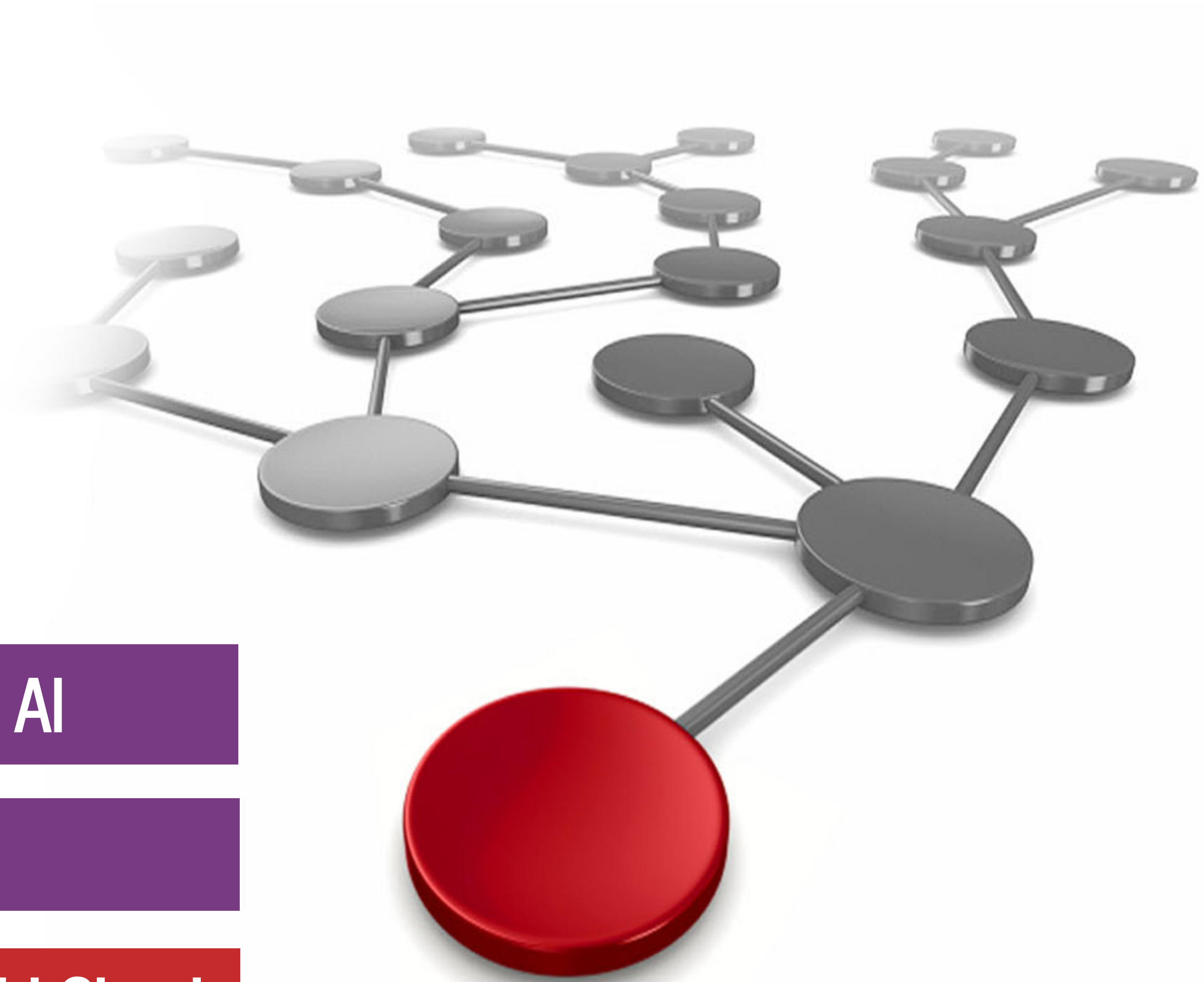
Draft Document for Review August 28, 2024 10:07 pm REDP-5743-00

Redbooks
ibm.com/redbooks

Accelerating AI and Analytics with IBM watsonx.data and IBM Storage Scale

Kedar Karmarkar
Chinmaya Mishra
Qais Noorshams
Gero Schmidt
Anna Greim
Dietmar Fischer



Data and AI

Storage

Hybrid Cloud

IBM

Redpaper

IBM Redbooks

Accelerating AI and Analytics with IBM watsonx.data and IBM Storage Scale

August 2024

REDP-5743-00

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

First Edition (August 2024)

This edition applies to Version 5, Release 2, Modification 1 of IBM Storage Scale.

This document was created or updated on August 28, 2024.

Contents

Notices

Trademarks

Preface

This IBM® Redpaper describes an IBM Data & AI solution for using IBM watsonx.data together with IBM Storage Scale. The paper showcases how IBM watsonx.data applications can benefit from the enterprise storage features and functions offered by IBM Storage Scale.

IBM watsonx.data empowers enterprises to scale their analytics and AI capabilities, leveraging an open lakehouse architecture. With its next generation query engines, a robust governance and open data frameworks, IBM watsonx.data facilitates seamless data access and sharing of data and metadata. With IBM watsonx.data, enterprises can swiftly connect to data wherever it resides, extract actionable insights, and optimize data warehouse or data lake expenses.

IBM Storage Scale is software-defined, high performance, scalable file and object storage that enables organizations to build a Global Data Platform for artificial intelligence (AI), high-performance computing (HPC), advanced analytics, and other demanding workloads.

IBM watsonx.data and IBM Storage Scale can be a powerful combination for building a scalable and cost-effective data lakehouse solution. This IBM Redbooks® publication delves into how IBM Storage Scale's robust storage capabilities and IBM watsonx.data's advanced analytics features come together to build a powerful data and AI platform. This platform empowers you to unlock valuable insights from your data and make data-driven decisions. This further helps organizations to expand from AI pilot projects to full-scale production systems by providing the right tools, platforms and software-defined storage on which to run it all.

This Redpaper is targeted toward technical professionals (customers, consultants, technical support staff, IT Architects, and IT specialists) who are responsible for delivering data lakehouse solutions optimized for data, analytics, and AI. This Redpaper is relevant for:

- technical professionals working on design and implementation of IBM watsonx.data solutions
- existing IBM Storage Scale customers looking to implement IBM watsonx.data solutions

Authors

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

Kedar Karmarkar is a Development Architect with the IBM Storage Scale development team and has contributed to Data Caching, Scale containerization, AI solutions and Storage Scale Development adoption teams. Kedar has over 25 years of infrastructure software, storage development experience in management, and architect roles. Prior to IBM, Kedar has led development of network-attached storage (NAS), Block level virtualization, replication, systems, and storage management products. Kedar has a Bachelor of Engineering (Computer Science) degree from University of Pune, India.

Chinmaya Mishra is a Software Architect in the Big Data & Analytics team within the IBM Storage Scale organization in IBM ISDL Labs. He joined IBM India in 2001 and has hold various technical and leadership roles in software products and solutions development teams across IBM India and IBM US. His area of expertise includes Transaction processing, Operating systems, Cloud Native solutions as a service, Clustered Filesystems and Analytics & AI solutions. He holds a Bachelor of Technology degree in Electrical Engineering from the Indian Institute of Technology, Kharagpur.

Qais Noorshams is an IBM Software Engineer in the Big Data & Analytics team within the IBM Storage Scale organization. Since he joined IBM Germany in 2015, he has been holding various technical and leadership positions in international software development projects. He is a certified Expert Developer, IBM Recognized Speaker, and IBM Recognized Teacher. His track record includes authoring more than 15 granted patents, more than 15 peer-reviewed publications, and various IBM-published newsletter and blog articles. He holds a PhD degree in Computer Science (Dr.-Ing.) from Karlsruhe Institute of Technology (Germany).

Gero Schmidt is a Software Engineer at IBM Germany R&D GmbH in the Big Data & Analytics team of the IBM Storage Scale development organization. He joined IBM Germany in 2001 as presales technical support engineer for enterprise storage solutions. He co-authored multiple IBM Redbooks® and has been a frequent speaker at IBM international conferences. In 2015 he joined the storage research group at the IBM Almaden Research Center in California, USA, where he worked on IBM Storage Scale, compression of genomic data in next generation sequencing pipelines and the development of a cloud-native backup solution for containerized applications in Kubernetes/Red Hat OpenShift. He holds a degree in Physics (Dipl.-Phys.) from the Braunschweig University of Technology in Germany.

Anna Greim is the scrum master of the IBM Storage Scale's Big Data and Analytics team. Anna has been with IBM for more than 8 years working as a software developer and enjoys working with a talented team and as part of that team developing great products and solutions.

Dietmar Fischer is the Manager of the IBM Storage Scale's Big Data and Analytics team. Dietmar has been with IBM for more than 25 years and has held several positions within the IBM Storage development organization including software test, development, project management and management. Dietmar has a strong technical computer science background and enjoys developing great solutions with a team of very talented experts.

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

Gopikrishnan Varadarajulu, Rohan Pednekar, Kevin Shen, Ted Hoover, Khanh Ngo, Gregory Kishi, Rene Orozco Martinez, Boda Devi Manikanta, Hariharan Ashokan, T K Narayanan, Sujith PS, Shafeek M, Renu Rajagopal, Prasad Kulkarni, Madhu Thorat, Pravin Ranjan, Rajan Mishra

Larry Coyne
IBM Redbooks, Tucson Center

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

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

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, 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

Overview

This IBM Redpaper describes the IBM solution for using IBM Storage Scale as enterprise storage with IBM watsonx.data. The paper showcases how IBM watsonx.data applications can benefit from the enterprise storage features and functions offered by IBM Storage Scale.

This chapter provides overview of IBM watsonx.data and IBM Storage Scale along with key features and use cases for these products. If you are already familiar with IBM watsonx.data and IBM Storage Scale, you may skip this chapter if appropriate.

1.1 Evolution of Data Analytics

This section describes the evolution of data lakes, the emergence of data lakehouses, and IBM watsonx.data lakehouse.

From Data Warehouse to Data Lake

Data warehouses aggregate data for business intelligence (BI) and Online Analytical Processing (OLAP) purposes. The typical strategy is to build upon a monolithic database, or data warehouse, and then analyze the data through an extract/ transform/load (ETL) processes. Data warehouses are often used for repeatable queries performed over large amount of historical data, such as transaction logs, website traffic, etc.

However, building a data warehouse (DW) comes up with a high up-front cost, and scaling a DW is an expensive affair both in terms of compute and storage. Moreover, DWs only work with structured data.

Moving data warehouses to the cloud does not solve the problem - it comes with vendor lock-in, sometimes with even higher costs, and with limited machine learning/AI use cases.

These limitations lead to the concept of data lakes, offering higher scalability and flexibility. Based on a scale-out architecture created on commodity servers, data lakes can store and process massive volumes of data in its original form - structured or unstructured. Adopters of data lakes looked to Hive, Impala and Spark together with Hadoop Distributed File System (HDFS) storage to simplify data engineering, real-time analytics, predictive analytics and machine learning tasks. Data lakes are also typically less expensive than data warehouses.

From Data lake to Data Lakehouse

As the volume, velocity and variety of data continues to grow, it exposes the monolithic nature of data lakes. Decoupling storage and compute for independent scaling has been an issue, for example with Hadoop.

Ever since, the AWS S3 API has been established as the de-facto standard to process unstructured data as objects. More and more enterprises are integrating S3 as the data access protocol of choice, in their data workflows. However, the processing layers in data lakes (e.g. Hive) are not well equipped to handle S3 based storages, even as S3 based cloud object stores became ubiquitous.

These limitations give rises to an emerging architecture called data lakehouse, that combines the flexibility of a data lake with the performance of a data warehouse. Lakehouse solutions provide a high-performance query engine over low-cost object storage in conjunction with a data governance layer. Data lakehouses are based around open-standard object storage and enable multiple analytics/AI workloads to operate on the same data simultaneously without requiring the data to be duplicated or transformed.

A key benefit of data lakehouses is that they address the needs both of traditional data warehouse analysts who curate and publish data for business intelligence and reporting purposes as well as those of data scientists and engineers who run more complex data analysis and processing workloads.

Data Lakehouse and AI

Data lakes have support for AI/ML since many years. For example, the support for machine learning using Spark ML library exists in Hadoop data lakes.

Enterprise AI doesn't work in isolation. Its typically part of a larger data pipeline. Starting from ingesting and acquiring data from various sources, data is curated, de-duplicated and cleansed in various ETL stages followed by further processing, all within a lakehouse, before being fed to AI based systems for training or inference. As is always the case, the quality and accuracy of the data is paramount for effectiveness of AI, therefore the importance of having a modern and integrated lakehouse can't be overstated, for the adoption of enterprise AI.

1.2 Introduction to IBM watsonx.data

Data is the fuel for AI. AI and Analytics based decision systems feed on massive amounts of data. The effectiveness and accuracy of insights generated by AI models depend on the quality of the data that they are trained on. The importance of clean, curated and accurate data can't be overstated. This is even more crucial with the emergence of generative AI.

Lakehouses are a step in that direction. However, fundamental challenges still remain:

1. First generation Lakehouses are still limited by their ability to address cost and complexity challenges. They are usually single query engines set up to support limited workloads e.g. just Business Intelligence (BI) or Machine Learning (ML).
2. Moreover, first generation Lakehouses typically deploy over cloud only with no support for multi-/hybrid cloud deployments.
3. Minimal governance and metadata capabilities to deploy across entire ecosystem remains an issue. The challenge is to bring analytics to data where it is generated and resides. Customers are looking for an easier migration path to a modern lakehouse with no migration or delayed migration of data and metadata.
4. And all this needs to be achieved while maintaining robust data governance and security policies in places, even as the usage and users of data become more varied than ever before.

To mitigate these issues, IBM designed the watsonx.data platform, positioning it as a modern Lakehouse platform to help organizations manage efficient use of their data. It's part of the broader IBM watsonx platform, an enterprise-ready AI and data platform designed to accelerate the adoption of enterprise AI. The watsonx family comprises of three platforms:

1. watsonx.ai - for generative AI and machine learning
2. watsonx.data - a next generation data lakehouse built on open architecture and open data formats
3. watsonx.governance - to enable AI workflows that are built with responsibility, transparency, and explainability.

watsonx.data empowers enterprises to scale Analytics and AI workloads. Based on a new open architecture lakehouse, it is a unique solution that allows co-existence of open source technologies and proprietary products. Its open architecture fully separates compute, metadata, and storage, and offers flexibility. This architecture provides a next-generation data query platform together with robust security, data governance, open data and table formats, allowing for seamless data access and sharing.

Users can store their enterprise data within watson.data or and make that data accessible directly for AI and Business Intelligence (BI). They can also attach existing enterprise data sources spread across cloud and on-premise environments to watsonx.data, which helps to reduce data duplication and cost of storing data in multiple places.

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 enterprises.

The following components provide the foundation of IBM watsonx.data architecture (see Figure 1-1):

Query engines The IBM watsonx.data platform natively includes Presto and Spark as the query engines. Presto is a distributed query engine designed to handle modern data formats that are highly elastic and scalable.

IBM watsonx.data query engines are fully modular and can be dynamically scaled to meet workload demands and concurrency. The engines can be attached to internal or external data stores in a plug-and-play configuration to access enterprise data in an open table format.

Milvus VectorDB Milvus is a vector database that stores, indexes, and manages embedding vectors used for similarity search and retrieval augmented generation (RAG). It is developed to empower embedding similarity search primarily for AI inferencing applications. Milvus is included in IBM watsonx.data as a service.

Metadata and Governance service The metastore included with watsonx.data is based upon the open-source Apache Hive Metastore (HMS). The metadata service enables the query engines to know the location, format, and read capabilities of the data. The metastore essentially manages table schemas as well as where to find them in object storage.

Data catalogs operate within the purview of the metadata and governance layer. Data catalogs assist query engines with finding the correct data and deliver semantic information for policies and rules specific to a particular data store. A data catalog is created specific to a data store when it is registered with watsonx.data and is managed by the HMS metadata service. The supported catalog types include Apache Iceberg, Hive, Apache Hudi or Delta Lake at the time of this writing.

IBM watsonx.data integrates with IBM Knowledge Catalog (IKC) and Apache Ranger for policy based governance and administration of data and metadata. The policy engine enables users to define and enforce rules for data protection.

One key aspect supporting the open architecture of IBM watsonx.data lakehouse is it's support for open table formats such as Apache Iceberg. As a vendor agnostic open table format, **Apache Iceberg** allows different engines to access the same data at the same time, thereby enabling data sharing across multiple repositories (e.g. data warehouses and data lakes). This allows using new technology with old data through metadata integration, and allows users to migrate data and workload at their own pace. Open formats and standards to ensure interoperability with future technology stacks.

As an example of the data sharing aspect, a IBM Db2® Warehouse has the option to read/write to/from a cloud bucket using open formats such as parquet and iceberg. The bucket metadata (table schemas et al.) can be exposed to watsonx.data using the 'Metadata Sync' feature provided by Apache Iceberg. This allows for seamless integration and sharing of data between IBM Db2 Warehouse and IBM watsonx.data without the need for deduplication or additional ETL operations, while allowing to offload some of the workloads to IBM watsonx.data.

IBM watsonx.data is delivered as containerized software as part of IBM Cloud Pak® for Data (PC4D) software bundle. IBM watsonx.data can be deployed on-premise, across multiple clouds, and is also as a managed service on AWS. An entry-level developer version is also available if you want to try it out.

For more information on IBM watsonx.data documentation here.

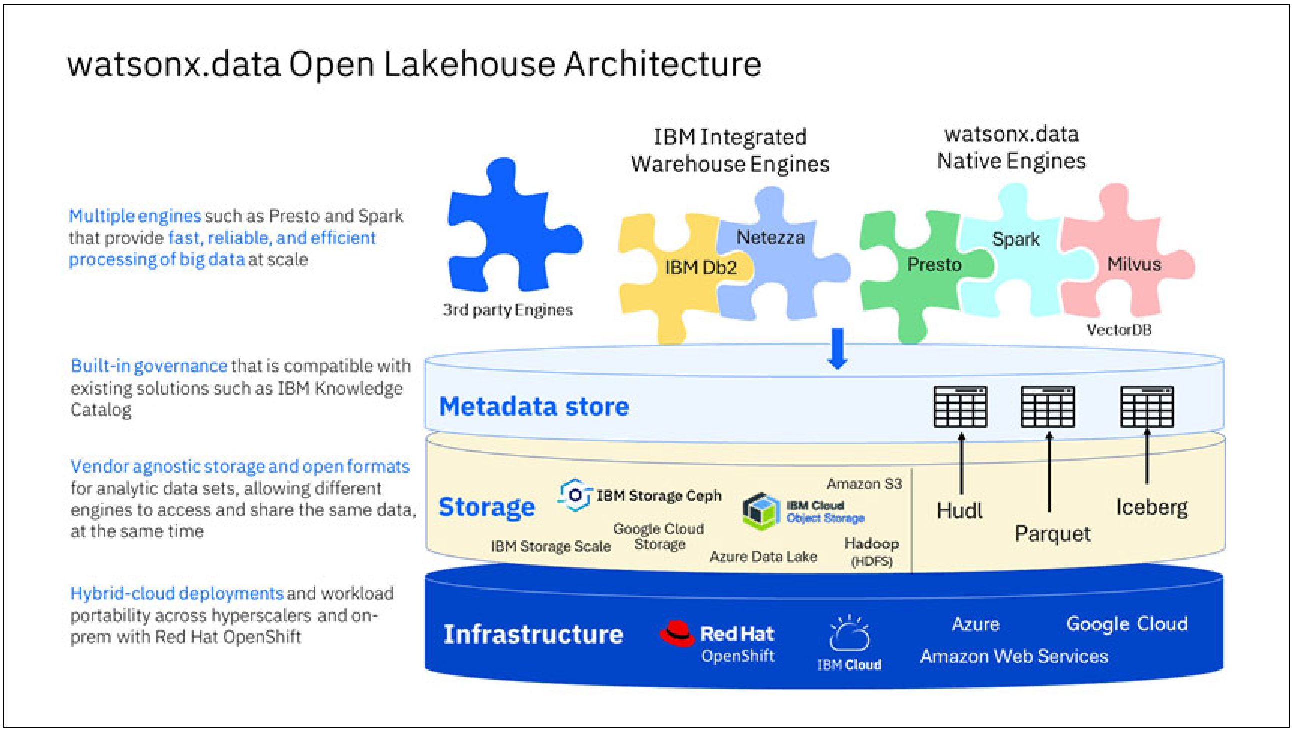


Figure 1-1 IBM watsonx.data architecture

1.3 Typical use cases for IBM watsonx.data

IBM watsonx provides a hybrid, open data lakehouse to power AI & Analytics workloads. Here are some key use cases to consider:

Rapid analytics with data virtualization

Query data in place with data virtualization in Presto, which has 35+ connectors to various external databases, HDFS, object stores and data store vendors.

Data warehouse optimization

Reduce the cost of expensive warehouses by “right sizing” workloads. Replace extract/transform/load (ETL) jobs with Spark, to reduce costs of your data warehouse through workload optimization. Discover data assets in your warehouse easily with the Apache Iceberg powered shared metadata and governance layer in watsonx.data.

Data Lake modernization

Accelerate modernization of your Data Lake with Apache Iceberg and object store. Replace or augment legacy Hadoop data lakes with an open data lakehouse and access better performance, security, and governance, without migration or ETL. Decoupling of compute and storage for independent scalability and lower costs.

Real-time analytics and business intelligence (BI)

Combine data from existing sources with new data to unlock new, faster insights without the cost and complexity of duplicating and moving data and metadata across different environments.

Streamline data engineering

Reduce data pipelines, simplify data transformation, and enrich data for consumption using Spark, SQL, Python, or an AI infused conversational interface.

Prepare Data for AI

Collect, curate and prepare data efficiently for use by AI with Spark and Milvus vector database. Build, train, tune, deploy, and monitor AI/ML models with trusted and governed data in IBM watsonx.data and ensure compliance with lineage and reproducibility of data used for AI. Integrated vectorized embedding capabilities in Milvus enable Retrieval Augmented Generation (RAG) use cases at scale across large sets of 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.

Figure 1-2 shows the positioning of watsonx.data within the IBM watsonx ecosystem.

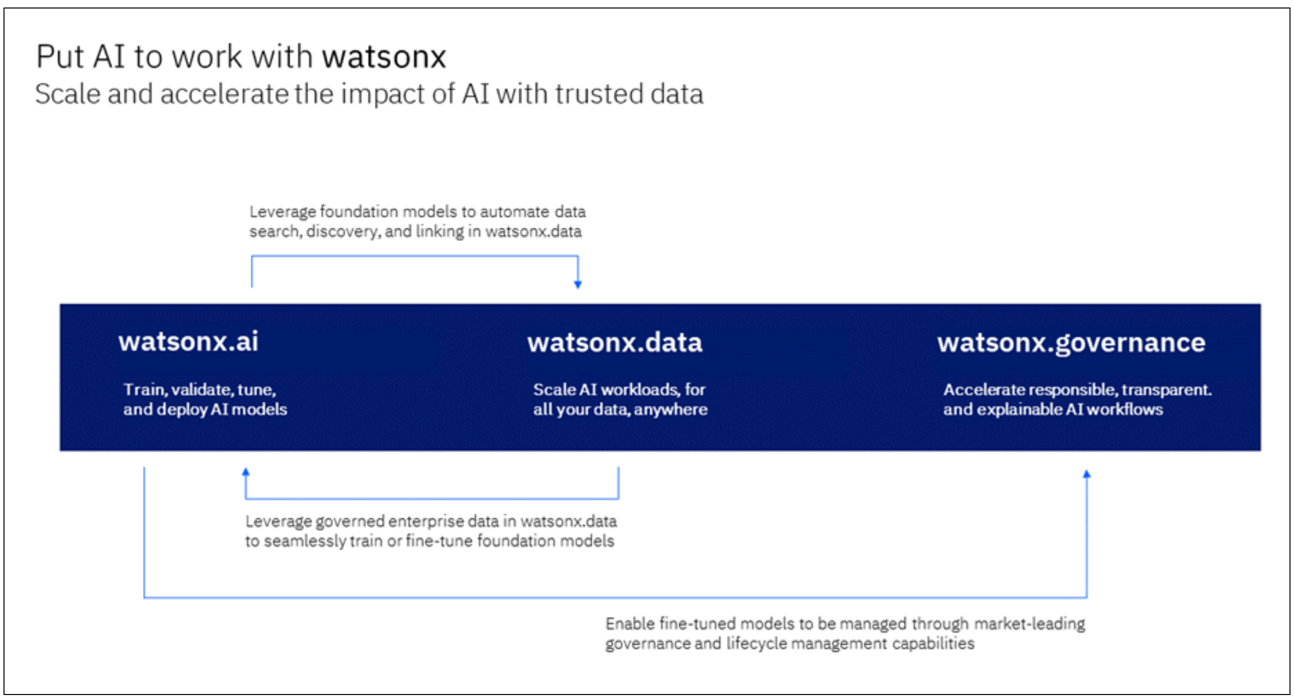


Figure 1-2 Positioning of watsonx.data within the IBM watsonx ecosystem

1.4 IBM Storage Scale

IBM Storage Scale (formerly known as IBM Spectrum Scale or IBM General Parallel File-System (GPFS)) is an industry-leading IBM storage software for file and object storage. It can be deployed as a software-defined storage solution that effectively meets the demands of AI, big data, analytics, and HPC workloads. It has market leading performance and scalability, and a wealth of sophisticated data management capabilities.

IBM Storage Scale System (formerly known as the IBM Elastic Storage® Server or ESS) is a fully integrated and tested IBM Storage Scale building block (Appliance) that provides enterprise grade performance, reliability, availability, and serviceability. It is an optimum way to deploy IBM Storage Scale storage for most IBM Storage Scale use cases. Alternatively, as a true software defined solution (SDS), customers can choose to deploy IBM Storage Scale over commodity servers (such as x86 based storage rich servers), whether on customer's on-premise environment or on a public cloud.

The ever-growing volume of data, multiple variety and formats of data and data silos being dispersed across on-premises or private/public clouds adds to the overall complexity and cost of building a modern Lakehouse solution built for the AI age. Enterprises who have invested in traditional data warehouses and data lakes are looking to simplify and modernize their applications. They are looking to integrate and unify the dispersed data sources for better data visibility, reducing duplication and controlling costs. These data sources could be cloud object storage, HDFS or even databases.

With IBM Storage Scale, customers can build a highly scalable Global Data Platform for their Lakehouse environments, offering higher performance, cost advantage and superior data management capabilities. IBM Storage Scale becomes the storage layer for the Lakehouse. Data may reside within Scale or be virtualized into Scale from any cloud, from any edge or from any legacy data silos, whether object, file or HDFS format. Data may be orchestrated to IBM Storage Scale to minimize the time to results. The Global Data Platform, powered by IBM Storage Scale offers the following differentiated data services. Refer to Figure 1-3.

- **Data Access Services**

  With a rich set of data protocols, IBM Storage Scale's Data Access Services provide unified and shared file and object access to any unstructured data stored anywhere across an organization. The data access services are “multi-lingual”, meaning some applications can create and access data with a certain protocol, and others may require access to the same data with a different protocol at the same time.

- **Storage Abstraction and Acceleration Services**

  IBM's global data platform provides high performance data access from where the data resides. By leveraging the IBM Storage Scale's Active File Management (AFM), it can abstract and virtualize remote data sources dispersed across the enterprise to be managed under a common storage namespace and accelerate them for high performance data access.

- **Data Management Services**

  IBM Storage Scale provides comprehensive Information life cycle management services including a highly flexible policy engine that allows customers to define rules for optimizing the storage of their unstructured data. These services transparently move data to the appropriate tier of storage, optimizing both cost and performance based on an organization's retention, archiving and data governance policies.

- **Data Resiliency Services**

  Data Resiliency Services provides comprehensive tools and capabilities to identify and detect threats, to protect an organization's data, and provide essential response and recovery capabilities when security breaches occur. These data resilience services align with all aspects of the NIST security framework, from practicing cyber hygiene before an event, all the way through detection, response, and recovery.

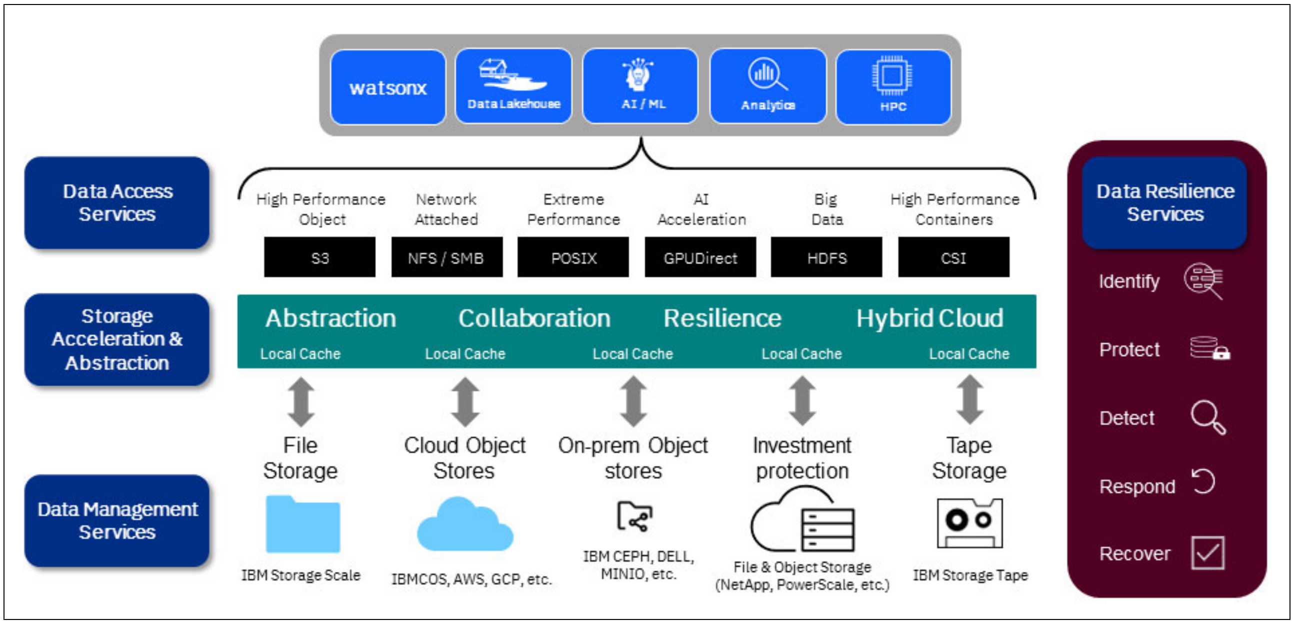


Figure 1-3 IBM Storage Scale, a Global Data Platform for Unstructured Data

For more information on IBM Storage Scale, see IBM Storage Scale

To know more about the IBM Storage Scale System (appliance), see IBM Storage Scale System.

1.5 IBM Storage Scale S3 Data Access

The AWS S3 API is established as the de-facto standard to process unstructured data as objects. More and more enterprises are integrating the S3 object access protocol in their workflows to acquire, process and manage unstructured data. To better support these evolving workloads, IBM added High Performance Object/S3 access to IBM Storage Scale Data Access Services which includes different protocol services such as NFS, HDFS and SMB already. This brings in standardization as to how the S3 service is deployed and managed within IBM Storage Scale as a data access protocol.

The Cluster Export Services (CES) infrastructure in IBM Storage Scale manages the following aspects for the Data Access services, including the S3 service:

- Manage high-availability: The participating nodes are designated as CES nodes or protocol nodes. A set of floating IP addresses, called CES address pool (CES IP Pool), is defined and distributed among the CES nodes. As nodes enter and leave the IBM Storage Scale cluster, the addresses in the pool can be redistributed among the CES nodes to provide high availability. Higher level application nodes access the S3 service over one or more of these floating IPs, which are assigned to active protocol nodes and moved to an inactive node in case of a failover.
- Monitoring the health of these protocols and raising events or alerts during failures

- Managing the floating IPs (CES IPs) that are used for accessing these protocols, by including failover and failback of these addresses which might be triggered by any protocol node failures

In a strategic approach, High Performance Object S3 service replaces Swift based Object S3 and Containerized S3 service which were the earlier implementations of the S3 protocol in IBM Storage Scale.

The new High Performance S3 service is still based on Red Hat Nooba and does not require a Red Hat OpenShift environment to be deployed which further simplifies the S3 service deployment and architecture. This simplified architecture allows for both containerized and non-containerized S3 applications to access the IBM Storage Scale S3 service.

The High Performance S3 service is optimized for multi-protocol data access to enable workflows which access the same instance of data using S3 and other access protocols. S3 objects are mapped to files and buckets are mapped to directories within IBM Storage Scale and vice versa.

For more information, see S3 support overview.

1.6 Storage Abstraction and acceleration with IBM Storage Scale AFM

IBM Storage Scale has the unique feature to abstract and virtualize remote S3 data sources dispersed across the enterprise. These S3 data sources could be local (on-premise) or could on various public clouds. By leveraging AFM and its enhanced local caching capabilities, data access to remote storage locations (e.g. S3 cloud stores or slow performing on-premises S3 stores) can be accelerated considerably, reducing data access times, while providing a common storage namespace for those dispersed storages.

IBM Storage Scale AFM virtualizes remote S3 buckets at fileset level. Cache relationships are created at a fileset level. Multiple such cache relationships can exist per file system, corresponding to remote buckets on various public clouds. Once the cache relationships are created, the remote S3 buckets appear as local buckets under the IBM Storage Scale file system, under a common storage namespace. This eliminates the need for data copy and greatly eases the management of those dispersed storages.

AFM uses user defined intelligent policies to accelerate data access including automatic eviction of data.

For more information, see Introduction to AFM to cloud object storage.

2

Solution Architecture and functional characteristics

This chapter provides an architecture overview of IBM watsonx.data with IBM Storage Scale and other IBM technologies associated with the solution.

2.1 Use cases for IBM Storage Scale with IBM watsonx.data

This section describes when to use IBM Storage Scale and IBM Storage Scale System based storage with IBM watsonx.data.

- **Disaggregated Compute and Storage infrastructure:** Customers like the flexibility to deploy, manage, monitor and grow their compute and storage infrastructures independent of each other. This provides them the flexibility to use different vendors, versions, architectures for compute and storage. IBM Storage Scale based software defined storage solution provides this flexibility as well enterprise level performance and features.
- **Existing IBM Storage Scale customers:** Existing customers of IBM Storage Scale who have already implemented Data warehouse or Data Lakehouse solutions can still modernize their Analytics solutions to use IBM watsonx.data while continuing to use IBM Storage Scale. The existing IBM Storage Scale environments can be upgraded to use High Performance S3 access to existing data as required by IBM watsonx.data
- **Modernize Storage infrastructure at small additional costs:** There are customers who are looking to modernize their Analytics solutions, however finding themselves significantly invested in legacy storage infrastructure. Such customers can still modernize, by adding IBM Storage Scale to their existing environment at a small additional cost, rather than having to modernize their entire storage infrastructure from scratch.
  - The disaggregated compute and storage architecture provides the flexibility to use existing investments in Compute/OpenShift for deploying IBM watsonx.data.
  - In addition, IBM Storage Scale can be configured over any of the existing legacy block or object storages and can easily virtualize legacy storages as accelerated S3 storage, hence protecting the existing investments while providing infrastructure modernization. This can be achieved by investing in High Performance IBM Storage Scale cluster (e.g.with NVME flash drives) that can be used to cache, accelerate frequently accessed data.

Customers looking for integrated compute and storage infrastructure solution for IBM watsonx.data in an appliance form factor, may consider IBM Fusion HCI based solution.

2.2 Solution architecture

This solution, as shown in Figure 2-1 consists of IBM watsonx.data software deployed on Red Hat OpenShift container platform. IBM Storage Scale storage environment is deployed outside the OpenShift cluster, in a non-containerized environment. The IBM Storage Scale infrastructure consists of IBM Storage Scale file systems that hold the data.
IBM Storage Scale provides the storage acceleration and abstraction feature for remote object buckets. It also offers the S3 Data Access protocol which provides high performance object access.

Depending on the customer's use case, IBM Storage Scale can be leveraged in this solution in either of the following two ways, or a combination of both:

1. As a high performance enterprise storage and as the primary object storage layer for the Lakehouse solution. The data buckets reside locally on the IBM Storage Scale file system itself.
2. As a persistent cache and storage acceleration layer for accessing remote object stores globally dispersed across various clouds, data centers and locations. For dispersed buckets, AFM abstracts and accelerates them in a way that these external buckets appear

as local buckets residing on the IBM Storage Scale file system itself. High-performance object access is delivered with intelligent caching service provided by AFM.

The S3 service then exposes the buckets (local or accelerated) to IBM watsonx.data for attachment to a query engine such as Presto or Spark.

This solution paves the way for complete separation of the compute and storage, which comes with the benefit of having to manage, operate, scale and grow the compute (Red Hat OpenShift/IBM watsonx.data) and storage (IBM Storage Scale) layers completely independent of each other. The storage and the S3 service stays outside OpenShift and is accessed by the S3 protocol from the compute layer in a plug-and-play configuration.

When processing engines within watsonx.data access data, the request reaches the IBM Storage Scale S3 service over S3 protocol. The S3 service interacts with the IBM Storage Scale client on the same node, which then hands off the I/O to IBM Storage Scale server nodes using the NSD protocol. If the bucket happens to be remote and has not been cached already onto the IBM Storage Scale file system, AFM gateways are engaged to access the remote object bucket. All this happens transparently without the IBM watsonx.data applications having to know details of the remote object bucket itself.

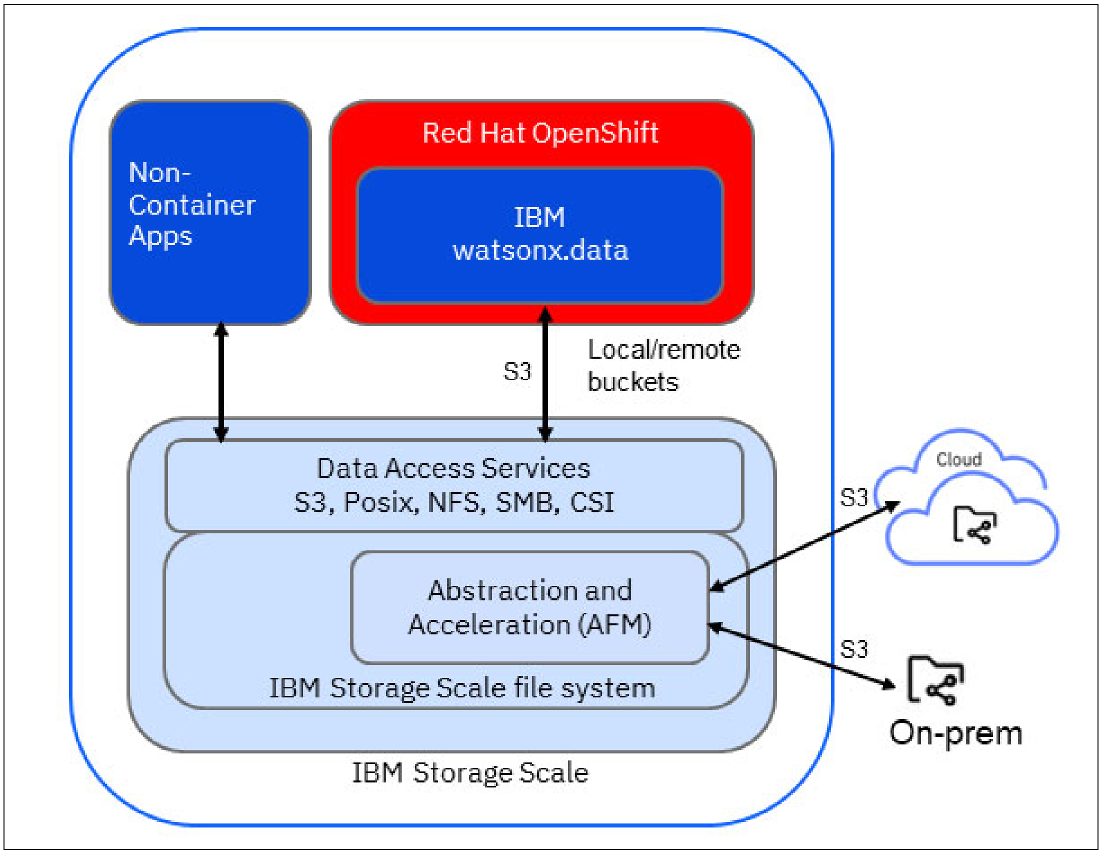


Figure 2-1 IBM watsonx.data with IBM Storage Scale Architecture

2.3 Lakehouses: Storage Pain Points

Even as IBM watsonx.data is capable of processing data from various sources as is where is, the data sprawl in terms of multiple storage silos pose a challenge to Storage Administrators from the perspective of management, visibility, governance and security of enterprise data. In this section, we look at the key customer pain points from a storage perspective.

1. Data Silos

   Enterprise data is distributed. Multiple data silos make it difficult to integrate structured and unstructured data into data lakehouse architectures. Users may be required to copy data, creating duplication and data management challenges. Customers need a storage solution that:

   – Can effectively integrate data across multiple data sources. Deliver the data closer to the application, while hiding the complexity and make it transparent to the workloads.

2. Escalating storage and data management costs

   As the volume of data being generated grows exponentially, customers need a storage solution that:

   – Can effectively manage the data lifecycle

   – Bring down costs using compression, data-deduplication.

   – Allows them to seamlessly manage data in different cost optimized tiers

   – Support YoY capacity growth story

3. Performance challenges

   Object and NAS filers often do not deliver the level of performance required for high performance. Businesses require that storage is not a bottleneck in the face of demanding compute and data intensive applications. They need a storage solution that:

   – Can accelerate storage performance for low latency cloud object storage

   – Offers superior storage performance, for customers expecting DW or OLAP type performance from their lakehouse at low cost.

   – Offers high IOPS and low latency performance for new-age applications such as IoT, Generative AI Training and Inference, or metadata intensive workloads involving large number of small files.

4. Security Challenges

   Data is a critical business asset. Customers need a storage solution that:

   – Protects their data from security threats, unplanned disasters and always available

   – Resilient to cyber threats, can be brought back online quickly and highly available to keep the business running

   – Enterprise security features including ransomware protection.

2.4 Lakehouses: The value proposition of IBM Storage Scale

These pain points together with the accelerated growth of AI across enterprises highlights the need for a high performant and hybrid Global Data Platform for enterprise data.
IBM Storage Scale and IBM Storage Scale System is well positioned within IBM's storage portfolio to address the market needs based on its leadership as a high-performance storage solution for Data and AI.

Following top benefits are realized by using IBM Storage Scale with IBM watsonx.data Lakehouse:

- Storage abstraction and virtualization, eliminate silos

Leveraging AFM, IBM Storage Scale can virtualize and abstract dispersed storages (islands) all over the enterprise and make them available under a common namespace. A single global namespace delivers a consistent, seamless experience for new or existing storage, making it easier to manage them from a single window of control. It reduces unnecessary data copies and improve efficiency, security and governance. Data may be virtualized and orchestrated into Scale from any cloud, from any edge or from any legacy data silos, whether object, file or HDFS format, thereby minimizing the time to result.

- Accelerated storage where performance matters

  IBM Storage Scale AFM performs as a tier 1 data caching service, performing automatic, transparent caching of back-end storage systems. It provides a high-performance persistent storage cache, together with low-capacity requirements. This has the effect of accelerating data queries and improve economics by fronting lower performance storage. With watsonx.data, a 5-15x improvement in query performance can be seen.

- Collapse layers and simplify data integration with multi-protocol data access

  IBM Storage Scale has the most comprehensive support for data access protocols. It supports data access by using S3, NFS, SMB, POSIX, HDFS and GPUDirect. This feature eliminates the need to maintain separate copies of the same data for traditional applications, analytics and AI, and enable globally dispersed teams to collaborate on data regardless of protocol, location or format

  While S3 is a must for lakehouses, multi-protocols provide the flexibility to ingest or access data from various legacy data sources. For example,

  – Data can be ingested into a bucket via NFS and same data is instantly available for processing by watsonx.data engines via S3.
  – Data may be curated and cleansed via Spark in IBM watsonx.data for AI model training or inference purposes. The curated data may be made available to AI workflows via POSIX and GPUDirect for highest performance access.

  This facilitates in-place analytics and simplifies the complexity of enterprise-wide data workflows starting from data cleansing all the way to AI.

- A Lakehouse optimized for AI

  IBM Storage scale with its rich set of data access protocols, provides a unified data platform for analytics and AI, reduces costs and simplifies data workflows. A high-performance storage platform, it minimizes the cost of training AI models by delivering a faster time to solution, as GPU resources are expensive. GPU Direct Storage (GDS) offers high bandwidth, low latency performance to train Generative AI models faster. Provides a Landing Zone for high-speed data ingest to AI training jobs.

> **Note:** The Granite series of Generative AI models shipped with IBM watsonx.ai were trained with large datasets residing on IBM Storage Scale.

- Lower costs

  – The IBM Storage Scale System provides much higher storage density than competition. This translates into cost savings in terms of Power, cooling and rack space needed in the data center. For customers requiring higher storage capacity and growth outlook, this can lower the Total Cost of Ownership (TCO) significantly over the years.
  – Multi-protocol access to same data, eliminates the need to maintain separate copies of the same data for traditional applications and for analytics or AI, thereby reducing data center footprint and associated costs.
  – Multiple performance tiers for storage, to optimize costs and performance. A high-performance tier for hot data along with a cost-effective tier or even tape for long

term storage and archival, together with automated policy driven placement across tiers makes it seamless and transparent to applications.

- Extreme scalability with parallel architecture

  IBM Storage Scale supports exabyte scale storage capacity. It is a storage platform that supports long term data growth story. Capacity be easily extended in a modular way, with linear scalability for future growth. It's distributed metadata architecture enables it to support billions of objects in terms of number without compromising on performance.

- A Proven Performance platform
  - Proven performance for HPC, Analytics and AI workloads. With a parallel architecture, every node in the cluster serves data and metadata, and no single node can become a bottleneck. This enables IBM Storage Scale to provide top-tier performance for demanding workloads, and retains the performance even as the capacities continues to grow.
  - High performance ingest via Posix and also via AFM and multiple protocols
  - Extreme performance for AI with GPU Direct Storage (GDS) for NVIDIA platforms
- A robust enterprise platform
  - A highly available and resilient storage platform, Six 9's for all apps: AI, Analytics, HPC, Backup, Archive, and Cloud
  - Well integrated with enterprise backup and restore solutions (IBM Storage Protect/IBM Storage Archive or 3rd party). Stretch Cluster, AFM DR provide tested DR solutions
  - Cyber resilient, encryption, WORM, data immutability support

2.5 Benefits and use cases of IBM Storage Scale AFM

The storage Acceleration feature in IBM Storage Scale AFM provides high performance data access by acting as a tier 1 data caching service for back-end storages. See Figure 2-2. AFM works together with Alluxio based in-memory caching in Presto, accelerating query performance significantly. In this context it is noteworthy to call out the differences between storage based caching as compared to in-memory caches. In-memory caches typically operate at a per-node basis. As a result, a query that may run faster on a given node while leveraging the cache, may not experience the same performance when run on another worker node, if there is a cache miss.

Key advantages of IBM Storage Scale based data cache are the following:

- It operates as a shared data cache, available to all the engines in IBM watsonx.data, whereas Alluxio is only available to Presto.
- The shared data cache is available to all the worker nodes in IBM watsonx.data at any given time.
- It provides a persistent data cache even for newly provisioned engines and survives engine restarts.
- Shared data cache is available to multiple protocols and not just to S3.

> **Note:** Many public cloud providers charge their customers data egress costs for moving data out of cloud. Therefore, having the data cached locally via AFM provides costs savings on the data egress costs. Storage acceleration also reduces the contention for bandwidth.

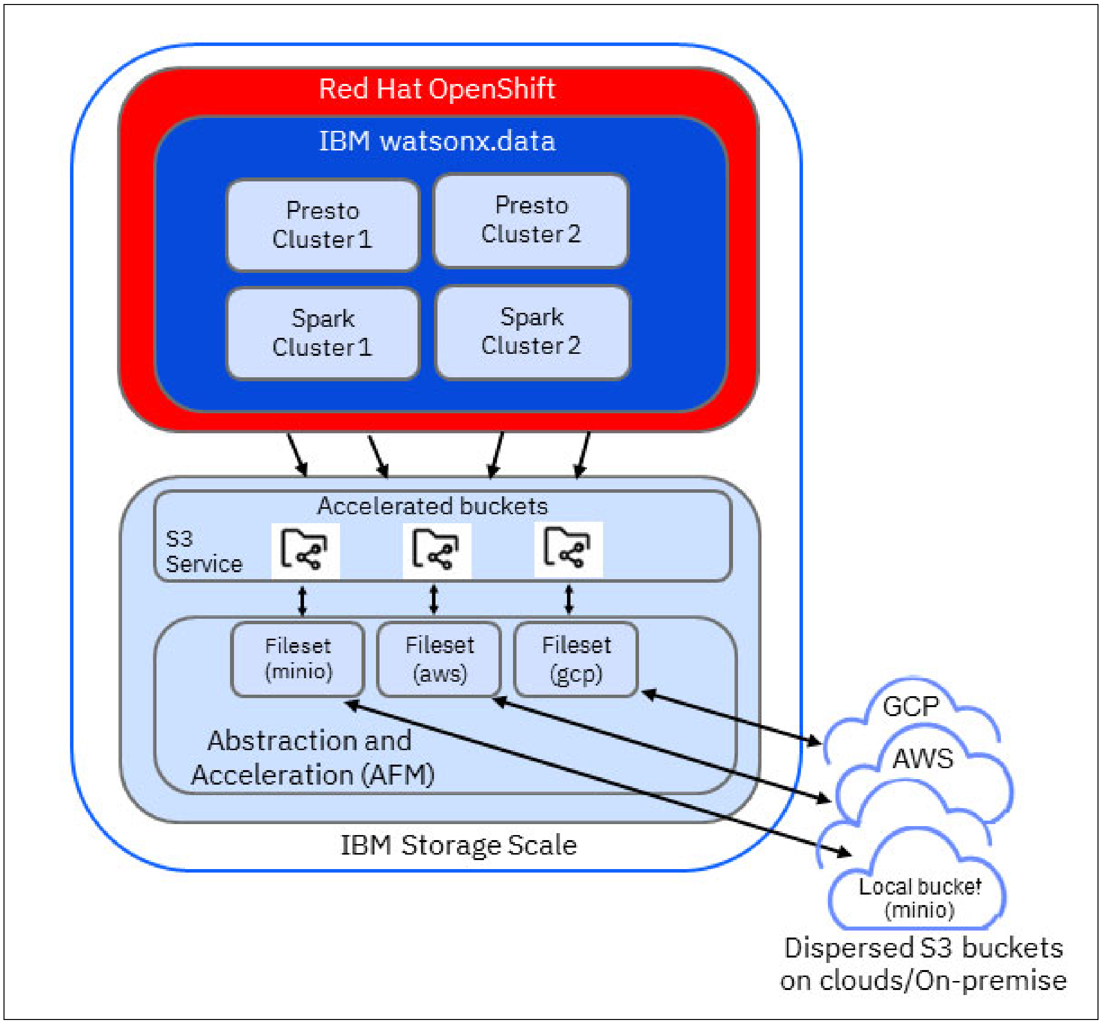


Figure 2-2 AFM aggregating dispersed storages under a common storage namespace

Here are some key use case of IBM Storage Scale AFM for IBM watsonx.data:

1. IBM watsonx.data is running on-premise and the applications (Presto/Spark) require high performance access to data stored in S3 buckets in a public cloud or in a different data center location. AFM transparently executes data caching services of the data from it's home location and accelerates the storage performance.
2. IBM watsonx.data is deployed on a public cloud. However, customer prefers to have their enterprise data on premise for security or regulatory reasons. In these scenarios as well, AFM can transparently execute data caching services of the data from on-prem location and accelerates the storage performance.
3. IBM watsonx.data is running on-premise accessing S3 data sources residing on-premise as well. However, the storage performance from these data stores is not adequate to meet the customer's query SLAs. In these scenarios, AFM can be used to accelerate IBM watsonx.data queries by fronting lower performance storage.
4. Customer wants to run IBM watsonx.data queries accessing data in a legacy NFS data source whether on-premise or in a cloud location. AFM can transparently virtualize those data sources and present them as S3 buckets to watsonx.data query engines while still accelerating the storage performance

3

Planning and sizing

This chapter describes planning and sizing guidelines for the licensed components and highlights several planning activities related to the solution in this Redpaper.

3.1 Supported configurations

Table 3-1 lists the physical platform and software component levels of the IBM watsonx.data with IBM Storage Scale architecture.

Table 3-1 IBM watsonx.data with IBM Storage Scale architecture components

Product name	Version
IBM Storage Scale	5.2.1
watsonx.data	2.0.2
Cloud Pak for Data (CP4D)	5.0.2
Architecture	x86_64

3.2 Planning

This section describes planning for IBM Storage Scale.Planning for IBM Storage Scale

You need to plan for IBM Storage Scale for capacity, performance and storage abstraction (AFM) and for advanced features such as Storage Tiering.

The storage capacity planning for the IBM Storage Scale cluster depends upon the customer's use case, whether Scale is being used as the primary storage for the S3 buckets or only for storage acceleration, or both.

For capacity planning as primary storage, take into account your current and the projected YoY growth storage requirements, together with the combined storage bandwidth offered by the system to plan for optimum performance. Visit the links Configuring and tuning your system for GPFS and Parameters for performance tuning and optimization to tune your cluster for optimum performance.

Planning for IBM Storage Scale AFM

AFM requires storage from a local Scale cluster to be used as persistent cache. Hence the AFM nodes must belong to the IBM Storage Scale server cluster. It's not possible to designate nodes from a Scale client cluster as AFM nodes. These nodes handle the outbound and inbound communication from a remote S3 data source. Avoid co-locating AFM nodes with Scale NSD nodes.

For storage acceleration, each bucket being accelerated maps to one AFM gateway node. In a I/O heavy production system with multiple accelerated buckets, it may be worthwhile to configure two or more AFM gateway nodes, for optimum performance and high availability.

It is also recommended that AFM filesets backing remote S3 buckets are configured over a fast storage tier, such as a Storage pool configured over NVME disks.

Planning for S3 service

The High Performance S3 service was added as of release 5.2.0 as a Technical Preview and as of release 5.2.1 as General Availability. For customers running a much older version of IBM Storage Scale System, its recommended that a separate Scale client cluster is configured as a Protocol cluster, unless the original IBM Storage Scale System itself can be upgraded to version 5.2.1.

As is the case with other CES-enabled protocols, S3 service is configured on designated protocol nodes. There are two possible architectures to deploy the S3 service:

- The S3 Protocol nodes can be part of the Scale (server) cluster itself. This may be the preferred approach for software-defined Scale environments including deployments on public clouds.
- Otherwise, the S3 Protocol nodes can be part of a separate Scale (client) cluster and accessing the Scale file system using remote mount access as described in IBM Storage Scale Documentation Mounting a remote GPFS file system

  In this configuration, the Scale server cluster (such as the IBM Storage Scale System or ESS) grants permission to the Scale client cluster for its owning file systems. The Scale client cluster remotely mounts the file systems and operates as a Protocol cluster to the application layer above. For very small or testing environments, the value of this additional administration effort might not become easily apparent. For production environments, however, this approach has some distinct advantages.

  – The Scale client cluster environment can be individually scaled, managed and upgraded, for example, to take advantage of new S3 service versions and improvements delivered with new IBM Storage Scale releases, with incurring minimal or no changes to existing IBM Storage Scale System infrastructure.
  – Higher level of storage isolation and multi-tenancy at storage level can be accomplished by constraining the IBM Storage Scale client clusters to access only the designated IBM Storage Scale filesystems or filesets, if the security policies demand so. For example, in an organization with multiple lines of business or departments, a dedicated Scale client cluster can be assigned to each of such department while keeping a common storage backend.
  – The Scale client cluster and the Scale server can run different versions of Scale software, if needed.
  – See Figure 4-1 on page 24 that shows the three tiered deployment for IBM watson.data with IBM Storage Scale architecture

Network Planning

The performance of the IBM watsonx.data IBM Storage Scale solution depends on the network provisioned for communication between watsonx.data and IBM Storage Scale. In case of a multi-tiered architecture defined in the prior section, it's important to plan for separate networks and network interfaces for

- network between the OpenShift cluster and the S3 Protocol node
- network between S3 Protocol cluster and IBM Storage Scale server (NSD) cluster, including the network between the Storage Scale server nodes itself.

3.3 Sizing guidelines

This Redpaper describes an open solution based on the software products used as components and not a dedicated appliance styled solution, thereby allowing higher flexibility in terms of how to size the solution keeping in view the customer's current business needs and growth projections. Hence the components of the solution, such as Red Hat OpenShift, IBM Storage Scale can be independently scaled as needed.

See the Red Hat OpenShift deployment Planning section of IBM watsonx.data product documentation before you begin. Especially review the System Requirements topic.

For production workloads, the following hardware configuration is recommended for the Red Hat OpenShift based worker nodes running IBM Watsxon.data:

Raw cores = 64
System memory (GB) = 1920
Local storage = 300 GB

If AFM based storage acceleration is needed, determine how large the persistent storage cache must be. This depends on various factors including:

- Total number of filesets (i.e. buckets) being cached
- Total size of the data on remote S3 storage, and size of the buckets that need to be cached at some point in time, based on access patterns by watsonx.data applications.
- Amount of data that needs to be read locally by watsonx.data applications during a short span of time to avoid multiple round-trip reads/writes to the remote S3 storage
- Type of caching used: whether it's read-only cache or read-write cache

Table 3-2 provides guidance on how to size the storage capacity to be configured for the cache. This is defined as a % of the actual storage size of the dispersed storage (remote/local) that is being accelerated.

Table 3-2 Sizing guidance for persistent storage cache for AFM based storage acceleration

Size of remote S3 data source	Cache size (as % of the remote storage size)
10 TB or smaller	30 %
10 TB to 1 PB	20 %
1 PB or larger	10%

4

Configuring the solution

This chapter shows how to configure the components of the solution i.e. IBM Storage Scale, the S3 service, and AFM. It also shows how to create regular and accelerated S3 buckets and registering those buckets to IBM watsonx.data.

4.1 Configuring IBM Storage Scale and IBM Storage Scale System

An example deployment following the recommended architecture as outlined in Figure 4-1. The deployment follows a three tiered architecture and specifically the separation of compute and storage infrastructure.

The tiers are further identified as follows:

- The compute tier contains the application layer. It is running the Red Hat OpenShift Container Platform, which hosts IBM Cloud Pak for Data and IBM watsonx.data. The latter, watsonx.data, uses S3 object protocol and accesses the storage infrastructure.
- The Scale client cluster contains the S3 protocol nodes. There are multiple Protocol nodes for high-availability and is an uneven number of nodes to achieve quorum, but an even number with Tiebreaker configuration can be used as well. The nodes of the Scale client cluster remotely mount the file system(s) of the Scale server cluster.
- The Scale server cluster consists of NSD server nodes and another node acting as AFM gateway. The Scale server cluster owns the file system and acts as an acceleration layer using AFM to connect with other S3, NFS data sources, or even another (remote) IBM Storage Scale file system.

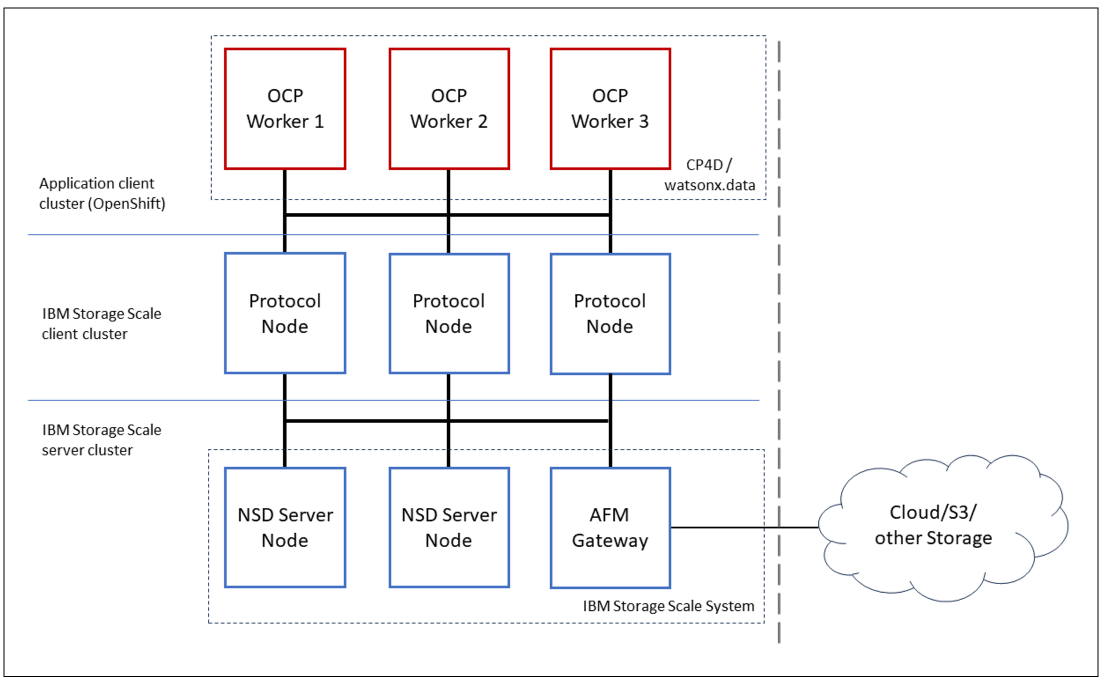


Figure 4-1 Three tiered deployment for IBM watson.data with IBM Storage Scale architecture

Example 4-1 listing shows the configurations of the Scale client cluster used as example for this paper.

Example 4-1 Configurations of the Scale client cluster used in this Redpaper

```
[root@fscc-sr650-46 ~]# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         cess3gpfs.bda.scale.ibm.com
  GPFS cluster id:           7776919622712034828
  GPFS UID domain:           cess3gpfs.bda.scale.ibm.com
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name           IP address  Admin node name            Designation
-------------------------------------------------------------------------------------
   1   node46s.bda.scale.ibm.com  10.10.1.86  node46s.bda.scale.ibm.com  quorum-manager-perfmon
   2   node47s.bda.scale.ibm.com  10.10.1.87  node47s.bda.scale.ibm.com  quorum-manager-perfmon
   3   node48s.bda.scale.ibm.com  10.10.1.88  node48s.bda.scale.ibm.com  quorum-manager-perfmon
```

The CES IPs are configured and assigned to the nodes of the Scale client cluster, in Example 4-2, we have two CES IPs setup.

Example 4-2 Listing the CES IP Addresses

```
[root@fscc-sr650-46 ~]# mmces address list --by-node
Node   Daemon node name           IP address   CES IP address list
------ -------------------------- ------------ ---------------------
1      node46s.bda.scale.ibm.com  10.10.1.86
2      node47s.bda.scale.ibm.com  10.10.1.87   10.10.1.121
3      node48s.bda.scale.ibm.com  10.10.1.88   10.10.1.120
```

Example 4-3 shows the Scale client cluster has mounted two file systems, one for data access, the other acting as CES shared root.

Example 4-3 Listing the remote mounted filesystems in the Scale client cluster

```
[root@fscc-sr650-46 ~]# mmremotefs show all
Local Name  Remote Name  Cluster name       Mount Point       Mount Options    Automount  Drive  Priority
essData     essData      ess3k5.bda.scale.com /gpfs/essData     rw               no         -        0
essCesRoot  essCesRoot   ess3k5.bda.scale.com /gpfs/essCesRoot  rw               no         -        0
```

The Scale server cluster owns the file systems and is described is described in Example 4-4.

Example 4-4 Configurations of the Scale server cluster used in this Redpaper

```
[root@fscc-sr650-36 ~]# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         ess3k5.bda.scale.com
  GPFS cluster id:           213597018859291206
  GPFS UID domain:           ess3k5.bda.scale.com
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR
```

```
 Node  Daemon node name            IP address    Admin node name             Designation
-------------------------------------------------------------------------------------
   1   ess3k5a.bda.scale.ibm.com  10.10.1.185   ess3k5a.bda.scale.ibm.com  quorum-manager-perfmon
   2   ess3k5b.bda.scale.ibm.com  10.10.1.186   ess3k5b.bda.scale.ibm.com  quorum-manager-perfmon
   3   ems3k5.bda.scale.ibm.com   10.10.1.76    ems3k5.bda.scale.ibm.com   quorum-manager-gateway-perfmon
```

The Scale client cluster, containing the S3 service, is used to configure S3 access for watsonx.data. This process is explained more elaborately over the next sections. First, an S3 account is needed, which can be created after the corresponding user and group have been defined within the operating system. For this account, a S3 bucket is created afterwards. The combination of CES IP and port, account credentials (access key and secret key), and bucket name are required to define the connection from watsonx.data

4.2 Configuring S3 access

This section describes the steps to install and configure the IBM Storage Scale S3 service, followed by how to create S3 buckets.

4.2.1 Install and configure the IBM Storage Scale S3 service

To setup and install IBM Storage Scale, a convenient method is to use the Installation Toolkit. For general steps to install IBM Storage Scale, refer to Installing. The following highlights steps to install and enable the S3 service specifically"

- Setup the basic information of the Scale cluster using the Installation Toolkit. Follow the above documentation.
- After the basic information is setup, define protocol nodes for the cluster using:

```
# cd /usr/lpp/mmfs/5.2.1.0/ansible-toolkit/
# ./spectrumscale node add hostname -p ...
```

- Define the CES IPs for the cluster to be exported using:

```
# ./spectrumscale config protocols -e <EXPORT_IP_POOL>
```

Where <EXPORT_IP_POOL> is a comma separated list of IP Addresses that would be used to access the S3 service by applications.

> **Note:** Reverse DNS lookup needs to be available for all CES IPs. The CES IPs must be unique and cannot be cluster node IPs.

- Configure the CES shared root file system, which is used for configuration and administration of the CES protocols, using:

```
# ./spectrumscale config protocols -f essCesRoot-m /gpfs/essCesRoot
```

> **Note:** It is recommended that the CES shared root is a separate file system. The CES shared root needs to be at least 4 GB.

- Enable the S3 protocol using:

```
# ./spectrumscale enable s3
```

- Review the configuration and perform a precheck of the deployment using

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

- Finally, deploy the changes using:

```
# ./spectrumscale deploy
```

- Verify that the S3 services are up and running on the designated S3 protocol nodes:

```
# mmces service list -a
node46s.bda.scale.ibm.com:  S3 is running
node47s.bda.scale.ibm.com:  S3 is running
node48s.bda.scale.ibm.com:  S3 is running
```

- To view the S3 protocol configuration, run

```
# mms3 config list
```

Example 4-5 shows the default configuration of the S3 service

Example 4-5 Default configuration of the S3 service

```
# mms3 config list
 S3 Configuration:
 =======================
 ALLOW_HTTP : false
 DEBUGLEVEL : default
 ENABLEMD5 : false
 ENDPOINT_FORKS : 2
 ENDPOINT_PORT : 6001
 ENDPOINT_SSL_PORT : 6443
 GPFSDLPATH : /usr/lpp/mmfs/lib/libgpfs.so
 NC_MASTER_KEYS_GET_EXECUTABLE : /usr/lpp/mmfs/bin/cess3_key_get
 NC_MASTER_KEYS_PUT_EXECUTABLE : /usr/lpp/mmfs/bin/cess3_key_put
 NC_MASTER_KEYS_STORE_TYPE : executable
 NSFS_DIR_CACHE_MAX_DIR_SIZE : 536870912
 NSFS_DIR_CACHE_MAX_TOTAL_SIZE : 1073741824
 NSFS_NC_CONFIG_DIR_BACKEND : GPFS
 NSFS_NC_STORAGE_BACKEND : GPFS
 UVTHREADPOOLSIZE : 16
```

4.2.2 Configuring filesets as a backing storage for S3 buckets

A fileset is a subtree of an IBM Storage Scale file system that in many respects appears like an independent file system. Filesets provide a means of partitioning the file system to allow administrative operations at a finer granularity than the entire file system.

Filesets can have their own defined quotas for data and inodes. The owning fileset becomes an attribute of each file for enforcing IBM Storage Scale based policies (such as automated tiering and placement, encryption, compression) as needed. Each fileset mounts at a regular directory path (called JunctionPath) within the Scale file system. A regular S3 bucket may be defined over the mount path.

For more information, see Filesets.

Here are some scenarios, where it may be preferred to create a S3 bucket over a fileset rather than over a regular directory:

- Assign a quota to the storage space that a bucket may consume
- Limit the total number of objects that a bucket may consume
- When it's needed to have all of the bucket's data (objects) to be automatically encrypted on the disk.
- Define automating tiering and placement policies for a bucket. For example, if cold data is uploaded to a bucket, automatically move it to a slower storage tier.
- Create time-based snapshots of the storage at a bucket level. Fileset snapshots can be created instead of creating a snapshot of an entire file system.

Example 4-6 shows how to create an independent fileset and create a S3 bucket on top of it.

Example 4-6 Creating an independent fileset

```
# mmcrfileset essData fset-1 --inode-space new
Fileset fset-1 created with id 1 root inode 524291.
```

Link the fileset to an IBM Storage Scale mount path.

```
# mmlinkfileset essData fset-1 -J /gpfs/essData/watsonx/b-wxd-fset1
Fileset fset-1 linked at /gpfs/essData/watsonx/b-wxd-fset1
```

Then proceed to create a S3 bucket over the fileset's mount path (directory), described in the following section. In the above example, the mount path is `/gpfs/essData/watsonx/` which would be the default bucket path (`--newBucketsPath`) for our S3 account.

4.2.3 Creating S3 buckets

This section explains how to configure S3 buckets over IBM Storage Scale. For every new bucket, a new directory is created under the IBM Storage Scale file system.

Alternatively, a bucket could be configured over a pre-existing directory. For example, a bucket could be configured over the mount point directory of a IBM Storage Scale fileset, so that the fileset becomes the backing storage of the S3 bucket.

The steps to create a S3 bucket is shown in the following command listings.

Create a S3 account first, associating the account to a system user.

/usr/lpp/mmfs/bin/mms3 account create <S3 account-name> --uid <uid> --gid <gid> --newBucketsPath <Path>

Where,

<uid> and <gid> are the Posix UID and GID associated with the S3 account. These parameters are not needed to be passed if an account name is passed. The account name should be a valid system username.

<Path> is filesystem absolute path, which will act as a base path for S3 buckets created using S3 API. This path can be overridden for buckets created with the **`mms3 bucket create`** command.

To view the details associated with this S3 account including its AWS access credentials, run

```
# mms3 account list <S3 account-name>
```

Then, create one or more S3 buckets, corresponding to this S3 account. There are two ways a bucket can be created.

1. Using the **mms3** command:

   ```
   # mms3 bucket create <S3 bucket-name> --accountName <S3 account-name>
   --filesystemPath <Path>
   ```

   Where

   <S3 account-name> is the name of account which should be used for the bucket

   <Path> is the filesystem absolute path including the directory for the bucket, which is to be used for bucket creation. This could be different than the default bucket path (--newBucketsPath) configured for the S3 account.

   The command will create a new directory with system path <Path> which will correspond to the S3 bucket.

2. Using the S3 API, e.g. the “aws” S3 client as shown in Example 4-7.

Example 4-7 Creating an S3 bucket using the S3 API

```
# wget https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip
# unzip awscli-exe-linux-x86_64.zip
# cd aws
# ./install
# alias s3u2='AWS_ACCESS_KEY_ID=<Your AWS_ACCESS_KEY_ID>
AWS_SECRET_ACCESS_KEY=<Your AWS_SECRET_ACCESS_KEY> aws --endpoint
https://10.11.94.182:6443 --no-verify-ssl s3'

# s3u2 mb s3:// b-watsonx2
make_bucket:  b-watsonx2
```

Example:

- Creating a S3 account named “watsonx”

  ```
  # mms3 account create watsonx --uid 2002 --gid 100 --newBucketsPath
  /gpfs/essData/watsonx/
  ```

- View the details associated with this S3 account including its AWS access credentials

  ```
  # mms3 account list watsonx

   Name     New Buckets Path      Uid  Gid Access Key              Secret Key
  -----     ----------------      ---  --- ----------              -------------
  watsonx  /gpfs/ess3k54/watsonx/ 2002 100 <Our AWS_ACCESS_KEY_ID>< Our
  AWS_SECRET_ACCESS_KEY>
  ```

- Creating a S3 bucket named under the default Bucket path

  ```
  # mms3 bucket create named b-watsonx --accountName watsonx --filesystemPath
  /gpfs/essData/watsonx/b-watsonx
  ```

- Creating a S3 bucket named b-watsonx2 not under the default Bucket path

  ```
  # mms3 create b-watsonx2 --accountName watsonx --filesystemPath
  /gpfs/essData/b-watsonx2
  ```

- Creating a bucket over an existing directory, containing data in it, in the following way:

  Change ownership of that directory to that of the S3 account first. For example.

  ```
  # chown watsonx:users /gpfs/essData/data-dir1
  ```

  For Fileset backed storage, the mount paths are created as owned by root. Change ownership of that directory to that of the S3 account.

  ```
  # chown watsonx:users /gpfs/essData/watsonx/b-wxd-fset1
  # mms3 bucket create b-watsonx3 -accountName watsonx -filesystemPath
  /gpfs/essData/data-dir1
  Note: The directory '/gpfs/essData/data-dir1' for bucket already exists.
  Skipping update of ownership and the setting of permissions of the directory
  for the user with uid:gid=2002:100
  Bucket b-watsonx3 created successfully
  ```

4.3 Configuring Data Abstraction and Acceleration

Configure IBM Storage Scale AFM to enable Storage Abstraction and Acceleration for dispersed buckets. This workflow involves:

- Creating a rule in AFM defining the connection to the remote S3 bucket. This will create a fileset in IBM Storage Scale corresponding to the remote S3 bucket.
- Creating a S3 bucket over that fileset. This bucket thus created abstracts or virtualizes the remote object bucket and is exposed through the IBM Storage Scale S3 interface.

Follow these instructions for configuring storage acceleration over remote buckets:

To start with, designate one or more nodes as AFM nodes. To designate a node as a AFM node, first ensure that the node has the AFM rpm (gpfs.afm.cos.*) installed, and the node has necessary connectivity to the remote cloud object S3 endpoint. Then run:

```
# mmchnode --gateway -N <AFM node hostname>
```

- Get the AWS access key ID and secret key for your remote bucket instance. For example, if using IBM COS, navigate to cloud.ibm.com → Instances → Storage → Service Credentials Tab → expand on down arrow. Get the details from cos_hmac_keys.
- Login to an AFM gateway node.
- Create the access keys in AFM corresponding to the remote object bucket.

  ```
  # mmafmcoskeys bucket[:{[Region@]Server|ExportMap}]  set {<access key>  <secret
  key> | --keyfile filePath}
  ```

- Create an AFM relationship for the remote S3 bucket as shown in Example 4-8.

Example 4-8 Creating AFM relationship for a remote S3 bucket

```
# mmafmcosconfig <Device> <FilesetName>  --endpoint
http[s]://{[Region@]Server|ExportMap}[:port]   --object-fs   --bucket
<BucketName>   --mode <AccessMode> --dir <Path> --debug
Where,
   <Device> is the name of your Storage Scale filesystem
   <FilesetName> is name of the fileset that you want created corresponding to
   the remote S3 object
   <BucketName> is the name of the actual remote S3 bucket
   <mode> is the AFM Access Mode
   <Path> is the relative directory path under the filesystem mount directory,
   where you want the fileset to be mounted (LinkPath).
```

Note the **`--dir`** parameter passed to the command. This is done to ensure that the fileset is created under the S3 “New Buckets Path” (from the command “**`mms3 account list`**”).

- To see the newly created fileset, run:

```
# mmlsfileset <Device>
```

To see the relationship of the fileset with the remote bucket, run:

```
# mmafmctl <Device> getstate
Where <Device> is the name of the Storage Scale filesystem
```

- Create a S3 bucket over the fileset's mount path. Change the ownership of the directory to that of the account corresponding to the S3 bucket.

```
# chown   <s3 account user>:<s3 account group> <fileset mount path>
```

Create the S3 bucket pointing to that directory:

```
# mms3 bucket create <bucket-name>--accountName <S3 account name>
--filesystemPath <fileset mount path>/<bucket-name>
```

Example:

In this example, there is a remote S3 bucket named “chm-cos-s3-bucket” residing on IBM Cloud Object Storage (IBMCOS). The following steps illustrate the steps for creating a virtual/accelerated S3 bucket named "b-watsonx" corresponding to the IBMCOS bucket.

Designate an AFM node:

```
# mmchnode --gateway -N ess3200b.bda.scale.ibm.com
```

Run the following commands on the AFM node.

- Create access keys in AFM corresponding to the IBMCOS object:

```
# mmafmcoskeys chm-cos-s3-bucket:s3.us-east.cloud-object-storage.appdomain.cloud \
set  <Your AWS_ACCESS_KEY_ID> \ <Your AWS_SECRET_ACCESS_KEY>
```

- View the AFM access key

```
# mmafmcoskeys all get --report version=1
chm-cos-s3-bucket:s3.us-east.cloud-object-storage.appdomain.cloud=COS:<Your
AWS_ACCESS_KEY_ID>:<Your AWS_SECRET_ACCESS_KEY>
```

- Create an AFM relationship for the IBMCOS bucket:

```
# mmafmcosconfig essData ibmcos-bucket \
--endpoint  http://s3.us-east.cloud-object-storage.appdomain.cloud  \
--object-fs \
```

```
--bucket chm-cos-s3-bucket \
--mode iw \
--dir watsonx/ibmcos-bucket --debug
```

Note: the value of **`-dir`** parameter above is chosen so, because the filesystem essData mount path is /gpfs/essData and the default path for S3 buckets (from the command **`mms3 account list`**) is /gpfs/essData/watsonx. This has the effect of creating the mount point of the fileset ibmcos-bucket at a relative path watsonx/ibmcos-bucket corresponding to the filesystem mount point, over which we will create a S3 bucket later.

The command produces following output shown in Example 4-9.

Example 4-9 Output of the mmafmcosconfig command

```
afmobjfs=essData fileset=ibmcos-bucket
bucket=chm-cos-s3-bucket newbucket= objectfs=yes dir=watsonx/ibmcos-bucket
policy= tmpdir= tmpfile= noDirectoyObject=no mode=iw remoteUpdate=no
xattr=no ssl=no autoRemove=no fastReaddir=no acls=no gcs=no vhb= cleanup=no
fastReaddir2=no lazyMigrate=no azure=no
bucketName=chm-cos-s3-bucket region=
serverName=s3.us-east.cloud-object-storage.appdomain.cloud cacheFsType=http
map= cacheHost=s3.us-east.cloud-object-storage.appdomain.cloud
Linkpath=/gpfs/essData/watsonx/ibmcos-bucket
target=http://s3.us-east.cloud-object-storage.appdomain.cloud/chm-cos-s3-bucket
endpoint=s3.us-east.cloud-object-storage.appdomain.cloud ENDPOINT=--endpoint
http://s3.us-east.cloud-object-storage.appdomain.cloud
XOPT= -p afmParallelWriteChunkSize=0 -p afmParallelReadChunkSize=0
```

- To view the newly created AFM relationship, run:

```
# mmafmctl essData getstate
Fileset Name    Fileset Target                              Cache State
Gateway Node    Queue Length   Queue numExec
------------    --------------                              -------------
------------    ------------   -------------
ibmcos-bucket
http://s3.us-east.cloud-object-storage.appdomain.cloud:80/chm-cos-s3-bucket
Active ems3000.bda.scale.ibm.com 0   3
```

Note: The output shows the AFM gateway node(s). The Cache State should be “Active” to indicate that the storage acceleration is working properly.

- Now create a S3 bucket over the directory /gpfs/essData/watsonx/ibmcos-bucket

```
# chown watsonx:users ibmcos-bucket/
# mms3 bucket create b-watsonx-cos --accountName watsonx --filesystemPath
/gpfs/essData/watsonx/ibmcos-bucket
Starting to create bucket with name b-watsonx-cos
Note: The directory '/gpfs/essData/watsonx/ibmcos-bucket' for bucket already
exists. Skipping update of ownership and the setting of permissions of the
directory for the user with uid:gid=2002:100
Bucket b-watsonx-cos created successfully
```

4.4 Define IBM Storage Scale S3 buckets to IBM watsonx.data

Install IBM watsonx.data using the documented procedure at Installing watsonx.data.

Use the following procedure to register a IBM Storage Scale S3 bucket to your watsonx.data instance as externally managed storage and associate a catalog for the bucket. This catalog serves as the query interface for watsonx.data for the data stored within the bucket. Refer to Figure 4-2.

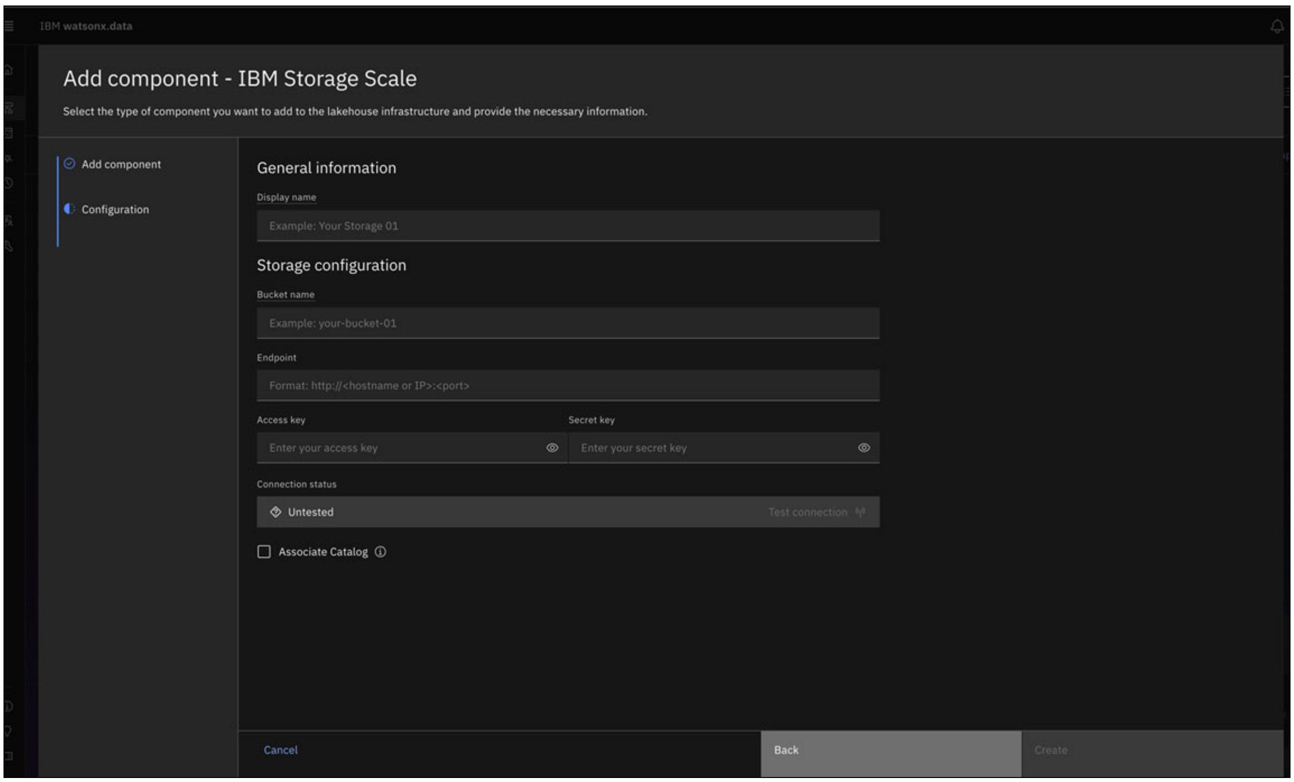


Figure 4-2 watsonX.data panel for adding IBM Storage Scale component

- Log in to watsonx.data console.
- From the navigation menu, select Infrastructure Manager.
- Click Add component.
- In the Add component window, select the IBM Storage Scale tile and provide the details of the S3 bucket.
- Bucket name - Enter the actual name of the S3 bucket as known to the IBM Storage Scale cluster. This could be a local or an accelerated bucket in IBM Storage Scale.
- Display name - Choose a display name of the bucket.
- Endpoint - Enter the endpoint URL. The URL is in the form of a http(s)://<IP Address>:<port>.
  - For <IP Address>, substitute this with the output of **`mmces address list`** command as explained in Example 4-2.
  - Refer to the output of **`mms3 config list`** command for the port number used by the S3 service, which is 6001 for http and 6443 for https by default.

> **Note:** For higher throughput and performance from the S3 service, a load balancer may be used, which works by distributing the workload among all protocol nodes. If a load balancer is configured, make sure to use the DNS name of the balancer in the endpoint URL instead of using the CES IP directly. For more details on using a load balancer, see IBM Storage Scale documentation Load balancing.

- Access key - Enter your access key.
- Secret key - Enter your secret key.
- Connection Status - Click the Test connection link to test the bucket connection.
- Associate Catalog - Select the check box to add a catalog for your storage.
- Catalog type - Select the catalog type from the list. The recommended catalog is Apache Iceberg. The other options for catalog are Apache Hive, Apache Hudi, and Delta Lake.
- Catalog name - Choose a name for the associated catalog.

To add a bucket-catalog pair, see Adding a storage-catalog pair.

Create an engine instance such as Presto and associate the catalog to that engine. This will make the S3 bucket discoverable through the catalog. Then continue to create schemas and tables under the storage catalog, as shown in the following command:

```
create schema <catalog name>.<schema name> with (location = 's3a://<bucket
name>/<directory for schema> ');
```

For example, if the name of the bucket is b-watsonx and the catalog name is c_scale, create a schema named “schema1” by running:

```
create schema c_scale.schema1 with (location = 's3a://b-watsonx/schema1');
```

This creates a directory called schema1 under the bucket's filesystem path in IBM Storage Scale. Data for all tables created under this schema would reside underneath this directory.

4.4.1 watsonx.data GUI view of the analytics infrastructure

The diagram, Figure 4-3, shows a sample view of the watsonx.data Infrastructure manager GUI including multiple engines, services, catalogs and IBM Storage Scale buckets as known to the environment. The diagram showcases the separation of compute, data and metadata layers and highlights the disaggregated nature of the architecture in terms of plug and play enabled modular components.

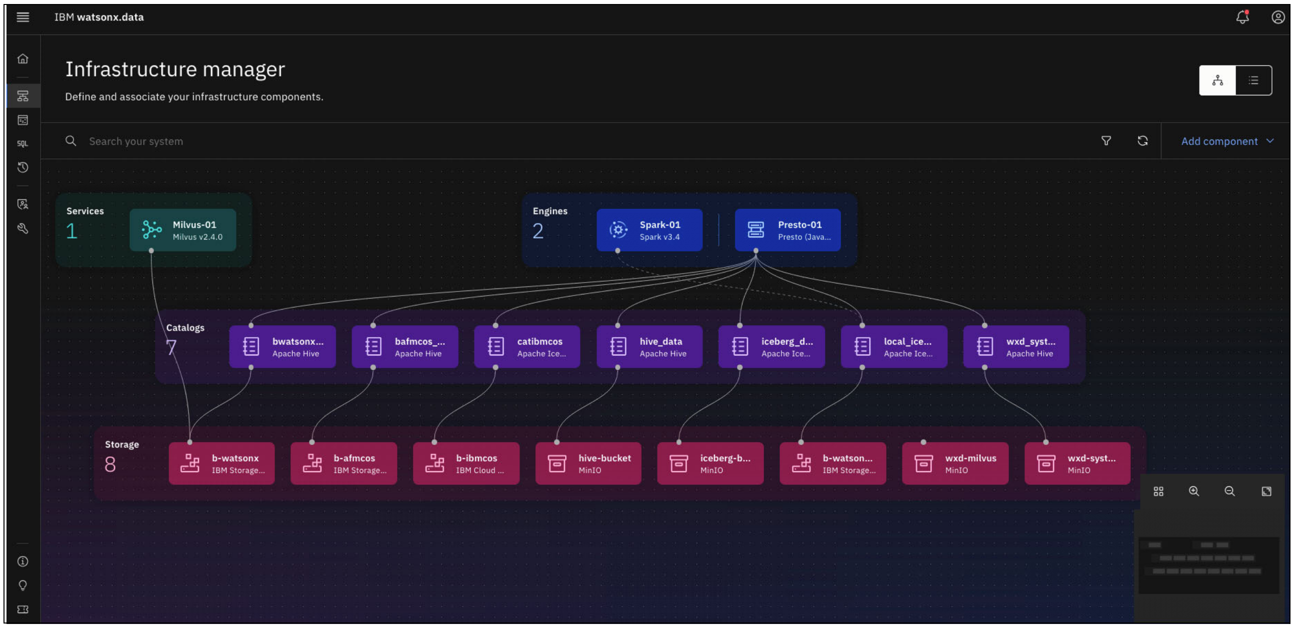


Figure 4-3 Sample view of the watsonx.data Infrastructure manager GUI

5

Monitoring

Since watsonx.data runs within OpenShift cluster, you can use all the standard monitoring features available within OpenShift to monitor watsonx.data projects or namespaces. Additionally, you can avail the monitoring capabilities natively available within watsonx.data and also with IBM Storage Scale.

5.1 Monitoring watsonx.data

watsonx.data has a built-in monitoring interface for the Presto engine. Routes are automatically created for each Presto (Java) engine that is provisioned. For details see Exposing secure route to Presto server.

Within the OpenShift cluster, the URL route can be queried - while passing the namespace of the watsonx.data installation “-n ${PROJECT_CPD_INST_OPERANDS}" - as follows:

```
# oc get route -n ${PROJECT_CPD_INST_OPERANDS} | grep presto
ibm-lh-lakehouse-presto-01-presto-svc
ibm-lh-lakehouse-presto-01-presto-svc-cpd-instance-test.apps.ocp4x.scale.ibm.com
ibm-lh-lakehouse-presto-01-presto-svc   8443                    reencrypt
None
```

The route's URL (in above example the URL ibm-lh-lakehouse-presto-01-presto-svc-cpd-instance-test.apps.ocp4x.scale.ibm.com) can be accessed via a browser to monitor running queries, query ID, query text, query state, percentage completed, username and source from which this query originated etc. The views are interactive, clicking on the queries will display further details. This is illustrated in Figure 5-1 which was captured during a TPCD-S query benchmark run in Presto.

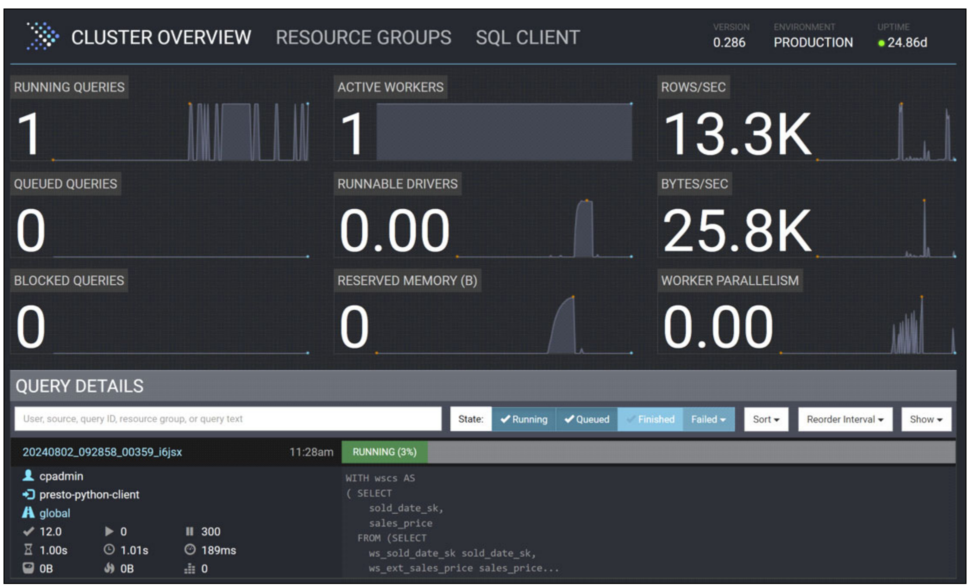


Figure 5-1 Presto UI for monitoring queries

5.2 Monitoring IBM Storage Scale S3 service

In addition to the standard storage monitoring available for IBM Storage Scale, there is a new monitoring endpoint for S3 service. These so-called Namespace-Filesystem (NSFS) metrics can be collected over HTTP and returns a well-defined metrics object in JSON.

The monitoring endpoint is specific for a S3 server, i.e., in an environment of multiple S3 protocol nodes, each node exposes such an endpoint. This allows for fine-grained monitoring and analysis, for example, when multiple S3 nodes are actively in use, the monitoring can show if the load is evenly balanced across the nodes.

The S3 monitoring endpoint is available at http://<host>:7004/metrics/nsfs_stats, so the data can be queried directly, for example:

```
# curl http://10.10.1.121:7004/metrics/nsfs_stats
{"nsfs_counters":{"noobaa_nsfs_io_read_count":0,"noobaa_nsfs_io_write_count":1,
"noobaa_nsfs_io_read_bytes":0,"noobaa_nsfs_io_write_bytes":4},"op_stats_counter
s":{"noobaa_nsfs_op_upload_object_count":1,"noobaa_nsfs_op_upload_object_error_
count":0}}
```

The output can be post-processed to be pretty-printed for better readability:

```
# curl -s http://10.10.1.121:7004/metrics/nsfs_stats | python3 -m json.tool
```

As the result is JSON, the output can be further parsed with a JSON parser to be post-processed, for example for scripting and automating purposes. Obtaining the write count in the above output is as simple as querying the respective field, for example:

```
# curl -s http://10.10.1.121:7004/metrics/nsfs_stats | jq -r
'.nsfs_counters.noobaa_nsfs_io_write_count'
1
```

These probes and metrics can be exploited by monitoring tools like Grafana to get an overview of the system, and further be extended into integrated monitoring frameworks. to build a much more complex analysis pipeline.

6

Configuring advanced storage functions

This chapter describes some advanced storage functions offered by IBM Storage Scale which may be leveraged for production use.

6.1 Enabling encryption of data at rest for S3 buckets

IBM Storage Scale allows encryption of data at a file system or at a fileset level. If the whole filesystem is enabled for encryption, all the S3 buckets created within, would have their respective objects encrypted. Otherwise, encryption may be enabled at a per-fileset level to allow only the specific S3 buckets created over those filesets to have their objects encrypted.

Follow the instructions outlined in Simplified setup: Using SKLM with a self-signed certificate to enable encryption for the IBM Storage Scale cluster.

- Follow Part 1 of the document: “Installing and configuring SKLM” to setup crypto servers. They serve as key managers for IBM Storage Scale nodes.
- Then follow Part 2: Configuring the Scale cluster for encryption.

The following example shows how to setup encryption at a fileset level.

- Create an independent fileset first:

```
# mmcrfileset essData fset1 --inode-space new
```

- Link the fileset to a directory. We will link to a directory under the default bucket path

```
# mmlinkfileset essData fset1 -J /ibm/essData/acc-user3/cmencrybuck/
```

In our environment, if we have the following configuration:

```
UUID = KEY-e030d65-665d3532-6aed-4727-828d-8be768e556b9
rkm id == crypto1_devG1
fileset name = fset1
filesystem name = essData
```

- Create a policy file (e.g. /root/enc.pol) with the following content (example). This defines a rule to encrypt any file under the fileset named fset1 with our crypto keys.

```
RULE 'p1' SET POOL 'system'
RULE 'Encrypt all files in file system with rule E1'
SET ENCRYPTION 'E1'
FOR FILESET ('fset1')
WHERE NAME LIKE '%'
RULE 'simpleEncRule' ENCRYPTION 'E1' IS
ALGO 'DEFAULTNISTSP800131A'
KEYS('KEY-e030d65-665d3532-6aed-4727-828d-8be768e556b9:crypto1_devG1')
```

- Apply the policy. The policy is applied to the filesystem and evaluation of each policy evaluates specific criteria. The policy file can have mix of placement rules, encryption rules, compression rules etc. together. We can also apply more than one key rule to encrypt files, e.g. encrypt fileset1 with key 1, fileset2 with key2 etc.

```
# mmchpolicy essData /root/enc.pol
```

- Create a file under the encrypted directory and verify that the file is encrypted.

```
# cp /etc/redhat-release /ibm/essData/acc-user3/cmencrybuck/
# mmlsattr -n gpfs.Encryption
/ibm/essData/acc-user3/cmencrybuck/redhat-release  |grep gpfs.Encryption
gpfs.Encryption:      "EAGC??????8/??????????? ????|?? ??5?????????????
??]~"??A?m???*?=p+[?3?c???@cn9?9y0g,???'H1?KEY-e030d65-665d3532-6aed-4727-828d-
8be768e556b9?crypto1_devG1?"
```

- For an un-encrypted file, the result would be as following:

```
# mmlsattr -n gpfs.Encryption /ibm/essData/redhat-release  |grep gpfs.Encryption
gpfs.Encryption:      No such attribute
```

- Once encryption has been enabled for the fileset, define a S3 bucket on the fileset's mount point directory using the regular procedure.

6.2 Enabling SSL for secure data transfer

In production environments, a secure connection between watsonx.data and IBM Storage Scale S3 service is fundamental. This can be realized with state-of-the-art Transport Layer Security (TLS) by establishing a https connection when connecting to the S3 service. This requires setting up the S3 service with a certificate signed by a certificate authority (CA). For in-house and typically firewall-protected environments, self-signed certificates can be an alternative without incurring the additional costs of a CA-signed certificate. The setup is done in two steps:

1. Creating and setting up an SSL certificate on the IBM Storage Scale S3 cluster
2. Configuring watsonx.data to accept the certificate

6.2.1 Enabling SSL for the IBM Storage Scale S3 cluster

Setup a self-signed SSL certificate for your Storage Scale cluster with the steps mentioned in Setting up self-signed SSL/TLS certificates.

A sample of the Subject Alternative Name (SAN) file as stated in the above documentation is shown in Example 6-1, containing a CES IP and the corresponding DNS name.

Example 6-1 A sample SAN file

```
# cat san.cnf
[req]
req_extensions = req_ext
distinguished_name = req_distinguished_name

[req_distinguished_name]
CN = localhost

[req_ext]
subjectAltName = DNS:localhost,DNS: cesip1.bda.scale.ibm.com,IP: 10.10.1.121
```

6.2.2 Enabling watsonx.data for SSL for secure data access

On the API node of the OpenShift Container cluster, the watsonx.data engines needs to be patched with the IBM Storage Scale certificate (tls.crt) created. section 6.2.1. After the login to the OpenShift cluster using **oc login**, the certificate can be patched with exporting its content in one line and masked line endings as follows

```
# export CERT="-----BEGIN CERTIFICATE-----\n<actual certificate
content>\n-----END CERTIFICATE-----\n"
```

Note: Remove the newline/line-break characters from the actual certificate content.

and the **oc patch** command

oc patch wxd/lakehouse --type=merge -n <namespace > -p "{ \"spec\": { \"update_ca_certs\": true, \"extra_ca_certs_secret\": \"$CERT\" } }"

The **oc patch** command restarts the compute engines. Wait until the restart is complete and continue to register IBM Storage Scale buckets using a secure (https) endpoint.

6.3 Data sharing for S3 workflows

IBM watsonx.data supports different workflows for shared data access across applications, for example, user-based access to processing engines like Presto. As the data resides on IBM Storage Scale, complex workflows using multi-protocols can be realized, for example, ingesting data using NFS or Posix and processing the same data using the S3 object protocol. Yet even onan S3 bucket level, the shared data access can be configured for extended workflows. This is illustrated in Figure 6-1.

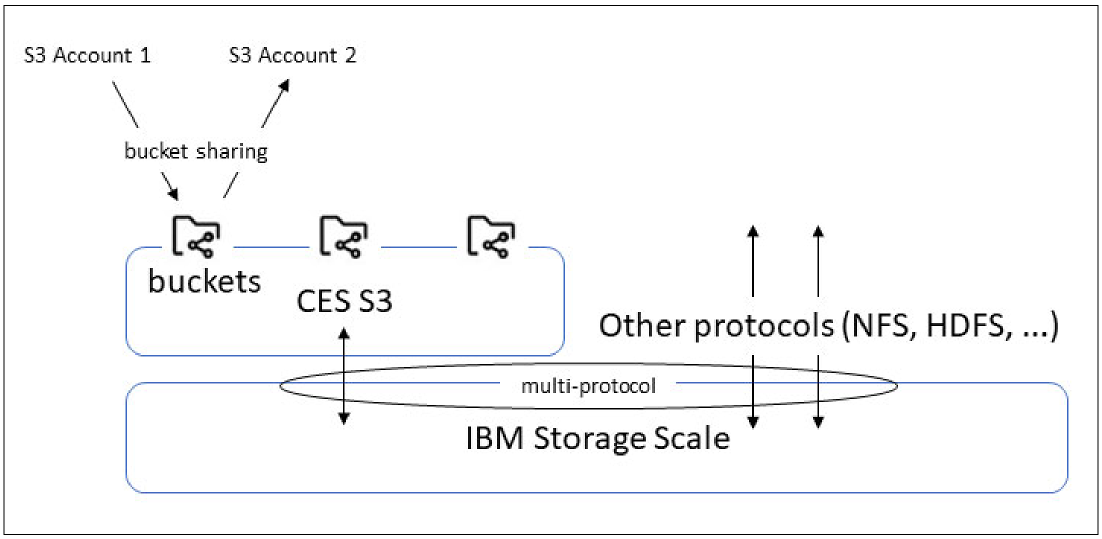


Figure 6-1 Data sharing for S3 workflows

6.3.1 Example A. Data sharing using multi-protocols

In an example of multi-protocol data access, a Hive external table (i.e. a non-managed table) may be defined over an IBM Storage Scale directory path underneath a bucket, while the directory acts as an external data repository containing existing table data. In the following example, customer has data files in .csv (flatfile) format, in a directory called “external_data” under the bucket b-watsonx that was created in “Creating S3 buckets” on page 28. Within Presto, a table named as “myexttable”is created under catalog b_watsonx_data and schema “myschema” with the following syntax, pointing to that directory

```
create table b_watsonx_data.myschema.myexttable (name varchar, id INT)
with (external_location = 's3a://b-watsonx/external_data',format='csv');
```

Once the table is defined, it is possible to view the existing data with SQL queries in Presto. Users may ingest more datafiles to the same directory via NFS/Posix or may even update the existing datafiles (e.g. update or append records) outside of Presto, and the same would be reflected in any subsequent SQL queries run from Presto. This feature can be leveraged to realize complex workflows within the enterprise data pipeline.

6.3.2 Example B. Data sharing at a S3 bucket level

In another example of data sharing within a bucket, a bucket can be defined that belongs to an ingest job and the same bucket can be made available for another S3 account, for example as read-only to process the data in the bucket. This can be achieved using typical S3 bucket policies and the s3api put-bucket-policy command, for example as the owner of the bucket activate a policy using:

```
aws --endpoint https://<CES-IP>:<port> s3api put-bucket-policy --bucket <bucket
name> --policy file://<path-to-file>
```

Allowing access to a bucket can be configured using the prior command with the following example policy.

```
$ cat policyReadWrite.json
{
  "Version":"2012-10-17",
  "Statement":[{
    "Sid":"policyReadWrite",
    "Effect":"Allow",
    "Principal": { "AWS": "userReadWrite" },
    "Action":["s3:*"],
    "Resource":"*"}]
}
```

or with the following example policy to allow for read only access:

$ cat policyReadOnly.json

```
{
  "Version":"2012-10-17",
  "Statement":[{
    "Sid":"policyReadOnly",
    "Effect":"Allow",
    "Principal": { "AWS": "userReadOnly" },
    "Action":["s3:GetObject", "s3:ListBucket"],
    "Resource":"*"}]
}
```

Data sharing at IBM Storage Scale S3 buckets level- example

Next, taking the steps together, the following steps show the complete flow. As example, users userMain and userReadOnly are created with their respective S3 accounts. For simplicity, define the following command aliases as

```
# access/secret as provided while creating the S3 accounts
$ alias s3uMain='AWS_ACCESS_KEY_ID=access... AWS_SECRET_ACCESS_KEY=secret...
aws --endpoint https://10.10.1.121:6443 s3'
$ alias s3uReadOnly='AWS_ACCESS_KEY_ID=access...
AWS_SECRET_ACCESS_KEY=secret... aws --endpoint https://10.10.1.121:6443 s3'
```

The commands can be run virtually from any system and as any console user, the access key and secret key combination define the S3 account user for the commands that are executed. On the system, there is a bucket for the main user as shown in Example 6-2.

Example 6-2 Viewing details of the bucket to be used for data sharing

```
$ mms3 bucket list

 Name
 ------
 b-userMain

$ mms3 bucket list --wide
{
"response": {
    "code": "BucketList",
    "reply": [
      {
        "_id": "6684fb63350732225d308367",
        "name": "b-userMain",
        "owner_account": "66841963b5280312429672fb",
        "system_owner": "userMain",
        "bucket_owner": "userMain",
        "versioning": "DISABLED",
        "creation_date": "2024-07-03T07:18:59.310Z",
        "path": "/gpfs/essData/userMain/b-userMain",
        "should_create_underlying_storage": false,
        "fs_backend": "GPFS"
    ]
  }
}
```

Access to this bucket by another user results in an Access Denied response:

```
$ s3uReadOnly ls s3://b-userMain
An error occurred (AccessDenied) when calling the ListObjectsV2 operation:
Access Denied
```

The main user can grant access, for example read only, to the bucket the user owns, this is done using the s3api call using the policy as shown in Example 6-3.

Example 6-3 Bucket owner granting read-only access to the bucket

```
$ alias s3uMainApi='AWS_ACCESS_KEY_ID=access... AWS_SECRET_ACCESS_KEY=secret... aws
--endpoint https://10.10.1.121:6443 s3api'
$ s3uMainApi put-bucket-policy --bucket b-userMain --policy file://policyReadOnly.json
```

After that, the other user can access the bucket:

```
$ s3uReadOnly ls s3://b-userMain
2024-07-10 10:44:50          4 myobj
```

But as the policy states, read only access is granted, so write access still results in an Access Denied response:

```
$ s3uReadOnly cp myobj s3://b-userMain
upload failed: ./myobj to s3://b-userMain/myobj An error occurred
(AccessDenied) when calling the PutObject operation: Access Denied
```

This example of data sharing at IBM Storage Scale S3 buckets level illustrates a simple scenario of data sharing at the level of S3 buckets.

Related publications

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

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only. For the current online list of IBM Storage Scale Redbooks select here.

- *IBM Storage Scale System Introduction Guide*, REDP-5729
- *IBM Hybrid Solution for Scalable Data Solutions using IBM Spectrum Scale*, REDP-5549
- *IBM Spectrum Scale and IBM Elastic Storage System Network Guide*, REDP-5484
- *Accelerating IBM watsonx.data with IBM Fusion HCI*, REDP-5720

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

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- IBM watsonx.data

  https://www.ibm.com/products/watsonx-data
- Product Documentation for IBM watsonx.data

  https://www.ibm.com/docs/en/watsonx/watsonxdata
- IBM Storage Scale

  https://www.ibm.com/products/storage-scale
- Product Documentation for IBM Storage Scale

  https://www.ibm.com/docs/en/storage-scale
- Product Documentation for IBM Storage Scale System

  https://www.ibm.com/docs/en/storage-scale-system
- How to sync externally managed Iceberg tables with the catalog integration in watsonx.data (blog)

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

REDP-5743-00

ISBN

Printed in U.S.A.