Red Hat Single Sign-on was used to integrate authentication and authorization within Ceph to existing customer systems.
- Red Hat Service Registry defined the data structure of the ingress and egress messages to the Hadoop applications.
- Red Hat Enterprise Linux is the underlying operating system for the Ceph Storage system.
- The existing hardware supplier would provide hardware support for the physical assets.
- Ceph Object Storage multisite replication would be 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 (Ceph) by adding new servers. Ceph automatically rebalances data across the cluster, ensuring efficient utilization.
- **Multi-Hadoop version support**: Different versions of Hadoop can access the same dataset. This allows data engineering and data science teams to manage their preferred Hadoop versions independently, fostering flexibility and innovation.
- **Improved efficiency and cost savings**: This approach enhances functionality and performance while significantly reducing storage costs.

5

IBM Storage Ceph with IBM watsonx.data

To help businesses unlock the true potential of their data, IBM created 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 and discusses the following topics:

- “Introducing IBM watsonx.data” on page 68
- “IBM watsonx.data infrastructure components” on page 69
- “Typical use cases for IBM watsonx.data” on page 70
- “IBM watsonx.data user interface” on page 71

5.1 Introducing IBM watsonx.data

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

However, most organizations continue to struggle with fundamental data challenges, such as the ever-growing volume of data, the variety of 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 in the hands of those that need it, with the uses and users of data becoming more varied than ever before.

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

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

Figure 5-1 shows the IBM watsonx platform.

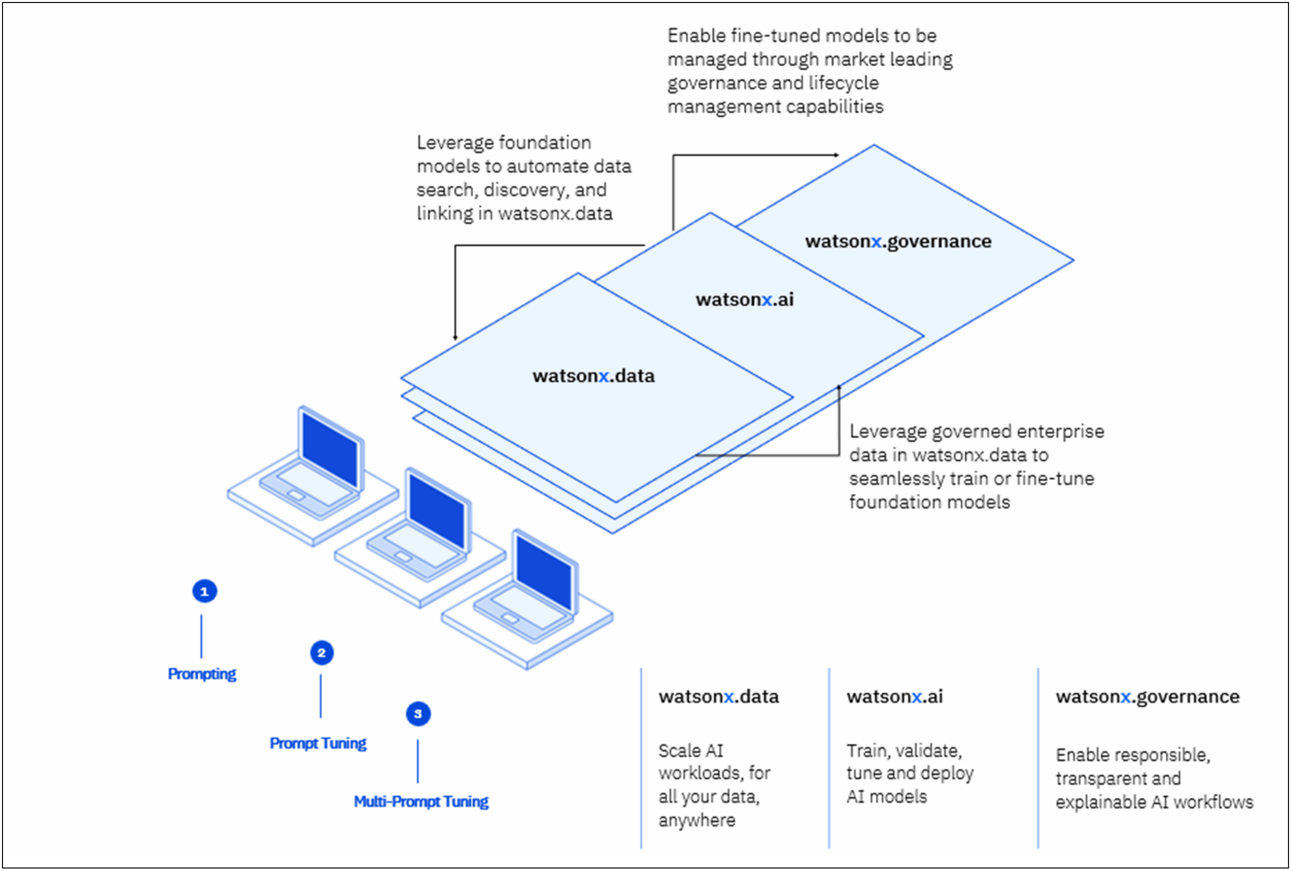
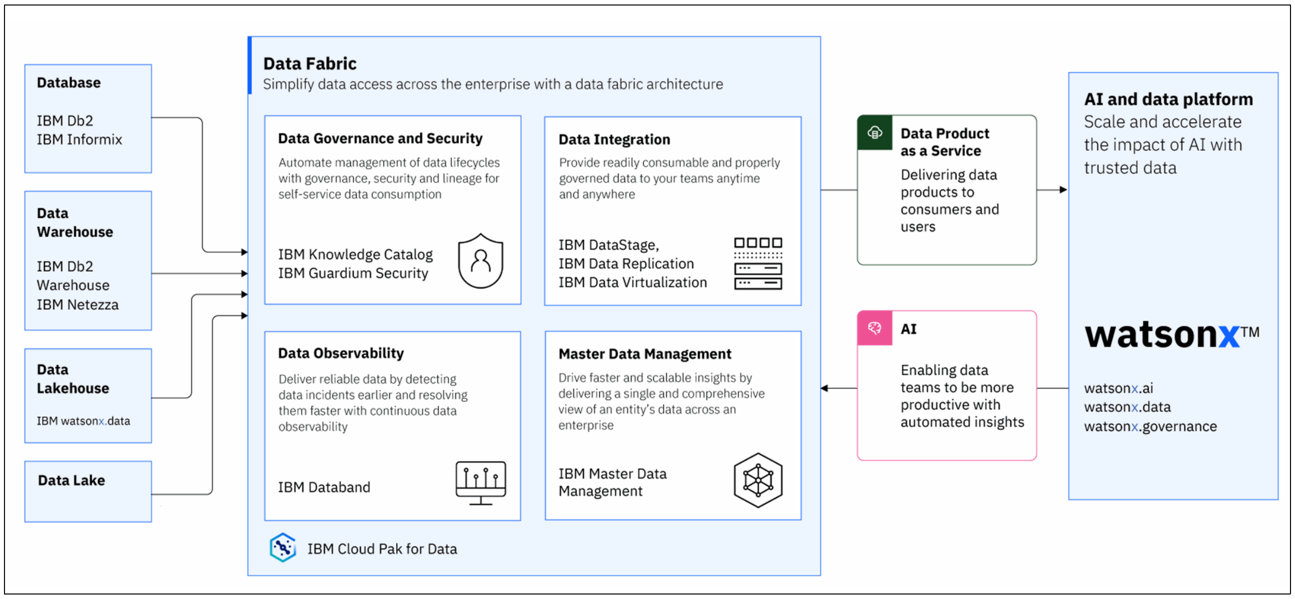

Enable fine-tuned models to be managed through market leading governance and lifecycle management capabilities
Leverage foundation models to automate data search, discovery, and linking in watsonx.data
watsonx.governance
watsonx.ai
watsonx.data
Leverage governed enterprise data in watsonx.data to seamlessly train or fine-tune foundation models
1
Prompting
2
Prompt Tuning
3
Multi-Prompt Tuning
watsonx.data
Scale AI workloads, for all your data, anywhere
watsonx.ai
Train, validate, tune and deploy AI models
watsonx.governance
Enable responsible, transparent and explainable AI workflows

Figure 5-1 IBM watsonx platform

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

With the foundation of IBM Cloud Pak® for Data's AI and data platform, watsonx.data 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.


Database
IBM Db2
IBM Informix
Data Warehouse
IBM Db2 Warehouse
IBM Netezza
Data Lakehouse
IBM watsonx.data
Data Lake
Data Fabric
Simplify data access across the enterprise with a data fabric architecture
Data Governance and Security
Automate management of data lifecycles with governance, security and lineage for self-service data consumption
IBM Knowledge Catalog
IBM Guardium Security
Data Integration
Provide readily consumable and properly governed data to your teams anytime and anywhere
IBM DataStage,
IBM Data Replication
IBM Data Virtualization
Data Observability
Deliver reliable data by detecting data incidents earlier and resolving them faster with continuous data observability
IBM Databand
Master Data Management
Drive faster and scalable insights by delivering a single and comprehensive view of an entity's data across an enterprise
IBM Master Data Management
IBM Cloud Pak for Data
Data Product as a Service
Delivering data products to consumers and users
AI
Enabling data teams to be more productive with automated insights
AI and data platform
Scale and accelerate the impact of AI with trusted data
watsonx™
watsonx.ai
watsonx.data
watsonx.governance
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 via data fabric services. The AI teams leverage generative AI and machine learning models which are tuned with responsible, transparent and explainable data workloads to improve the productivity with automated insights.

5.2 IBM watsonx.data infrastructure components

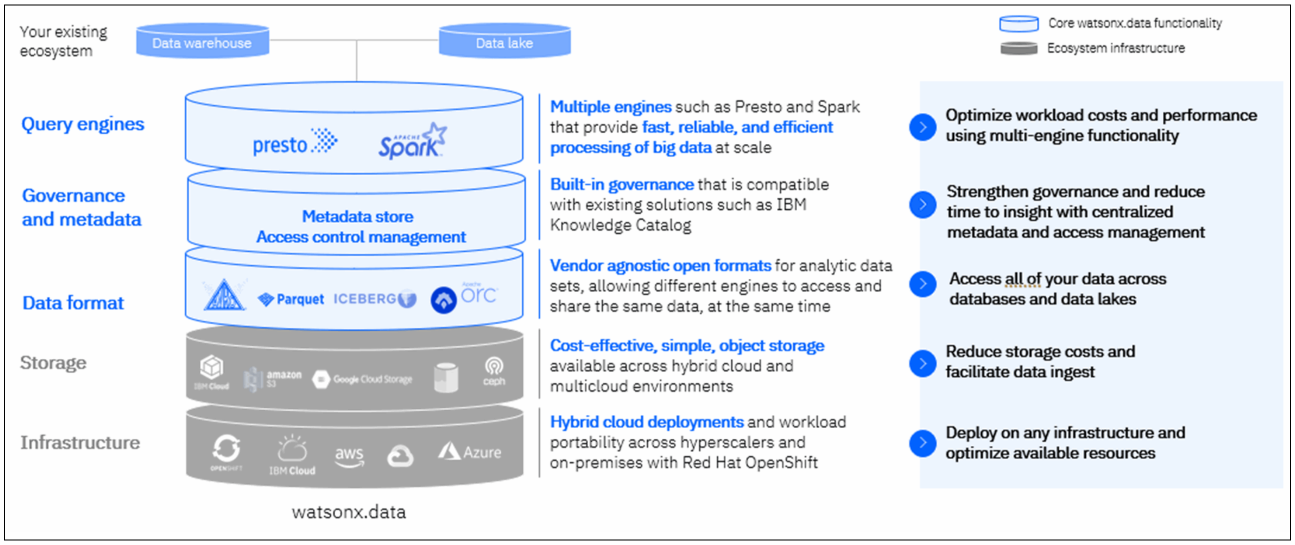
Built on Red Hat OpenShift, watsonx.data is cloud-agnostic, supporting deployment on-premises or any major cloud provider like IBM Cloud, AWS, Azure, and GCP.

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 Netezza®.
- **Catalogs**: Metadata catalogs are used to manage table schemas and metadata for the data residing in watsonx.data.
- **Buckets**: watsonx.data stores data in object storage at a fraction of the cost compared to the traditional block storage found in high-performance data warehouse. Specifically, data is stored in S3-compatible buckets, which are identified storage areas within object storage, similar to file folders. AWS Simple Storage Service (S3), IBM Storage Ceph, IBM Cloud Object Storage (COS), 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's lakehouse architecture and its value proposition.



Multiple engines such as Presto and Spark that provide fast, reliable, and efficient processing of big data at scale

Built-in governance that is compatible with existing solutions such as IBM Knowledge Catalog

Vendor agnostic open formats for analytic data sets, allowing different engines to access and share the same data, at the same time

Cost-effective, simple, object storage available across hybrid cloud and multicloud environments

Hybrid cloud deployments and workload portability across hyperscalers and on-premises with Red Hat OpenShift

Optimize workload costs and performance using multi-engine functionality

Strengthen governance and reduce time to insight with centralized metadata and access management

Access all of your data across databases and data lakes

Reduce storage costs and facilitate data ingest

Deploy on any infrastructure and optimize available resources

Figure 5-3 IBM watsonx.data's 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:

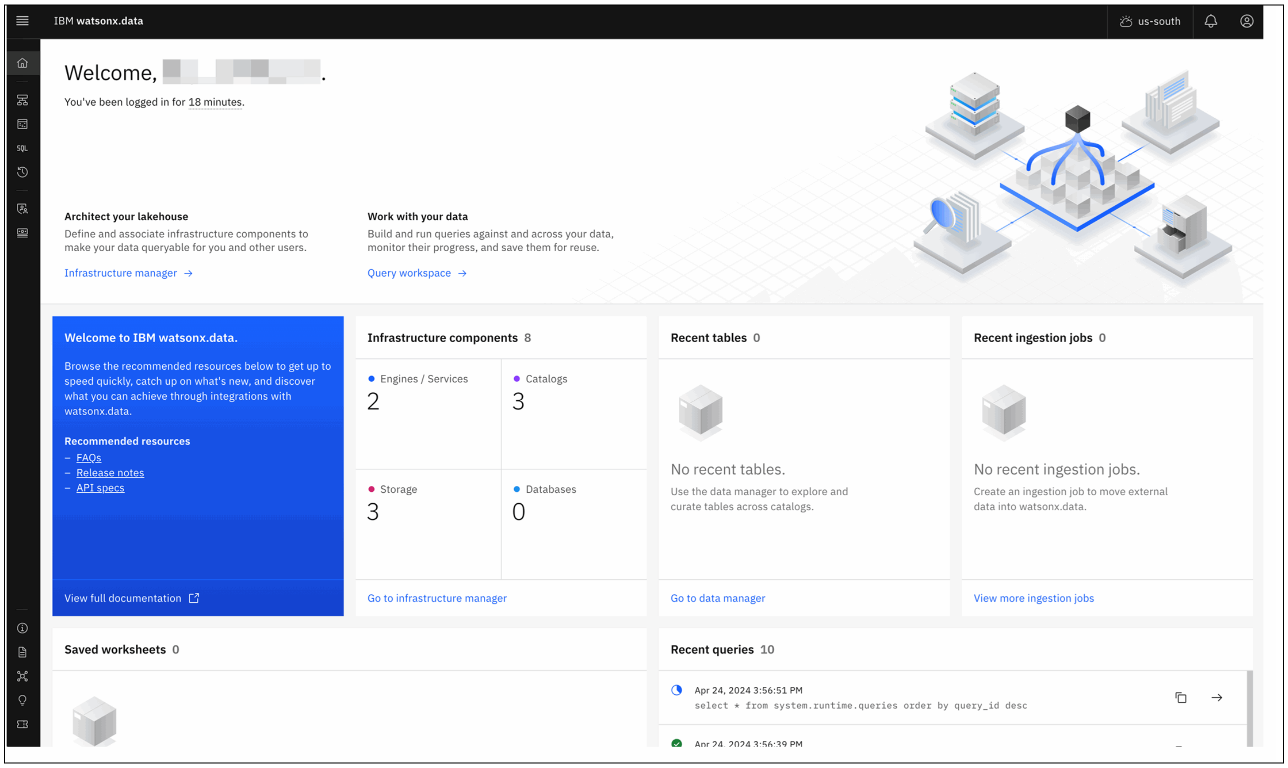
- **Data warehouse optimization**: Optimize workloads from the data warehouse by choosing the right engine for the right workload at the right cost, replace *extract/transform/load (ETL)* jobs and reduce costs of your data warehouse through workload optimization.
- **Data lake modernization**: Augment Hadoop data lakes 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 business intelligence (BI)**: Connect existing data with new data in minutes and unlock new insights. Integrate with IBM Cognos® and other third-party BI and dashboarding 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**: Easily virtualize or replicate data to Iceberg tables for analytics.
- **Datastore 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 infused in watsonx.data to find, augment, and visualize data and unlock new data insights through a conversational interface - no SQL required. Unleash cryptic structured data using auto-generated semantic metadata in natural language for easy self-service access to data.

5.4 IBM watsonx.data user interface

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

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

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



Welcome, [redacted].

You've been logged in for 18 minutes.

Architect your lakehouse
Define and associate infrastructure components to make your data queryable for you and other users.
Infrastructure manager →

Work with your data
Build and run queries against and across your data, monitor their progress, and save them for reuse.
Query workspace →

Welcome to IBM watsonx.data.
Browse the recommended resources below to get up to speed quickly, catch up on what's new, and discover what you can achieve through integrations with watsonx.data.

Recommended resources
– FAQs
– Release notes
– API specs

View full documentation

Infrastructure components 8
Engines / Services 2
Catalogs 3
Storage 3
Databases 0
Go to infrastructure manager

Recent tables 0
No recent tables.
Use the data manager to explore and curate tables across catalogs.
Go to data manager

Recent ingestion jobs 0
No recent ingestion jobs.
Create an ingestion job to move external data into watsonx.data.
View more ingestion jobs

Saved worksheets 0

Recent queries 10
Apr 24, 2024 3:56:51 PM
`select * from system.runtime.queries order by query_id desc`
Apr 24, 2024 3:56:39 PM

Figure 5-4 IBM watsonx.data console home

This home page contains the following panels:

- **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 have recently been explored.
- **Recent ingestion jobs**: Frequently run queries saved as worksheets, for easier reuse.
- **Recent queries**: Queries that have recently been run or are in the process of being run.

Tip: Left side icon menu give access to the different console pages.

5.4.1 Infrastructure manager

Adding buckets is done through the Infrastructure manager page. This page includes a graphical canvas view of the different infrastructure components currently 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 residing in an embedded object storage service, and catalogs corresponding to those buckets.

Figure 5-5 shows the Infrastructure manager page.

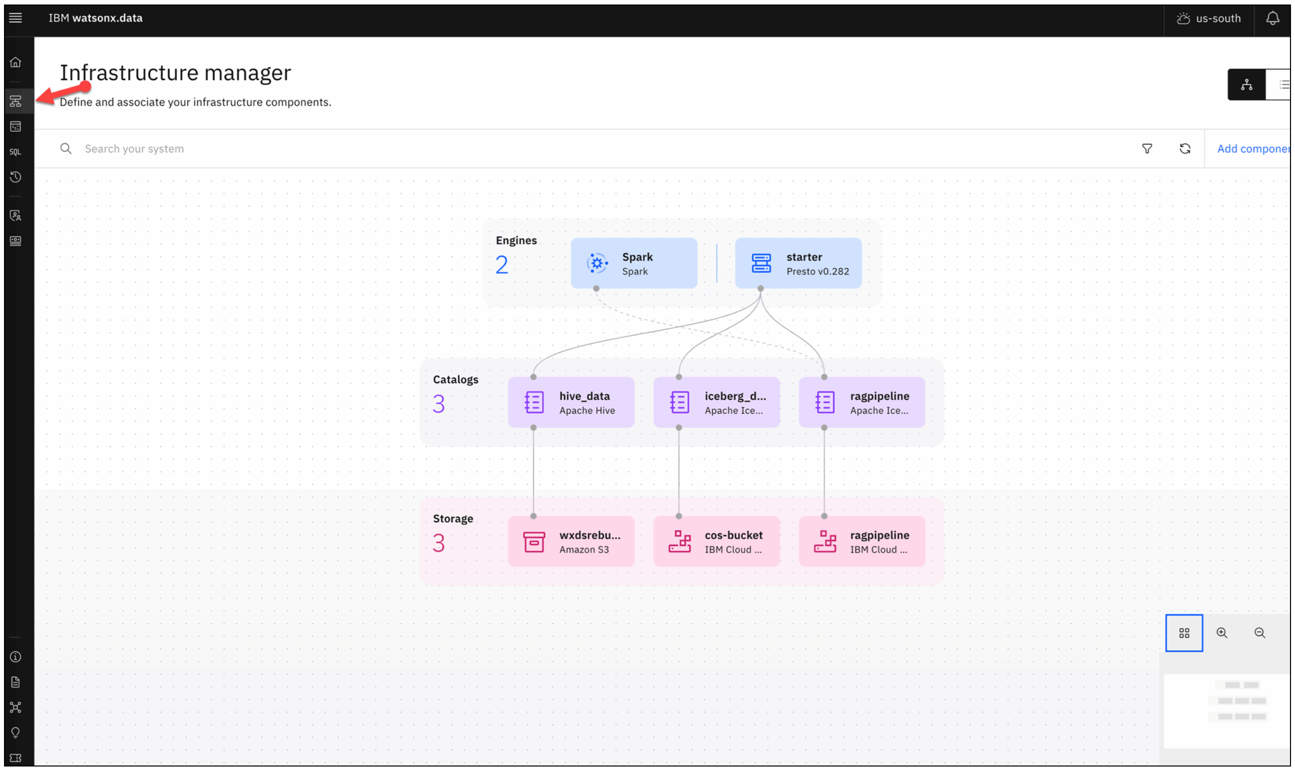


Figure 5-5 Infrastructure manager

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 at the same time. The catalog is how you reference the bucket within the user interface and in Structured Query Language (SQL) statements.

In addition to adding buckets, you can also add databases. This lets Presto access 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 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 as well, similar to adding a bucket. This allows you to reference that database's data within watsonx.data's user interface and from SQL queries.

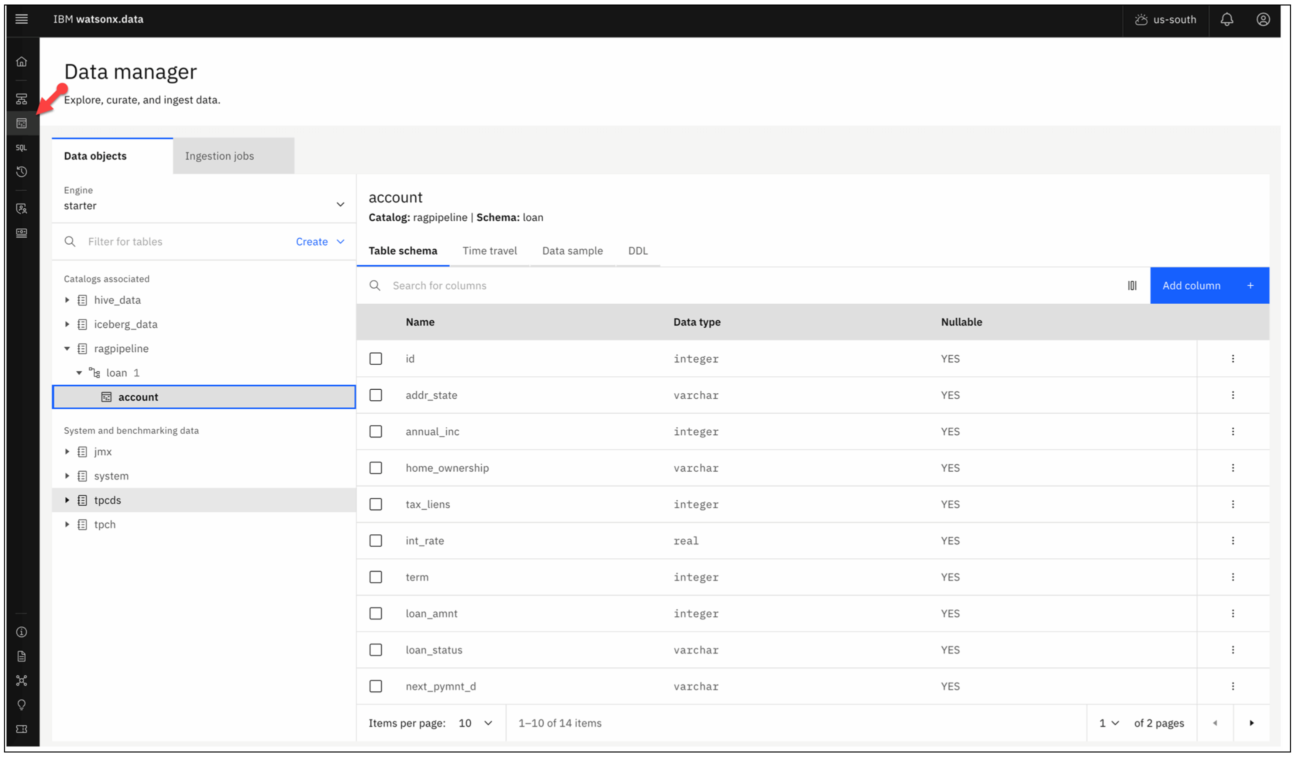
Catalogs can be Apache Iceberg, Apache Hive or Delta Lake. Generally speaking, tables created and populated with watsonx.data should use the Iceberg open table format because they inherit beneficial capabilities like transactional consistency, schema and partition evolution, snapshots for time travel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

5.4.2 Data manager

Within the watsonx.data user interface, the Data manager page can be used to explore and curate your data. This includes the ability to create new schemas and tables, drop schemas and tables, and load data into tables.

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

Figure 5-6 shows the Data manager page.



Data manager

Explore, curate, and ingest data.

Catalogs associated
- hive_data
- iceberg_data
- ragpipeline
  - loan 1
    - account

System and benchmarking data
- jmx
- system
- tpcds
- tpch

account

Catalog: ragpipeline | **Schema:** loan

Name	Data type	Nullable
id	integer	YES
addr_state	varchar	YES
annual_inc	integer	YES
home_ownership	varchar	YES
tax_liens	integer	YES
int_rate	real	YES
term	integer	YES
loan_amnt	integer	YES
loan_status	varchar	YES
next_pymnt_d	varchar	YES

Items per page: 10 1–10 of 14 items 1 of 2 pages

Figure 5-6 Data manager page

Catalogs correspond to the data sources added to watsonx.data. Specifically, this includes the object storage buckets and remote data sources added on the infrastructure manager page. In Presto, tables can be referenced in queries and other structured query language (SQL) statements by specifying the catalog, schema, and table name, separated by periods. For example: `SELECT COUNT(*) FROM DATA_CATALOG.SALES_SCHEMA.CUSTOMER_TABLE`.

The top-level navigation point allows to select the engine to work with. In Figure 5-6 only one Presto engine is available. However, in environments with multiple Presto engines defined, administrators can choose any one, while non-administrators can select any engine they have been granted access to.

Once the engine is selected, we can navigate through the catalogs associated with it. When a catalog contains a dataset, we can navigate through schemas and tables.

Data manager page can also be used to create schema and upload a data file to define and populate it.

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

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

Iceberg's *time travel feature* leverages snapshots, 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 new ones. Additionally, 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, the workspace allows you 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 assist in building new ones.

- **Build schemas and tables**: Define the structure for your data 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 navigation menu on the left. This menu allows you to easily select objects of interest and generate corresponding SQL statements. For instance, you can quickly create a `SELECT` statement to retrieve data from a specific table.

The right side of the page features the SQL editor pane. Here, you can create and edit your SQL statements, execute them using Presto, and view the resulting data.

Figure 5-7 on page 75 shows the Query workspace.

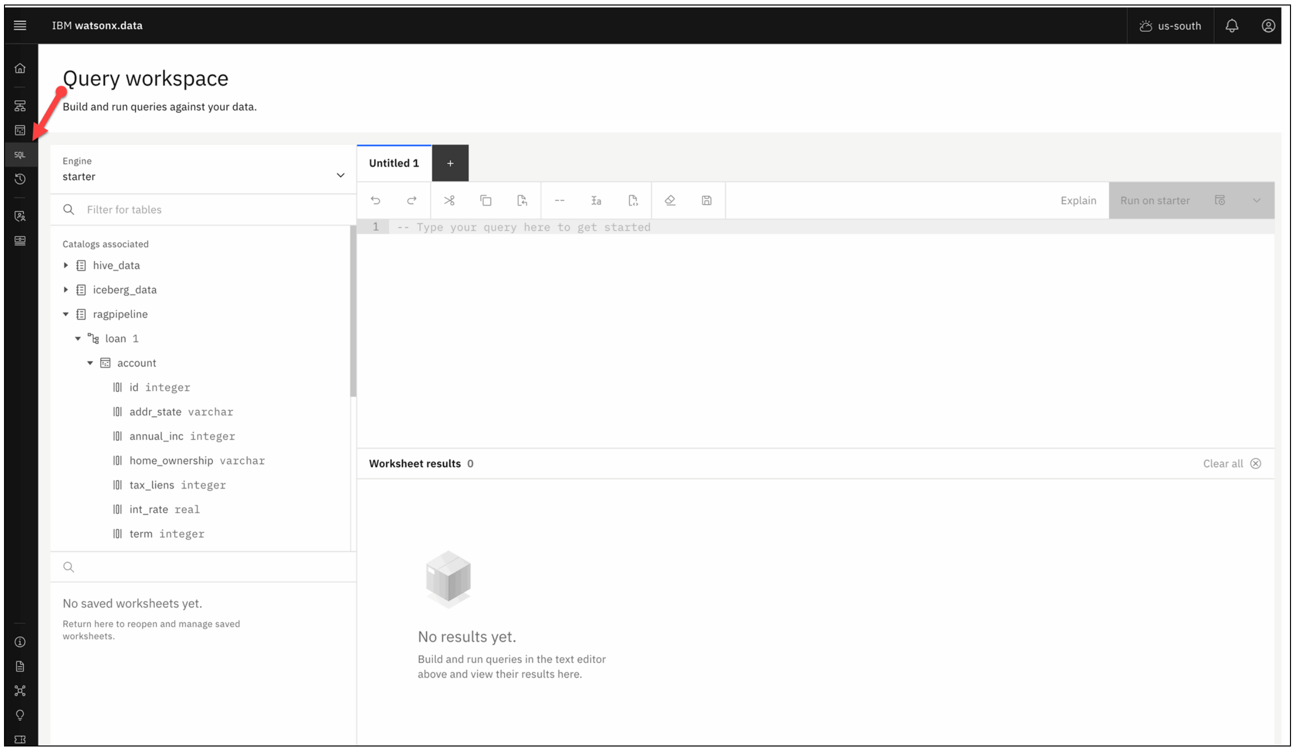


Figure 5-7 Query workspace

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

- **Presto command-line interface (CLI)**: For advanced users, Presto offers its own terminal-based CLI for executing SQL queries directly.
- **JDBC drive**r: Developers can leverage Presto's Java Database Connectivity (JDBC) driver to integrate "homegrown" and third-party applications seamlessly with watsonx.data tables and data. This allows these applications to directly interact with your data using SQL.

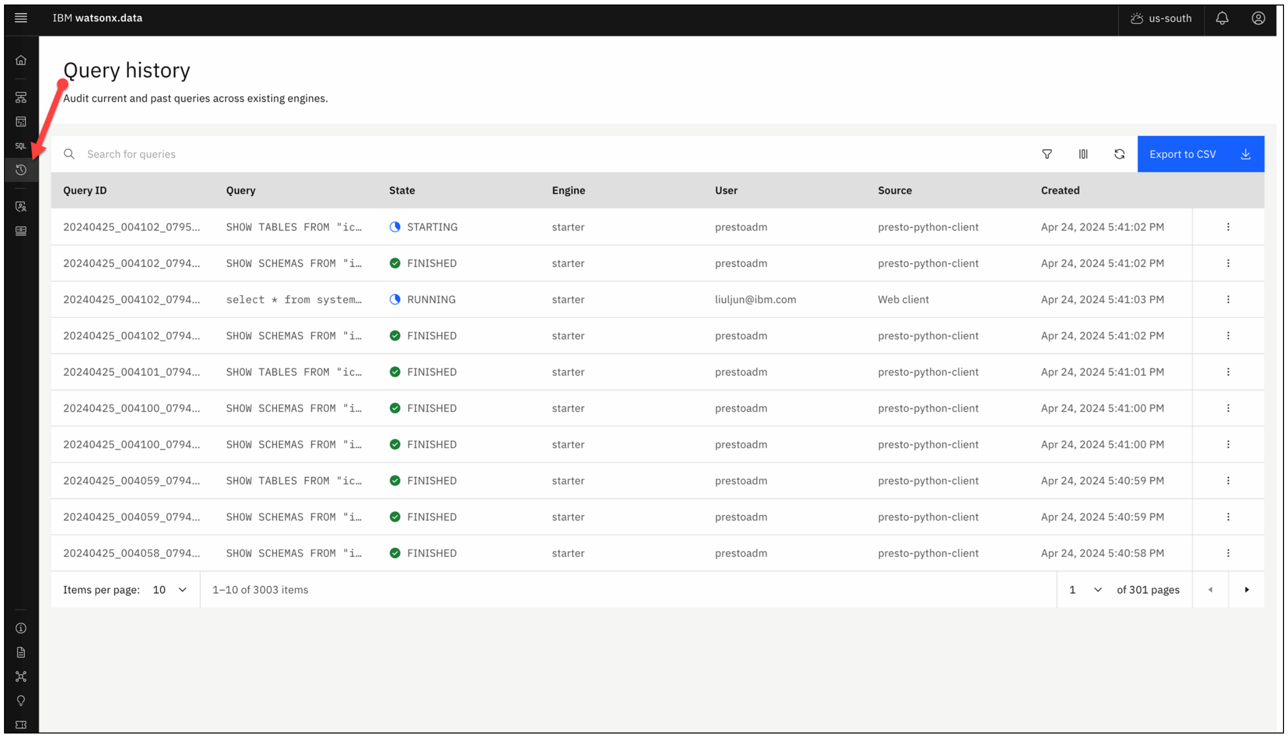
Note: You can also run SQL statements using Spark SQL.

5.4.4 Query history

Using the Query history page, we can audit currently running queries and queries that ran in the past, across all the engines defined in the environment. This includes queries that users have explicitly run, as well as queries used in the internal management and function of the environment.

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

Figure 5-8 on page 76 shows the Query history page.



Query history

Audit current and past queries across existing engines.

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	liuljun@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 page

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

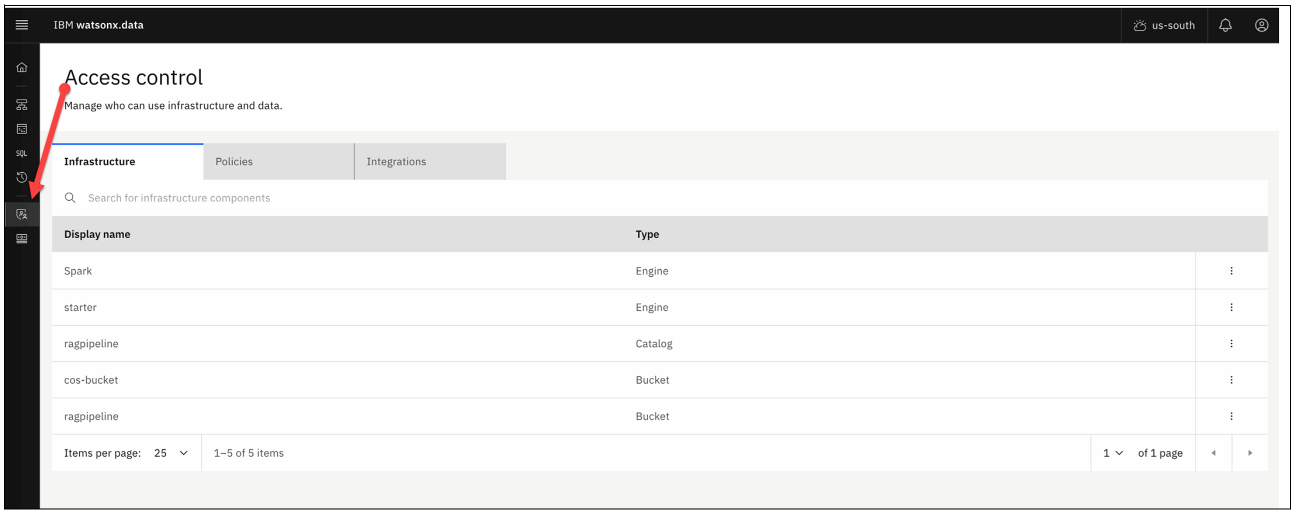
5.4.5 Access control

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, offering a tiered security system that safeguards your data while maintaining an intuitive and user-friendly experience:

- **Infrastructure access control:** It defines role based access control to resources, such as engines, catalogs, buckets and external databases.
- **Engines:** These resources represent compute engines (currently only Presto) that are appropriately configured and sized for a particular set of data access use cases. These engines are associated with *catalogs* that describe the data artifacts present inside physical storage - buckets and external databases.
- **Catalogs:** Catalogs represent the core *container* of data accessible via engines using SQL. Catalogs also identify further organization into schema tables and columns.
- **Buckets:** Object Storage buckets are physical locations where data files are maintained. Typically, these are also organized using specific table formats (Apache Iceberg is the primary one). These table formats and file formats such as Parquet/Avro/Orc also include additional metadata about the contents of these tables. Buckets are represented by a *logical catalog* for use via SQL within an engine. Buckets can be located in different sites/regions/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 using federated SQL. This allows you to seamlessly access and query data from these databases alongside your data in buckets. Additionally, these

external databases are represented as logical catalogs, enabling them to be attached to multiple engines simultaneously. This means 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 page.


IBM watsonx.data
us-south
Access control
Manage who can use infrastructure and data.
Infrastructure
Policies
Integrations
Search for infrastructure components

Display name	Type
Spark	Engine
starter	Engine
ragpipeline	Catalog
cos-bucket	Bucket
ragpipeline	Bucket

Figure 5-9 Infrastructure Access control page

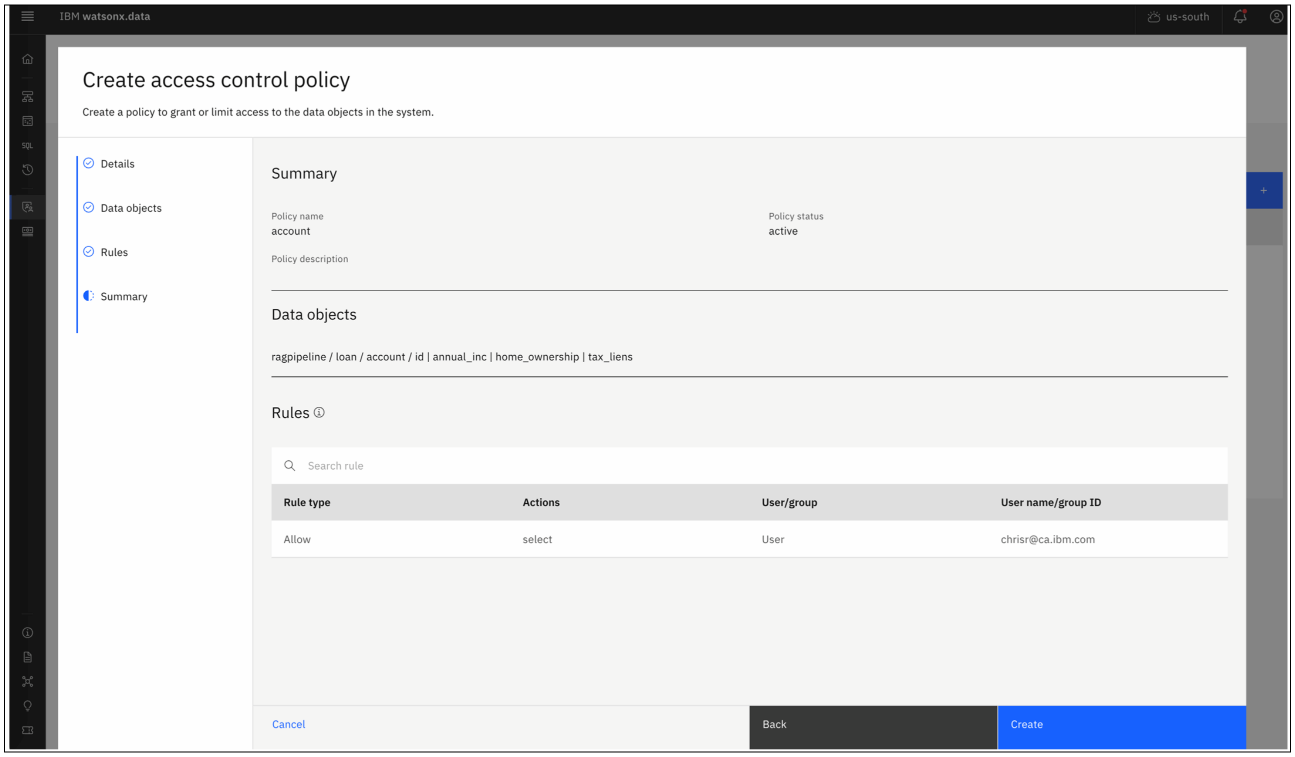
IBM watsonx.data empowers you to define access policies that govern what data users can see and manipulate. These policies determine:

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

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

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

Figure 5-10 on page 78 shows the Access policies page.



Create access control policy

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

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

Figure 5-10 Access policies

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

6

Retail use case

This chapter dives into a practical example of a data pipeline leveraging an S3 data lake. We will use IBM Storage Ceph Object Storage as the central repository for all your analytical datasets.

We will guide you through the pipeline's stages: *Ingest*, *Transform*, and *Consume*. Hands-on examples will be provided at each step to illustrate the process.

This chapter explores the various zones and their functionalities within a data lake architecture. We will demonstrate how IBM Storage Ceph's capabilities address the key requirements of modern data lakes. By showcasing the integration between IBM Storage Ceph and IBM watsonx.data, we'll unveil how this powerful combination effectively stores, manages, and unlocks the potential of your analytical data.

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:

- **Deepen 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

Our objective is to develop a data-driven strategy by analyzing client purchases:

- **Dashboards:** To visualize data to help execs make better decisions.
- **SQL query:** Curated table views per business unit line.
- **Online app:** More innovative customer e-commerce with personalized offers.

We will use basic datasets to make our examples easy to follow and understand. These simplified retail datasets are generated daily from the branch's physical stores in CSV format. There will be two types of CSV files that will be ingested from the branches. One will contain personally identifiable information (PII), while the other will provide 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 our data lake architecture.

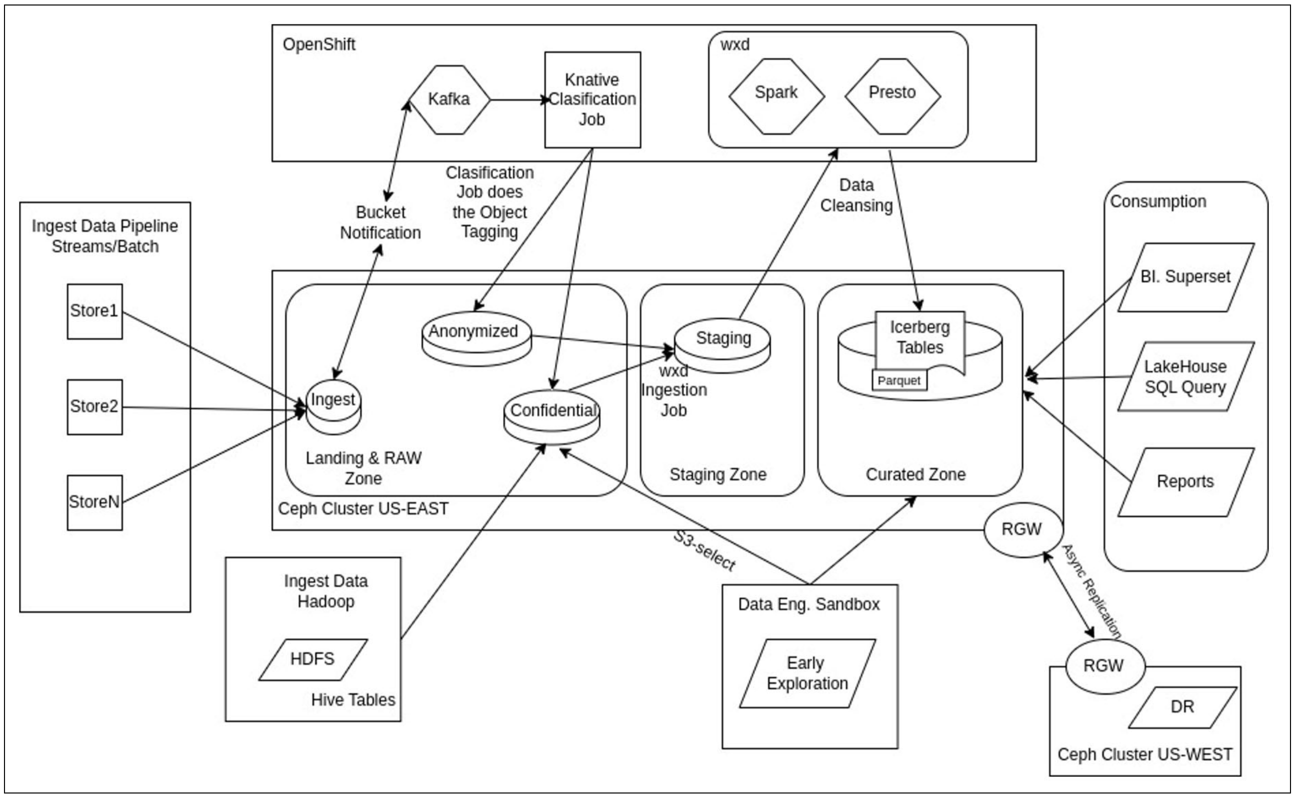


Figure 6-1 Point of Sales (POS) example use case for the modern data lake

From left to right, we have the following sections:

- **Ingest**: Landing and raw zone
  - We are ingesting 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 sessions data are generated here.
    - **Customer feedback:** Data collected from in-store kiosks, online surveys, and feedback forms.
  - The characteristics of the ingested data are:
    - Sales records.
    - Product information.
    - Customer details.
    - Inventory status.
    - On-line browsing logs.
  - The central site utilizes an IBM Storage Ceph S3 bucket as a temporary landing zone to store daily data received from branches.
  - Ingested data gets classified based on its confidentiality and stored in buckets within the raw zone.

   - The raw zone confidential bucket ingests data from Hive tables available on Hadoop, as well:
      - Some of this data is copied with **`distcp`** into the confidential bucket directly.
      - Presto in watsonx.data seamlessly transfers other Hive tables into the curated zone using efficient `INSERT INTO` operations.
   - Data engineers require early exploration of data in its raw format (CSV).
- **Transform:** Staging and curated zone
   - Using our buckets and datasets in the raw zone, we leverage the two engines available in watsonx.data, Spark and Presto, to run Data Quality Assurance and Refinement.
   - Spark can handle the heavy lifting of data transformations, aggregations, and cleaning.
   - Presto empowers data enrichment by efficiently joining tables from your staging zone dataset. This allows you to create new views or tables in the curated zone, combining and transforming data for deeper analysis.
   - Once the import and data cleansing process finishes, we end up with curated and optimized Iceberg data tables tailored for each business line. These tables, 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 Tableu, Power BI, Superset, and so forth.
   - Retail departments can leverage the watsonx.data SQL client to independently query their curated business line data. This enables them to gain faster insights and make data-driven decisions without relying on IT.

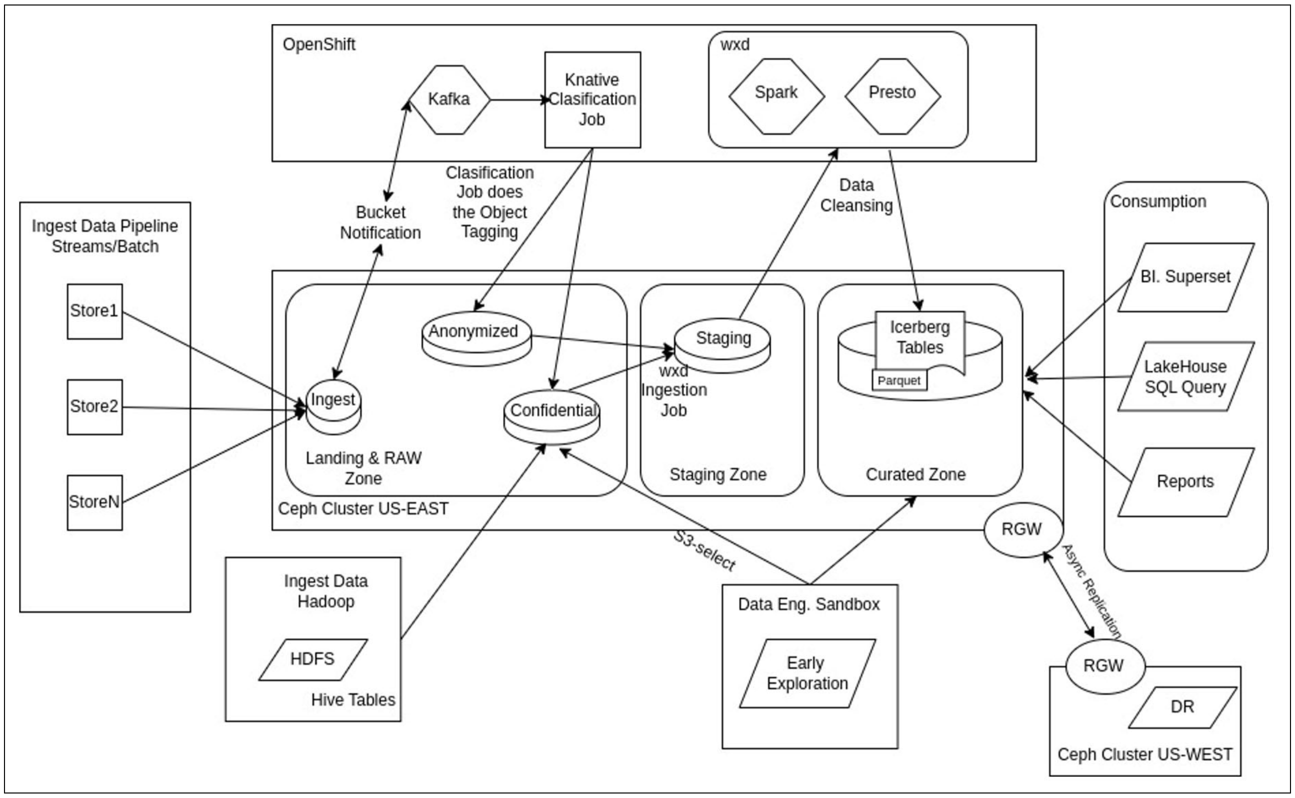


Figure 6-2 Point of Sales example use case for the modern data lake

7

Ingest: Landing and raw zones

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

In this chapter we cover how data from the different retail sources (physical stores, E-commerce, Hadoop (Surveys)) is ingested into the data lake.

This chapter has the following sections:

7.1 Ingestion from stores or branches to the landing zone

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

1. **Daily batch jobs:** Each store runs a batch job at the end of the day, collecting 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 called `ingest`. This acts as a landing zone for the data pipeline.
4. **Automatic deletion:** Since this is a temporary storage location, the data in the `ingest` bucket once processed is automatically deleted after 24 hours using S3's lifecycle policy feature.

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

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

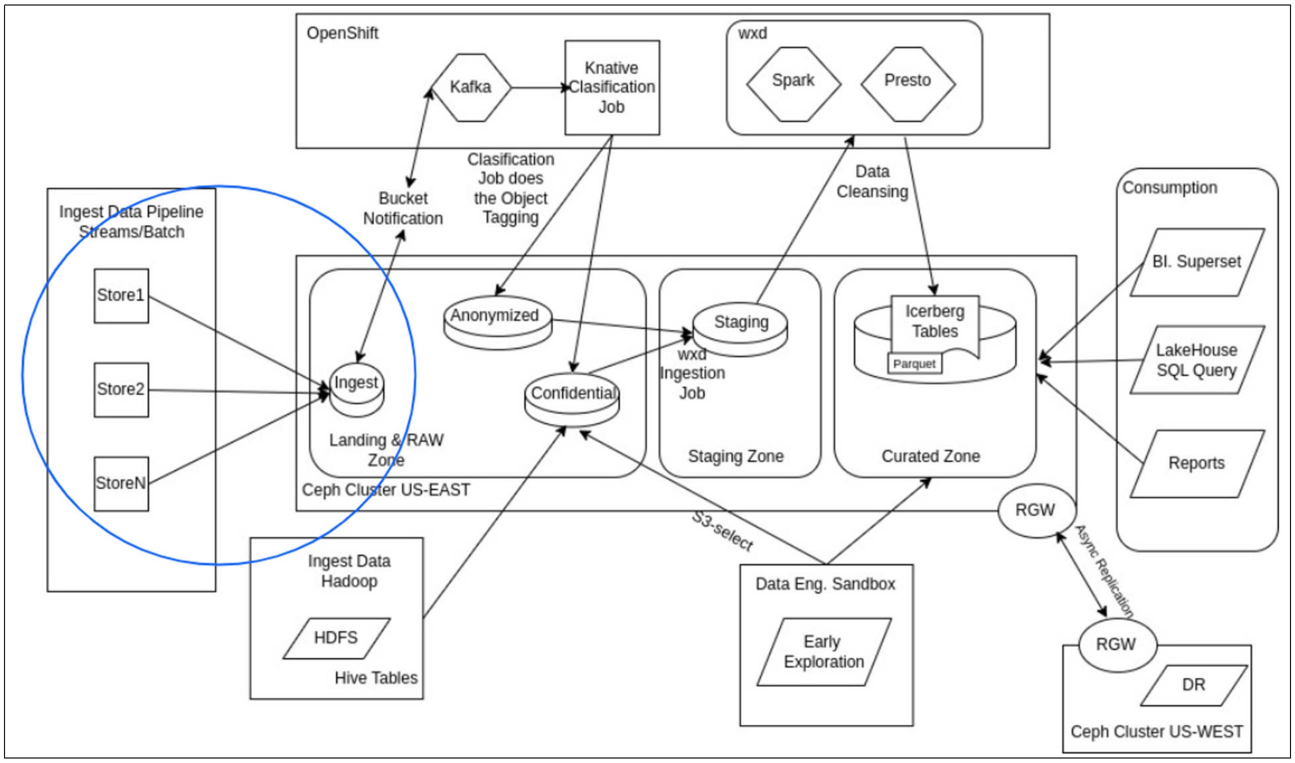


Figure 7-1 Landing zone diagram

Here is a step-by-step walkthrough of this scenario:

1. Let us start with a practical example by setting up the landing zone bucket.

   We have a local Object Gateway user who is the administrator responsible for the physical stores ingestion data pipelines. See Example 7-1.

Example 7-1 Local Object Gateway user

```
# radosgw-admin user info --uid shopingest | jq .keys
```

```
  {
    "user": "shopingest",
    "access_key": "D1YOUYZ8EPY98VAIKHW6",
    "secret_key": "Ct5Xx3eiUOfFW1h5pajJzqvM8PZSPh3mHi16QVav"
  }
```

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

   Let us configure the AWS CLI for the ingest user so we can start getting work done. Remember that the AWS region equates to the zonegroup we have configured in IBM Storage Ceph. See Example 7-2.

Example 7-2 Configure 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]: Ct5Xx3eiUOfFW1h5pajJzqvM8PZSPh3mHi16QVav
Default region name [None]: default
Default output format [None]: json
```

3. We will call our landing zone bucket `ingest`. See 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, let us apply the lifecycle (LC) policy to delete the contents of buckets that are older than 24 hours.

1. Run the following command as shown in Example 7-4.

Example 7-4 Delete 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. We can check that the lifecycle configuration for bucket `ingest` is in place and still in a `not initialized` state, since it has not run yet. Once it executes, we will see it in the COMPLETED state.

Example 7-5 Checking the life cycle 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 make sure our lifecycle is working as expected we will run a quick test. We will use the `rgw_lc_debug_interval` parameter and change the interval to 60 in the Ceph RGW config database, followed by the object gateway service restart. To expedite testing of the `ingest` bucket's lifecycle configuration, we will temporarily enable the debug interval. In this mode, one day within the lifecycle configuration is treated as 60 seconds. This allows us to verify if the lifecycle is functioning correctly within a minute.

Note: Remember to revert this change after the test is complete.

Example 7-6 Change the interval to 60 in the Ceph RGW config database, followed by the 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. We will upload a file called `testlc` without the S3 tag set to `processed=true`, so this file will not get deleted. See Example 7-7.

Example 7-7 Upload a file called "testlc" without the S3 tag set to processed=true

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

5. Next, we will upload an object called `shop4_02_04_2024.csv` with the tag set to `processed=true`. See Figure 7-2 on page 89.

Example 7-8 Upload an object 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 testlc
$ 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 our temporary debug mode), let us confirm that the lifecycle job has successfully finished. We should see that the object `shop4_02_04_2024.csv` has been deleted from the bucket. This indicates that the LC job identified the object based on the "processed" tag and performed the configured deletion. The object named `testlc` should remain in the bucket unchanged. This verifies that the LC job correctly identified objects without the "processed" tag and excluded them from deletion. See Example 7-9 on page 87.

Example 7-9 Confirm that the lifecycle job has successfully finished

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

7. After successfully testing the lifecycle configuration, it is crucial to remove the temporary debug interval setting we implemented (`rgw_lc_debug_interval`). This ensures the lifecycle process operates with its normal timing parameters. See Example 7-10.

Example 7-10 Testing the LC 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.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'
```

7.1.2 Ingest: Data at rest encryption SSE-KMS

Our data encryption strategy leverages IBM Security Guardium Key Lifecycle Manager (GKLM), a KMIP-compliant key manager, integrated with IBM Storage Ceph. This ensures encryption at rest for our datasets. In Example 7-11 we can see the Ceph configuration options that we are using to integrate GKLM with Ceph.

Example 7-11 Ceph configuration options that we are using to integrate GKLM with 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
```

We will configure an S3 bucket encryption policy to automatically encrypt all objects uploaded to the bucket, ensuring data privacy for users. See Example 7-12.

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

```
$catkms-config.json
"Rules": [
    {
         "ApplyServerSideEncryptionByDefault": {
              "KMSMasterKeyID": "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 IDP authentication with single-sign-on

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

Here is the existing setup:

- We have an Active Directory (AD) group named `stores` that includes a service user for each branch.
- Our goal is to allow the branch applications running at each store to authenticate with the central S3 endpoint provided by IBM Storage Ceph. Once authenticated, these applications should be authorized to upload their daily generated CSV files to the `ingest` bucket.
- We have Red Hat Single Sign-On or RHSSO (built on top of the Keycloak project) running on Red Hat OpenShift, we have federated the RHSSO with our enterprise IDP (Active Directory), where all our company users are managed, and we have an AD group called `stores` that is now accessible within RHSSO. See Figure 7-2.

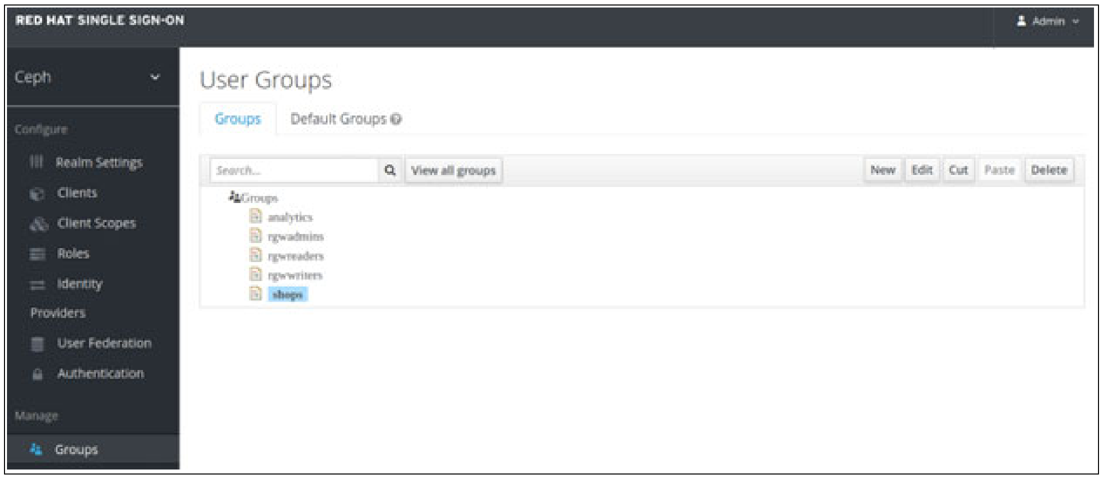
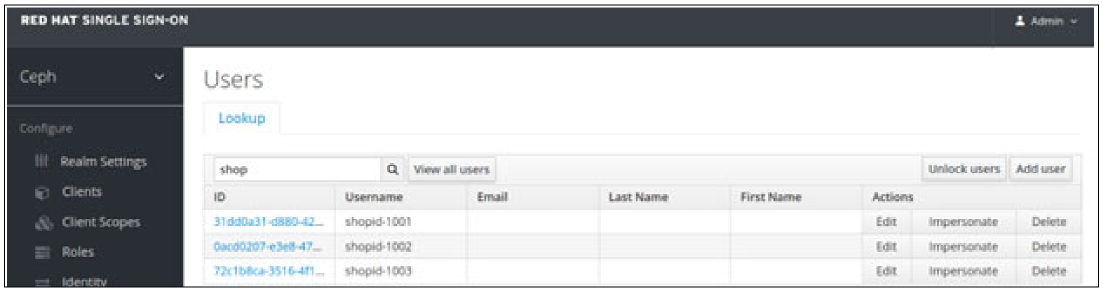


Figure 7-2 Red Hat Single Sign-On

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



ID	Username	Email	Last Name	First Name	Actions		
31dd0a31-d880-42...	shopid-1001				Edit	Impersonate	Delete
0acd0207-e3e8-47...	shopid-1002				Edit	Impersonate	Delete
72c1b8ca-3516-4f1...	shopid-1003				Edit	Impersonate	Delete

Figure 7-3 User look up

For a step-by-step guide on setting up RHSSO as an OIDC provider for IBM Storage Ceph, refer to "*Chapter 7"* of the IBM Redbooks *IBM Storage Ceph Solutions Guide,* REDP-5715.

IBM Storage Ceph supports other SSO through OIDC; the following article Ceph ISV integration using Open ID Connect, provides an example of setting up the IBM Storage Ceph Object Authentication/Authorization with IBM Security Verify.

Once RHSSO is set up, we can see that the group membership of the users is part of the JWT token. See 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 referenced in this section are available in "*Chapter 7"* of the IBM Redpaper *IBM Storage Ceph Solutions Guide,* REDP-5715. Also, all code examples like bucket policies, IAM roles, LC, and so forth are available in this book's GitHub repository.

7.1.4 Ingest: IAM role RBAC-based authorization

Next, we will create an IAM role that allows users belonging to the `shop` group to upload objects (PUT permissions) to the `ingest` bucket. This 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 using the `radosgw-cli` or RESTful API as a Ceph administrator.
- **shopingest user permissions**: Alternatively, you can grant the shopingest user specific permissions to manage roles and policies relevant to their tasks.

In this example, we will follow the second approach.

1. So, we give the shopingest user full access to managing roles. See Example 7-14.

Example 7-14 Give the shopingest user full access to managing roles

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

2. We first create the IAM role with its policy doc, allowing any user that belongs to the group `shops` to assume this IAM role. See Example 7-15.

Example 7-15 Create 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-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"
        }
      }
    }
  ]
# 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 JSON policies, refer to "*Chapter 7*" of the IBM Redpaper: *IBM Storage Ceph Solutions Guide,* REDP-5715.

3. Now that we have established the IAM role, let us configure its policy. This policy is a crucial element that dictates what resources users can access after assuming the role. See Example 7-16.

Example 7-16 Configure 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, let us test the entire workflow. This involves a shop user assuming the role and using it to write data to the `ingest` bucket. See Example 7-17.

Example 7-17 Test out the entire workflow

```
$ . ./test-assume-role.sh shopid-1001 Shopid-1001 shop-ingest
Getting the JWT from SSO...
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. We perform a quick test to confirm successful object uploads while verifying that deletion is restricted. See Example 7-18.

Example 7-18 Check 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 it, and routing it to the appropriate storage location within the raw zone.

- Landing zone and S3 bucket notifications:
  - Data from branch stores is uploaded to a designated landing zone bucket, serving 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.
  - Upon receiving a notification (triggered by a new file upload), Knative, a serverless platform, automatically spins up a container to execute a classification job.
- Data classification and routing:
  - The classification job analyzes the uploaded data to determine if it contains Personally Identifiable Information (PII).
  - PII Data: Data containing 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" for easy recognition.
- Legal hold with S3 object lock (Optional):

– The data pipeline can incorporate an S3 object lock legal hold functionality (optional). This feature can be activated in case of customer disputes related to specific purchase transactions.

Key points:

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

We will be focusing the components inside the blue circle in Figure 7-4.

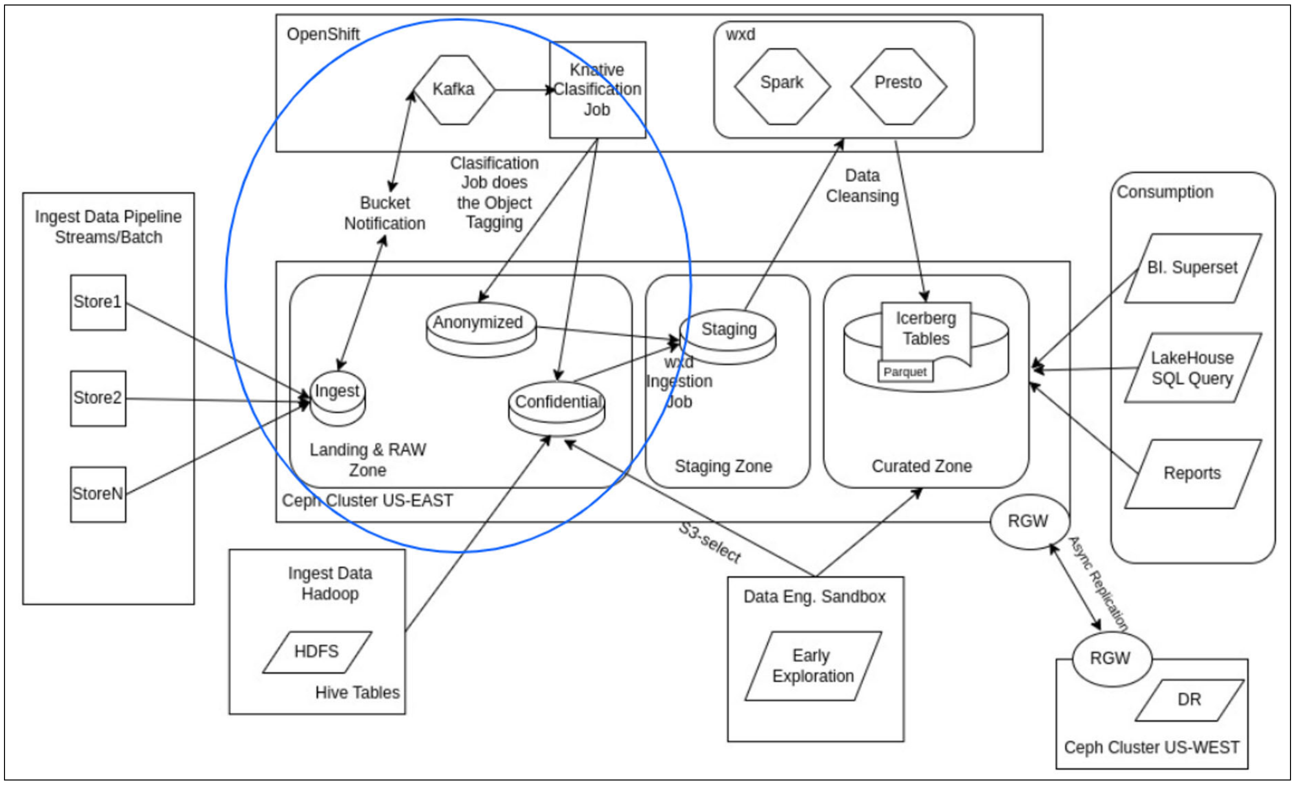


Figure 7-4 Data pipeline

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

We will begin by configuring S3 bucket notifications on the ingest buckets. The purpose of this step is:

- Create a Kafka Topic called `shop-ingest-pipe`.
- Configure a topic on bucket `ingest` that points to our Kafka deployment running on OCP.
- Configure a bucket notification on the ingest bucket to create an event for every object for our `shop-ingest-pipe` Kafka topic.

"*Chapter 5"* of the IBM Redpaper *IBM Storage Ceph Solutions Guide,* REDP-5715 provides a detailed step-by-step description of how to set up Kafka and S3 bucket notifications notifications from scratch.

7.2.1 Raw zone: OpenShift Kafka setup

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

Example 7-19 Kafka deployment running on OCP

```
# 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
#ocgetsvckafka-kafka-routeplain-bootstrap-nkafka
1
NAME                                   TYPE       CLUSTER-IP        EXTERNAL-IP
PORT(S)          AGE
kafka-kafka-routeplain-bootstrap   NodePort   172.21.169.227   <none>
9095:30131/TCP   17d
```

We will create the `shop-ingest-pipe` using a Custom Resource (CR). A CR is a custom object in Kubernetes that allows us to describe and manage non-standard resources specific to our application. See Example 7-20.

Example 7-20 Create the 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

We will continue configuring the ingest bucket and adding a Simple Notification Service (SNS) topic with the OCP Kafka configuration details. See Example 7-21.

Example 7-21 Configuring the ingest bucket and add 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"
    }
  ]
}
```

Now, let us configure the notification event on the ingest bucket. See Example 7-22.

Example 7-22 Configure 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:*"
      ]
    }
  ]
```

We have everything in place. Now, let us verify that the configuration for the Kafka topic and S3 bucket notification integration is working as expected. Here is what we will do:

1. **Start the Kafka consumer:** Execute the **`kafka-console-consumer.sh`** script with the necessary parameters to subscribe to the relevant Kafka topic.
2. **Upload a test object:** Upload a new object to the ingest bucket.
3. **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 indicates that the bucket notification triggered successfully and the message was published to the Kafka topic.

We will upload a test object to the `ingest` bucket using a PUT request. See Example 7-23.

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

```
# aws --profile ingestuser s3 cp /etc/hosts s3://ingest/testlc8
upload: ../../etc/hosts to s3://ingest/testlc8
```

As a final step, let us confirm that the notification event has been successfully sent to Kafka. See Example 7-24.

Example 7-24 Check we have 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","bucket":{"name":"ingest","ownerId
entity":{"principalId":"shopingest"},"arn":"arn:aws:s3:default::ingest","id":"fcab
df4a-86f2-452f-a13f-e0902685c655.47553.1"},"object":{"key":"testlc8","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

We can now proceed with our work. Our next step involves responding to event notifications that we receive from our Kafka topic whenever a store uploads a new .csv file. Our objective is to initiate a server-less process using Knative for every event triggered. The process will check the uploaded object for any personal information and store it in either the `anonymized` or `confidential` buckets of the raw zone.

So, let us describe an outline of the steps we need to accomplish in this section:

1. Create and configure the `anonymized` and `confidential` buckets.
   - Enable object lock on the bucket.
     - Enable S3 versioning (Versioning gets enabled with object lock).
   - S3 SSE-KMS encryption at rest.
   - Data tiering: move cold data from the raw zone buckets to the HDD tier after 120 days.

   - Multi-Factor Authentication Delete: Deleting objects from the raw zone is only possible with MFA authentication.
   - Add S3 tags representing 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 server-less Knative service to run our Python script for every incoming bucket notification on the Kafka topic.

7.2.4 Raw zone: S3 object lock and object versioning configuration

Now, let us create the two buckets for storing classified data:

- `confidential`: This bucket will store data containing Personally Identifiable Information (PII).
- `anonymized`: This bucket will hold data without PII.

1. For managing object lifecycle and legal hold requirements, we will continue using the `shopingest` user. This user has Ceph Object Storage admin privileges for both the landing and raw zones. Additionally, we will enable object locking on buckets containing confidential and anonymized data. This allows our classification script to initiate legal holds on specific objects when flagged by the legal team. See Example 7-25.

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

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

2. We will do a quick test to verify that uploads of objects to this bucket do not get any object lock retention policy. See Example 7-26.

Example 7-26 Verify that uploads of objects to this bucket do not get any 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": "j81o6rO9eUOlNEb9FaOyyOTPOh8Xfnc",
  "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": "j81o6rO9eUOlNEb9FaOyyOTPOh8Xfnc",
  "ContentType": "binary/octet-stream",
  "Metadata": {},
  "ObjectLockLegalHoldStatus": "ON"
```

3. Since object locking is now functioning as intended, let us discuss 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. See 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, you need to 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 only need object versioning you can enable it with the following command. See 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 Ceph storage cluster, let us configure encryption at rest. We will leverage the IBM Security Key Lifecycle Manager (SKLM), formerly known as GKLM, which is already integrated with our Ceph environment.

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

Example 7-29 Configuring encryption at rest

```
$catkms-config.json
"Rules": [
    {
         "ApplyServerSideEncryptionByDefault": {
              "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": "S4mQBMzO9jb3NHPuHKYZWXQjRkMOuVa",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "aws:kms",
    "Metadata": {},
    "SSEKMSKeyId": "rgw00dc42a9b000000000"
```

7.2.6 Raw zone: S3 Multi-Factor Authentication delete (MFA delete)

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

To further bolster the security of our datasets, we will be enabling Multi-Factor Authentication (MFA) specifically for deletion actions. This additional layer of verification helps prevent unauthorized access and accidental data loss.

In this example, we will be using the OAuth tools package to implement *TOTP (Time-based One-Time Password)* as the MFA method. However, you have the flexibility to leverage any TOTP application that suits your needs. Popular options include Google Authenticator, IBM Security Verify, and Authy.

1. To proceed, we will need to install the OAuth tools on your RHEL OS server, designated as linux1. We will use the *DNF (Dandified YUM)* package manager, the recommended tool for managing software on RHEL systems, to install the required OAuth tools. See Example 7-30.

Example 7-30 dnf install oathtool

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

2. We create a seed using the systems “urandom” module. See Example 7-31.

Example 7-31 Create a seed using the systems “urandom” module

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

3. Since the `shopingest` user will be responsible for occasional deletions requiring MFA verification, let us configure a Time-Based One-Time Password (TOTP) entry for their account. We give a name to the totp-serial in this case `MFAdatalake`, and we add the previously created totp-seed. See Example 7-32.

Example 7-32 Configure a TOTP entry for their 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 we created, we can create TOTP PINs for the MFA. We can validate that our config is working using 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 config is in sync. See Example 7-33.

Example 7-33 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 successful MFA setup, let us enable the S3 MFA delete feature on our `confidential` bucket. Since we will not have life cycle expirations configured, we will only allow object deletions through Multi-Factor delete.

   We first create a TOTP pin with the oathtool. See Example 7-34.

Example 7-34 Create a TOTP pin with the oathtool.

```
$ oathtool -d6 --totp 2ed8dcdd34fe31b2d9174c6b25db98
709585
```

6. We use this pin with the MFA ID to enable the `MFADelete` option on the `confidential` bucket. See Example 7-35.

Example 7-35 Enable 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. Let us do a quick test to verify that MFA is in place and that we are not allowed to delete a file through the standard S3 API. We will upload a test object called `mfa_delete_test`. See Example 7-36.

Example 7-36 Verify 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":
"eyJhd3M6czM6YXJuIjoiYXJuOmF3czpzMzo6c2hvcGluZ2VzdDpjb25maWRlbnRpYWwvbWZhX2RlbGV0Z
V90ZXN0In0="
```

8. We attempted to delete the object using a standard s3api delete-object call, but it failed (as expected). See 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, we implemented the `--mfa` flag with the `s3api delete-object` call. This 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 Add 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, we can leverage data tiering for the less frequently accessed cold data residing in our raw zone buckets. Here is how we will achieve this:

- **Lifecycle policy configuration**: We will configure a lifecycle policy that automates the transition of objects to a more cost-effective storage class. This policy will be applied to all objects within the `anonymized` and `confidential` buckets.
- **Transition to HDD tier**: After a predefined period of inactivity (for example, 90 days), objects will be transitioned from the standard storage class to a slower HDD tier. This tier is ideal for data that is infrequently accessed but still needs to be retained for compliance or historical purposes.
- **Tag-based filtering (Optional)**: Similar to the filtering approach 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 could configure the lifecycle policy to only tier objects with a specific tag (indicating processed data) to the HDD storage class. This ensures frequently accessed, recently ingested data remains in the higher-performing standard tier.

1. Each of our Ceph OSD hosts has 2 NVMe OSDs and 4 HDD OSDs. This is represented in Ceph with device classes. Ceph detects device types and marks the device class accordingly. We can also configure custom device classes when creating the OSDs.

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

Example 7-39 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. The following link offers in-depth details on Ceph's Crush algorithm. On a high level, CRUSH rules define data placement within the 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 SSD.

Example 7-40 Configuring the failure domain to host and setting the device class to use to 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 gets placed.

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

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

Our objective for the next section is to create a new RGW data pool that consumes the 4 HDD drives available on each host.

4. We first create the rule that uses the HDD device class. See Example 7-41.

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

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

5. Now, we create a new pool that uses the cold CRUSH rule we previously created. See Example 7-42 on page 103.

Example 7-42 Create a new 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'
```

6. Once we have the pool ready we need to create a new storage class. Let us call the new Storage Class `COLD`.

Note: For compatibility with AWS SDKs it is recommended to use storage classes names that are available in AWS S3.

7. Our zone utilizes a single placement target with a single configured storage class named `STANDARD`. This standard class leverages pools comprised of NVMe drives.. See 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
    }
  }
```

8. We will now add the new storage class to the zonegroup. See Example 7-44.

Example 7-44 Add the new 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"
            ]
        }
    }
```

9. Next, we configure the data pool we will be using for the `COLD` storage class. See Example 7-45.

Example 7-45 Configure the data pool 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
```

10. This configuration optimizes storage costs and performance. It provides two storage classes:

- **STANDARD:** Utilizes high-performance all-flash drives for frequently accessed data.
- **COLD:** Leverages cost-effective spinning disks for less frequently accessed data.

A lifecycle policy (Figure 7-5 on page 105) automatically transitions objects in raw zone buckets from STANDARD to COLD storage after 90 days of inactivity. This ensures 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 day",
      "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

Having completed the bucket creation and configuration, let us move on to the next task: *Creating an IAM role that grants read access to the ingest bucket and write access for processed data to the raw zone buckets*.

1. Following our approach with storage, we will configure user access at the Identity Provider (IdP) level. In our case, that's Active Directory (AD). We have created a user named user-ops-01 who belongs to the `rawzoneadmin` LDAP/AD group. Since our RHSSO federates with AD, these users are automatically mapped to RHSSO, as shown in Figure 7-5.

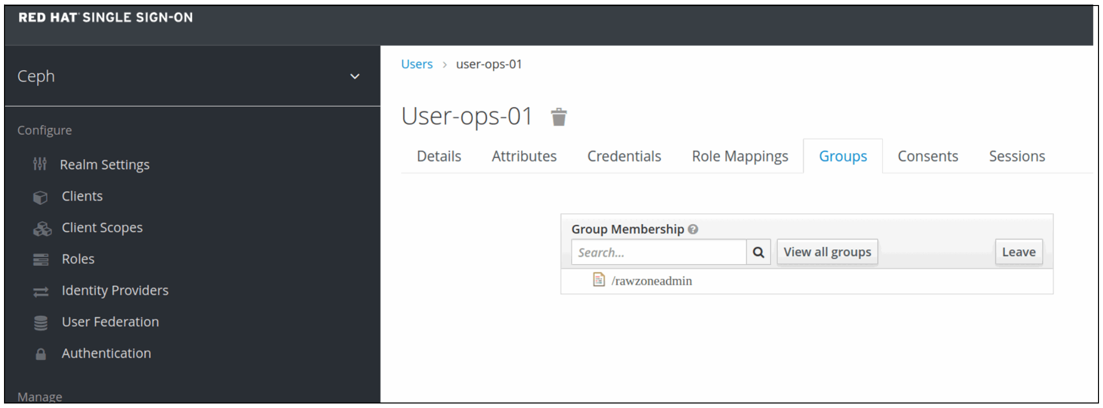


Figure 7-5 User map

In the same fashion as with the IAM setup we used for the stores, we will create an IAM role that will allow 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

2. The trust policy associated with our IAM role specifies that only members of the LDAP group `rawzoneadmin` can assume this role. See 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-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": "rawzoneadmin"
        }
      }
    }
  ]
$ radosgw-admin role create --role-name raw-zone-admin
--assume-role-policy-doc=$(jq -rc . /root/iam_role_policy_shop_read.json)
```

3. Now that we gave established the IAM role for data ingestion, let us define the specific permissions it will have. IAM roles can be assigned one or more policies, allowing for granular control over user access. We will start by configuring a policy specific to accessing the `ingest` bucket. See Example 7-48.

Example 7-48 Configuring a policy 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)
```

4. We will add a second policy that covers the PUT access to the raw zone buckets. See Example 7-49.

Example 7-49 Add 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)
```

5. Now that we have created the IAM role `raw-zone-admin`, we will perform a quick test to ensure it functions as intended. We will assume the role and attempt to access relevant resources. See 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 SSO...
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..
```

6. We will test the `ingest` bucket. See 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
```

7. We will test the `anonymized` and `confidential` buckets. See 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 used in this scenario are available at this GitHub link.

7.2.9 Raw zone: Serverless data pipeline workflow

Our IAM role is working as expected, so we can move on to the following task: *Create a serverless Knative service to run our 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 will categorize CSV files based on data confidentiality. In our example, files containing personal information will be classified as confidential and stored in the designated bucket. Conversely, files without personal information will be classified as non-personal and stored in the `anonymized` bucket. Additionally, the script will tag objects with "red" for personal data and "green" for non-personal data. Finally, to preserve data for potential legal issues, the script will leverage 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 data is stored securely based on confidentiality.
- **Enhanced search and access:** Data tagging facilitates easier identification and retrieval of specific data types.
- **Legal compliance:** S3 object lock safeguards sensitive data, potentially aiding in legal matters.

Figure 7-6 provides a high-level description of the workflow.

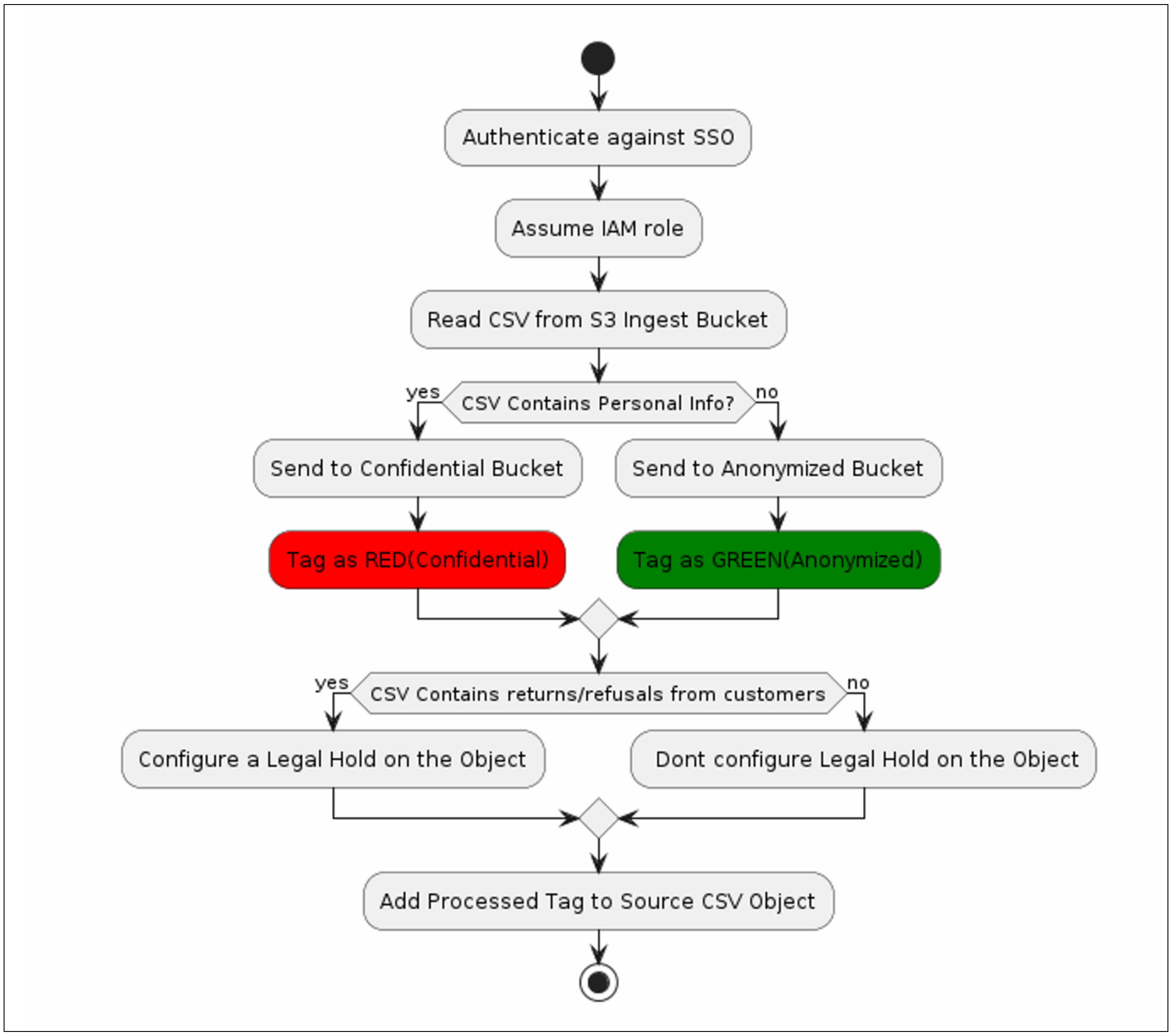

Authenticate against SSO
Assume IAM role
Read CSV from S3 Ingest Bucket
yes
CSV Contains Personal Info?
no
Send to Confidential Bucket
Send to Anonymized Bucket
Tag as RED(Confidential)
Tag as GREEN(Anonymized)
yes
CSV Contains returns/refusals from customers
no
Configure a Legal Hold on the Object
Dont configure Legal Hold on the Object
Add Processed Tag to Source CSV Object

Figure 7-6 High-level description of the ingest data workflow

The Python script can be found on the following GitHub repository: link.

Our Python classification script is ready for deployment. The next step is to integrate the script within the pipeline.

Here is the flow (See Figure 7-7 on page 110):

1. Branch stores upload end-of-day CSV files.
2. An S3 bucket notification triggers upon upload.
3. The notification sends an event to Kafka.
4. The Kafka event triggers a Knative serverless job.
5. This job executes the script, classifies the data, and moves it to the appropriate bucket (`confidential` or `anonymized`).
6. Additionally, the script tags objects based on data type ("red" for personal, "green" for non-personal) and applies legal holds to relevant files based on customer complaints.

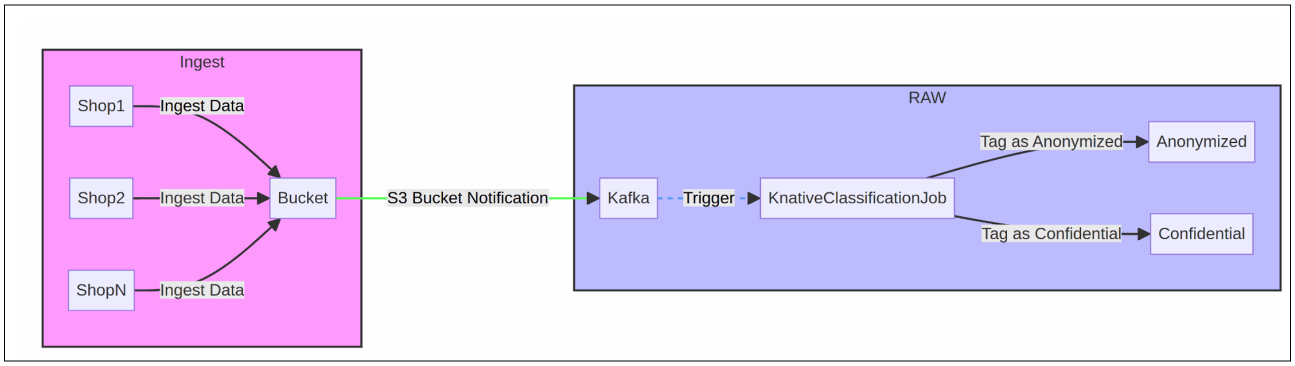


Figure 7-7 Data corresponding bucket flow

With a clear understanding of the workflow, let us begin the implementation process. We already have the `ingest` bucket with S3 bucket notifications configured against our Kafka topic called `shop-ingest-pipe`. So, we need to work on the serverless (Knative) implementation.

To ensure smooth execution by the serverless function, our Python script/app needs to 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 here. From that Dockerfile, we created a container image and uploaded it to the `quay.io` image registry: `https://quay.io/dparkes/ingest_to_raw`.

7.2.10 Raw zone: Knative and Kafka service deployment

Next, we need to configure a Knative service.

1. We already have Knative operators deployed and running on our OCP cluster. See Example 7-53.

Example 7-53 Knative operators deployed and running on our OCP 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. We deployed the Knative Eventing, Serving, and Kafka operators using their default configurations. These operators offer customization options for advanced configurations if needed. See 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, we will create a Knative serving service, as shown in Example 7-55 on page 111. This service will be responsible for:

- **Instantiation:** Upon receiving a request, the service will automatically create an instance of our Python classification application.
- **Data classification:** The instantiated application will then process the incoming data and perform the classification task.

  This approach ensures efficient resource utilization by dynamically provisioning application instances based on incoming data.

Example 7-55 Create 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 the Knative service gets a request to the following URL: `https://ingest-to-raw-rawtest.ocp-eu-redbook-ce869a544b369f1dfd24beec10027762-i000.eu-es.containers.appdomain.cloud`, it will spawn a pod with our container image and process the data. For example, when we run a **`curl`** command that issues a `GET` request, you can see how the two pods execute immediately. See Example 7-56 on page 112.

Example 7-56 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 CRD:** We defined a custom resource definition (CRD) 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 then delivers these CloudEvents to the Knative service we created earlier (See Example 7-55 on page 111). This Knative service acts as the sink in this scenario.

Key takeaways:

- Knative's 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, allowing the script to receive and process the CloudEvents containing S3 bucket notification details.

Figure 7-8 shows a visual representation of the workflow.

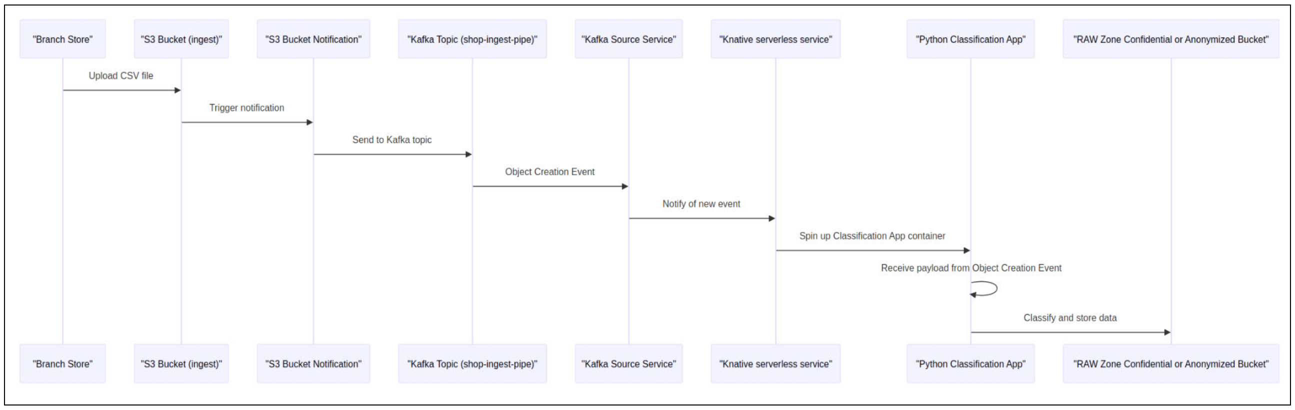

"Branch Store"
"S3 Bucket (ingest)"
"S3 Bucket Notification"
"Kafka Topic (shop-ingest-pipe)"
"Kafka Source Service"
"Knative serverless service"
"Python Classification App"
"RAW Zone Confidential or Anonymized Bucket"
Upload CSV file
Trigger notification
Send to Kafka topic
Object Creation Event
Notify of new event
Spin up Classification App container
Receive payload from Object Creation Event
Classify and store data

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

At this point, we have our full workflow setup, the raw zones buckets created and configured, and 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 dataset generator in the GitHub repo that we will use to create an example CSV file and upload it to the ingest bucket, starting the pipeline. See Example 7-58.

Example 7-58 Sample dataset 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 has arrived. See Example 7-59.

Example 7-59 Object created event has 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":"ingestnotificationknative","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":"","sequence
r":"253E196640A8371D","metadata":[{"key":"x-amz-content-sha256","val":"UNSIGNED-PA
YLOAD"},{"key":"x-amz-date","val":"20240412T135900Z"},{"key":"x-amz-security-token
","val":"B5CXfbOuiSSiFPST8k6fs9rncQfzK47fdnxSIDEOOuwJrlAqJyS4SIOOyu7LudYIN8/8cMUuc
Ob9QhRsAvOVqVat6HrU6Vf7Vhlg8yOcKtNz83A1/1q9MsMmFjSfJw8sc61RhQ7WGmsExkWHQdv1CdF+FHQ
sxj+y/SuVb8CM5RQ9+XhW/BLr4jJjxz/vt4rx5SOtgf9TG6LzhTs5wzwkOwyjXl29xohsQbVsinSExarhq
ISdpWfcMnx/BuhdMPDaWh3P8QesPj/5xZ16auidFszEGxyrAEuLK8L4BFnMlrOLGn1FA5SIsm2K+c2e/+u
wNyut8mLciIdm5LHYST6WzmuwP+We3tCgBh5dcbOO+Ej76OJXYfJjy1A5ZLOkX1l1SOj4HslvXwDHU9l9J
jE7mApusVsmkMwR932pmKV53Plk5m9sWcA5LwVpGLOTrlOFHxc3v2+8BQZNtDicUhFvdGJc9GL4G9C7aCu
ooAvu8DrUpkwd+pSnU3Be9ScOHpMYAprqM6SVbOxmM3nONhTgK1s9h4EFAnc4wBbbwJKURwi++bvkMy1Xc
sfWPZVPd7pT"}],"tags":[]}},"eventId":"1712930341.490186.0f735216f002184af224946f8d
451018","opaqueData":""}]}
```

3. By examining the application pod logs, we can confirm that the serverless application has initiated file processing. See Example 7-60

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 let us verify that the CSV file containing personal information has arrived at its secure destination. We will use the head-object call to the `confidential` bucket, using the object name as the key. This confirms the file's presence without downloading it. Since the `confidential` bucket utilizes SSE-KMS, the data remains encrypted at rest. See Example 7-61.

Example 7-61 Check if the .csv has 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": "rgw00dc42a9b000000000"
```

5. We will verify the object has an S3 tag, a label used to categorize data. Since it contains PII (Personally Identifiable Information), the tag value for data security classification should be red. See Example 7-62.

Example 7-62 Check if 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. The final step is to confirm whether S3 Object Legal Hold has been activated for the ingested CSV files containing legal conflicts from the stores. See Example 7-63.

Example 7-63 Final check

```
$greplegalshop4_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": "j81o6r09eU0lNEb9Fa0yy0TP0h8Xfnc",
  "ContentType": "binary/octet-stream",
  "Metadata": {},
  "ObjectLockLegalHoldStatus": "ON"
```

7.2.12 Raw zone: Hadoop data ingest into the raw zone

We have 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. For that, we must configure an on-demand ingest of Hadoop 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 datasets. This flexibility allows users to leverage various tools for analysis, including watsonx.data and other SQL engines.

As Chapter 5, “IBM Storage Ceph with IBM watsonx.data” on page 67 offers a comprehensive hands-on example of S3A driver/connector usage for data movement between HDFS and IBM Storage Ceph Object, we will refrain from duplicating those steps here. Once the data resides within a Ceph bucket, you can import it into your watsonx.data curated collection, following the same process 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. Some examples of how the data may be transmitted to the main site include:

- **Direct upload:** Data is sent directly from the source (For example, web servers, client-side browsers) to the data lake.
- **Log shipping**: Server logs are periodically collected and sent to the data lake using automated scripts.
- **Data streaming**: Real-time data streaming using technologies like Apache Kafka allows for 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 datasets are transformed into CSV files and uploaded to a designated bucket named "ecommlandlogs" within our storage system. This upload triggers an automated data pipeline, facilitated by bucket notifications and Kafka, which performs some initial data processing.

Figure 7-9 shows the data pipeline architecture for e-commerce.

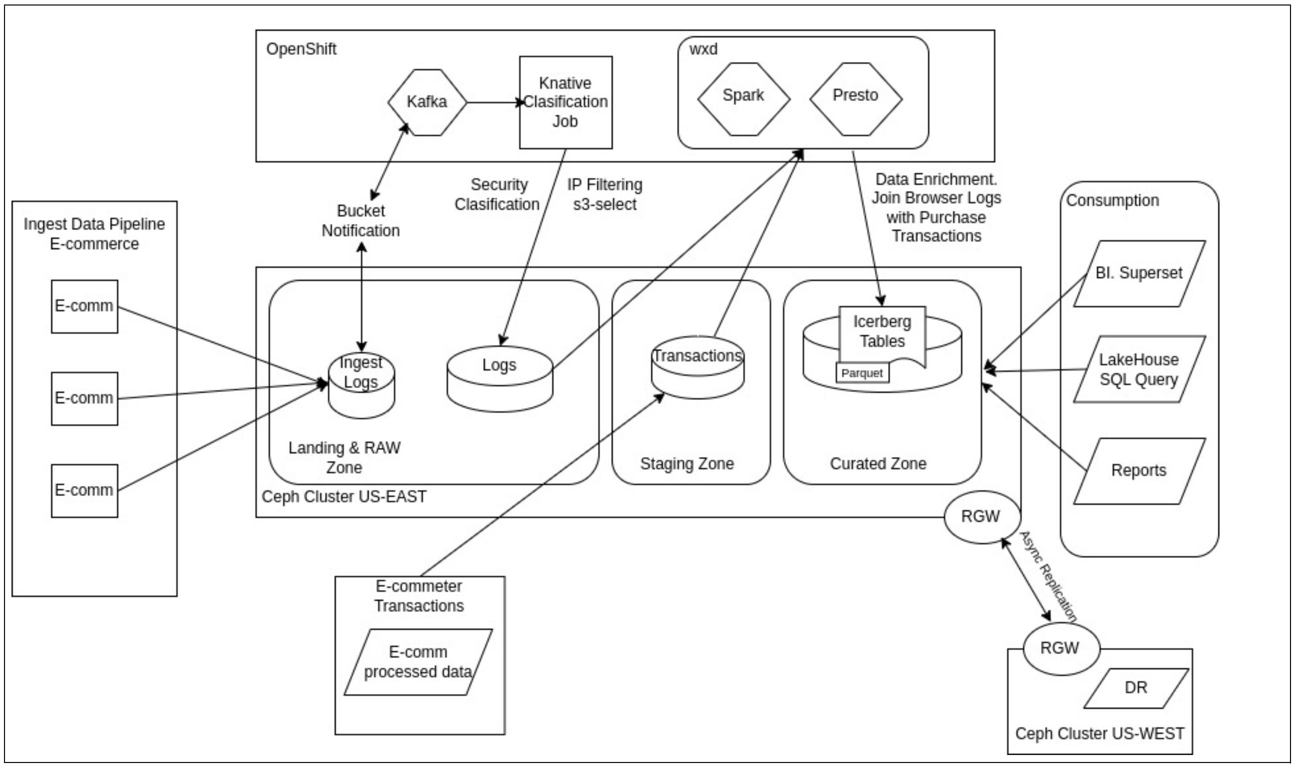


Figure 7-9 Data pipeline architecture for e-commerce

To avoid repetition, we will skip the detailed workflow explanation as it mirrors the physical stores use case covered in 7.2, “Raw zone: Branch stores classification data pipeline” on page 92. For those interested, the OpenShift deployment details for the e-commerce pipeline, including Knative serverless application example code and sample 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. While this data offers a wealth of insights into customer behavior, it can also contain unwanted elements such as activity from malicious IP addresses. Processing and storing such massive datasets can strain storage resources and significantly extend processing times and costs during the analytics phase.

Ceph's *S3 Select* feature offers a strategic advantage to address these challenges. S3 Select allows for querying and retrieving specific subsets of data directly within S3 objects stored on the Ceph layer. By applying S3 Select before data reaches the data lake's raw zone, retailers can filter out browsing logs associated with known malicious IP addresses. Discarding irrelevant or harmful data early reduces the volume of data written to and subsequently read from the data lake.

S3-Select: Faster data retrieval for Ceph Storage:

S3-Select is a powerful tool that optimizes data retrieval from Ceph storage. It utilizes two key technique:

- **Predicate pushdown:** Instead of transferring all the data to the client and then filtering it, S3-Select pushes the filtering logic (predicates) to Ceph itself. This allows Ceph to identify and transfer only the relevant data, significantly reducing the amount of data transferred and improving 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 Ceph. This simplifies querying complex datasets and streamlines the data retrieval process.

Benefits:

- **Reduced network traffic:** By filtering data at the source, S3-Select significantly minimizes the amount of data transferred, leading to faster query execution times.
- **Improved efficiency:** Utilizing Ceph's processing power for filtering reduces the workload on the client, improving overall system efficiency.
- **Simplified queries:** The SQL-like interface allows for easy and intuitive querying of complex datasets stored in Ceph..

The full Python script is available in GitHub here. In Example 7-64 we provide 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 dataset that contain IPs on known malicious CIDR (Classless Inter-Domain Routing) network ranges. This is just a very basic query for our example use case. You can perform very advanced SQL queries with the help of the S3 Select feature available with IBM Storage Ceph.

If we check the logs for our running e-commerce example app, we can see S3 Select in action. See 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 has triggered a new event on the pipeline, and a new browsing log file "`browsing_data_20240507202409.csv`" has arrived. Our Knative serverless application triggers upon data arrival and initiates processing. One part of the processing script uses S3 Select, that runs a SQL query on the dataset: `'SELECT * FROM S3Object WHERE NOT (ip LIKE '45.%' OR ip LIKE '210.%')`.

Note: Crucially, the entire filtering process executes on the IBM Storage Ceph Object Gateways themselves and only return the results to the client. This offloads processing from the client application and improves overall efficiency.

You can see this on the next output line of the logs: "`{'BytesScanned': 14088969, 'BytesProcessed': 14088969, 'BytesReturned': 4677325}`". It scanned `14088969` bytes, but after the filtering, it only returned `4677325` bytes to the application. If you are interested in the IBM Storage S3 Select feature, check out this blog, where we provide TPC-DS Benchmark numbers when using S3 Select.

After processing, the data will be stored in a raw zone bucket named `ecommraw` with the same configuration (SSE encryption, versioning, and so forth) as the buckets used for the physical branch stores described in 7.2, “Raw zone: Branch stores classification data pipeline” on page 92.

Two different datasets will be stored: one for the e-commerce transactions (`ecommtrans`) and the other for the e-commerce logs (`ecommlogs`).

- **E-commerce logs**: Once 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 could 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 specifically designed for big data analytics. This conversion optimizes data storage efficiency and accelerates future processing within the data lake.
  c. **Transfer to staging zone**: Finally, 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 dataset 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 will enrich the e-commerce dataset by joining browsing logs and transaction datasets in the staging zone. By combining these datasets, retailers gain a more holistic view of customer behavior, enabling:

- **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.

8

Transform: Staging and curated zones

In Chapter 7, “Ingest: Landing and raw zones” on page 83, we covered how the different retail sources (physical stores, e-commerce, Hadoop (Surveys)) ingest data into the data lake.

In this chapter, we will continue providing examples and detailed hands-on instructions on how we can transform and enrich our datasets so they can provide higher value for our users at the consume layer.

In this chapter we cover the following:

- “Points of Sale (physical stores)” on page 122
- “E-commerce” on page 134

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 needs to be split into two different types of data, such as anonymized and confidential.

Anonymized data covers the non-confidential data such as the items that have been sold from a specific store and the item details such as their type, category, payment type and so forth.

To be able to analyze the different categories of data, you need to create separate buckets for anonymized and confidential information with separate access policies so that only the allowed people 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 S3 buckets: bucket policies and object tags. We will explore how to assign bucket policies to control user access and leverage object tags to categorize your data effectively.

To illustrate these concepts, we will create two example buckets:

- **Confidential bucket:** This bucket will store user-related confidential data, requiring strict access controls.
- **Anonymized bucket:** This bucket will house anonymized, non-confidential customer data. We will utilize object tags to further classify this data within the bucket.

Once we have established these buckets, we will delve into the commands for creating and assigning bucket policies and object tags.

To create the buckets, you can use the following commands, as 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 creating the buckets, you need to assign bucket policies to these buckets to limit the access. This will allow to limit who can access the data within those buckets. An example bucket policy is shown in Example 8-2.

Example 8-2 Assign 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/shopingest"]},
    "Action": "s3:GetObject",
    "Resource": [
      "arn:aws:s3:::confidential/*"
    ]
  }]
}
```

Tip: You can assign policies for multiple users at once. For example, since shopingest user will also need access, we are giving permission to that user as well.

With this policy, only the `confidential` user will be able to retrieve objects from the `confidential` bucket. As we will be utilizing another bucket to store anonymized user data, we will also need to assign a bucket policy to it. This ensures consistent access control across all our data storage buckets. See Example 8-3 on page 123.

Example 8-3 Assign 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 creating the buckets and assigning policies, we will add them to watsonx.data for ingesting the data inside those buckets.

1. First, you need to navigate to Infrastructure manager dashboard of the watsonx data and click **Add bucket** option on the Add component menu. See Figure 8-1.

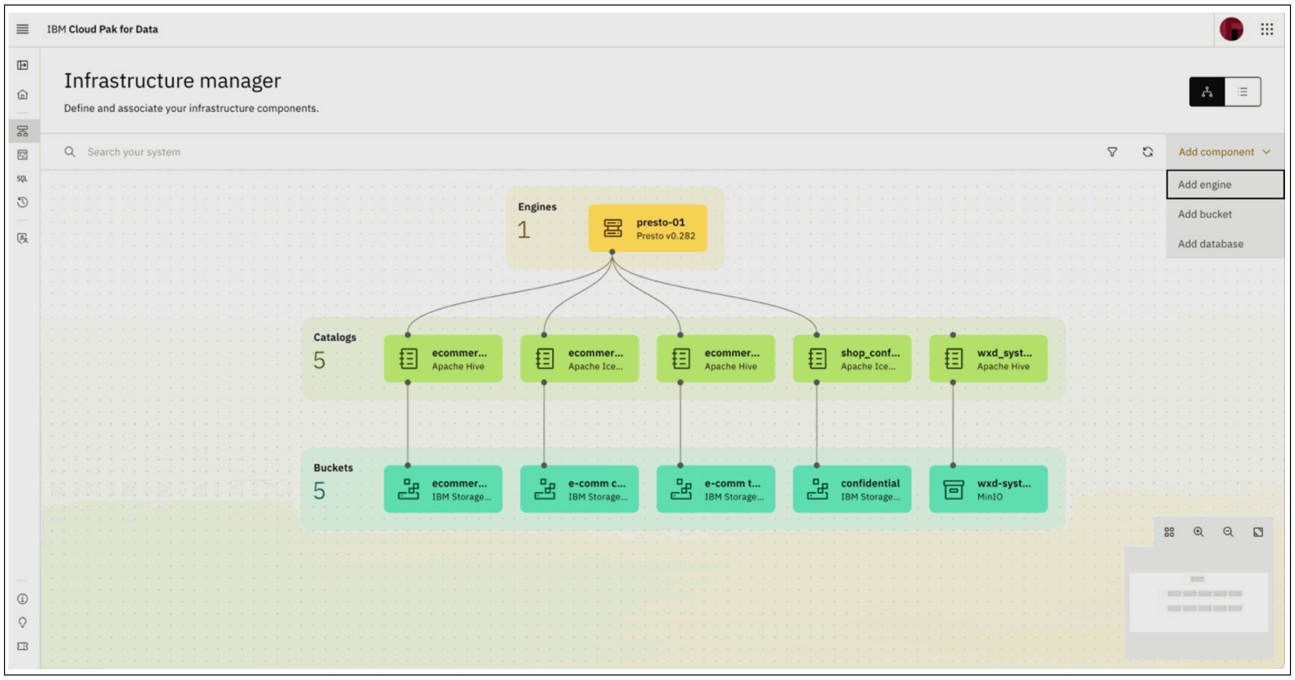
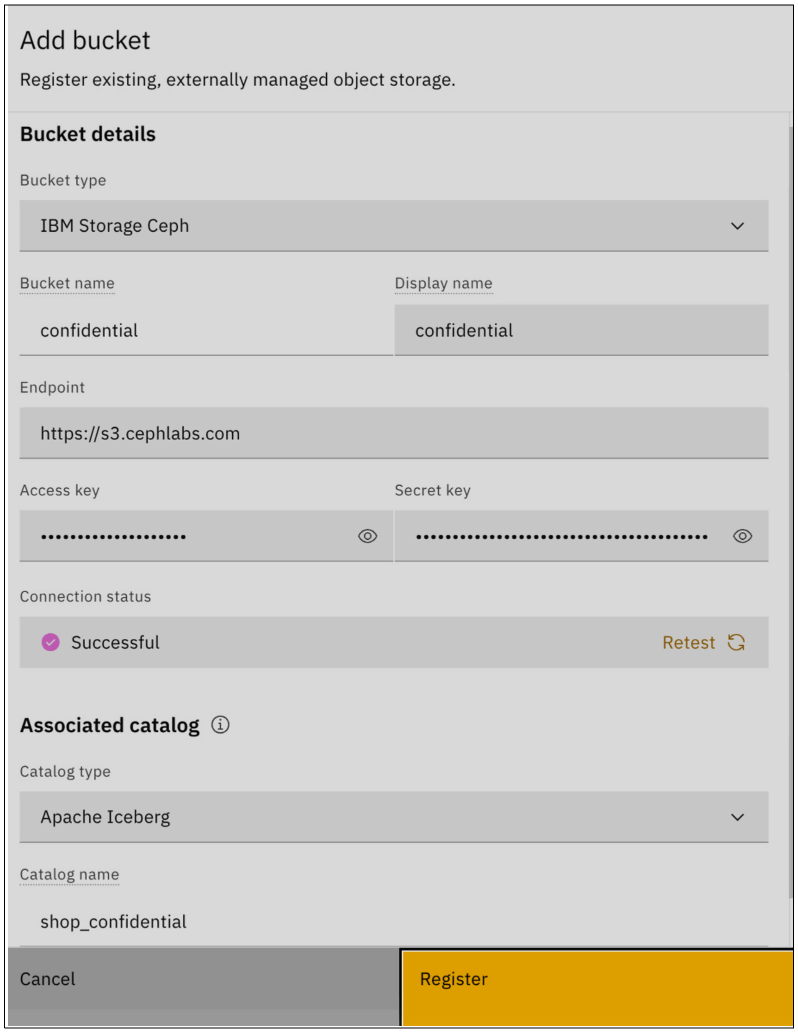


Figure 8-1 Add bucket option

2. Select **IBM Storage Ceph** as the Bucket type and enter the details for accessing your Ceph bucket. An example can be seen 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

Associated catalog

Catalog type: Apache Iceberg

Catalog name: shop_confidential

Figure 8-2 Add bucket

3. Once the connection test is successful, you can click **Register** and add the bucket to your watsonx.data environment. Note that the Access key and Secret key values belong to the confidential user on Ceph.
4. After adding the buckets, we will integrate them with the Presto engine running on watsonx.data to be able to ingest and consume the data. To do this, you can hover over the bucket that you would like to add and click **Manage associations** option. See Figure 8-3 on page 126.

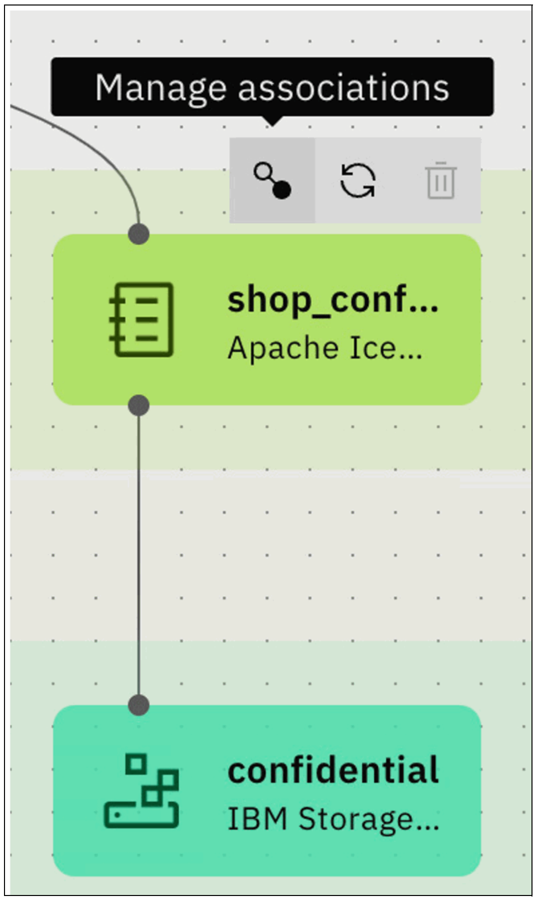
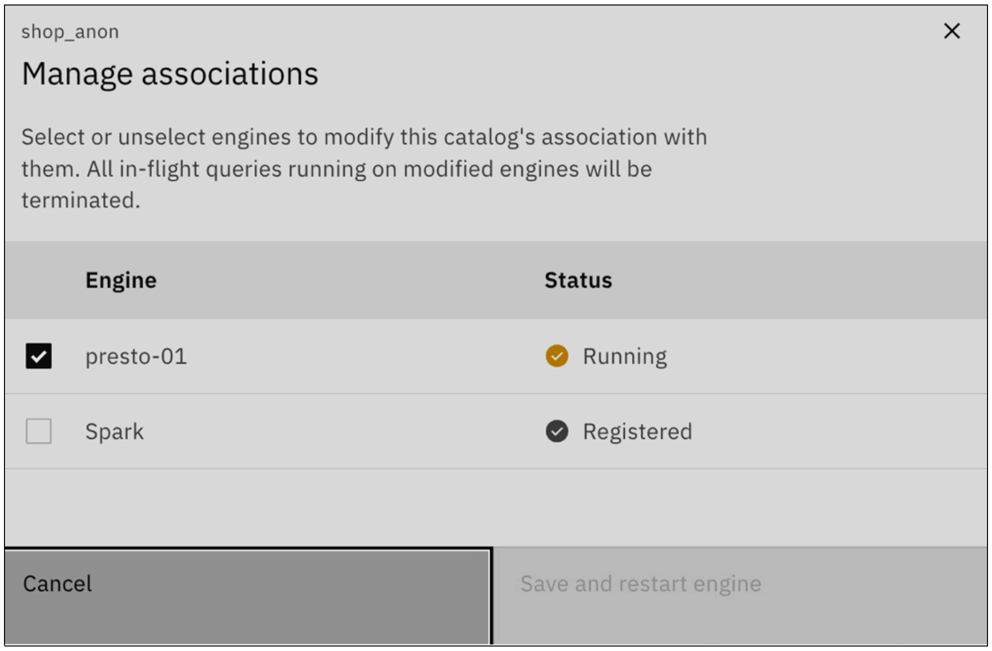


Figure 8-3 Manage associations

5. From the options menu, select **Presto**. The Presto engine will restart. This process may take a few moments. See Figure 8-4 on page 127.


shop_anon

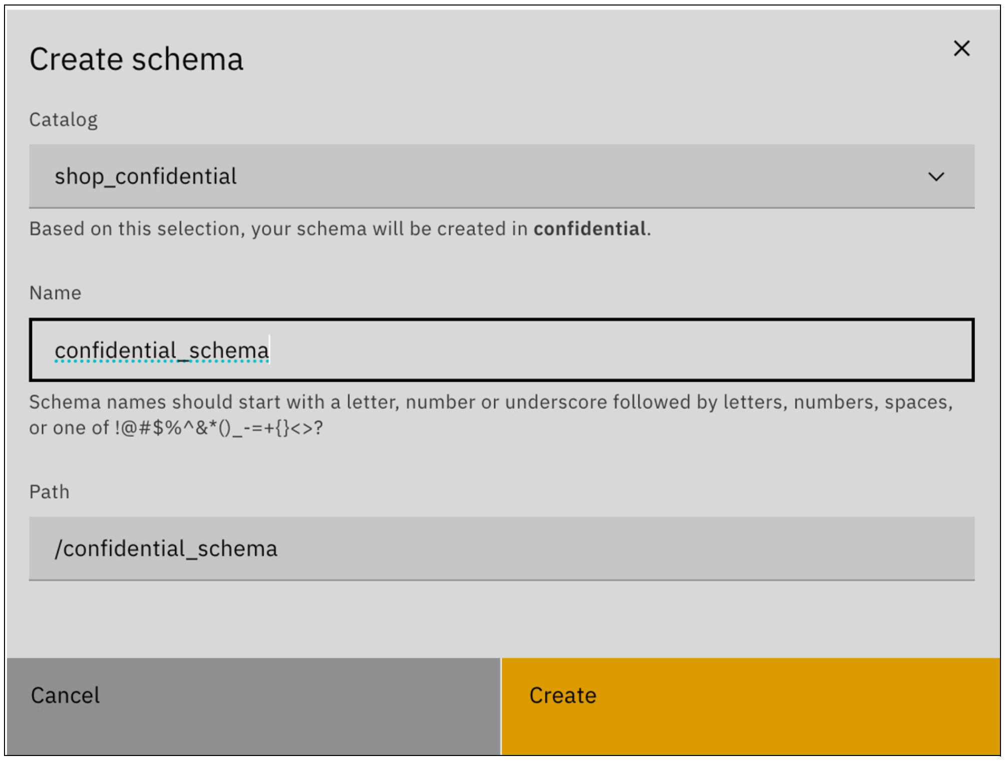
Manage associations

Select or unselect engines to modify this catalog's association with them. All in-flight queries running on modified engines will be terminated.

	Engine	Status
✓	presto-01	Running
☐	Spark	Registered

Figure 8-4 Presto engine will then restart

6. After integrating the `confidential` bucket, we will integrate the `anonymized` bucket using the watsonx.data CLI. To do this, you need to install the watsonx.data command line tool.
7. After the completion of integration, we will create schemas and tables under the catalogs that were created after integrating the buckets with Presto. To create a schema, navigate to **Data manager** from the left-side menu and click **Create** and select **Create schema** option from the dropdown options.
8. Enter the details for the schema you would like to create and click **Create**. See Figure 8-5 on page 128.


Create schema

Catalog

shop_confidential

Based on this selection, your schema will be created in **confidential**.

Name

confidential_schema

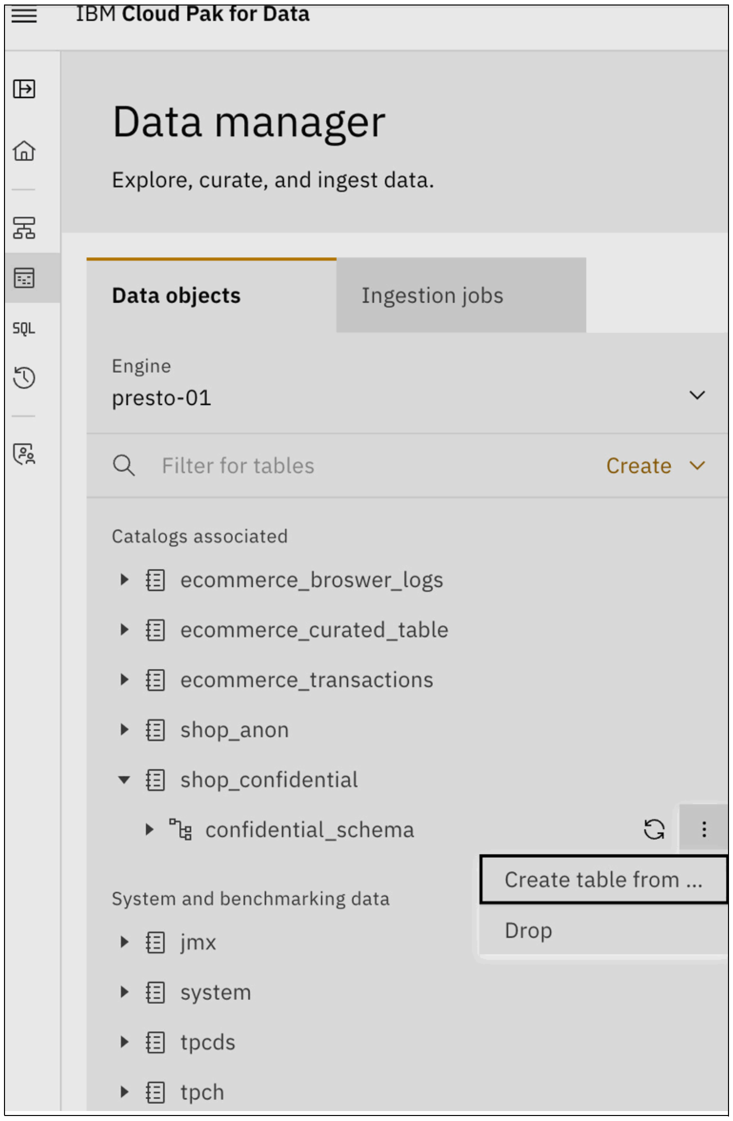
Schema names should start with a letter, number or underscore followed by letters, numbers, spaces, or one of !@#$%^&*()_-=+{}<>?

Path

/confidential_schema

Figure 8-5 Create schema

9. Then, you need to create tables under the schemas that were created in the previous steps. To do this, navigate to your schemas and select **Create table from** option from the three-dot menu on your schema. See Figure 8-6 on page 129.



Data manager

Explore, curate, and ingest data.

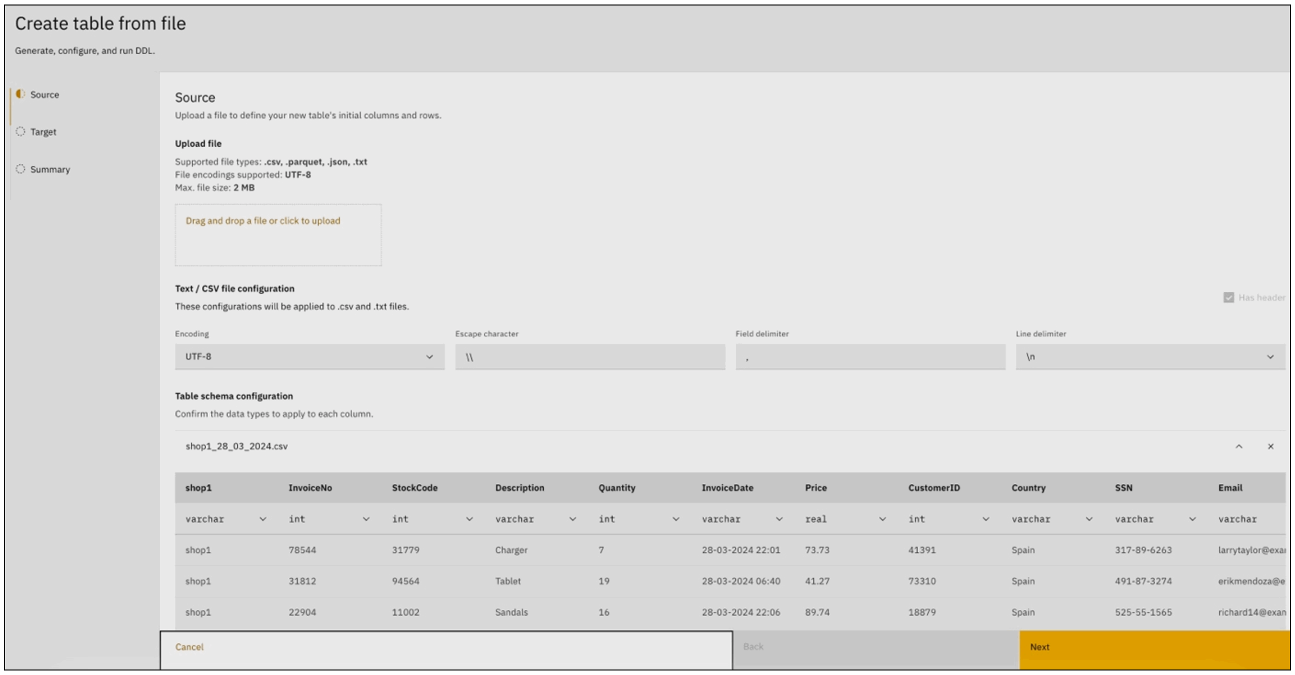
Engine
presto-01

Catalogs associated
- ecommerce_broswer_logs
- ecommerce_curated_table
- ecommerce_transactions
- shop_anon
- shop_confidential
  - confidential_schema

System and benchmarking data
- jmx
- system
- tpcds
- tpch

Figure 8-6 Create table from option

10. Next, you need to 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. See Figure 8-7 on page 130.


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.

Has header

Encoding	Escape character	Field delimiter	Line delimiter
UTF-8	\\	,	\n

Table schema configuration

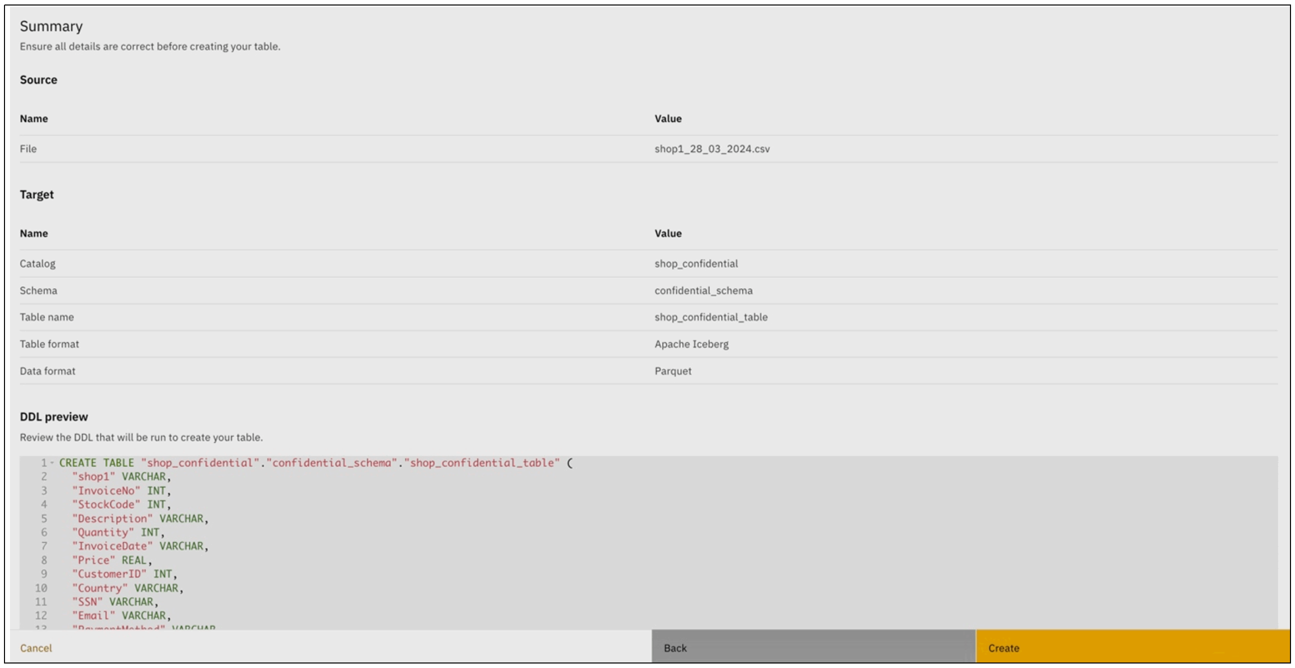
Confirm the data types to apply to each column.

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@exan

Figure 8-7 Upload a sample file from your computer to watsonx.data

11. You need to select your target schema and give your table a name and click **Next** to continue with the summary screen and table creation. See Figure 8-8 on page 130.


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

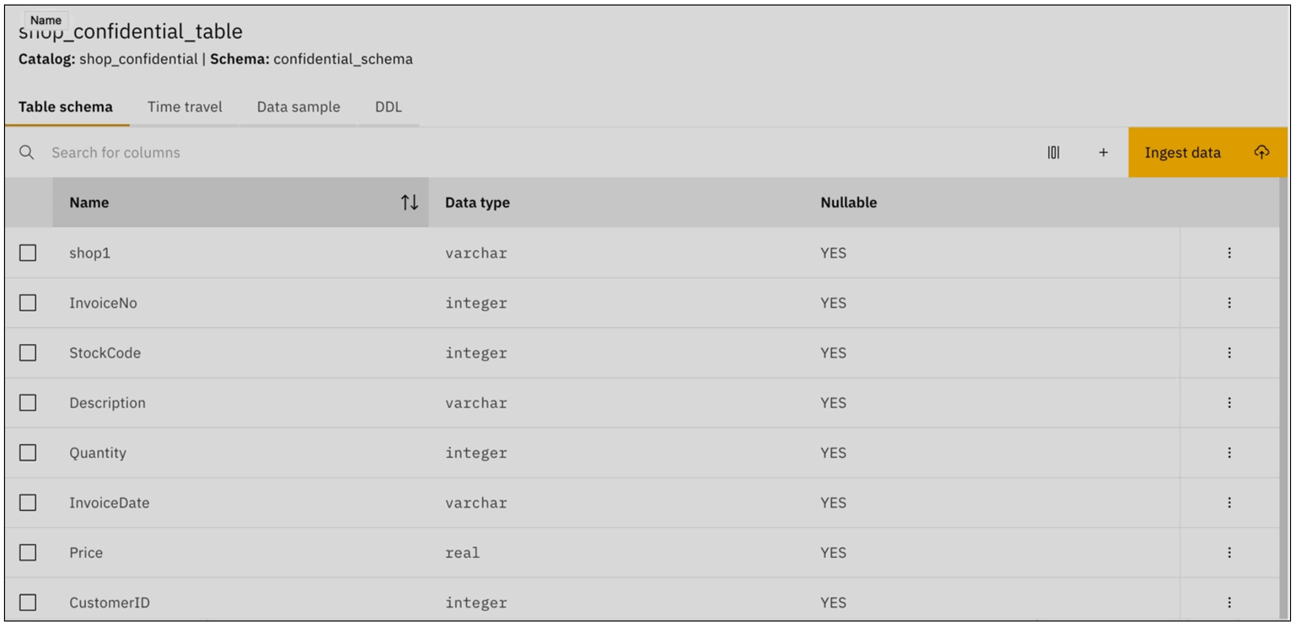
DDL preview

Review the DDL that will be run to create your table.

```
CREATE TABLE "shop_confidential"."confidential_schema"."shop_confidential_table" (
  "shop1" VARCHAR,
  "InvoiceNo" INT,
  "StockCode" INT,
  "Description" VARCHAR,
  "Quantity" INT,
  "InvoiceDate" VARCHAR,
  "Price" REAL,
  "CustomerID" INT,
  "Country" VARCHAR,
  "SSN" VARCHAR,
  "Email" VARCHAR,
  "PaymentMethod" VARCHAR
```

Figure 8-8 Summary screen

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 from the **Ingest data** option on your table. See Figure 8-9 on page 131.

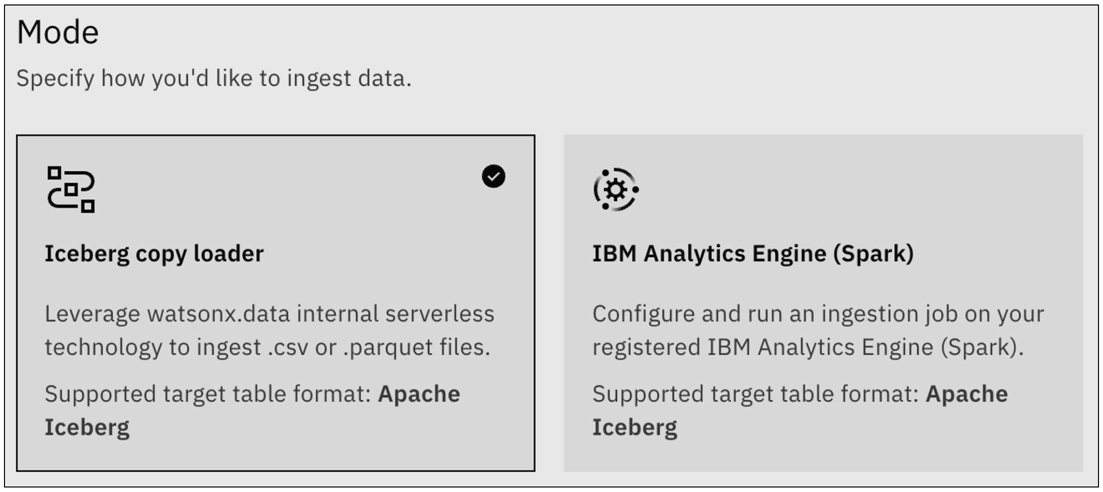

shop_confidential_table

Catalog: shop_confidential | **Schema:** confidential_schema

Name	Data type	Nullable
shop1	varchar	YES
InvoiceNo	integer	YES
StockCode	integer	YES
Description	varchar	YES
Quantity	integer	YES
InvoiceDate	varchar	YES
Price	real	YES
CustomerID	integer	YES

Figure 8-9 Ingest data option

13. You can also review the uploaded data from the Data sample tab on the same screen. Clicking the **Ingest data** button opens a dialog where you can choose your preferred ingestion mode. You can select your ingest mode and click **Next**. We select **Iceberg copy loader** for this demonstration. See 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)

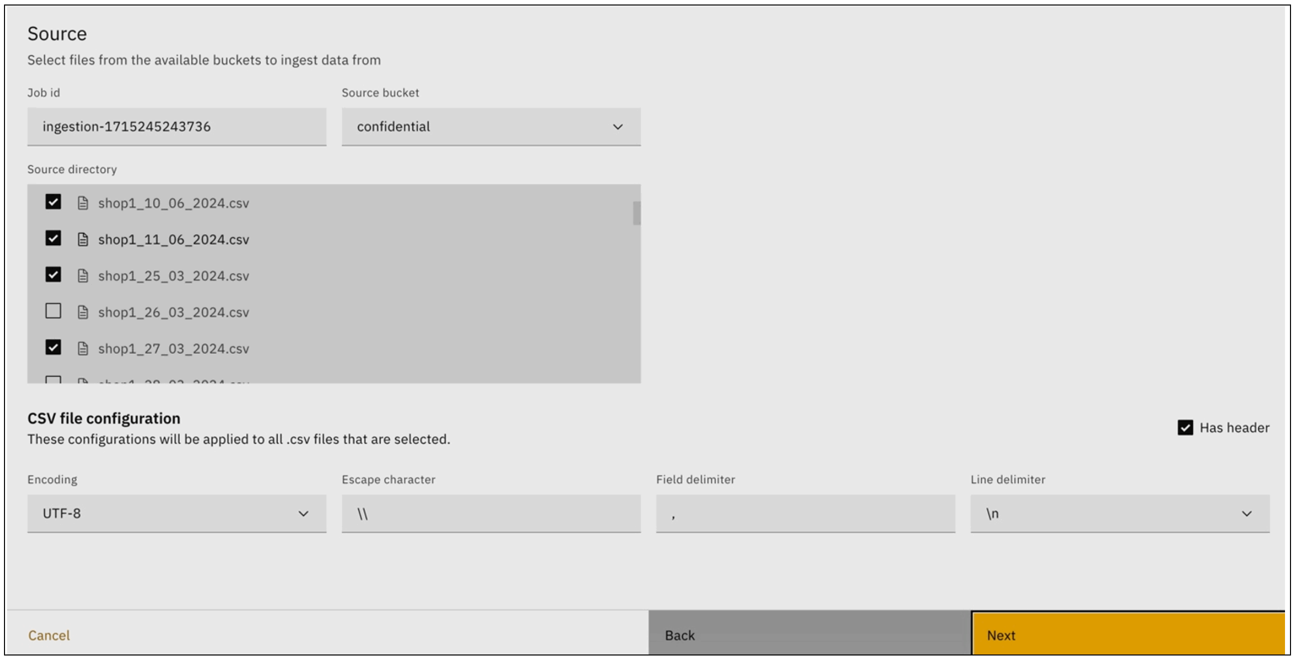
Configure and run an ingestion job on your registered IBM Analytics Engine (Spark).

Supported target table format: **Apache Iceberg**

Figure 8-10 Iceberg copy loader selected

14. Select the files within the `confidential` bucket that you would like to use. See Figure 8-11 on page 132.

> **Note:** We are selecting the files in the `confidential` bucket to be able to use analytics capabilities for user-confidential files. If you would like to expand your analysis with other files in other buckets you can select from other buckets, as well


Source

Select files from the available buckets to ingest data from

Job id: ingestion-1715245243736

Source bucket: confidential

Source directory

- [x] shop1_10_06_2024.csv
- [x] shop1_11_06_2024.csv
- [x] shop1_25_03_2024.csv
- [ ] shop1_26_03_2024.csv
- [x] shop1_27_03_2024.csv

CSV file configuration

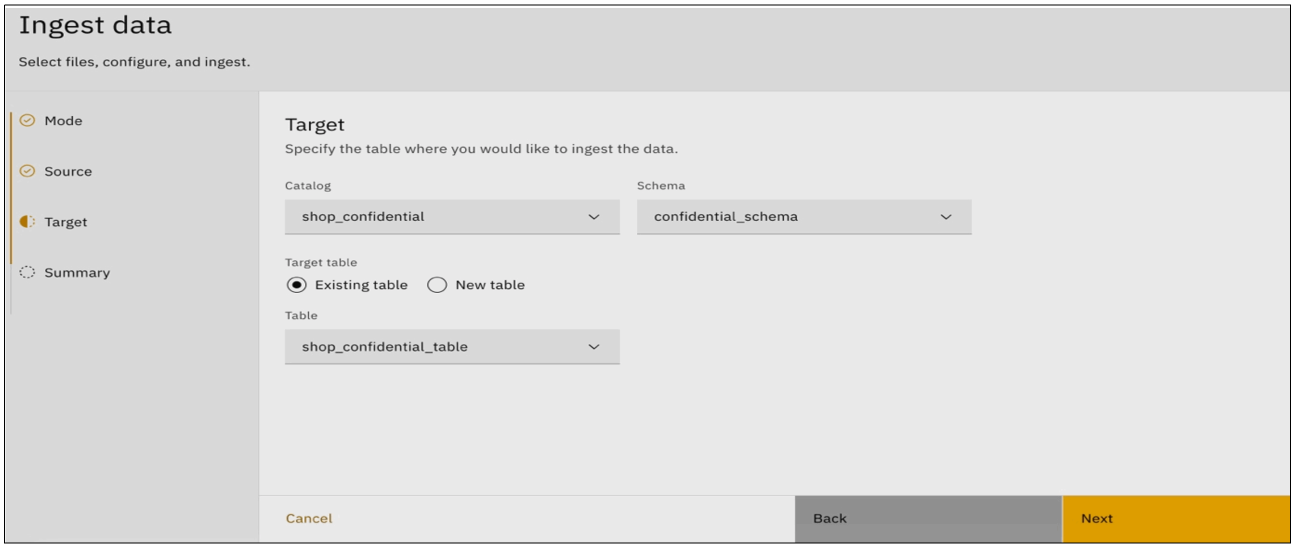
These configurations will be applied to all .csv files that are selected.

[x] Has header

Encoding	Escape character	Field delimiter	Line delimiter
UTF-8	\\	,	\n

Figure 8-11 Select the files within the confidential bucket

15. Click **Next** after selecting the files to be used and specify your target for the ingestion. You can use the same table or create a new one at this step. See Figure 8-12.


Ingest data

Select files, configure, and ingest.

Target

Specify the table where you would like to ingest the data.

Catalog: shop_confidential

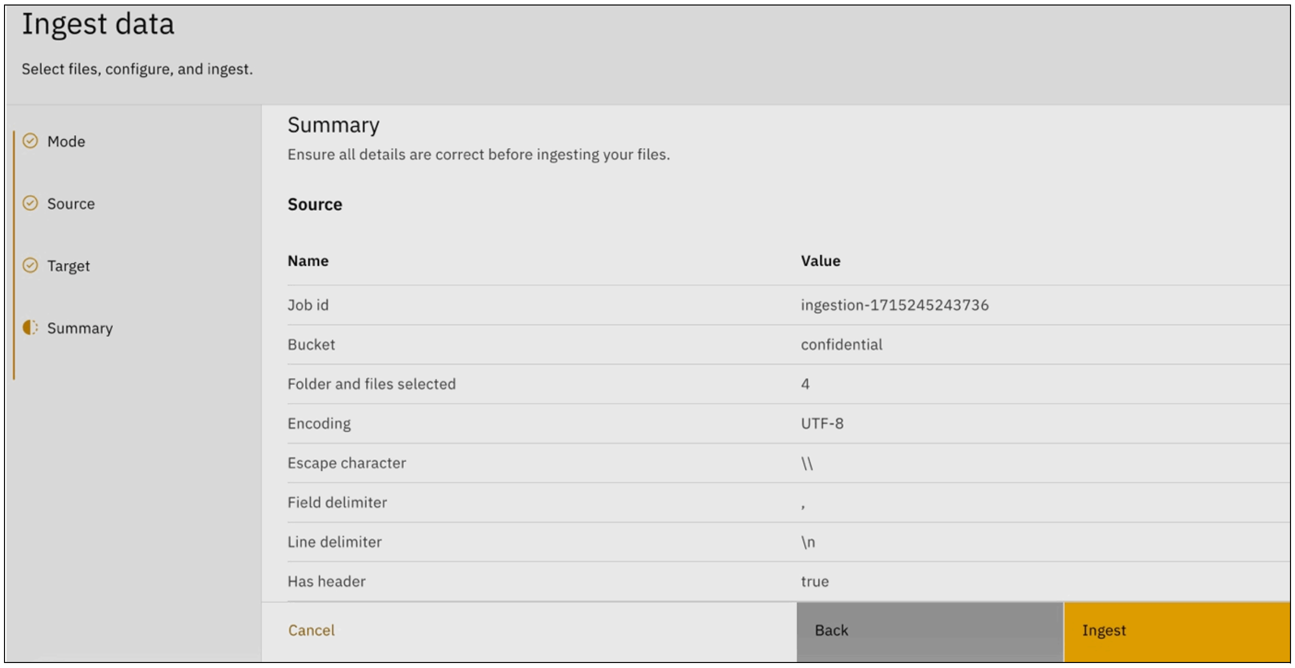
Schema: confidential_schema

Target table: [✓ Existing table] New table

Table: shop_confidential_table

Figure 8-12 Specify your target for the ingestion

16. You will be presented with the Summary screen to review the data. Click **Ingest** to continue with the data ingestion. See Figure 8-13 on page 133.



Ingest data

Select files, configure, and ingest.

Summary

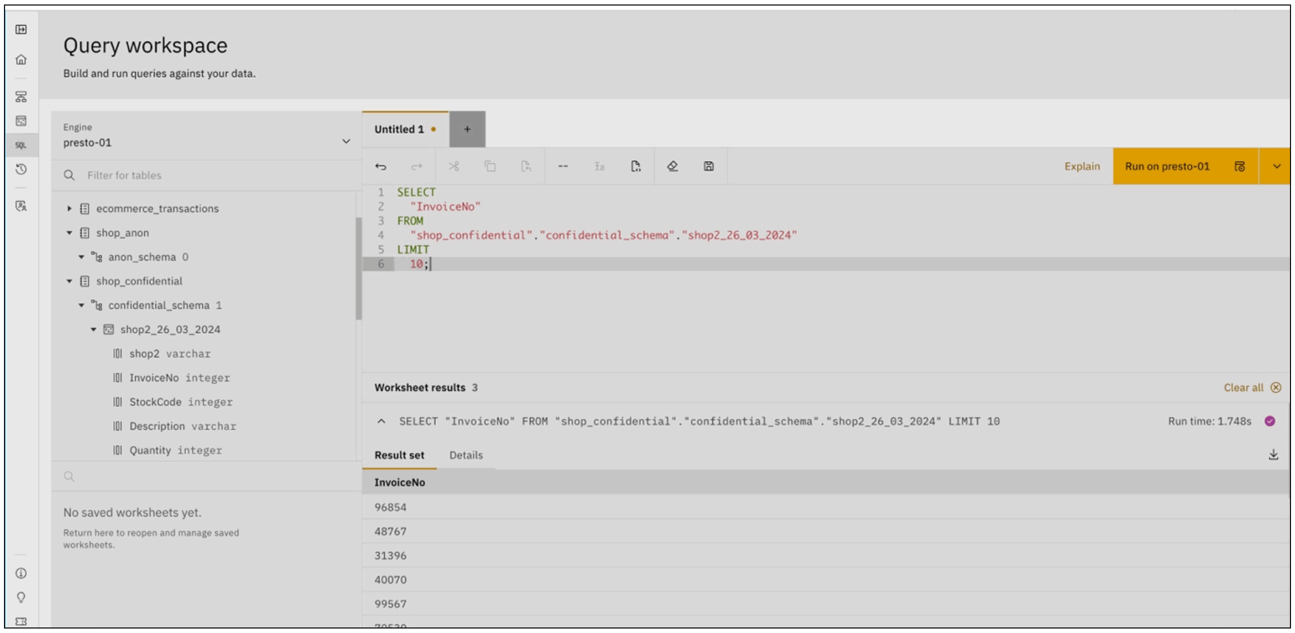
Ensure all details are correct before ingesting your files.

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

Figure 8-13 Summary screen

17. You will be notified when the data ingestion is complete. You can review the status from the **Ingestion jobs** tab on the Data Manager screen.
18. You can query your data tables with the SQL option on the left-side 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.

```
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
```

```
SELECT
  "InvoiceNo"
FROM
  "shop_confidential"."confidential_schema"."shop2_26_03_2024"
LIMIT
  10;
```

No saved worksheets yet.

Return here to reopen and manage saved worksheets.

Worksheet results 3 · Clear all

```
SELECT "InvoiceNo" FROM "shop_confidential"."confidential_schema"."shop2_26_03_2024" LIMIT 10
```

Run time: 1.748s

InvoiceNo
96854
48767
31396
40070
99567

Figure 8-14 Query workspace

8.2 E-commerce

E-commerce platforms are the backbone of modern retail, generating massive amounts of data that can be incredibly 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:

- User sessions (how long users stay on the site).
- Page views (which pages users visit).
- Clicks (what elements users interact with).
- Time spent on specific pages.
- Shopping cart activity (items 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.

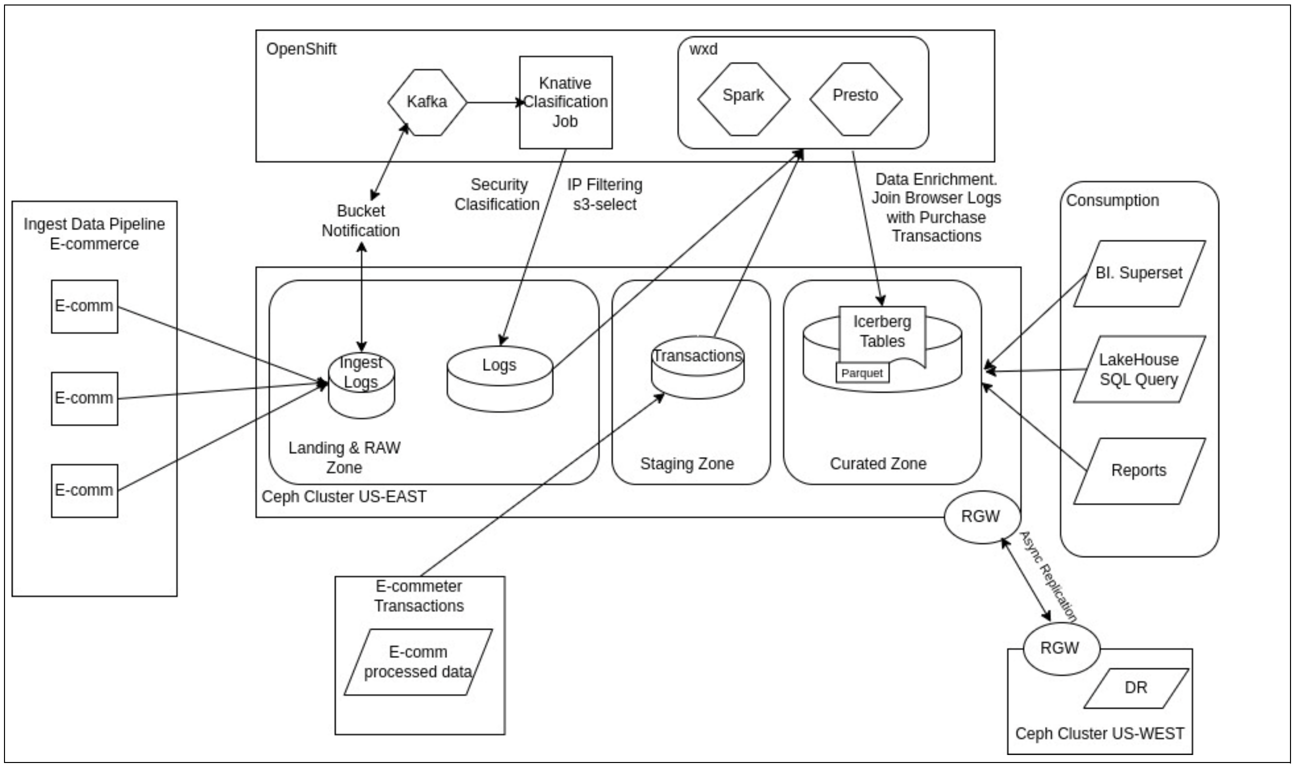


Figure 8-15 Data pipeline architecture for e-commerce

8.2.1 E-commerce: Ingest

To understand the upcoming data processing steps, let us follow the path of our e-commerce data before it lands in the raw zone. We will examine two example datasets: user browsing logs and customer online purchase transactions:

- *Browser logs* from various web servers are shipped to the data lake and ingested as CSV files within a landing zone bucket. This triggers a processing event that scans all customer IPs within the logs. Any IP addresses identified as malicious based on predefined IP ranges are filtered out and removed from the data set.

  Once processed, they are stored on a raw zone bucket called `ecommlogs`.
- *Transaction logs* reach the data lake already processed and converted into Parquet format. For optimized storage and retrieval, we store them in a dedicated bucket named `ecommtrans`.

You can refer to 8.2, “E-commerce” on page 134 for a more detailed description of the Ingest e-commerce pipeline.

8.2.2 E-commerce: Browser log workflow

Upon storing CSV browser logs in the `ecommlogs` bucket, a Spark cleansing job automatically executes. This job removes duplicates or corrects invalid data, transforms the dataset into a Parquet format, and stores the resulting Parquet objects in a date-partitioned structure. This format ensures easy consumption by any SQL engine, like Presto (WXD).

Figure 8-16 on page 135 shows the configured high-level workflow.

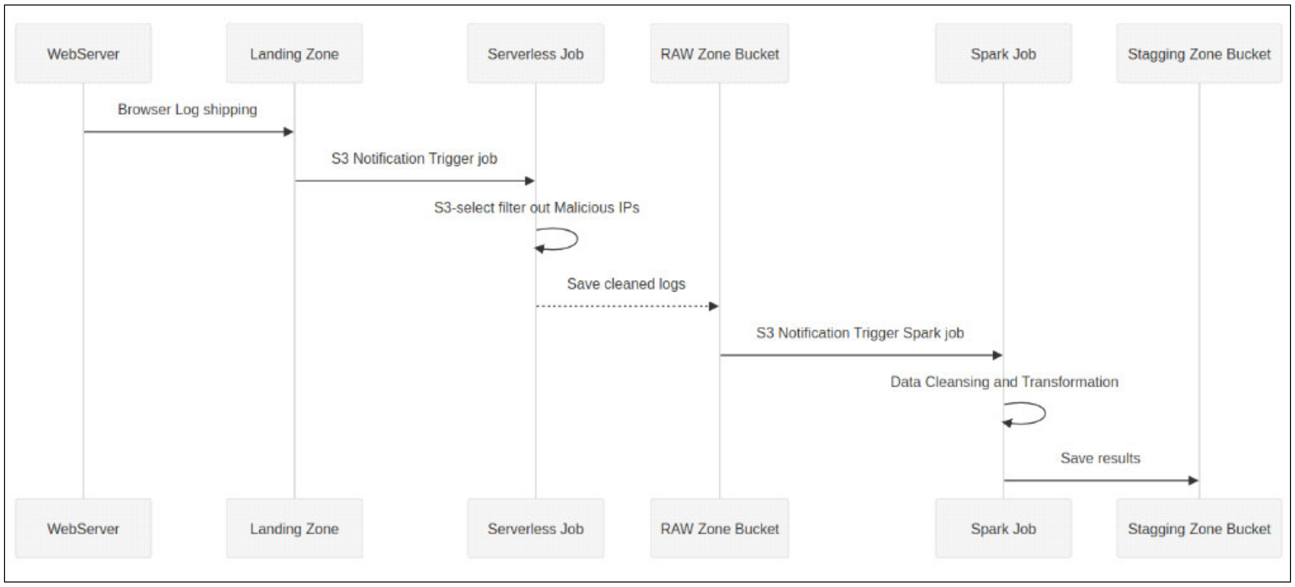


Figure 8-16 E-commerce ingest high-level workflow

8.2.3 E-commerce: Transactions workflow

Unlike browser logs, e-commerce transaction data arrives in the data lake and is already processed and stored in Parquet format within the staging zone's `ecommtrans` bucket. This streamlined approach is possible because:

- **External processing**: The processing of transaction logs occurs outside the data lake pipelines themselves. This might involve a separate system dedicated to transaction data management.
- **Pre-optimized format**: By arriving in Parquet format, a columnar storage format optimized for data lakes, the data is ready for efficient storage and retrieval within the data lake. This eliminates the need for further processing upon landing in the staging zone.

8.2.4 E-commerce: Initial data cleansing

We already have a configured bucket called `ecommlogs` that is part of the raw zone. Data arrives in the `ecommlogs` bucket, the designated landing zone within the raw zone for our e-commerce browser logs. Example 8-4 shows a sample of the dataset.

Example 8-4 Sample dataset

```
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 has changed within the bucket.
2. **Kafka topic delivery**: The S3 bucket notification is then delivered to a designated Kafka topic.
3. **Spark job activation**: The message within the Kafka topic triggers the execution 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 allows the Spark job to identify and process the specific new data that needs to be handled.

As a reference, the example Spark job code is available on the following GitHub link. The Spark Job uses the S3A connector to access the buckets containing the datasets. We will use the S3A IAM AssumeRole integration to access the source and destination buckets. For this example, we are using AssumeRole, Currently, Hadoop/S3A lacks built-in support for `AssumeRoleWithWebIdentity` authentication. However, this functionality 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 `AssumeRoleWithWebIdentity` authentication method so that it can be used from any product that uses the S3A connector like Spark, Hadoop, Dremio, and so forth. The link to the code for the Custom Credentials Provider is `https://github.com/pritha-srivastava/CredentialsProvider.`

1. Continuing with our example, we need to create an IAM role with a policy that will allow the Spark job to access the bucket in the raw zone called `ecommraw` and store the processed objects on a bucket called `ecommlogs`.

If it does not already exist, we will create a bucket named `ecommlogs` within the raw zone. This bucket will be owned by the `ecommadmin` RGW user. The `ecommadmin` user will be our S3 IBM Storage Ceph Object administrator for the e-commerce raw and staging zones.

Since the `ecommadmin` user will be managing the creation and maintenance of the IAM roles, we will provide the user with the roles capability, allowing them to manage the required IAM roles. See Example 8-5.

Example 8-5 Provide the user with 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, let us 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 made by users on our website. This data arrives through an event-driven data pipeline as discussed in 8.2.4, “E-commerce: Initial data cleansing” on page 136.

Example 8-6 Confirm 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 will use the IAM AssumeRole call to access a specific IAM role that will provide us with the required access to the `ecommraw` and `ecommlogs` buckets. The assumed role call checks if a local user from the RGW user list can assume a specific role. The first step for this setup will be creating a local RGW user that we will use to assume the IAM role and gain access to the S3 buckets through a set of temporary secure token service credentials. See Example 8-7.

Example 8-7 Creating a local RGW user

```
# radosgw-admin user create --uid wxdecomm --display-name 'wxdecomm'
```

   The IAM role we will assume will grant the necessary permissions for our Spark job to process e-commerce data efficiently. Here is a breakdown of the specific permissions:

   – Read access to `ecommraw` bucket: This allows the Spark job to retrieve data files from the raw zone for processing.

   – Write access to `ecommlogs` bucket: This 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:

   – Access the raw browsing log data stored in the `ecommraw` bucket.

   – Process the data as needed (for example, cleaning, transforming).

   – Write the processed data to the designated `ecommlogs` bucket within the staging zone for further use.

4. Our JSON IAM role doc policy specified who is allowed to access and assume our new "spark-ecomm" role. In the policy, we are allowing the local rgw user "wxdecomm" to assume the role. See 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. While the radosgw-admin CLI offers IAM role management, the S3 API is another option. For instance, Example 8-9 demonstrates listing recently created roles using the S3 API.

Example 8-9 List the recently created role through 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. Once the IAM role has been successfully created we need to attach the role policies that will 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 will use two separate policies for different buckets.

- **First role policy:** This policy will grant the role access to perform GET and LIST operations on objects within the `ecommraw` bucket. See Example 8-10.
- **Second role policy (Optional):** Additionally, a separate policy can be created to allow the role to put object tags on the processed objects in the `ecommraw` bucket. This helps identify processed objects and prevent redundant processing.

Example 8-10 First role policy

```
# cat iam_policy_ecommerce_spark_read_from_ecommraw.json
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging",
        "s3:Get*",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads"
      ],
      "Resource": [
      "arn:aws:s3:::ecommraw/*",
      "arn:aws:s3:::ecommraw",
      "arn:aws:s3:::ecommraw/"
      ]
    }
  ]
# radosgw-admin role policy put --role-name=spark-ecomm
--policy-name=read-from_ecommraw  --policy-doc=$(jq -rc .
/root/iam_policy_ecommerce_spark_read_from_ecommraw.json)
Permission policy attached successfully
```

7. Our second role policy will take care of providing our Spark job the required access to the `ecommlogs` stagging zone bucket. See 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 dataset. See Example 8-12.

Example 8-12 Assume the “spark-ecomm” role

```
$ aws --profile=wxdecomm sts assume-role --role-arn arn:aws:iam:::role/spark-ecomm
--role-session-name testecommuser | jq .
  "Credentials": {
    "AccessKeyId": "SnFcrc5Mch5C3tymAlW",
    "SecretAccessKey": "VSGPR3W1KESNIWPG1HYHLUFWFYJKM7AY9YYTO6B",
    "SessionToken":
"iyrTmKgTKOezePM6QsYeCikCwrcAoJ27vHNaBA3w7l9KY1Lwv3KccK3GLwRIHOEJI+SInTYMFmhLGrkPA
PkeaiYsOmxv5KcLJK736LqDJcyQub3eEPBsuEoujpfUHYIgTs3Q1v4X7PwogdKy4jASiDJEzCoofJcfaTm
ls9mFljpZe5+iEydNjTtuELn1ve4MY59axk29PzOUqGmB+IjfTHf6rtZQ3632AsSP7uJsyhVY3VbKypQhe
R29lzNK7R16gWTBhsKSbJ4BtClWuXrgtK/BchgSP28z76j+2a+5ihpjXai5x4L6atsskVlhGxghgYwUMNo
Wu15QEdSmXR2wvA==",
    "Expiration": "2024-04-25T12:44:31.236872+00:00"
  },
  "AssumedRoleUser": {
    "Arn": "arn:aws:sts:::assumed-role/spark-ecomm/testecommuser"
  },
  "PackedPolicySize": 0
$ aws --profile=wxdecomm s3 ls s3://ecommlogs
                           PRE browsing/
```

9. Example 8-13 shows a snippet of the Spark job S3A configuration 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, once the full pipeline is triggered, the actual key names will be 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 involved (for example, `ecommraw` and `ecommlogs`).
- **IAM role:** The role to assume for accessing the buckets ("spark-ecomm").
- **RGW STS endpoint:** The endpoint used to obtain temporary credentials.
- **STS session duration:** The lifespan of the temporary credentials 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 only be used to access the specified S3 buckets within that timeframe. See Example 8-14 on page 141.

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 are ready to execute our Spark job and start our script. The script will perform data cleansing, convert the data to Parquet format, and optimize the S3 object layout for efficient querying by partitioning based on log dates. See Example 8-15.

Example 8-15 Executing our 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 if a specific tag has been applied to the processed objects within the `ecommraw` bucket. This tag can serve as an indicator that the data has been successfully processed by the Spark job. See 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 stagging zone `ecommlogs` bucket, where the data has been saved in Parquet format with the partitioning “by-date” scheme in place. See 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 have finished processing the browser logs in the staging zone.

    We also have the e-commerce transaction dataset in Parquet format in the `ecommtrans` bucket. This transaction dataset is stored in the staging zone by a different system not part of our data analytical workloads. See Example 8-18.

Example 8-18 Transaction dataset

```
$aws--profileecommadmins3lss3://ecommtrans/transactions/
main
2024-04-20 20:21:40     794937 transaction_data_20240420201736.parquet
```

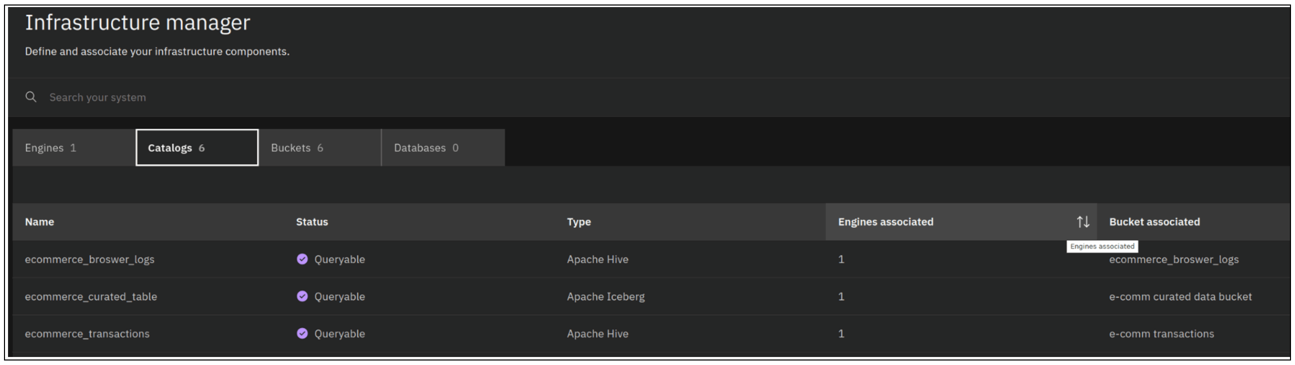
15. Our next objective is to enrich the data by running SQL queries that will JOIN and correlate the transaction dataset with the browser log dataset. The resulting JOIN queries will create views or new tables in the curated zone, providing high-quality data for final users to consume.

    Our data processing pipeline involves two key steps:

- **Staging zone setup:** We will configure watsonx.data to recognize the staging zone buckets `ecommlogs` and `ecommtrans`.
- **Catalog and Presto integration:** We will create a catalog within watsonx.data and connect it to the Presto engine. This will allow us to run queries on the data stored in these buckets.

    We will not cover these steps in detail as they have already been shown in 8.1, "Points of Sale (physical stores)" on page 122. You just need to follow the same steps.

    We will show how our catalogues (Figure 8-17 on page 143) and buckets (Figure 8-18 on page 144) look like once configured from the watsonx.data UI interface.

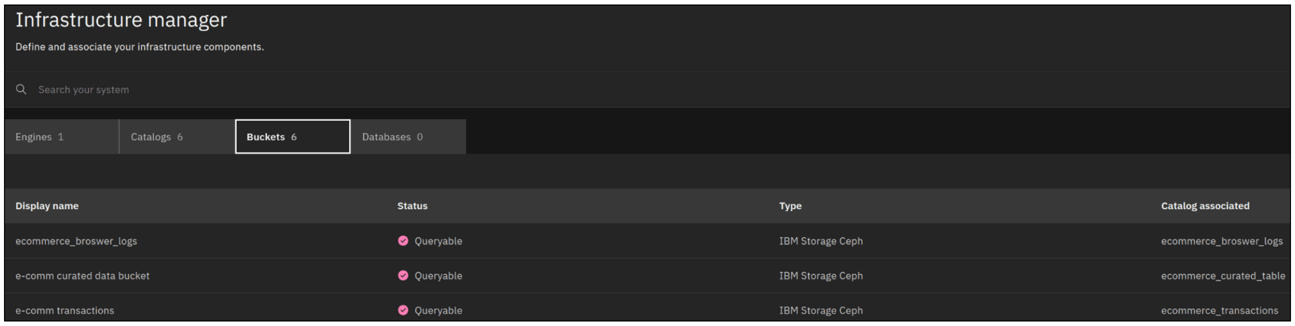


Infrastructure manager

Define and associate your infrastructure components.

Name	Status	Type	Engines associated	Bucket associated
ecommerce_broswer_logs	Queryable	Apache Hive	1	ecommerce_broswer_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

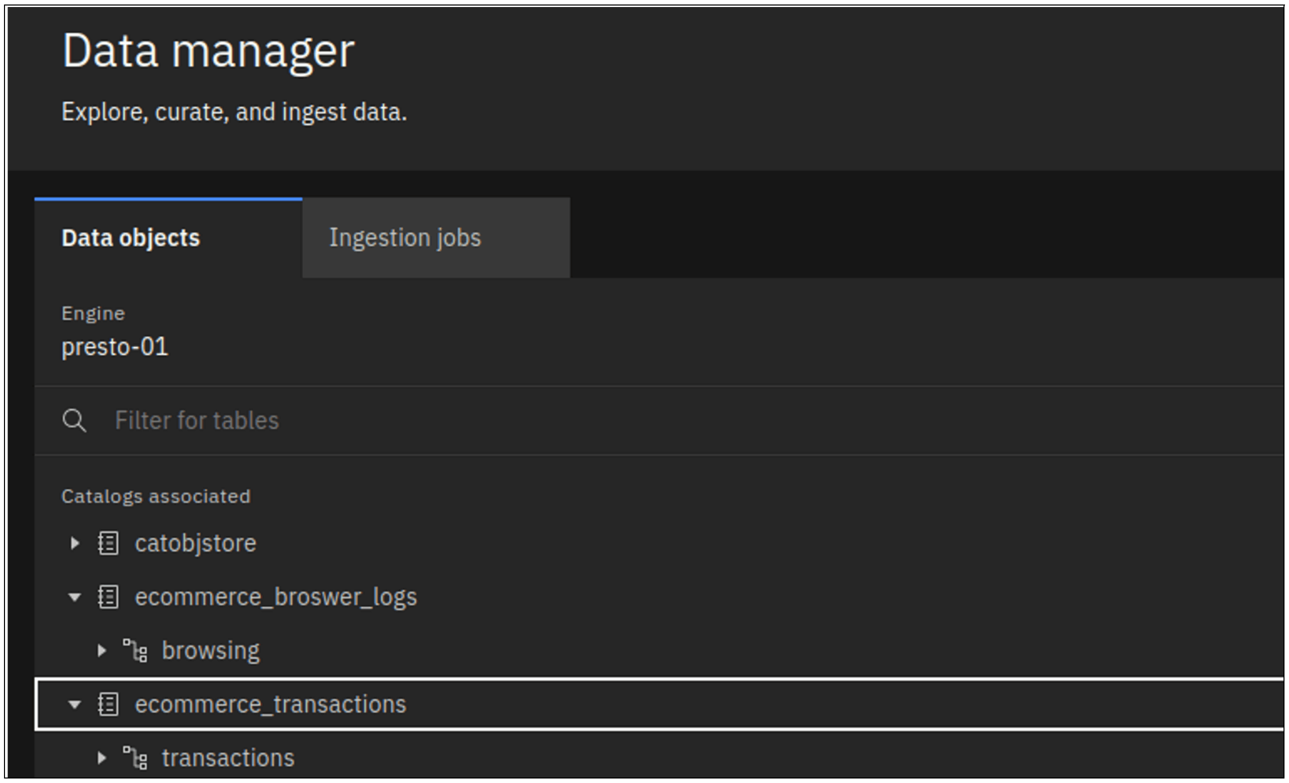

Infrastructure manager

Define and associate your infrastructure components.

Display name	Status	Type	Catalog associated
ecommerce_broswer_logs	Queryable	IBM Storage Ceph	ecommerce_broswer_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 shows the Data manager window.


Data manager

Explore, curate, and ingest data.

Figure 8-19 Data manager objects

Note: While watsonx.data offers a powerful SQL query interface, for this demonstration, we continue working from the presto-cli. This choice provides several benefits:

- **Improved readability:** Presto CLI outputs the query results directly in the terminal window. This format can be easier to read and navigate compared to screenshots from a graphical user interface like watsonx.data.
- **Code focus:** The CLI environment allows us to focus solely on the SQL code itself, potentially improving clarity for users familiar with command-line tools.

Connecting to a Presto instance

For connecting to a IBM Cloud Pak for Data (CP4D) watsonx.data Presto instance, you can follow the detailed instructions at this link. The process can be summarized in the following steps:

1. Downloading and installing the watsonx.data client package. See Appendix B., "Configuring the command line tools for IBM watsonx.data" on page 185 for detailed instructions.
2. Getting the Presto Route from OCP (if connecting from outside OCP). See Example 8-19.
3. Authenticating against the Presto server using the presto-cli binary.

Example 8-19 Connecting to a CP4D WXD 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

Presto CLI provides a terminal-based interactive shell to run queries. In this section we will show you some examples.

1. We will select a default catalogue and schema on the Presto CLI for our session. See Example 8-20.

Example 8-20 Select a default catalogue 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://ecommtrans/transactions/' );
CREATE TABLE
```

2. Before proceeding, we will confirm that our data is accessible for querying through Presto. This ensures we can interact with the data using Presto commands. We will run a select query on the transactions database to check our valid entries. See Example 8-21.

Example 8-21 Confirm 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
----------+--------------------------------------+------------------------------------
------+------------------+-------------+----------+--------------------+----------
-----------+------------------------
    151147 | d326021d-7d9a-4d07-ad8a-cbcb98e9bb8b |
a0cc7c88-598f-419a-9865-731050470e82 | Anklet           | Jewelry     |        2 |
179.45093533953303 | 4826793988494781    | 2024-01-31 00:16:53.000
     67774 | 1cdaed40-24af-470f-b490-6b65ea1d7137 |
584d0685-3e4f-4120-aea1-3feb3cd81480 | Briefcase        | Accessories |        4 |
235.32103048225355 | 213188716095993     | 2024-04-09 07:14:46.000
    348332 | ade78612-6e17-468d-bca1-d13d6c9b5ac8 |
3d401c57-7e9b-415d-8ee1-d590ef44c3f6 | Camera           | Electronics |        1 |
15.713681097442546 | 060432004103        | 2024-04-23 15:11:16.000
    544221 | db8b77ff-411d-473e-87db-eb3ebb886f66 |
03c0d802-018d-4f4e-bbed-6d6097356ffe | Wallet           | Accessories |        2 |
19.473549788852125 | 4878117633003       | 2024-02-11 02:38:36.000
     21300 | 088fced0-b3fc-46bb-a7b7-080d6bfc7e15 |
b4b6733c-7c8e-48be-88bb-58b565e7919c | Boots            | Footwear    |        3 |
29.992012929257477 | 5389913698923269    | 2024-02-08 05:36:40.000
    500227 | 732dcf31-256f-481e-9866-c6b85b2eaa66 |
b9b26b2a-7e66-4c8c-8657-15b34e9b8749 | Shoes            | Footwear    |        3 |
236.57968778431743 | 4761099383053405    | 2024-04-03 04:55:50.000
    889201 | 3d307269-3808-42f9-9f88-3d504cb64ed3 |
cd979a56-1589-4adf-bb60-f195f26c6e53 | Shoes           | Footwear    |        4 |
70.53589141219959 | 4688405084778750813 | 2024-04-25 08:54:06.000
     29812 | c0707ddf-4957-49ce-9711-11eda4a033a0 |
dcca6ae3-74cf-472e-91c5-3abcad14e326 | Keyboard         | Electronics |        3 |
194.46344960197496 | 213184954150426     | 2024-01-15 20:05:03.000
    414108 | e4b572ad-f72b-46ad-a864-38a77c74444c |
1ee8c74b-0de8-410a-b645-5f0e4d7f8033 | TV               | Electronics |        4 |
107.01796367659708 | 4888985346196620    | 2024-01-17 04:43:12.000
    155841 | 61793d8b-b482-4dc1-8074-f991a440829c |
040dfe0c-9de5-496e-883e-3bb1a7899c37 | Sneakers         | Footwear    |        3 |
244.61225511805344 | 4995183362357975    | 2024-03-20 11:27:32.000
(10 rows)
Query 20240425_182004_00064_3yss9, FINISHED, 1 node
Splits: 18 total, 18 done (100.00%)
[Latency: client-side: 0:01, server-side: 0:01] [15 rows, 694KB] [29 rows/s,
1.31MB/s]
```

3. Our transactions table is ready. Let us create our browser log table next. We log into our Presto catalog `ecommerce_broswer_logs` and schema `browsing` and create a new table with the name `browsing_data_partitioned`. We are configuring the database's external

location to "`s3://ecommlogs/browsing`" in Parquet format and partitioned by the "ds" column. See Example 8-22.

Example 8-22 Create our browser log table

```
./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 browsing --catalog ecommerce_broswer_logs
Password:
presto:browsing> CREATE TABLE IF NOT EXISTS browsing_data_partitioned ( 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 DOUBLE, i_category VARCHAR,
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)
```

4. Once the table creation is complete, we will issue a SELECT statement to fetch data from the newly created table. This allows us to verify the table schema and ensure data population. This is common with partitioned tables that need to update the schema metadata. See Example 8-23.

Example 8-23 Issue 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>
```

5. We can run a **CALL** command to synchronize or refresh the table partition metadata. After that, we can see that our SELECT query does provide the expected database entries. See Example 8-24.

Example 8-24 Run a CALL command to synchronize or refresh the table partition metadata

```
presto:browsing> CALL
ecommerce_broswer_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_3yss9, 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. Once we have both of our e-commerce tables ready, we can perform a JOIN SQL operation combining the detailed transaction records with the corresponding browsing behavior to create a unified data view. By linking these two datasets, we can get a complete view of the customer journey from initial interest (browsing) to purchase (transaction).

   By combining browsing data with transaction records, we can uncover valuable patterns that would be hidden in isolated datasets. This enriched view allows for data-driven decision making:

- **Understanding purchase behavior:** Analyze which items were browsed the most before purchasing, revealing customer interest and potential buying triggers.
- **Purchase journey insights:** Explore how long customers took to make a purchase, allowing you to 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, enabling you to 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_broswer_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
 Jewelry     |        17 |          2401 | 0.71
 Electronics |        12 |          2807 | 0.43
```

7. Here is a simple SQL JOIN query that merges the transactions and `browsing_data_partitioned` tables on the `client_id` and `item_id`. See Example 8-26.

Example 8-26 SQL JOIN query

```
presto:browsing> SELECT *
             -> FROM ecommerce_transactions.transactions.transactions t
             -> JOIN ecommerce_broswer_logs.browsing.browsing_data_partitioned b
             -> ON t.client_id = b.customer AND t.item_id = b.resource_fk;
 client_id |           transaction_id             |             item_id
| item_description |  category   | quantity |    total_amount    |
credit_card_number |   transaction_date    |      ip       |         ts
| tz  | verb | resource_type |             resource_fk              | response |
browser |   os    | customer | d_day_name | i_current_price | i_category  |
i_description | c_preferr>
----------+--------------------------------------+--------------------------------
------+------------------+-------------+----------+--------------------+----------
-----------+-------------------------+-----------------+------------------------+
-----+------+---------------+--------------------------------------+----------+---
------+---------+----------+------------+-----------------+-------------+---------
------+---------->
    474819 | d8a9151f-c0f7-4015-a571-39b3536cfbb0 |
2e98168b-e2d5-418d-a0c9-bb8145636abb | Smartwatch       | Electronics |        2 |
175.45342227620418 | 379978089772404     | 2024-03-15 10:57:14.000 |
24.248.204.119  | 2024-03-21 21:10:48.000 | UTC | GET  | Item          |
2e98168b-e2d5-418d-a0c9-bb8145636abb |      200 | Safari  | iOS     |   474819 |
Thursday   |           37.67 | Electronics | Smartwatch    | true     >
    633908 | b5832c2c-4e14-416d-b4ee-49bc484a12bd |
c18bfec0-4b6c-4996-8e8a-e01942292723 | Tiara            | Jewelry     |        1 |
194.3392690088152 | 4882206094378740    | 2024-04-03 11:21:03.000 | 88.217.59.186
| 2024-03-21 20:13:22.000 | UTC | GET  | Item          |
c18bfec0-4b6c-4996-8e8a-e01942292723 |      200 | Opera   | iOS     |   633908 |
Thursday   |           21.63 | Jewelry     | Tiara         | true     >
    485640 | b9388077-cf55-4a8e-b7e4-761038c8b658 |
e372e7d5-a236-4e0d-8b6a-541ba6265655 | Belt             | Accessories |        1 |
89.09583039228963 | 3597179776167723    | 2024-03-07 11:41:25.000 |
173.150.144.221 | 2024-03-21 06:59:22.000 | UTC | GET  | Item          |
e372e7d5-a236-4e0d-8b6a-541ba6265655 |      200 | Edge     | Linux    |   485640 |
Thursday    |            54.5 | Accessories | Belt          | true
```

Strategies for complex SQL queries involving JOINs: Complex SQL queries involving JOINs can be challenging to write, maintain, and can lead to slow performance. There are two main strategies to address this challenge, each with its own advantages and disadvantages:

- We could create a VIEW that will make accessing our complex JOIN query easy for the final users.
- We could also create a new table out of the JOIN query, so further queries can be done on a single table.

Views and tables have their pros and cons. At a high level:

A view is a virtual table based on the result set of an SQL statement. It contains rows and columns, just 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 as it queries the underlying tables dynamically.
  - Simplifies complex SQL into simple queries for end users.
- Cons:
  - Performance can be slower than querying from a physical table because the view must execute the underlying SQL each time 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. Once you have created 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 upon each query against the view. This allows you to 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_broswer_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]
```

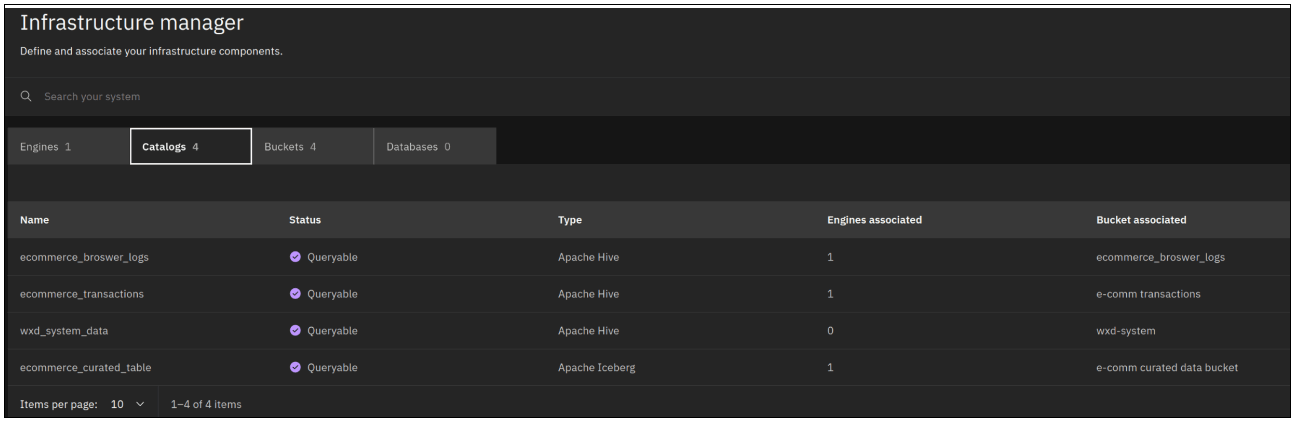
9. The next step is to create a new external table that holds this joined data. This table can be used for more efficient queries since it avoids repeatedly joining these tables and allows quicker access to the combined data set. This new table will be stored in a new bucket that is part of the curated zone. This is the final destination of our e-commerce data. We have cleansed and enriched the dataset, and this new destination bucket will be using the Iceberg table format.

   Before creating the table, we must create and configure the bucket in watsonx.data. As in the previous examples, the buckets will be created by the ecommadmin userid and we will give access to the wxdecomm user through bucket policies. Since there is no current support in watsonx.data for IAM roles, we are using bucket policies to control access to the buckets. See Example 8-28.

Example 8-28 Create and configure the bucket in watsonx.data

```
$ aws --profile ecommadmin s3 mb s3://ecommcurated
$ cat bucket_policy-for-wxd-to-use-ecommcurated.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:::ecommcurated/*",
                "arn:aws:s3:::ecommcurated/",
                "arn:aws:s3:::ecommcurated"
            ]
        }
    ]
$ aws --profile ecommadmin s3api put-bucket-policy --bucket ecommcurated --policy 
file://bucket_policy-for-wxd-to-use-ecommcurated.json
```

10. Once we have the bucket and policy, we will create the new catalogue for watsonx.data using the Iceberg connector and connect it to the Presto engine through the UI. See Figure 8-20 on page 152.


Infrastructure manager

Define and associate your infrastructure components.

Search your system

Engines 1 | Catalogs 4 | Buckets 4 | Databases 0

Name	Status	Type	Engines associated	Bucket associated
ecommerce_broswer_logs	Queryable	Apache Hive	1	ecommerce_broswer_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.Once the catalog and schema are defined, you have two options for utilizing the joined data:

- **Create a permanent table:** Execute a CREATE TABLE statement based on the join query. This creates a new table storing the combined results for future use. See Example 8-29.
- **Temporary analysis:** If the joined data is only needed for immediate analysis, consider using the join query directly within Presto. This avoids creating a separate table.

Example 8-29 Create 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,
    t."transaction date" AS transaction_date
FROM
    ecommerce_transactions.transactions.transactions t
JOIN
    ecommerce_broswer_logs.browsing.browsing_data_partitioned b
ON
    t."client id" = b.customer AND t."item id" = b.resource_fk;
presto:browsing> select * from
"ecommerce_curated_table"."joint_table_logs_trans".combined_browsing_transactions;
       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
|            transaction_id            | item_description |  category   | quantity
|    total_amount     | credit_card_number  |    transaction_date
----------------+-------------------------+-----+------+---------------+----------
----------------------------+----------+---------+---------+----------+-----------
-+-----------------+-------------+---------------+-----------------------+--------
----+--------------------------------------+------------------+-------------+-----
-----+--------------------+--------------------+-------------------------
 25.179.76.191   | 2024-03-16 04:10:47.000 | UTC | GET  | Item          |
dee454d7-07c7-4af3-b33e-a8f3cbe56f7c |      200 | Safari  | MacOS   |   156197 |
Saturday   |           62.48 | Clothing    | Gloves        | true
| 2024-03-16 | 22047b90-48a2-411d-92c5-cc897838f1e6 | Gloves          | Clothing
|        2 | 213.86670698592872 | 3580656663566595    | 2024-03-22 15:26:41.000
 94.100.164.132  | 2024-03-28 21:07:57.000 | UTC | GET  | Item          |
dee454d7-07c7-4af3-b33e-a8f3cbe56f7c |      200 | Firefox | Windows |   156197 |
Thursday   |           81.24 | Clothing    | Gloves        | true
| 2024-03-28 | 22047b90-48a2-411d-92c5-cc897838f1e6 | Gloves          | Clothing
|        2 | 213.86670698592872 | 3580656663566595    | 2024-03-22 15:26:41.000
 188.92.198.17   | 2024-01-04 01:18:37.000 | UTC | GET  | Item          |
dee454d7-07c7-4af3-b33e-a8f3cbe56f7c |      200 | Firefox | Windows |   156197 |
Thursday   |           12.55 | Clothing    | Gloves        | true
| 2024-01-04 | 22047b90-48a2-411d-92c5-cc897838f1e6 | Gloves          | Clothing
|        2 | 213.86670698592872 | 3580656663566595    | 2024-03-22 15:26:41.000
 148.158.37.246  | 2024-04-15 13:33:36.000 | UTC | GET  | Item          |
dee454d7-07c7-4af3-b33e-a8f3cbe56f7c |      200 | Opera   | Android |   156197 |
Monday     |           17.42 | Clothing    | Gloves        | true
| 2024-04-15 | 22047b90-48a2-411d-92c5-cc897838f1e6 | Gloves          | Clothing
|        2 | 213.86670698592872 | 3580656663566595    | 2024-03-22 15:26:41.000
```

12. We finally have all our data cleansed and curated, using the Parquet format for our data files and Iceberg as our table format.

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 allows additions, deletions, and updates to table schemas without breaking old readers. This means queries will run correctly even as the table schema changes. Iceberg also offers hidden partitioning, query optimization, and ACID transactions, which ensure atomicity, consistency, isolation, and durability of data operations.
- Another benefit of using Iceberg is snapshot isolation, which allows readers to access a consistent snapshot of the data, even as the data is being updated, providing a consistent view at all times.
- 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 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 processing into curated Iceberg tables, the previously stored data in the staging zone will be automatically cleaned and removed using an S3 lifecycle policy. This 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 allows for rerunning the entire data pipeline if necessary. Consequently, 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 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 using scheduled jobs that will incrementally update the table data through SQL INSERTS, to more complex scenarios for large datasets that will use a data pipeline tool for the orchestration like Apache Airflow.

During this walkthrough, we have seen examples where data gets copied across different data lake zones. While this approach can be useful in some cases, it is not always necessary.

IBM watsonx.data empowers you to directly query data residing in its raw zone. This eliminates the need for creating additional copies of the data, saving storage space and processing time. This strategy is particularly beneficial for:

- **Real-time decision making**: When immediate insights are required, querying the raw zone data directly allows for 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 by enabling direct raw zone queries.

Here, we finish our Transformation layer section. We will continue our journey in Chapter 9, “Consume” on page 155, where we will show some basic examples of consuming the cleansed and enriched dataset.

9

Consume

The curated zone acts as the bridge between the raw data you ingest and the valuable insights 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.

In this chapter we will share some examples of consuming data from the curated zone through visualization tools like Apache Superset, and optimized SQL tables through watsonx.data SQL query engine.

This chapter has the following sections:

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 end-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 required for analytical purposes. This zone contains data that has been cleansed, enriched, transformed, and catalogued 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 ready for analysis. This data undergoes a rigorous cleaning process to eliminate inaccuracies, inconsistencies, and redundancies. It adheres to strict organizational data quality standards, ensuring 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, JSON, TXT), curated data is typically structured and optimized for efficient querying and analysis. This often involves transforming it into columnar formats like Parquet, Avro, or ORC (Optimized Row Columnar), which are specifically 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 tools, machine learning models, and other analytics applications without requiring further transformation. In this chapter we will share some examples of consuming data from the curated zone through visualization tools like Apache Superset, and optimized SQL tables through watsonx.data SQL query engine.

9.2 Visualization: E-commerce example

In our e-commerce retail example, we have processed the raw data obtained from the server logs into the raw zone. The logs have undergone multiple processes, including security checks to remove fake entries from malicious or non-human customers. The logs have also been cleansed to remove duplicate and inconsistent entries and transformed into parquet file format. Now, they are available for easy table partitioning by date if required.

In the staging zone, the browser logs merge with the e-commerce transaction dataset 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 datasets into clear and compelling visuals, empowering decision-makers to:

- Grasp difficult concepts at a glance.
- Spot hidden patterns with ease.
- Make faster and more informed decisions.

In this section, we will demonstrate an example of visualizing curated data using Apache Superset. This section provides a high-level example using Apache Superset to visualize a sample of our curated dataset. For a deeper dive into Superset's functionalities, refer to the official documentation.

Apache Superset is a powerful, open-source web application designed for data exploration and visualization. It caters to enterprises by offering a user-friendly 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, easy integration with different databases, customizable dashboards with a simple drag-and-drop 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 besides Apache Superset, here is list of some of them in no particular order:

- **Tableau:** Tableau is a leading BI software that allows users to create and share interactive charts, graphs, dashboards, and complex visual stories. It is renowned for its powerful data visualization capabilities and user-friendly interface.
- **Microsoft Power BI:** 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 is known for its deep integration with other Microsoft products.
- **Looker:** Looker, now part of Google Cloud, offers a data exploration and discovery business intelligence platform. It provides powerful analytics and insights through its robust data modeling language.
- **Sisense:** Sisense empowers builders to manage, analyze and visualize complex data, to deliver insights at the right time, on the right device. It is known for its ability to be embedded into apps to deliver analytics at the point of decision.
- **Metabase:** Metabase is an open-source business intelligence tool that lets you ask questions about your data, and displays answers in formats that make sense, whether that's a bar graph or a detailed table. Its ease of setup and use are notable.
- **Redash**: Redash helps you make sense of your data. It allows you to connect and query your data sources, 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 a variety of deployment methods to suit your needs. These include local installations, containerization with Docker or Podman, and deployments on Kubernetes using Helm charts.

We have a running RHEL 9 OS system with Podman available, where we will spin-up a container called superset based on the "apache/superset" container image. Additionally, we will configure the Superset secret key using an environment variable. See Example 9-1.

Example 9-1 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 will use the **`podman exec`** command to connect to the running container and execute commands within its environment. Once inside the container, we will run the superset **`create-admin`** command. This will guide you through setting up an administrator account, including defining a password you will use to access the Superset dashboard. See Example 9-2 on page 158.

Example 9-2 Setting up 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 will be available through a web browser on port 8088 of the RHEL OS host. See Example 9-3.

Example 9-3 Superset UI will be 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 screen.

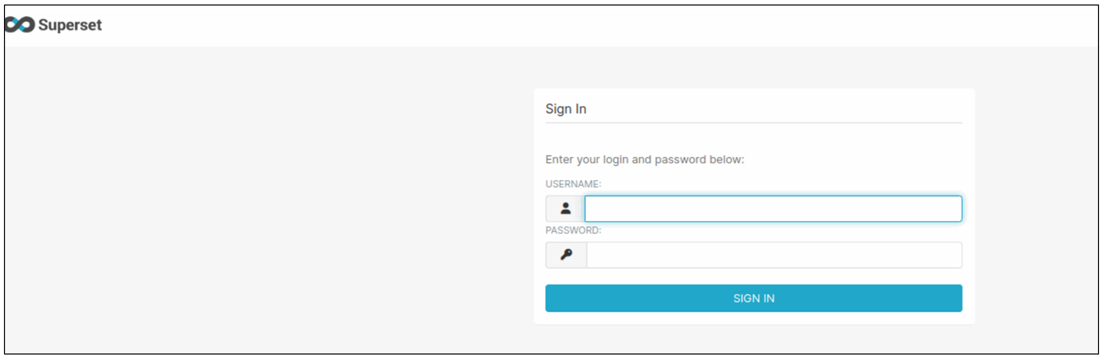


Figure 9-1 Sign In screen

Now that we have logged in with our administrator credentials, it is time to connect Superset to your data source. In this example, we will configure Superset to access the PrestoDB SQL engine we have set up in watsonx.data. See Figure 9-2 on page 159.

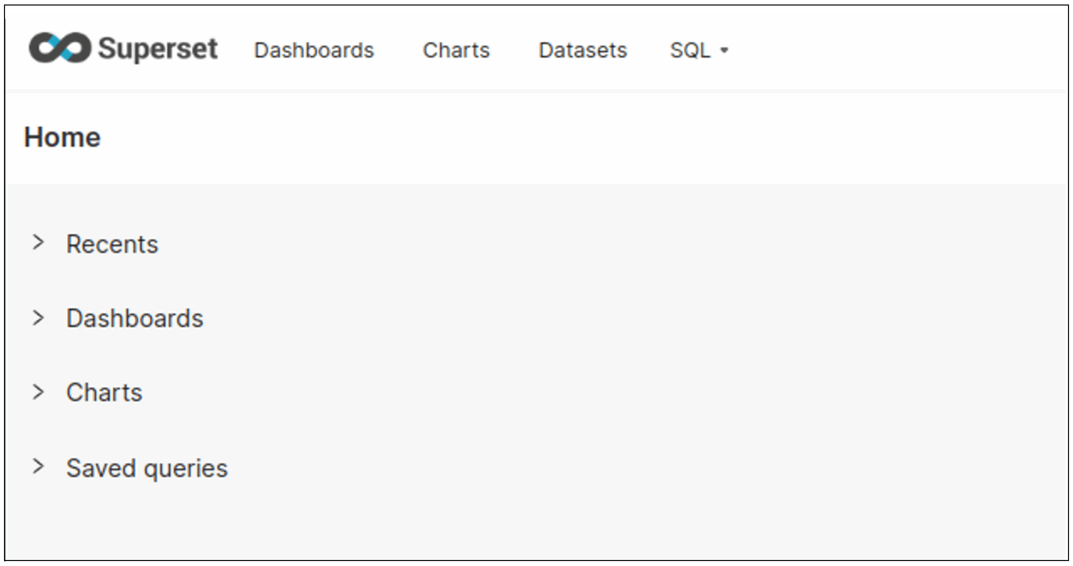


Figure 9-2 Configure Superset to access the PrestoDB SQL engine

9.2.2 Connection Apache Superset to PrestoDB (watsonx.data)

In this next step we will go through an example of connecting Apache Superset to our running PrestoDB instance.

1. We will first select **Settings** and **Database Connections**. See Figure 9-3

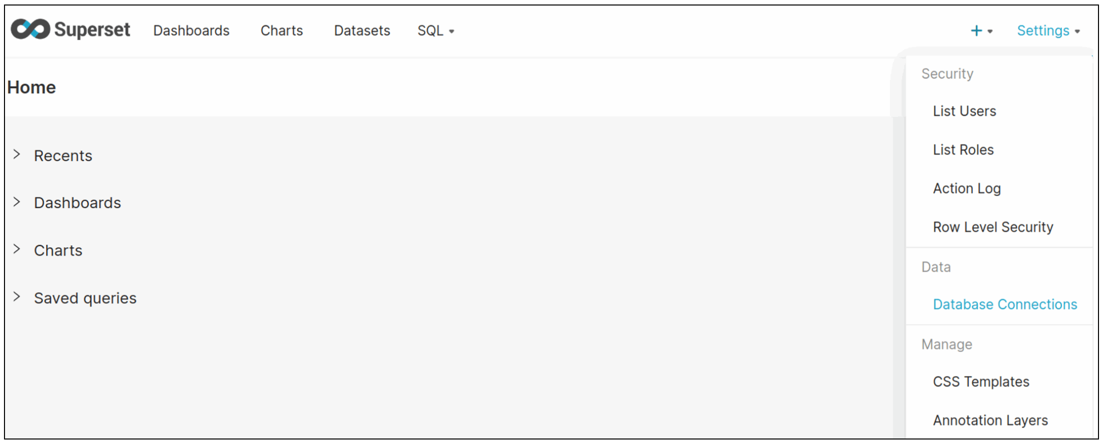


Figure 9-3 Select Settings and Database Connections.

2. In the Databases section we will select **+ DATABASE** to add a new database connection. See Figure 9-4 on page 159.

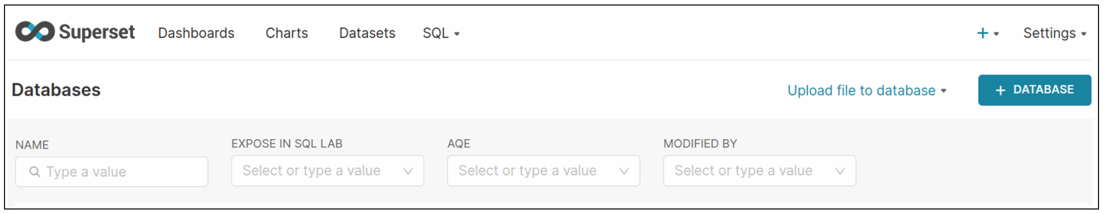


Figure 9-4 Add a new database connection

3. We select **Presto** from the list. See Figure 9-5 on page 160.

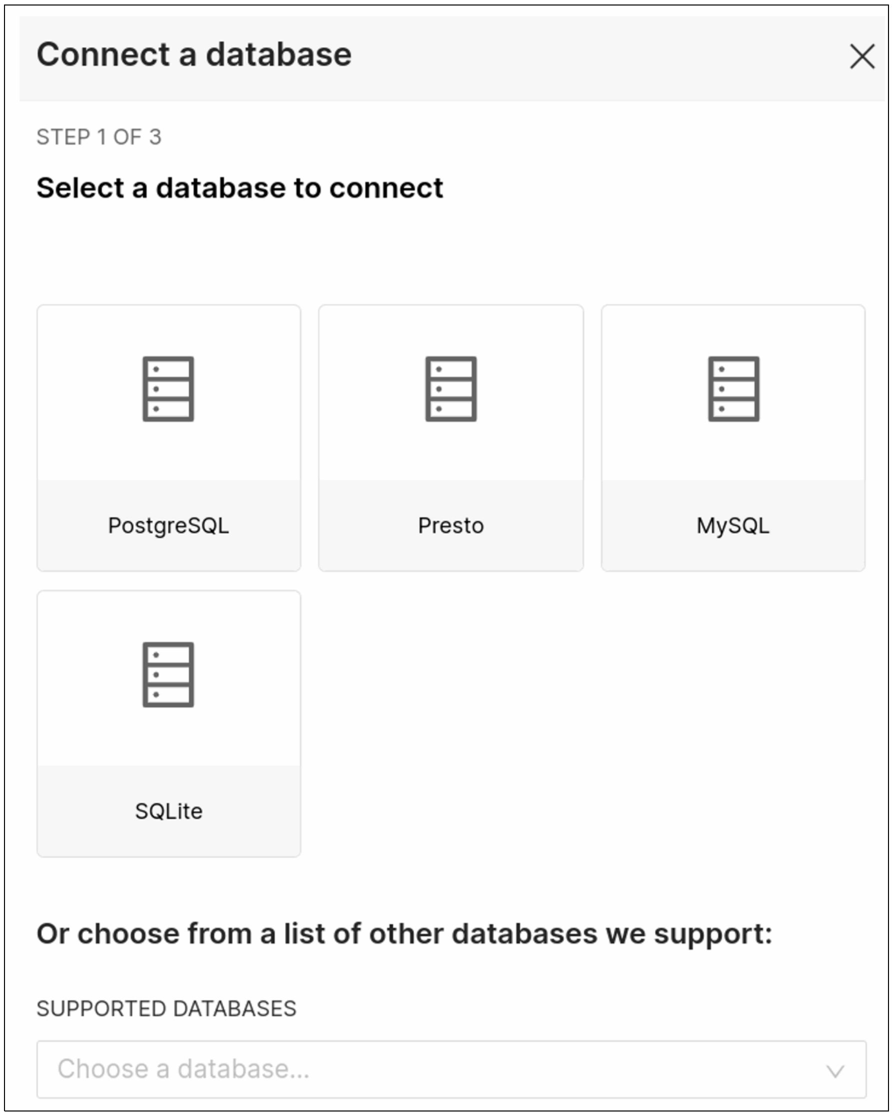
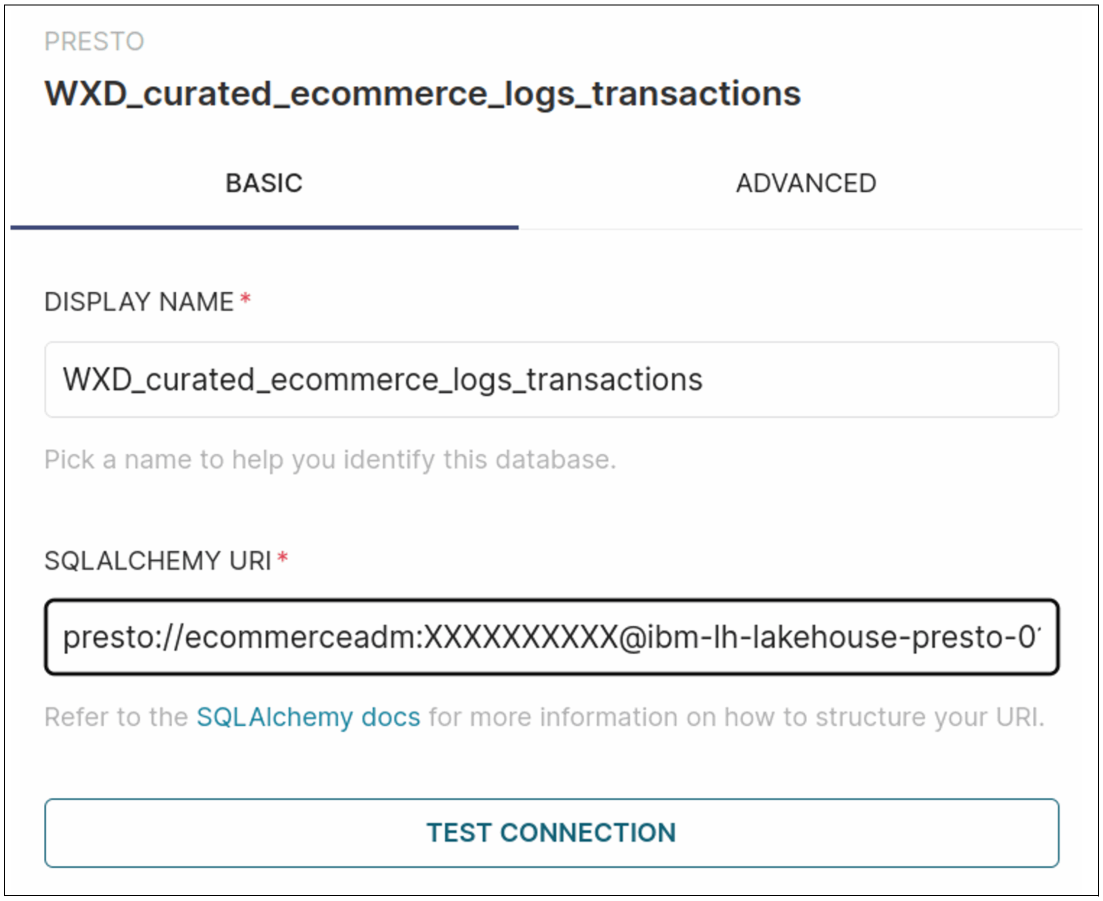


Figure 9-5 Select Presto from the list

4. The BASIC Configuration tab lets you define key details for your data source. Here, we will provide a user-friendly display name and the SQLAlchemy connection URI, which specifies how Superset connects to our database. See Figure 9-6 on page 161.


PRESTO

WXD_curated_ecommerce_logs_transactions

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.

Figure 9-6 BASIC Configuration tab

5. We will connect to a curated dataset in the catalog. This dataset resides in a bucket within the curated zone and combines our e-commerce browsing logs and transaction data in a single table for easier analysis.

   So, the URI is to use for this example is shown in Example 9-4.

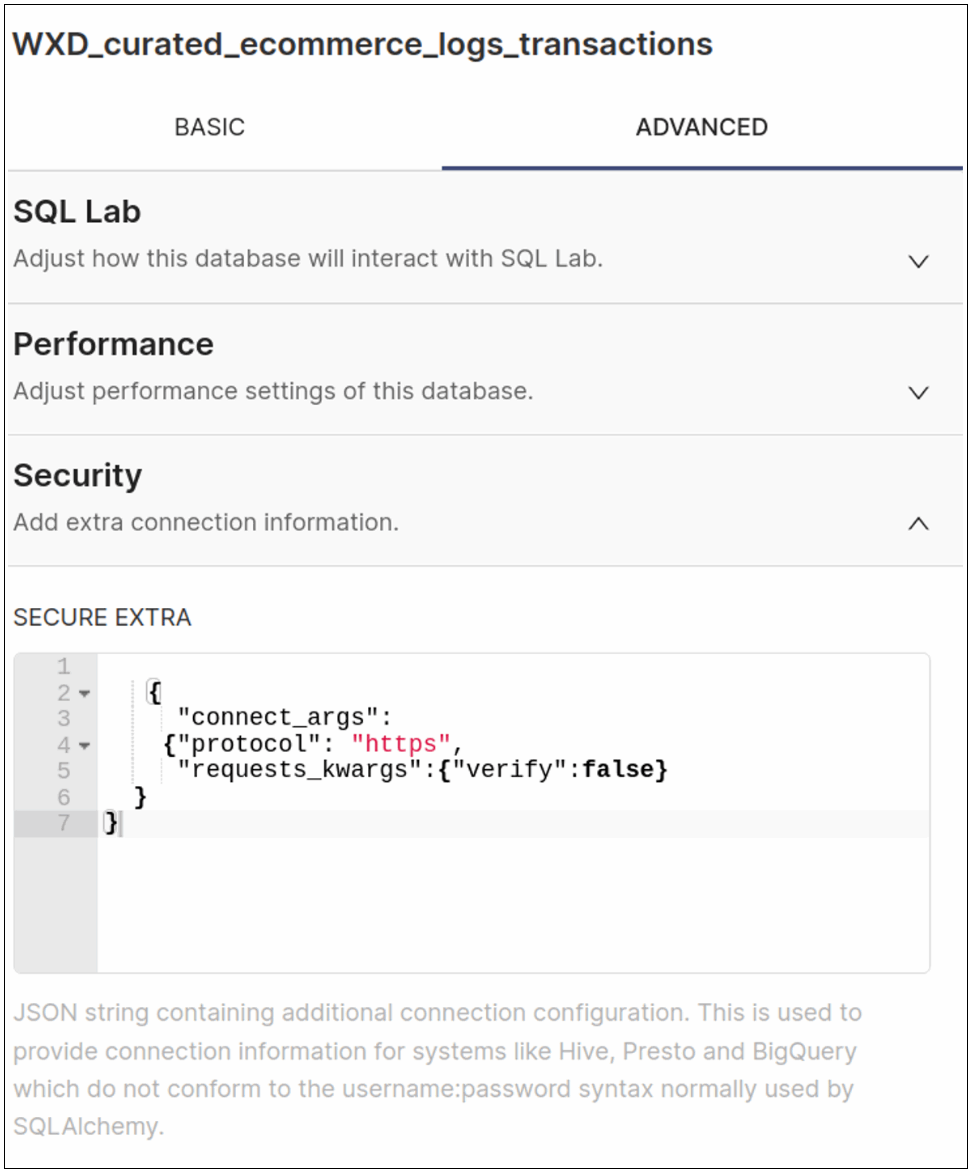
Example 9-4 URI used for this example

```
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 will use a username and password combination to access the PrestoDB catalog named `ecommerce_curated_table`.

   Since we are using SSL for the PrestoDB connection, but the Superset deployment does not trust the PrestoDB SSL certificate, we will need to add an entry "`verify:false`" in the Advanced security settings. See Figure 9-7 on page 162.

Tip: In production, *ensure your Superset deployment trusts the SSL certificate used by your PrestoDB instance*. This can be achieved by installing the trusted CA certificate or configuring mutual TLS authentication. Bypassing this check (`verify:false`) introduces security vulnerabilities and should be avoided in a production environment.



WXD_curated_ecommerce_logs_transactions

SQL Lab

Adjust how this database will interact with SQL Lab.

Performance

Adjust performance settings of this database.

Security

Add extra connection information.

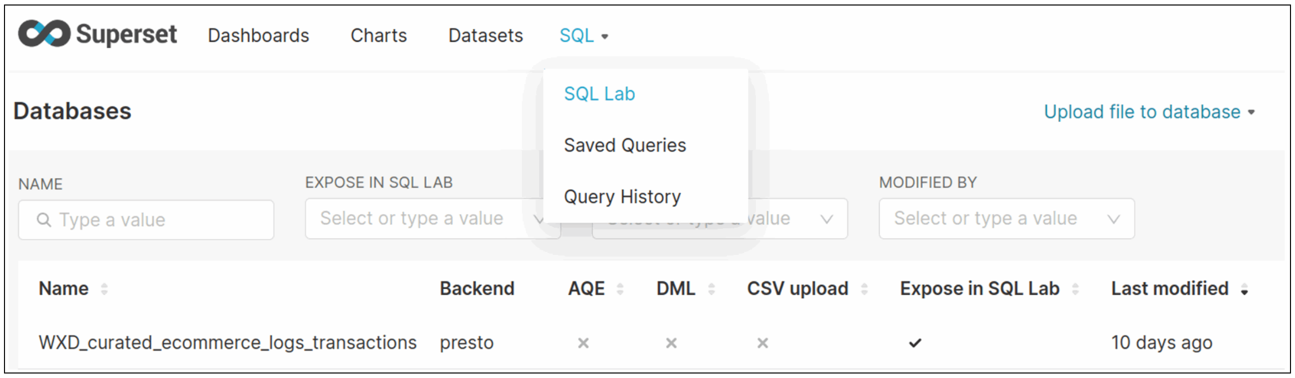
SECURE EXTRA

```
    {
      "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 Add the entry "verify:false" in the Advanced security settings

7. After saving your configuration, click the **Test connection** button to confirm that Superset can successfully connect to your PrestoDB instance on OpenShift.
8. At this point we can go into the Superset **SQL Lab** to start creating our charts. See Figure 9-8 on page 163.



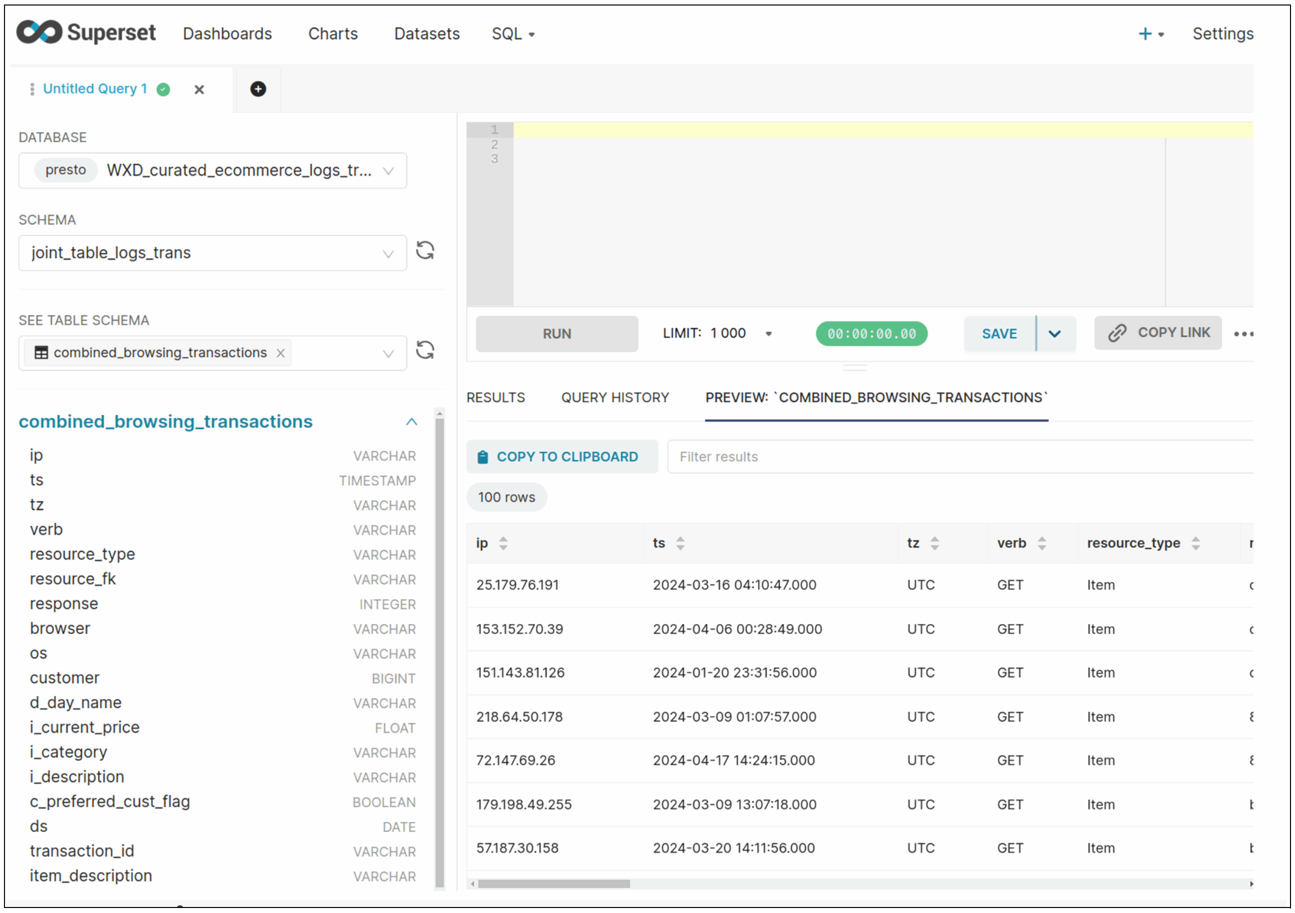
Name	Backend	AQE	DML	CSV upload	Expose in SQL Lab	Last modified
WXD_curated_ecommerce_logs_transactions	presto	×	×	×	✓	10 days ago

Figure 9-8 Superset SQL Lab

9.2.3 Creating a graph or dataset from a SQL query

In this section we will walk you through an example of creating a chart from a SQL query. We will then populate our E-commerce visualization dashboard with this chart.

1. In the Superset SQL Lab, we can select our database or catalog, schema and table. In our case, we will choose the **combined_browsing_transactions table** that we created in Chapter 8, “Transform: Staging and curated zones” on page 121.
2. Superset will automatically display the table schema, providing a detailed overview of the data structure. Additionally, we will be able to preview a sample of the data to get a quick glimpse into its contents. See Figure 9-9 on page 164.



DATABASE: presto WXD_curated_ecommerce_logs_tr...

SCHEMA: joint_table_logs_trans

SEE TABLE SCHEMA: combined_browsing_transactions

LIMIT: 1 000

00:00:00.00

combined_browsing_transactions

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

100 rows

ip	ts	tz	verb	resource_type	r
25.179.76.191	2024-03-16 04:10:47.000	UTC	GET	Item	c
153.152.70.39	2024-04-06 00:28:49.000	UTC	GET	Item	c
151.143.81.126	2024-01-20 23:31:56.000	UTC	GET	Item	c
218.64.50.178	2024-03-09 01:07:57.000	UTC	GET	Item	8
72.147.69.26	2024-04-17 14:24:15.000	UTC	GET	Item	8
179.198.49.255	2024-03-09 13:07:18.000	UTC	GET	Item	b
57.187.30.158	2024-03-20 14:11:56.000	UTC	GET	Item	b

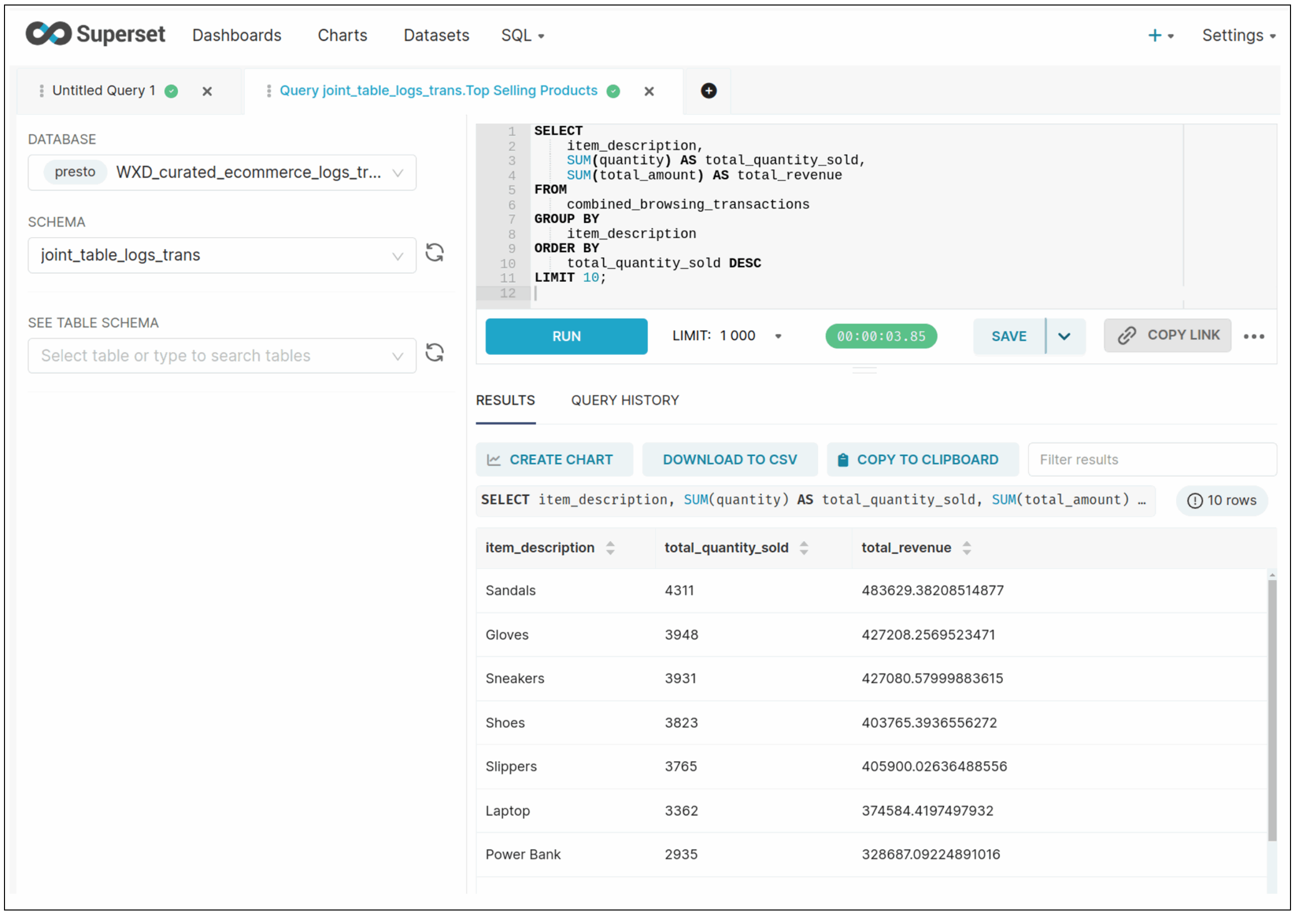
Figure 9-9 The table schema, displaying a detailed overview of the data structure

3. We will insert a query to get the top 10 selling items. See Example 9-5.:

Example 9-5 Query to get 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 executing our query in SQL Lab, we will see the results displayed. Superset also provides an option to “Create a chart” from the query data. We will leverage this feature to create visual representations of your analysis. See Figure 9-10 on page 165.



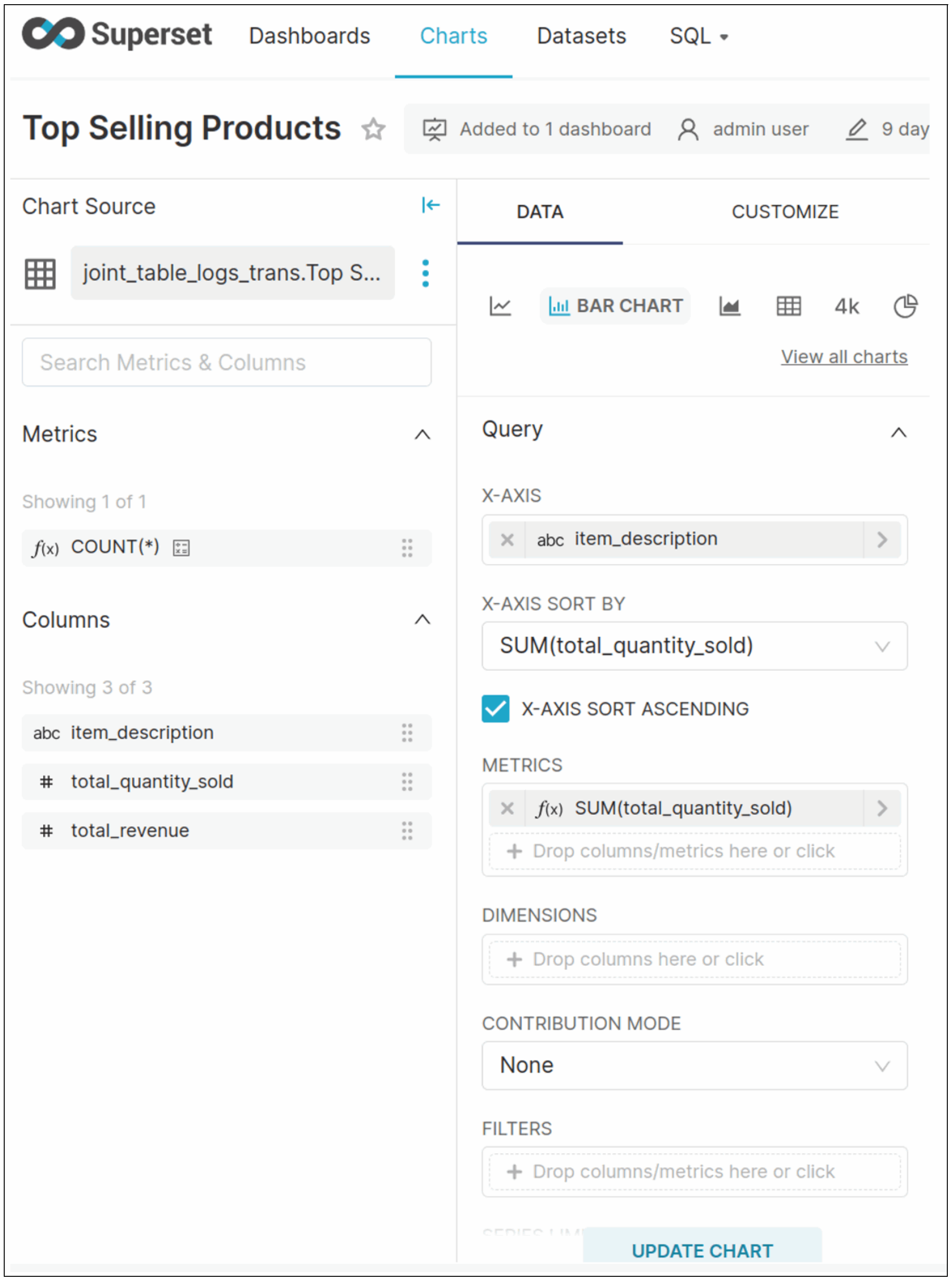
```
SELECT
    item_description,
    SUM(quantity) AS total_quantity_sold,
    SUM(total_amount) AS total_revenue
FROM
    combined_browsing_transactions
GROUP BY
    item_description
ORDER BY
    total_quantity_sold DESC
LIMIT 10;
```

```
SELECT item_description, SUM(quantity) AS total_quantity_sold, SUM(total_amount) …
```

item_description	total_quantity_sold	total_revenue
Sandals	4311	483629.38208514877
Gloves	3948	427208.2569523471
Sneakers	3931	427080.57999883615
Shoes	3823	403765.3936556272
Slippers	3765	405900.02636488556
Laptop	3362	374584.4197497932
Power Bank	2935	328687.09224891016

Figure 9-10 Superset displays the results

5. Once in the Charts section, we will use a bar chart to visualize the data. We will configure the chart's elements as follows (See Figure 9-11 on page 166):
   - **X-axis**: We will assign the `Item_description` field to the X-axis. This will display individual item descriptions along the horizontal axis of the chart.
   - **Sorting**: To prioritize the most frequently sold items, we will sort the data by `Total_quantity_sold` in descending order. This ensures items with the highest sales volumes appear at the top or right side of the chart.
   - **Metric**: We will use the `SUM(total_quantity_sold)` metric to aggregate the total quantity sold for each item. This will provide a clear view of sales performance across different items.



joint_table_logs_trans.Top S...

Metrics
Showing 1 of 1
f(x) COUNT(*)

Columns
Showing 3 of 3
abc item_description
# total_quantity_sold
# total_revenue

Query
X-AXIS
abc item_description
X-AXIS SORT BY
SUM(total_quantity_sold)
✓ X-AXIS SORT ASCENDING
METRICS
f(x) SUM(total_quantity_sold)
DIMENSIONS
CONTRIBUTION MODE
None
FILTERS

Figure 9-11 Visualize the data

6. This will provide us with an easy to visualize graph that shows the top-selling e-commerce products. See Figure 9-12 on page 167.

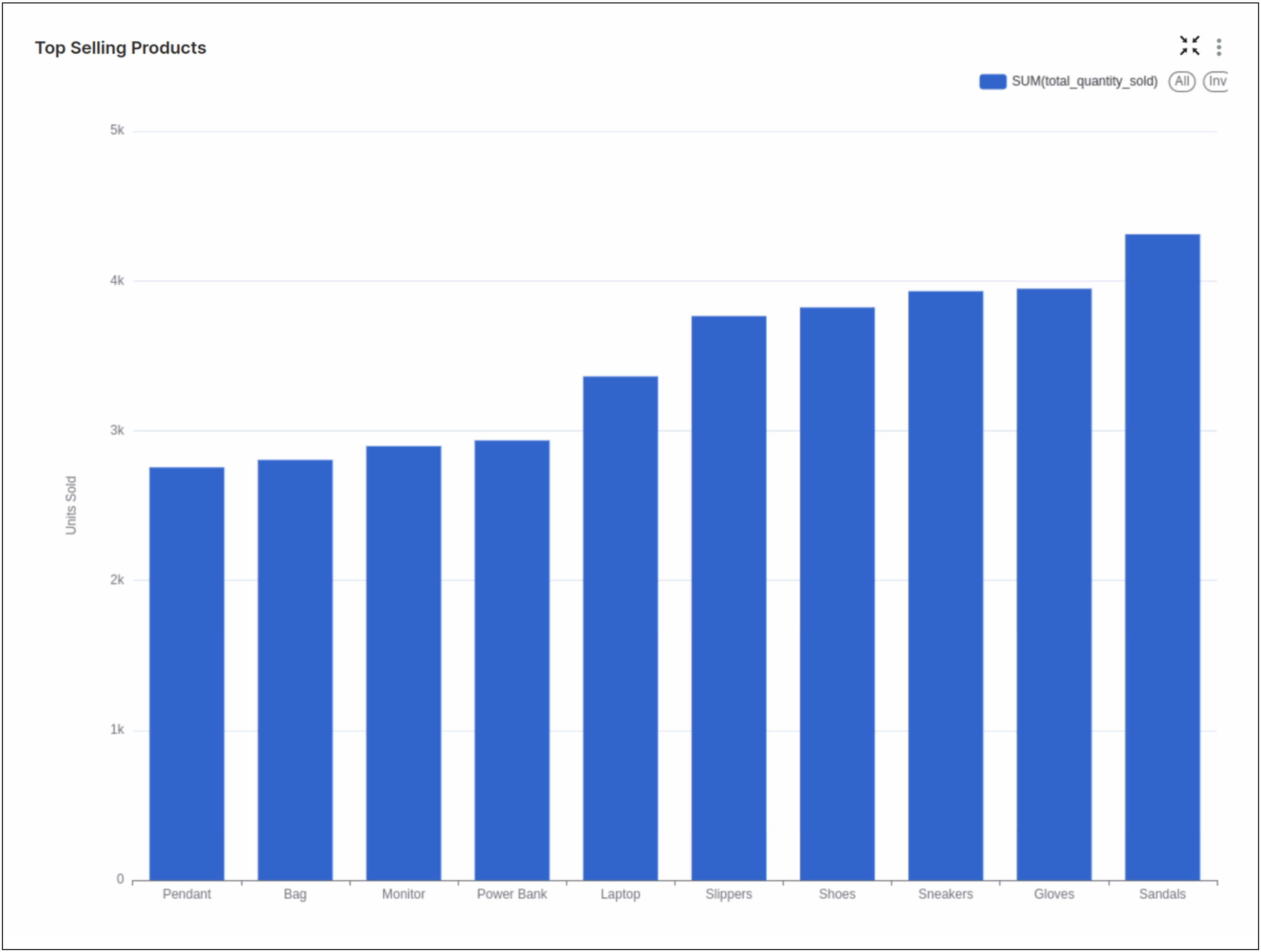
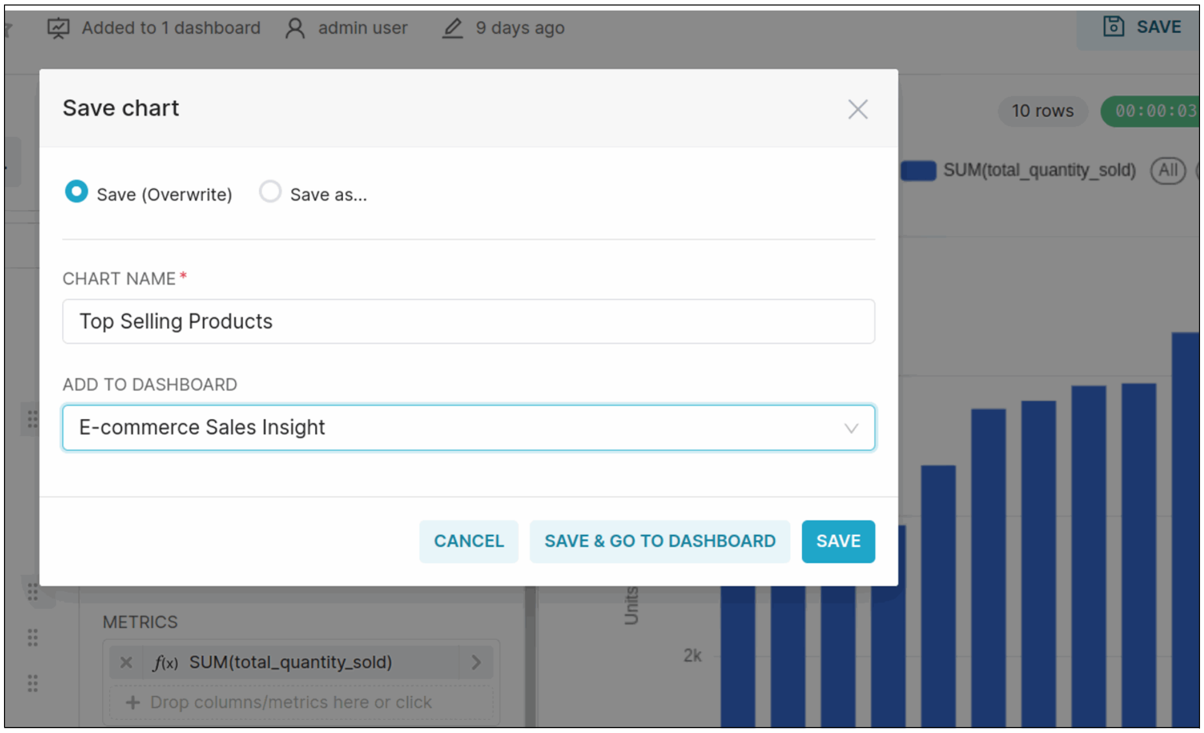


Figure 9-12 Top-selling e-commerce products

7. Once we have crafted a chart that effectively visualizes our findings, we can use the **Save** button on the right save it. We can choose to directly save the chart to a dashboard. In our example, we will start building a dedicated e-commerce dashboard that we will call “E-commerce Sales Insight”. See Figure 9-13 on page 168.



Save chart

[✓ Save (Overwrite)] Save as...

CHART NAME *

Top Selling Products

ADD TO DASHBOARD

E-commerce Sales Insight

CANCEL SAVE & GO TO DASHBOARD SAVE

Figure 9-13 Save chart

8. This example demonstrates how to create a single chart in Superset. The next section of the dashboard leverages the same principles to build additional charts. These charts will delve deeper into your e-commerce data, providing easy-to-consume high-value data around our e-commerce business. See Figure 9-14 on page 168.

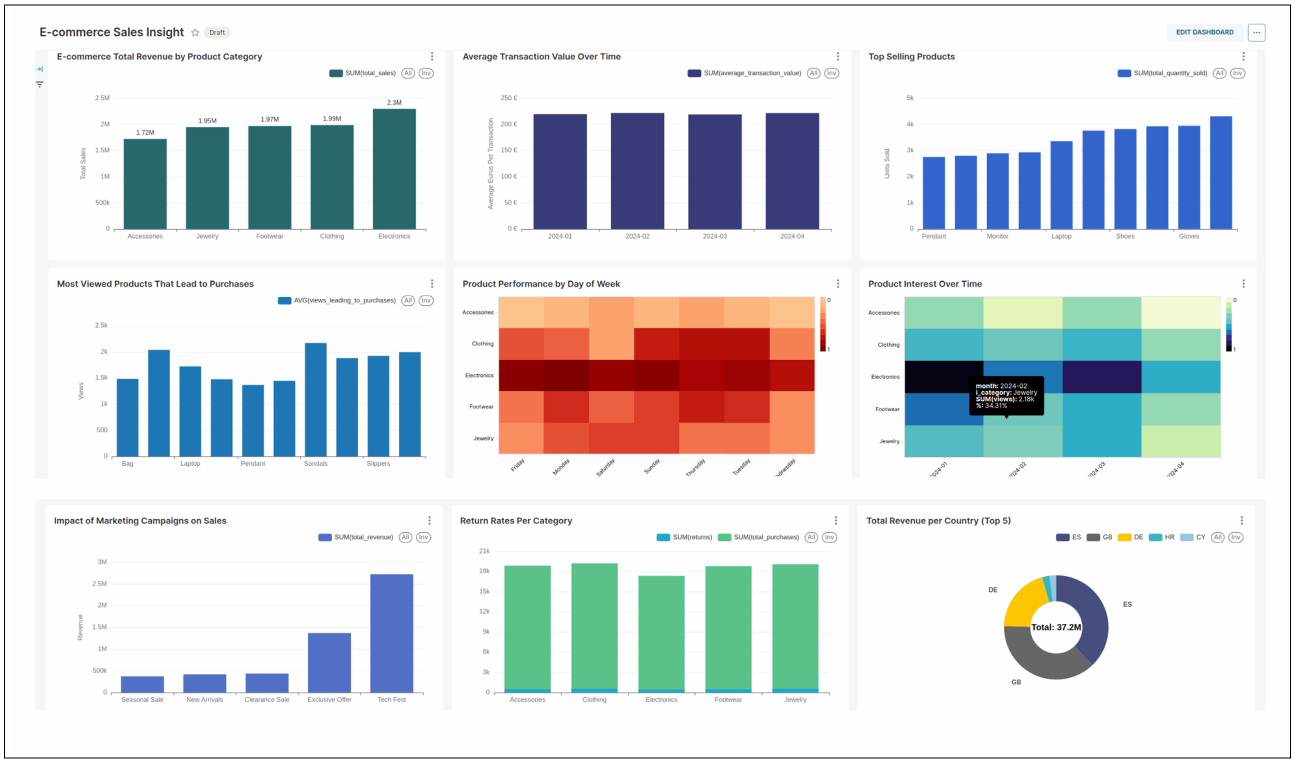

E-commerce Sales Insight
Draft
EDIT DASHBOARD
E-commerce Total Revenue by Product Category
SUM(total_sales)
Accessories
Jewelry
Footwear
Clothing
Electronics
Average Transaction Value Over Time
SUM(average_transaction_value)
Top Selling Products
SUM(total_quantity_sold)
Most Viewed Products That Lead to Purchases
AVG(views_leading_to_purchases)
Product Performance by Day of Week
Product Interest Over Time
Impact of Marketing Campaigns on Sales
SUM(total_revenue)
Seasonal Sale
New Arrivals
Clearance Sale
Exclusive Offer
Tech Fest
Return Rates Per Category
SUM(returns)
SUM(total_purchases)
Total Revenue per Country (Top 5)
ES
GB
DE
HR
CY
Total: 37.2M

Figure 9-14 Building additional charts

9. Once we are happy with our dashboard setup and configuration, we can move it from draft to published. See Figure 9-15.

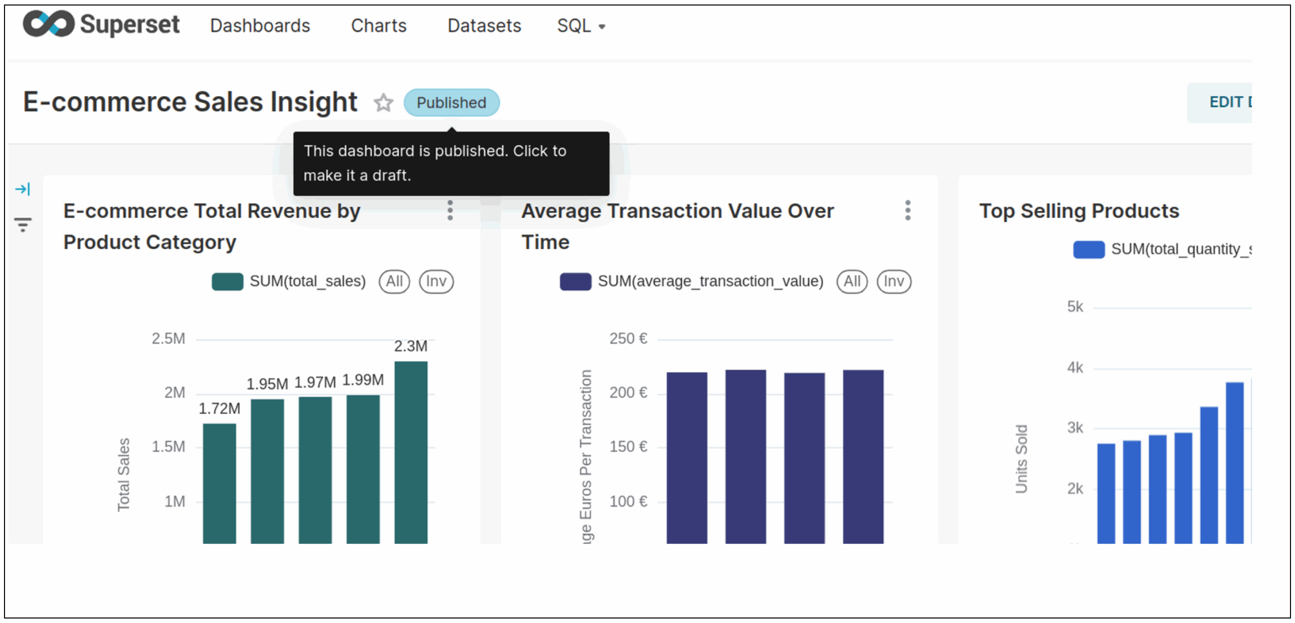


Figure 9-15 Move the chart from draft to published

Apache Superset implements a robust *role-based access control (RBAC)* model. This authorization system offers granular control, allowing you to define permissions for individual users and roles. It even extends to row-level access for the datasets you configure, providing maximum flexibility in data security.

Superset's RBAC system allows you to define permissions for dashboards and datasets, ensuring users only see the data relevant to their roles:

- **Owner access**: Assign "owner" permissions to users responsible for maintaining and updating dashboards.
- **View access**: Grant "view" access to specific teams requiring insights for their business unit.

This granular control lets you share data securely. For example, the marketing team can view customer behavior data, while the finance team sees sales data relevant to their department.

For a deeper dive into what you can achieve regarding authorization with Apache Superset, explore 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. In this section we will give you some examples.

9.3.1 IBM watsonx.data Single Sign-On (SSO) authentication setup

You can bring your own SSO to the watsonx.data platform. IBM watsonx.data platform supports LDAP, SAML, and OIDC providers to be integrated with your instance.

1. We will integrate our Keycloak OIDC provider to the watsonx.data instance. To do this, navigate to the **Identity providers** option from the left-side menu. See Figure 9-16.

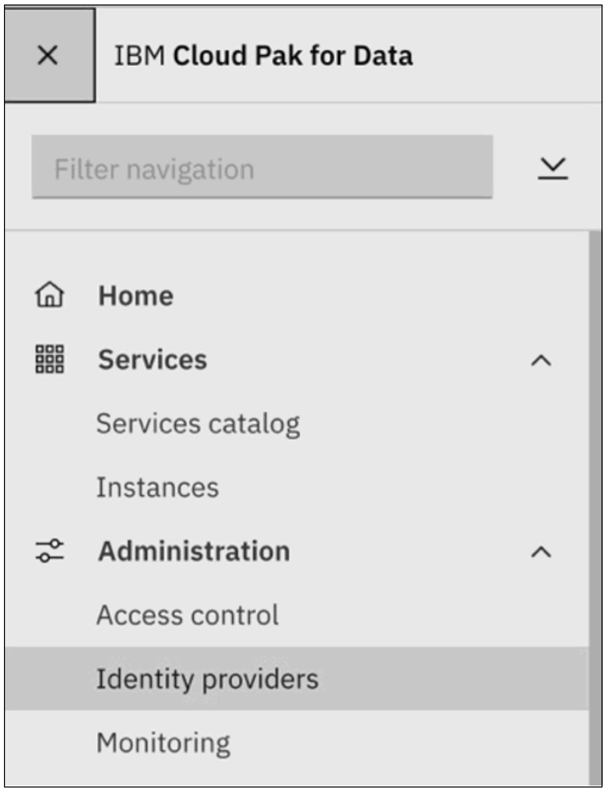
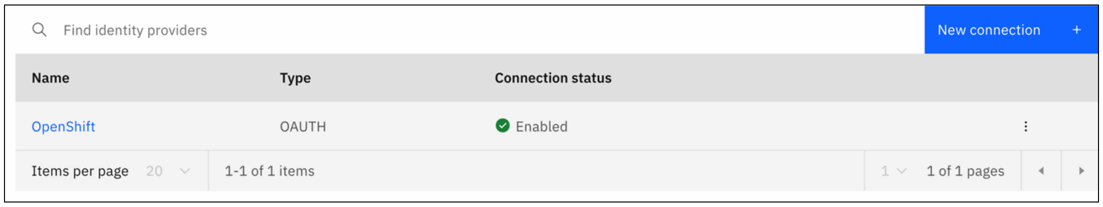


Figure 9-16 Identity providers option

2. Click **New connection** button on the interface. See Figure 9-17 on page 170.

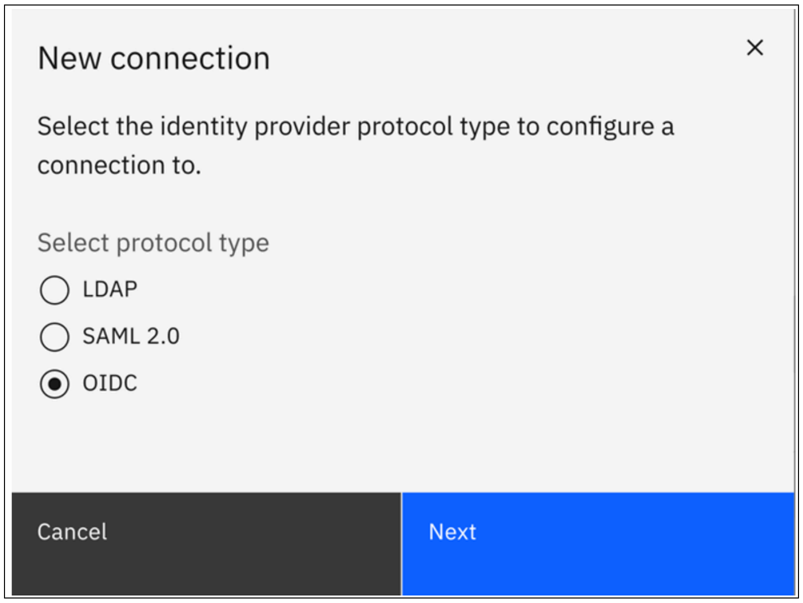


Name	Type	Connection status
OpenShift	OAUTH	Enabled

Items per page 20 1-1 of 1 items 1 1 of 1 pages

Figure 9-17 New connection

3. You will be prompted to choose a protocol. We will use **OpenID Connect (OIDC)** for this example. See Figure 9-18 on page 171.


New connection

Select the identity provider protocol type to configure a connection to.

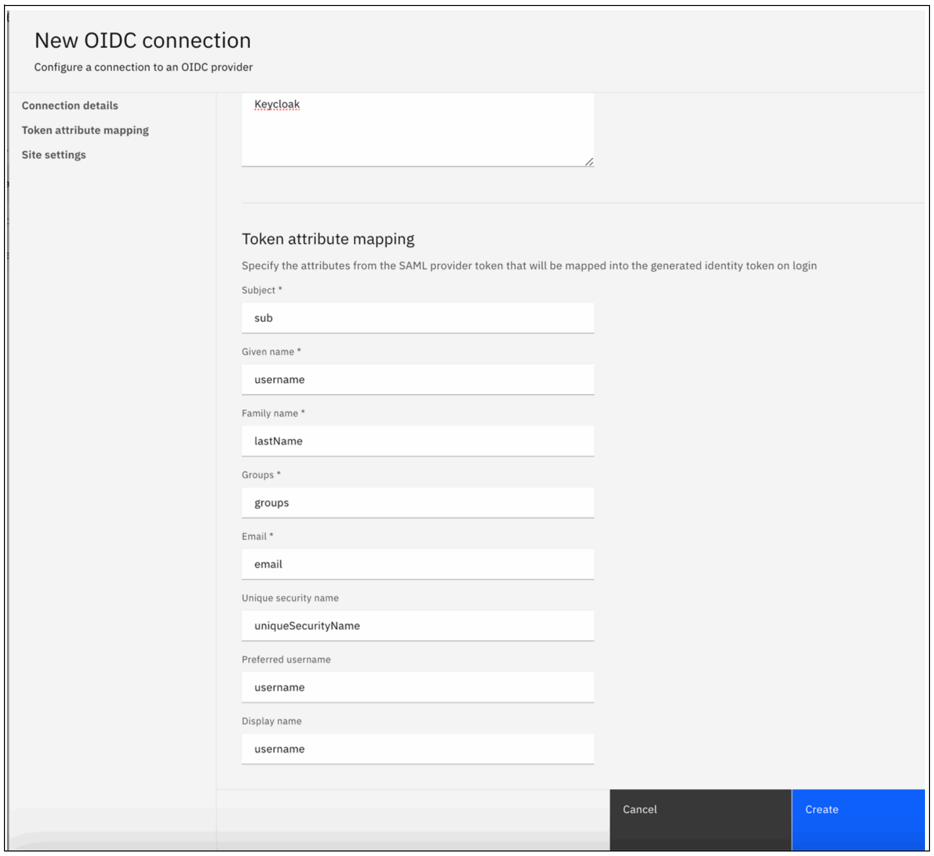
Select protocol type

- ○ LDAP
- ○ SAML 2.0
- ● OIDC

Figure 9-18 OIDC option

4. To proceed, provide the following Keycloak credentials (See Figure 9-19):

- ► The token details
- ► The well-known configuration URL


New OIDC connection

Configure a connection to an OIDC provider

Keycloak

Token attribute mapping

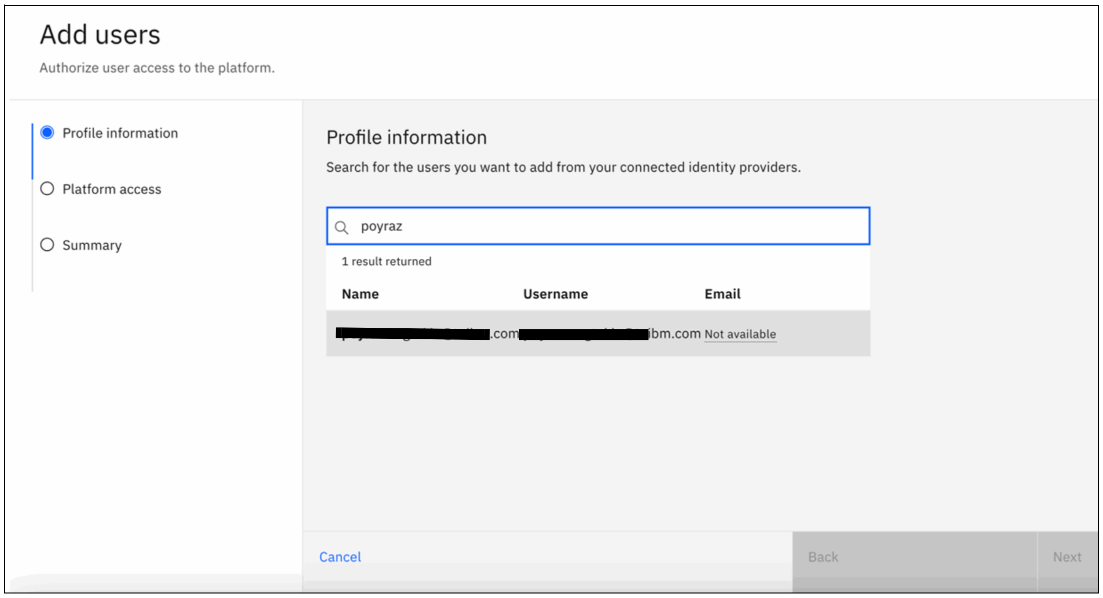
Specify the attributes from the SAML provider token that will be mapped into the generated identity token on login

Subject *	sub
Given name *	username
Family name *	lastName
Groups *	groups
Email *	email
Unique security name	uniqueSecurityName
Preferred username	username
Display name	username

Figure 9-19 Enter the details

5. Click **Create** to save your Keycloak configuration. The connection with your SSO provider will be verified, and upon success, the settings will be saved. For more details on Identity Provider configurations in watsonx.data, refer to IBM Documentation.

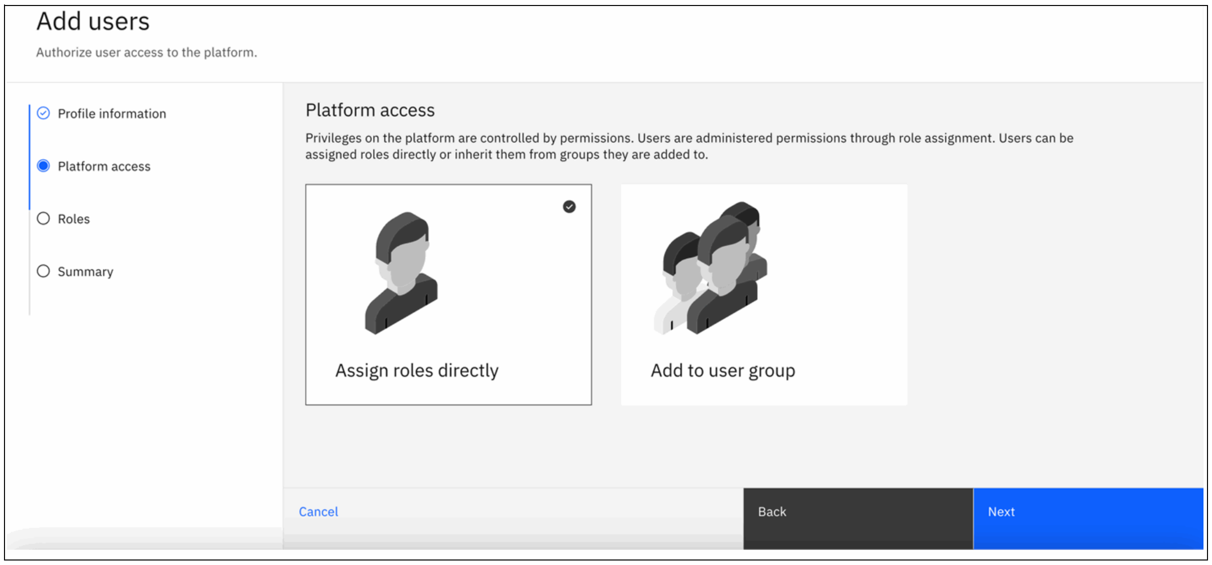
6. Add users from your SSO provider directly within watsonx.data. Navigate to the **Access control** section from the left-side menu, click **Add User**, and search for users using your SSO credentials. See Figure 9-20 on page 172.


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.
poyraz
1 result returned

Name	Username	Email
[redacted].com	[redacted]ibm.com	Not available

Figure 9-20   Add users

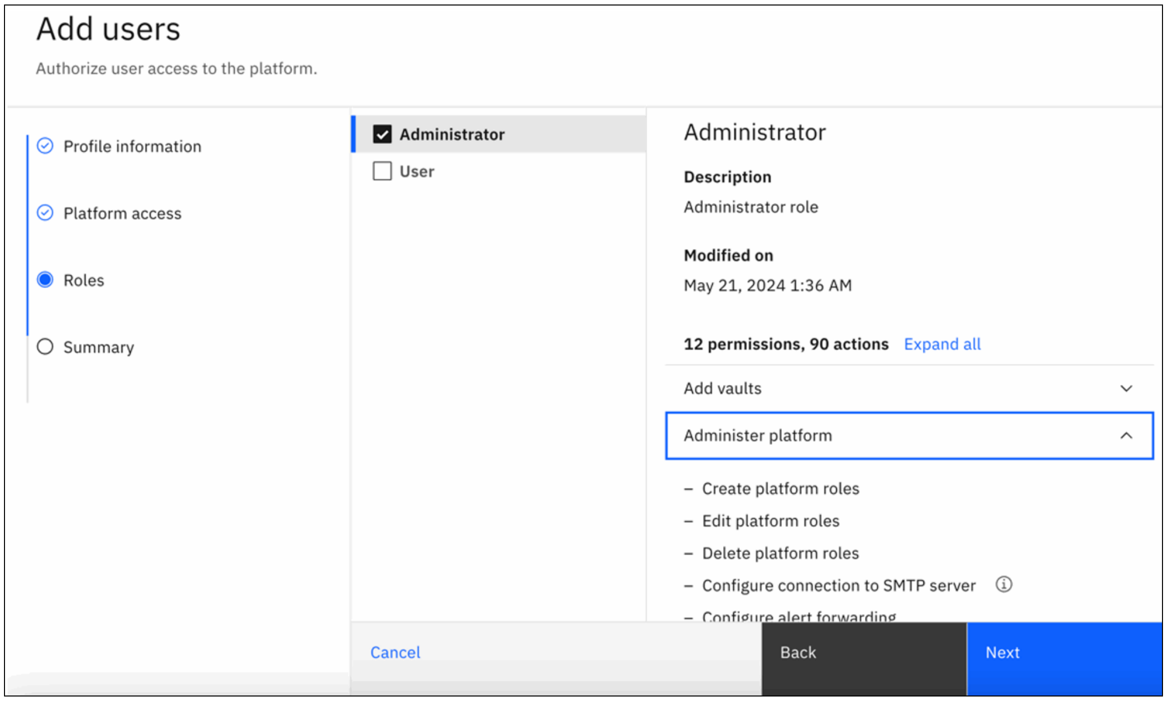
7. After selecting the user, click **Next** to set the platform access type for the user. See Figure 9-21 on page 172.



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.

Figure 9-21   Platform access

8. There are two ways to grant users access in watsonx.data: direct assignment or group membership. We will focus on directly assigning roles to the specified user in this example. See Figure 9-22 on page 173.


Add users

Authorize user access to the platform.

☑ Administrator

☐ User

Administrator

Description

Administrator role

Modified on

May 21, 2024 1:36 AM

12 permissions, 90 actions Expand all

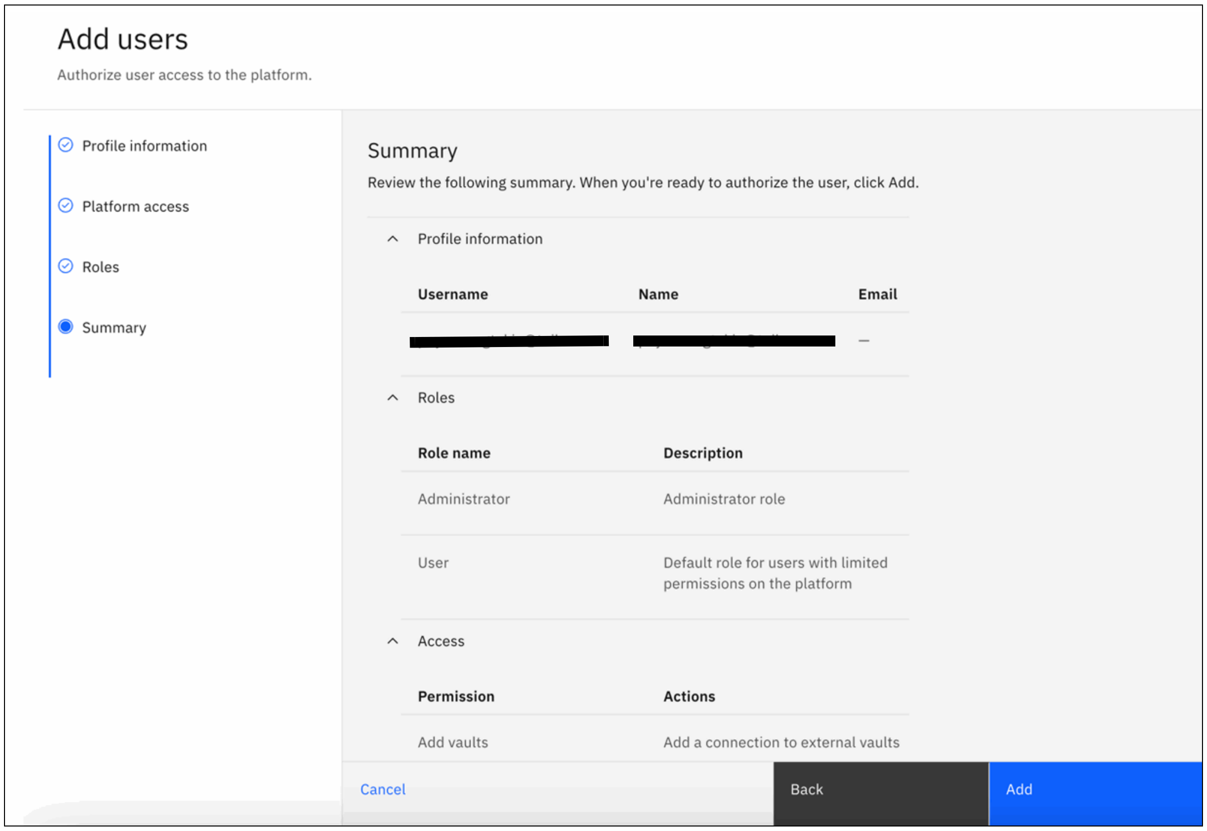
Add vaults

Administer platform

- Create platform roles
- Edit platform roles
- Delete platform roles
- Configure connection to SMTP server
- Configure alert forwarding

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 have assigned to your user and then click **Add** to add the user to your watsonx.data instance. See Figure 9-23 on page 174.


Add users

Authorize user access to the platform.

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

Figure 9-23 Summary

Added user can now benefit from secure SSO for logging into the watsonx.data platform user interface.

9.3.2 Consuming the curated data from watsonx.data

IBM watsonx.data allows you to run SQL queries with the Presto engine using the graphical user interface. In this way, you can run your SQL queries on the go without a need for any additional setup.

1. To run the queries from the GUI, navigate to the **Query workspace** from the left-side navigation menu. You will see the query screen with your data catalogs shown on the left and the SQL query terminal on the right along with the results section at the bottom of the screen. See Figure 9-24 on page 175.

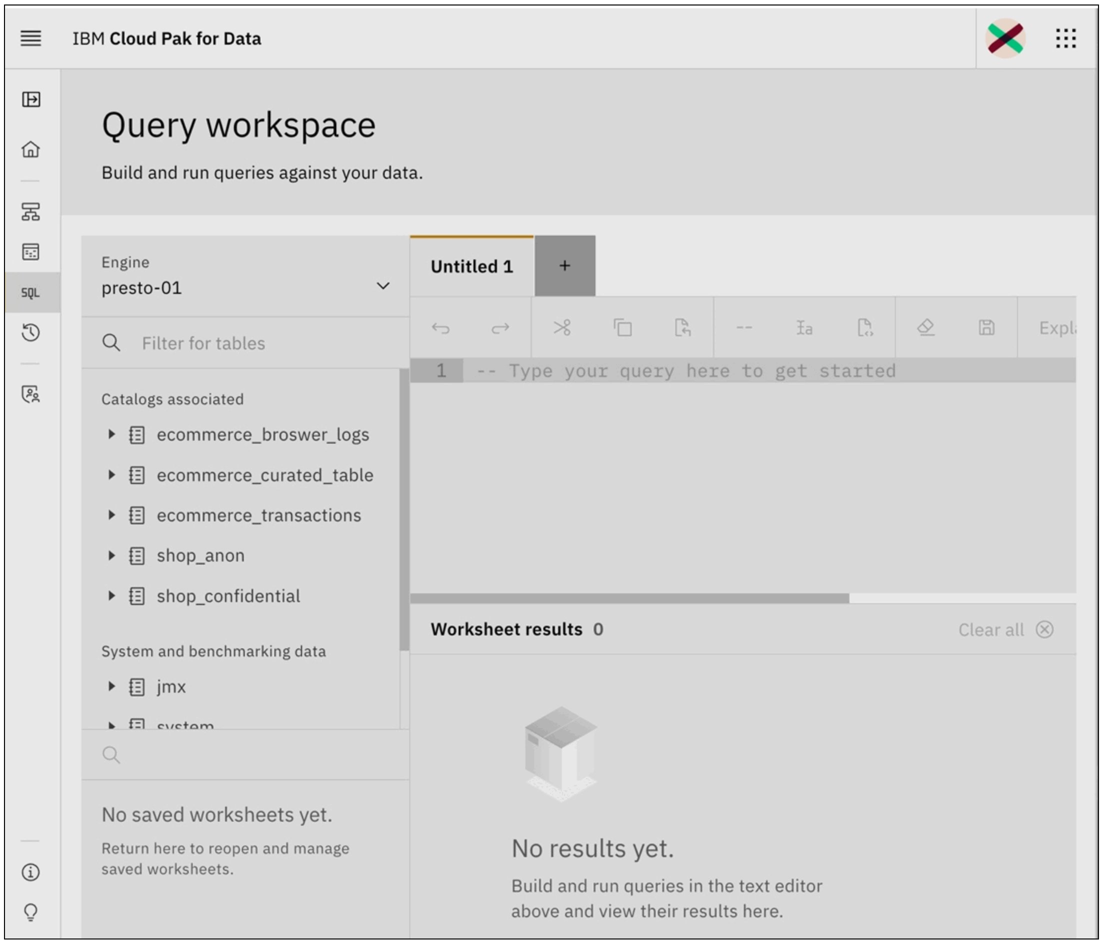
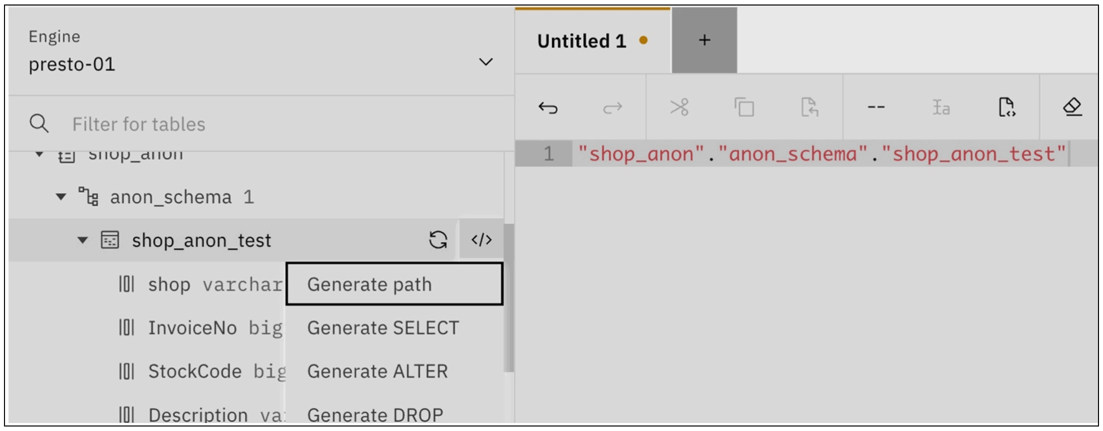


Figure 9-24 Query workspace

2. In watsonx.data's SQL Query workspace, imagine you want to analyze detailed invoice information and total quantity sold per product for each customer. To target a specific table, you will need its path. Right-click the desired table and select **Generate path**. This path will automatically populate in the SQL query section.

   Tip: You can also retrieve paths for individual columns within tables.

   See Figure 9-25 on page 176.



```
"shop_anon"."anon_schema"."shop_anon_test"
```

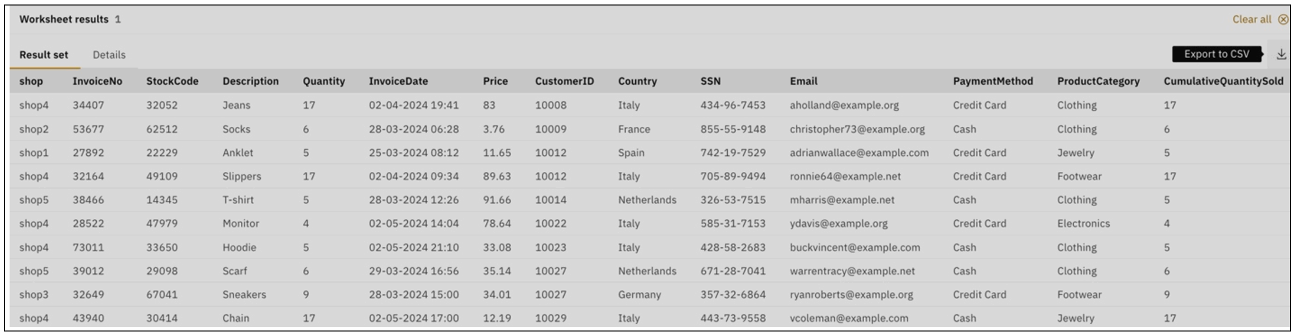
Figure 9-25 Generate path option

3. Since we have our path now, we can run the following SQL query to retrieve the data. See Example 9-6.

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;
```

4. Once the query finishes, the results will be displayed. You can then export them as a CSV file by clicking the **download** button on the right. See Figure 9-26 on page 177.



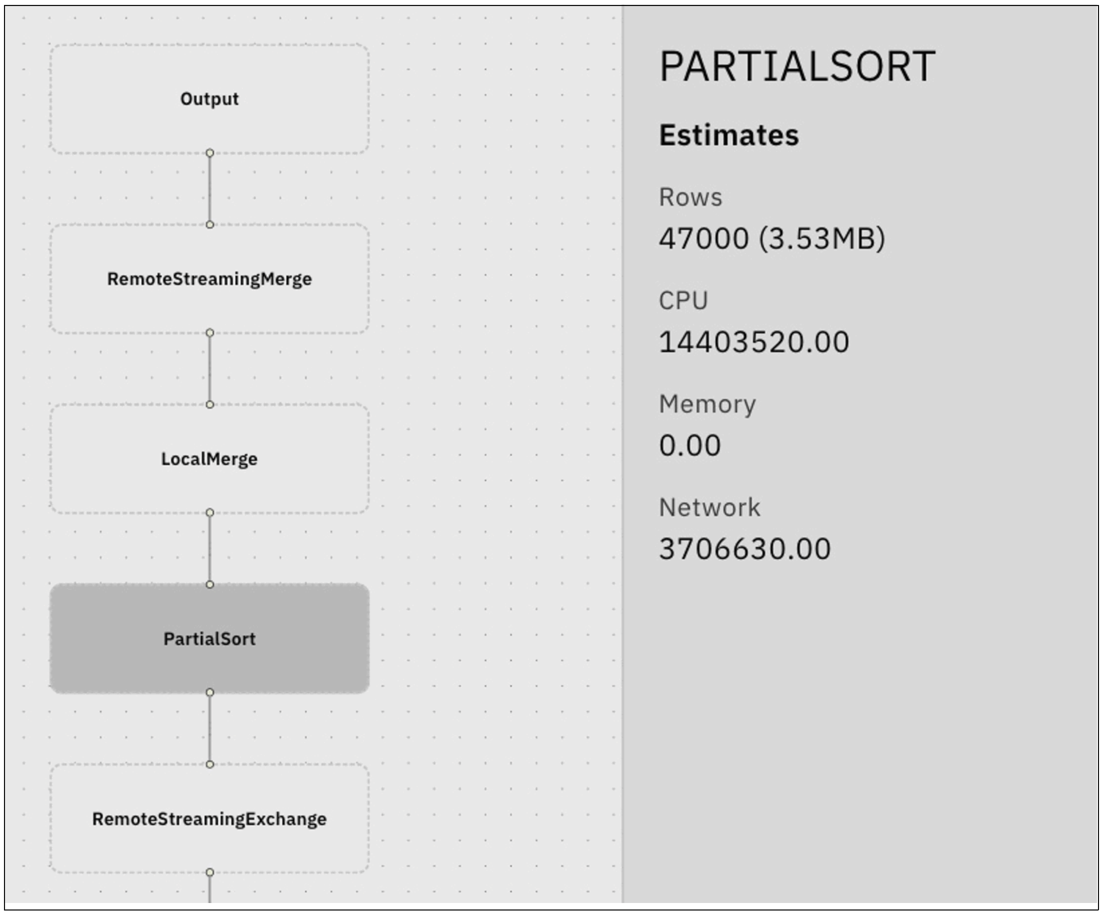
shop	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	Price	CustomerID	Country	SSN	Email	PaymentMethod	ProductCategory	CumulativeQuantitySold
shop4	34407	32052	Jeans	17	02-04-2024 19:41	83	10008	Italy	434-96-7453	aholland@example.org	Credit Card	Clothing	17
shop2	53677	62512	Socks	6	28-03-2024 06:28	3.76	10009	France	855-55-9148	christopher73@example.org	Cash	Clothing	6
shop1	27892	22229	Anklet	5	25-03-2024 08:12	11.65	10012	Spain	742-19-7529	adrianwallace@example.com	Credit Card	Jewelry	5
shop4	32164	49109	Slippers	17	02-04-2024 09:34	89.63	10012	Italy	705-89-9494	ronnie64@example.net	Credit Card	Footwear	17
shop5	38466	14345	T-shirt	5	28-03-2024 12:26	91.66	10014	Netherlands	326-53-7515	mharris@example.net	Cash	Clothing	5
shop4	28522	47979	Monitor	4	02-05-2024 14:04	78.64	10022	Italy	585-31-7153	ydavis@example.org	Credit Card	Electronics	4
shop4	73011	33650	Hoodie	5	02-05-2024 21:10	33.08	10023	Italy	428-58-2683	buckvincent@example.com	Cash	Clothing	5
shop5	39012	29098	Scarf	6	29-03-2024 16:56	35.14	10027	Netherlands	671-28-7041	warrentracy@example.net	Cash	Clothing	6
shop3	32649	67041	Sneakers	9	28-03-2024 15:00	34.01	10027	Germany	357-32-6864	ryanroberts@example.org	Credit Card	Footwear	9
shop4	43940	30414	Chain	17	02-05-2024 17:00	12.19	10029	Italy	443-73-9558	vcoleman@example.com	Cash	Jewelry	17

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 on the **Explain** button located on the left-side of the **Run on Presto** button.

With the Explain option, you can see the steps of your SQL query along with the resources that would be required for each step. See Figure 9-27.



PARTIALSORT

Estimates

Rows
47000 (3.53MB)

CPU
14403520.00

Memory
0.00

Network
3706630.00

Figure 9-27 Visual Explain

For example, you can preview the estimated costs associated with the PartialSort operation when you execute 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 using Presto CLI. To do this, you need to have the Presto CLI installed on your environment.

To run the commands using the Presto CLI, you first need to connect to your Presto engine. An example command is shown in Example 9-7.

Example 9-7 Connect 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:
```

Once you log in to Presto using the CLI and provide your credentials, you can directly execute SQL queries. For instance, in Example 9-8, we will 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 using the IBM Storage Ceph GUI

This appendix details configuring the RADOS Gateway service within an IBM Storage Ceph cluster using the graphical user interface.

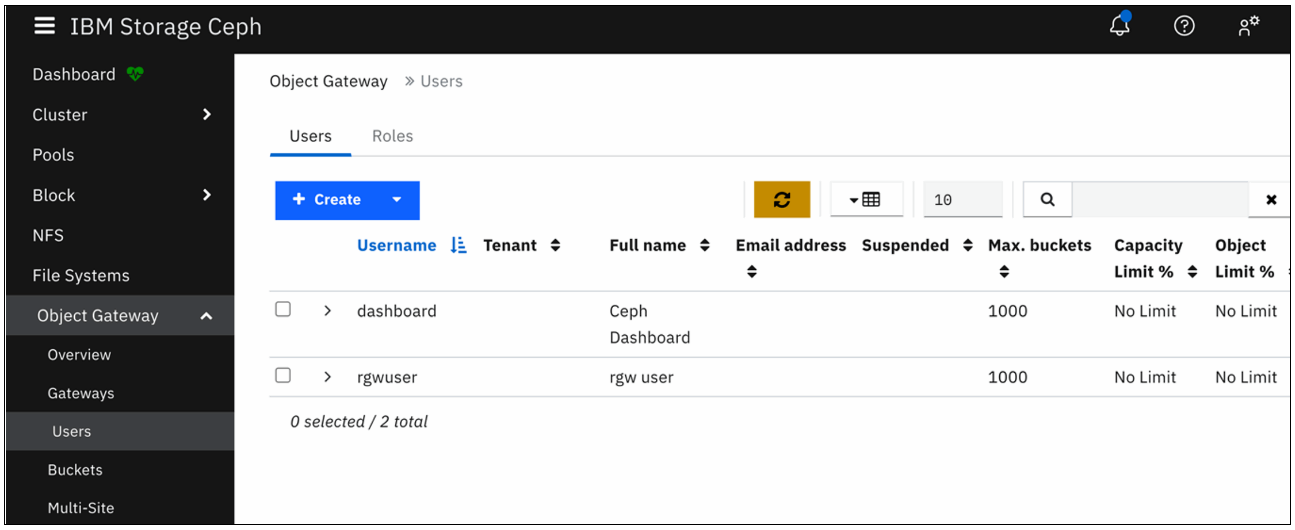
Configuring RADOS Gateway using IBM Storage Ceph Graphical user interface

The IBM Storage Ceph GUI provides a user-friendly interface for creating and configuring RADOS Gateway services. To begin, locate the **Object Gateway** option within the left navigation menu

RADOS Gateway user creation

Perform the following steps to create a RADOS Gateway user.

1. The **Users** section of the Object Gateway interface allows you to view existing users, create new ones, and modify their access permissions See Figure A-1.



	Username	Tenant	Full name	Email address	Suspended	Max. buckets	Capacity Limit %	Object Limit %
☐	dashboard		Ceph Dashboard			1000	No Limit	No Limit
☐	rgwuser		rgw user			1000	No Limit	No Limit

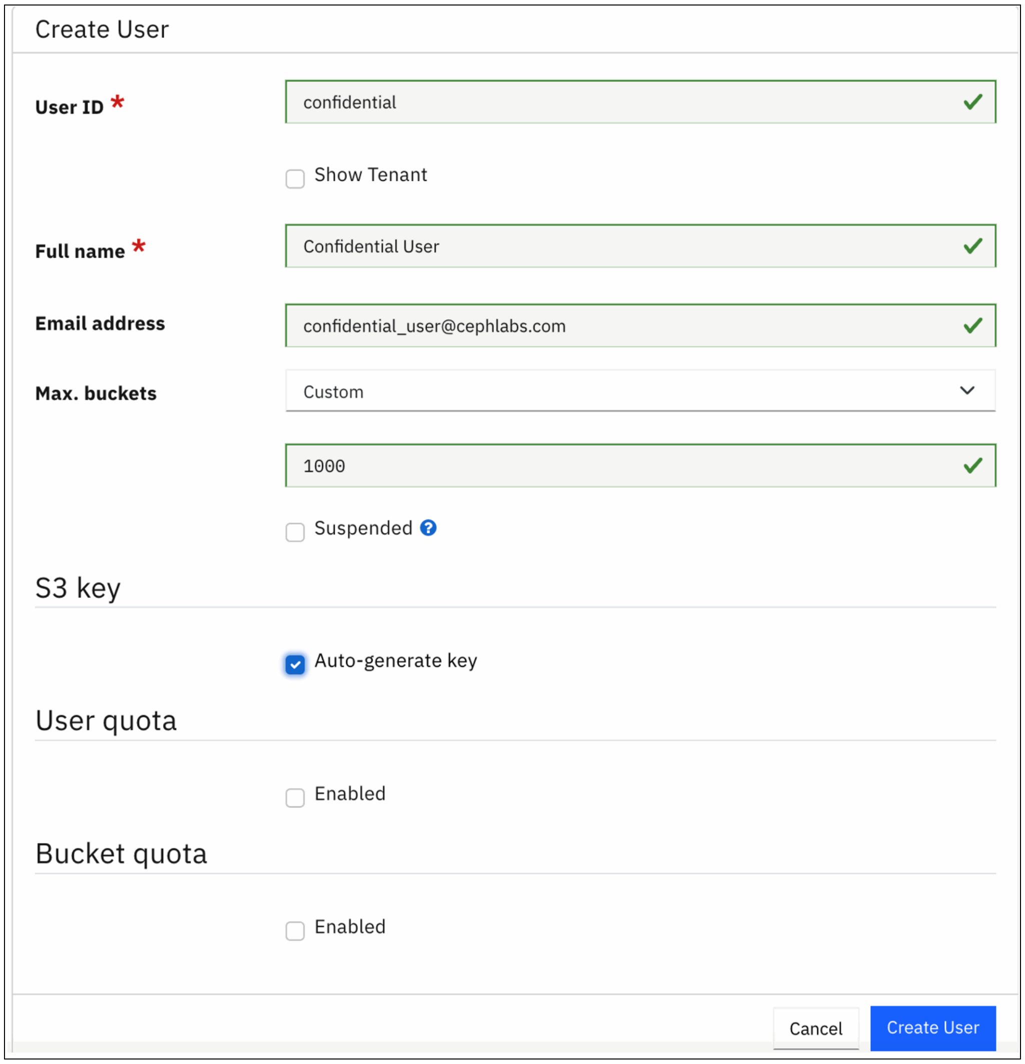
0 selected / 2 total

Figure A-1 Users option on the Object Gateway interface

2. To create an RGW user, click **Create** button and fill in the required information. This form allows you to create a new user with two options:

- Auto-generate secure S3 credentials for the user.
- Set storage quotas to limit their data usage.

  See Figure A-2 on page 183.


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

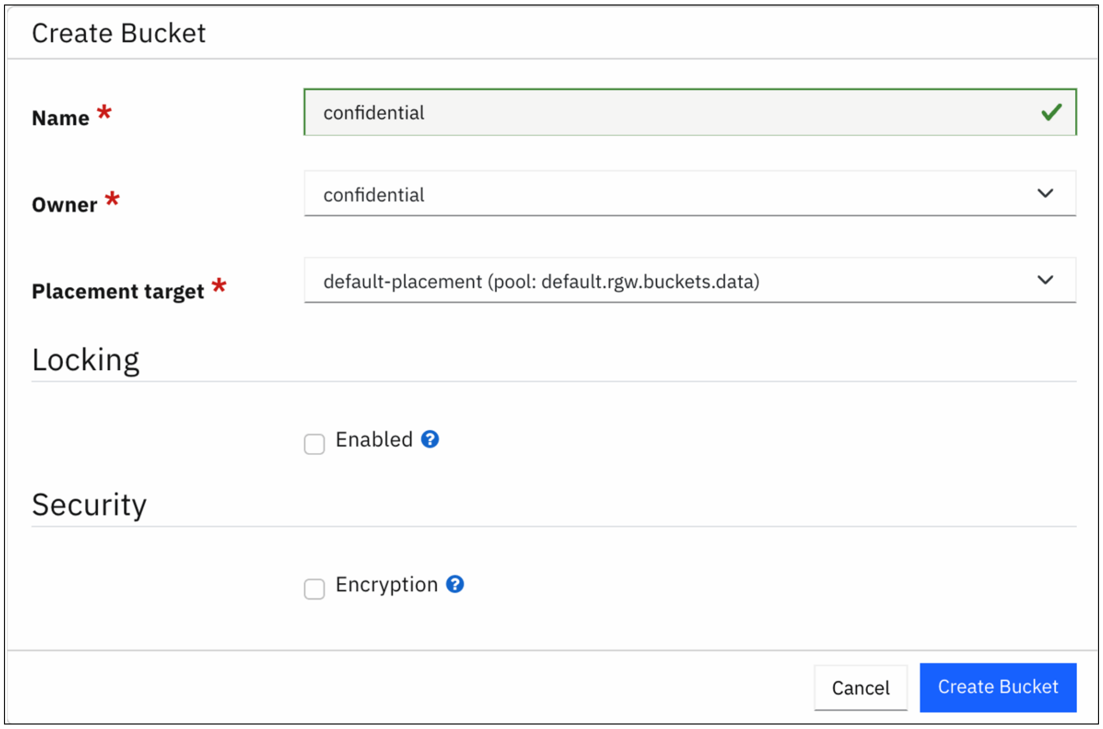
Bucket quota

[ ] Enabled

Figure A-2 Create an RGW user

Bucket creation

You can create buckets and assign owners to these buckets using the IBM Storage Ceph GUI. Navigate to **Buckets** from the left-side menu and click **Create** to create a bucket. You can lock and encrypt your bucket using this form. An example for creating the confidential bucket using the GUI is shown in Figure A-3 on page 184.


Create Bucket

Name * confidential

Owner * confidential

Placement target * default-placement (pool: default.rgw.buckets.data)

Locking

☐ Enabled

Security

☐ Encryption

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.

Configuring command line tools for watsonx.data

To interact with watsonx.data from the command line, you will need the `ibm-lh-client` utility. However, before installing it, ensure your environment meets the prerequisite package requirements. The `ibm-lh-client utility` can be installed on the following platforms. See Table B-1.

Table B-1 ibm-lh-client utility

Operating system	**x86-64**	**Docker / Podman / Colima installation instructions**
Linux	Yes	Docker, Podman
Windows	Yes	Docker, Podman
Mac OS x86	Yes	Docker, Podman
Mac with Apple Silicon with Rosetta Emulation	No	Docker, Colima

Depending on your platform, you will find installation instructions for the prerequisite packages by clicking these links.

Installation of the ibm-lh-client utility

Perform the following steps to install the ibm-lh-client utility:

1. Create an installation directory and navigate to it. See Example B-1 on page 186.

Example B-1 Create an installation directory and navigate to it

```
mkdir Documents/wxd
cd Documents/wxd
```

2. Set up the environment variables, as shown in Example B-2.

Example B-2 Set 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
```

Table B-3 shows the internal tag (represented by `$LH_RELEASE_TAG`) 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	v1.1.4 (latest)
4.8.4	v1.1.3
4.8.3	v1.1.2
4.8.1	v1.1.1
4.8.0	v1.1.0

3. If you have Docker installed on your environment, run the following command to set an environment variable named DOCKER_EXE with the value `docker`.

   ```
   export DOCKER_EXE=docker
   ```

4. You then need to pull the watsonx.data client package and then copy them into your environment. See Example B-4.

Example B-4 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 `/tmp` directory and compare the checksum value with the `bom.txt` file. See Example B-5.

Example B-5 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. See Example B-6.

Example B-6 Authenticate 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, navigate to your installation folder and start with adding username and password for authentication. See Example B-7.

Example B-7 Adding 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 using the following command:

```
./cert-mgmt --op=add --host=<hostname> --port=<port>
```

To add your CloudPak for Data instance connection to your ibm-lh utility, you can 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 the command above.

You can connect to your presto instance scheme with the following command:

```
./presto-cli --server <your instance URL>  --user cpadmin --schema
confidential_schema --catalog shop_confidential
```

Creating schema

Perform the following steps:

1. To create a schema on your watsonx.data instance, you need to first log in with presto-cli. See Example B-8 on page 188.

Example B-8 Log in with 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 will be asked to provide the `cpadmin` user's password and then you will get the presto prompt:

```
presto>
```

3. You can now create your schema using the Presto CLI. We will create the anonymized bucket's schema with the CLI. See Example B-9.

Example B-9 Create your schema 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 the schema creation by asking Presto to show the created schemas. See Example B-10.

Example B-10 Show 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 table and ingesting data

To create a table inside your schema using the presto cli, you need to manually define the column names instead of importing a table from the GUI.

An example command to create a table is shown in Example B-11:

Example B-11 Create 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);
```

You need to define your S3 credentials as environment variables before ingesting the data into your Presto engine. You can use the following command for defining these variables. See Example B-12 on page 189.

Example B-12 Define 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"
```

Abbreviations and acronyms

ABAC	Attribute-Based Access Control
AD	Active Directory
API	application programming interface
ASF	Apache Software Foundation
AWS	Amazon Web Services
CA	certificate authority
CD-ROM	compact-disc read-only memory
CFTC	Commodity Futures Trading Commission
CIDR	Classless Inter-Domain Routing
CLI	command-line interface
COS	Cloud Object Storage
CP4D	Cloud Pak for Data
DR	Disaster Recovery
DRAM	Dynamic Random Access Memory
EBS	Elastic Block Store
EC	erasure coded
EMR	Elastic MapReduce
EOL	end of life
ETL	Extract, Transform, and Load
FINRA	Financial Industry Regulatory Authority
FIPS	Federal Information Processing Standards
GA	general availability
GCP	Google Cloud Platform
GDPR	General Data Protection Regulation
GKLM	IBM Guardium® Key Lifecycle Manager
GUI	graphical user interface
HDD	hard disk drive
HDFS	Hadoop Distributed File System
HIPAA	Health Insurance Portability and Accountability Act
HSM	Hierarchical Storage Management
HTTP	Hypertext Transfer Protocol
I/O	input/output
IAM	Identity and Access Management
ID	identifier
IDP	Identity Provider (IdP)
IP	Internet Protocol
ISV	independent software vendor
ITIL	Information Technology Infrastructure Library
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
JWT	JSON Web Token
KMIP	Key Management Interoperability Protocol
KMS	Key Management Service
LC	lifecycle
LDAP	Lightweight Directory Access Protocol
MFA	Multi-Factor Authentication
NFS	Network File System
NL-SAS	Near Line SAS
NOSQL	Not Only SQL or Non-Relational SQL
NVMe	Non-Volatile Memory Express
OCP	OpenShift Container Platform
OIDC	OpenID Connect
ORC	Optimized Row Columnar
OS	operating system
OSD	Object Storage Daemon
PCI-DSS	Payment Card Industry Data Security Standard
PII	Personally Identifiable Information
PIN	Personal Identification Number
POSIX	Portable Operating System Interface
QAT	Intel QuickAssist Technology
RADOS	Reliable Autonomic Distributed Object Store
RADOSGW	RADOS Gateway
RAG	Retrieval-Augmented Generation
RAID	Redundant Array of Independent Disks
RAM	random access memory
RBAC	Role-Based Access Control
RBD	RADOS Block Device
RDBMS	relational database management system
RGW	RADOS Gateway
RHEL	Red Hat Enterprise Linux

RHSSO	Red Hat Single Sign-On
RPO	Recovery Point Objective
S3A	Amazon S3A File System
S3N	Amazon S3 Native File System.
SAML	Security Assertion Markup Language
SAS	Serial Attached SCSI
SATA	Serial ATA (Advanced Technology Attachment)
SDK	software development kit
SEC	U.S. Securities and Exchange Commission
SKLM	Security Key Lifecycle Manager
SNS	Simple Notification Service
SQL	Structured Query Language
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
SSL	Secure Sockets Layer
SSO	single sign-on
STS	Security Token Service
TB	terabyte
TLS	Transport Layer Security
TOTP	Time-based One-Time Password
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Universal Time Coordinated
WXD	watsonx.data
WORM	Write Once Read Many

C

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-I
BM-watsonx.data-and-Beyond.git
```

Related publications

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

IBM Redbooks

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

- *IBM Storage 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, draft 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

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine.fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

Redbooks

Unlocking Data Insights and AI: IBM Storage Ceph as a Data

SG24-8563-00

ISBN DocISBN

IBM

(1.5" spine)
1.5"<-> 1.998"
789 <->1051 pages

Redbooks

Unlocking Data Insights and AI: IBM Storage Ceph as a Data

SG24-8563-00

ISBN DocISBN

IBM

(1.0" spine)
0.875"<->1.498"
460 <-> 788 pages

Redbooks

IBM Storage Ceph as a Data Lakehouse Platform for IBM

SG24-8563-00

ISBN DocISBN

IBM

(0.5" spine)
0.475"<->0.873"
250 <-> 459 pages

Redbooks

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

(0.2"spine)
0.17"<->0.473"
90<->249 pages

(0.1"spine)
0.1"<->0.169"
53<->89 pages

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine.fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

Redbooks

Unlocking Data Insights and AI: IBM Storage Ceph

SG24-8563-00

ISBN DocISBN

IBM

(2.5" spine)
2.5"<->nnn.n"
1315<-> nnnn pages

Redbooks

Unlocking Data Insights and AI: IBM Storage Ceph as a Data Lakehouse Platform for IBM

SG24-8563-00

ISBN DocISBN

IBM

(2.0" spine)
2.0" <-> 2.498"
1052 <-> 1314 pages

SG24-8563-00

ISBN DocISBN

Printed in U.S.A.